University of Ottawa

PhD Thesis

Analysis of IoT Spatial and Spatiotemporal Data: a Smart Farming Use Case

Author:
Patrick Killeen

Supervisor:
Dr. Iluju Kiringa
Dr. Tet Yeap

A thesis submitted to the University of Ottawa in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Smart IoT Lab
Faculty of Engineering, School of Electrical Engineering and Computer Science

© Patrick Killeen, Ottawa, Canada, 2025

Dedication

I dedicate this thesis to my best friend Polly, my pet quaker (or monk) parrot, who was perched by my side during my entire journey as a graduate student. His encouraging (not distracting) squawks and 'what-chya-doing's guided and inspired me towards great research.

List of Publications

Journal

1. Patrick Killeen, Ci Lin, Futong Li, Iluju Kiringa, and Tet Yeap. "Nitrous Oxide Emission Prediction Using IoT Soil and Weather Sensor Data," *ACM Journal on Computing and Sustainable Societies*, 2025, just accepted.

2. Patrick Killeen, Ci Lin, Futong Li, Iluju Kiringa, and Tet Yeap. "IoT-Based Smart Farming Architecture Using Federated Learning: a Nitrous Oxide Emission Prediction Use Case," *ACM Journal on Computing and Sustainable Societies*, vol. 3, no. 2, 2025.

3. Ci Lin, Futong Li, Patrick Killeen, Tet Yeap, and Iluju Kiringa. "Agriculture-informed Neural Networks for Predicting Nitrous Oxide Emissions," *ACM Transactions on Internet of Things*, 2024, pp. 1–24.

4. Patrick Killeen, Iluju Kiringa, Tet Yeap, and Paula Branco. "Corn Grain Yield Prediction Using UAV-Based High Spatiotemporal Resolution Imagery, Machine Learning, and Spatial Cross-Validation," *Remote Sensing*, vol. 16, no. 4, 2024, p. 683.

5. Patrick Killeen, Iluju Kiringa, and Tet Yeap. "Unsupervised Dynamic Sensor Selection for IoT-based Predictive Maintenance of a Fleet of Public Transport Buses," *ACM Transactions on Internet of Things*, 2022, pp. 1–36.

Conferences

1. Patrick Killeen, Ci Lin, Iluju Kiringa, and Tet Yeap. "Nitrous Oxide Emission Prediction by Combining Process-based Models and Neural Networks," *2025 12th International Conference on Future Internet of Things and Cloud (FiCloud)*, Istanbul, Turkiye, 2025, pp. 334–341.

2. Gautam Vira, Patrick Killeen, Tet Yeap, and Iluju Kiringa. "Predictive Maintenance by Detection of Gradual Faults in an IoT-Enabled Public Bus," *in 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).* IEEE, 2024, pp. 570–576.

3. Tet Yeap, Iluju Kiringa, Paula Branco, Patrick Killeen, Ci Lin, Bernard Asare, David Owusu, and Futong Li. "Predicting Crop Yields and Nitrous Oxide Emissions Using Data Collected at the Area X.0 Smart Farming Research Fields: A Progress Report," *in Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering*, 2023, pp. 245–247.

4. Patrick Killeen, Iluju Kiringa, Tet Yeap, and Paula Branco. "Using UAV-Based Multispectral Imagery, Data-Driven Models, and Spatial Cross-Validation for Corn Grain Yield Prediction," in *2023 IEEE International Conference on Data Mining (ICDM).* IEEE, 2023, pp. 220–228.

5. Moswa, A.B., Killeen, P., Kiringa, I., Yeap, T. (2023). Corn Yield Prediction Using Crop Growth and Machine Learning Models. In: Chinara, S., Tripathy, A.K., Li, KC., Sahoo, J.P., Mishra, A.K. (eds) *Advances in Distributed Computing and Machine Learning. Lecture Notes in Networks and Systems*, vol 660. Springer, Singapore.

6. Patrick Killeen, Iluju Kiringa, and Tet Yeap. "Corn Grain Yield Prediction Using UAV-based High Spatiotemporal Resolution Multispectral Imagery," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2022, pp. 1054–1062.

7. Patrick Killeen, Bo Ding, Iluju Kiringa, and Tet Yeap, "IoT-based predictive maintenance for fleet management", *Procedia Computer Science*, vol. 151, 2019, pp. 607–613.

Submitted to Journal

1. Patrick Killeen, Tet Yeap, and Iluju Kiringa. "UAV Imagery-Based Yield Prediction Using Federated Learning and Model Sharing for Smart Farming Applications".

Abstract

Farmers will be expected to produce more food due to an increasing population despite challenges like climate change. Precision agriculture (PA) and smart farming can be used to help farmers achieve this goal by reducing input costs and enabling agriculture resource optimizations. Smart farming can enable PA by gathering site-specific agriculture data (agri-data) from a) sensors using in-field gateways and/or b) other data sources. The high spatiotemporal variability of agri-data, privacy concerns, and high system deployment costs act as challenges for PA. PA model performance evaluation risks being over-optimistic if spatial (and/or temporal) structure (such as spatial autocorrelation) is not carefully considered in the model evaluation process. Block cross-validation (CV) can be used to address this to create folds of spatially (or temporally) disjointed blocks of data, although data from new locations (or time periods) may lead to pessimistic model extrapolation performance when using this evaluation technique. Cloud computing-based centralized learning (CL) could be used to train PA models, but CL does not scale well in the Internet of Things (IoT) setting and suffers from poor privacy. In addition to high system deployment costs, ignoring farmers' privacy concerns will lead to poor smart farming system (SFS) adoption rates. Limiting a SFS's farmer user-base would result in less available training data, and this could in turn negatively impact model performance. Fog computing-based local learning (LL) could instead be used, by training many local models at the edge of networks using only local data, but unfortunately, applying LL to agri-data may lead to loss of useful geographical trends. Distributed machine learning (ML) can be applied to address these challenges by only sharing model updates to train models.

We proposed an IoT SFS architecture that uses privacy-aware distributed ML to train PA models without having to share farmers' private data. Expensive sensing equipment is first required to train the models using expensive-to-sense ground truth data and affordable sensors' data, but once the training is complete, new farmers can join the system to benefit from the models without needing any expensive equipment. By leveraging data from a Canadian smart farm, we performed yield prediction and nitrous oxide (N_2O) emission prediction experiments as use cases to showcase the proposed architecture. We used various forms of spatial and temporal block CV for evaluating PA model performance using datasets of varying heterogeneity (independent and identically distributed (IID) and non-IID datasets). We performed experiments using CL, LL, federated learning, and distributed ensemble learning, where clients/nodes were simulated on a single machine. Our results showed that when using IID datasets, distributed ML could do reasonably well and even compete with CL in terms of model performance. However, the IID dataset experiment results may have been over-optimistic due to the stronger presence of spatial/temporal autocorrelation. When using non-IID datasets (which represents the more realistic scenario of having high spatiotemporal variability in agri-data), we found that distributed ML did more poorly and failed on multiple occasions. Despite this, by using distributed ML and non-IID datasets, we were able to generate useful yield precision maps for most clients. The results reported in this thesis demonstrate that the proposed smart farming IoT architecture combined with distributed ML can potentially be used for achieving high spatiotemporal resolution agri-data sensing in a manner that is a) privacy-aware, b) affordable, and c) scalable, at the expense of reduced sensing accuracy.

Acknowledgments

I would like to thank Invest Ottawa - Area X.O and GPS Ontario. Without their help, our yield prediction and nitrous oxide emission prediction research performed at the Area X.O smart farm would not have been possible. Furthermore, I thank my parents, Richard Killeen and Suzanne Kettley. Without their support, graduating from my PhD would likely have not been possible. I especially thank my mom for helping me proof read each and every research paper I submitted for publication during my graduate studies. I would also like to thank my supervisors, Iluju Kiringa and Tet Yeap, and I would like to thank those that were directly involved with my research, namely Audrey Moswa, Paula Branco, Gautam Vira, Ci Lin, and Futong Li.

Table of Contents

List of Publications iii

Abstract v

List of Tables xii

List of Figures xx

Nomenclature xxiv

1 Introduction 2
- 1.1 Motivation and Challenges . . . 2
  - 1.1.1 Yield Prediction . . . 4
  - 1.1.2 Nitrous Oxide (N_2O) Prediction . . . 5
- 1.2 Solution Outline . . . 6
- 1.3 Contributions . . . 7
  - 1.3.1 Computer Science . . . 7
  - 1.3.2 Crop Yield Prediction . . . 8
  - 1.3.3 N_2O Emission Prediction . . . 9
- 1.4 Outline of Thesis . . . 9

2 Background and Related Work 11
- 2.1 Internet of Things . . . 11
- 2.2 Cloud, Fog, and Edge Computing . . . 11
- 2.3 Machine Learning . . . 12
  - 2.3.1 Model Evaluation . . . 12
    - 2.3.1.1 Dataset Resampling Methods . . . 13
    - 2.3.1.2 Evaluation Metrics . . . 13
  - 2.3.2 Data-driven Models . . . 15
    - 2.3.2.1 K-means . . . 15
    - 2.3.2.2 Linear Regression . . . 15
    - 2.3.2.3 Partial Least Squares Regression . . . 16
    - 2.3.2.4 Random Forest . . . 16
    - 2.3.2.5 Support Vector Machine . . . 16
    - 2.3.2.6 ZeroR . . . 16
    - 2.3.2.7 Neural Networks . . . 17

2.3.3 Ensemble Learning . . . 19
2.3.4 Federated Learning . . . 19
2.3.4.1 Privacy Preserving Technologies . . . 20
2.3.4.2 Federated Averaging Algorithm . . . 21
2.3.4.3 Other Federated Learning Aggregator Algorithms . . . 23
2.4 Spatiotemporal Data Analysis . . . 23
2.4.1 Spatial Data . . . 23
2.4.2 Time Series Data . . . 24
2.4.3 Machine Learning on Spatiotemporal Data . . . 25
2.4.4 Block Cross-Validation . . . 26
2.5 Smart Farming . . . 26
2.6 Remote Sensing . . . 27
2.7 Crop Yield Data . . . 27
2.7.1 Cleaning . . . 27
2.7.2 Post-processing . . . 28
2.7.3 Prediction . . . 28
2.8 Nitrous Oxide (N_2O) . . . 29
2.8.1 Creation . . . 29
2.8.2 Measuring . . . 30
2.8.3 Predicting . . . 30
2.9 Related Work . . . 31
2.9.1 Smart Farming IoT Architectures . . . 32
2.9.2 Federated Learning in Agriculture . . . 34
2.9.3 UAV Imagery-based Yield Prediction . . . 36
2.9.4 Nitrous Oxide (N_2O) Emission Prediction . . . 39

3 IoT-based Smart Farming Architecture Using Privacy-aware Distributed Machine Learning 42
3.1 Architecture Layers . . . 43
3.2 Smart Farming System Components . . . 44
3.2.1 Predictor Node . . . 44
3.2.2 Root Node . . . 45
3.2.3 Application Protocols . . . 45
3.2.4 Remote Administration . . . 47
3.3 Predictive Analytics . . . 48
3.3.1 Distributed Machine Learning Methodology . . . 49
3.4 Applications . . . 53
3.5 Threat Model . . . 54
3.6 Meeting IoT-based Smart Farming Architecture Requirements . . . 55

4 Crop Yield Prediction 57
4.1 Introduction . . . 57
4.2 Adapted IoT Architecture . . . 58
4.2.1 System Deployment and Costs . . . 59
4.2.2 Benefits of Using the System . . . 60

4.2.3 Adopting Barriers . . . 61
4.3 Materials and Methods . . . 62
4.3.1 Study Site . . . 62
4.3.2 Data Collection . . . 62
4.3.2.1 Weather . . . 63
4.3.2.2 Management . . . 63
4.3.2.3 Imagery . . . 64
4.3.2.4 Yield . . . 65
4.3.3 Corn Growth Stages . . . 66
4.3.4 Vegetation Indices . . . 66
4.3.5 Model Selection . . . 68
4.3.5.1 Features Selection . . . 68
4.3.5.2 Hyperparameter Tuning . . . 68
4.3.6 Significance Testing . . . 69
4.4 Yield Prediction Study: Using UAV Imagery and Spatial Cross-validation . . . 70
4.4.1 Materials and Methods . . . 71
4.4.1.1 Datasets . . . 71
4.4.1.2 Corn Growth Stages . . . 71
4.4.1.3 Feature Extraction . . . 72
4.4.1.4 Model Hyperparameters . . . 73
4.4.2 Experiments . . . 73
4.4.2.1 Location-only Standard K-Fold Cross-validation . . . 74
4.4.2.2 Standard K-Fold Cross-validation . . . 74
4.4.2.3 Spatial Cross-validation . . . 74
4.4.3 Analysis and Results . . . 75
4.4.3.1 Imagery Type Analysis . . . 76
4.4.3.2 Effects of Spatial Autocorrelation on Performance . . . 79
4.4.3.3 Model Comparison . . . 80
4.4.3.4 High vs. Low Spatial Resolution Imagery Analysis . . . 81
4.4.3.5 Evaluation Method Comparison . . . 81
4.4.3.6 Imagery Acquisition Date Analysis . . . 84
4.4.3.7 Data Processing Time Analysis . . . 85
4.4.4 Discussion . . . 90
4.4.5 Conclusion . . . 92
4.5 Yield Prediction Study: Using UAV Imagery, Spatial Cross-validation, and Model Selection . . . 94
4.5.1 Materials and Methods . . . 94
4.5.1.1 Datasets . . . 94
4.5.1.2 Feature Extraction . . . 94
4.5.2 Experiments . . . 95
4.5.2.1 Model Selection . . . 96
4.5.3 Analysis and Results . . . 96
4.5.3.1 Small vs. Large Number of Features . . . 97
4.5.3.2 RGB vs. MS Imagery . . . 98
4.5.3.3 Model Spatial Generalizability . . . 98

4.5.3.4 Model Comparison . . . 99
4.5.3.5 Data Processing Time Analysis . . . 100
4.5.4 Conclusion . . . 104
4.6 Yield Prediction Study: Using UAV Imagery, Spatial Cross-validation, Model Selection, and Distributed Machine Learning . . . 105
4.6.1 Materials and Methods . . . 106
4.6.1.1 Datasets . . . 106
4.6.1.2 Feature Extraction . . . 106
4.6.1.3 Machine Learning Models . . . 106
4.6.1.4 Layered Block Cross-validation . . . 107
4.6.2 Experiments . . . 108
4.6.2.1 Model Selection . . . 109
4.6.2.2 Federated Learning and Model Sharing . . . 110
4.6.3 Analysis and Results . . . 112
4.6.3.1 Centralized Learning with Hyperparameter Tuning . . . 113
4.6.3.2 Baseline Model Performance . . . 113
4.6.3.3 Model Comparison . . . 119
4.6.3.4 Learning Method Comparison . . . 120
4.6.3.5 Resource Utilization . . . 123
4.6.4 Conclusion . . . 124

5 Nitrous Oxide (N_2O) Emission Prediction **128**
5.1 Introduction . . . 128
5.2 Adapted IoT Architecture . . . 130
5.2.1 System Deployment and Costs . . . 130
5.2.2 Benefits of Using the System . . . 131
5.2.3 Adopting Barriers . . . 131
5.3 Materials and Methods . . . 132
5.3.1 Study Site . . . 133
5.3.2 Data Collection . . . 133
5.3.3 Predictor Node Prototype . . . 135
5.3.4 Evaluation Methods . . . 136
5.3.4.1 Block Cross-validation . . . 136
5.3.4.2 Seasonal Performance . . . 137
5.4 N_2O Emission Prediction Study: Using Temporal Cross-validation, Single-year Interpolation, and Multi-year Extrapolation . . . 138
5.4.1 Materials and Methods . . . 139
5.4.1.1 Datasets . . . 139
5.4.1.2 Data Pre-processing . . . 139
5.4.1.3 Machine Learning Models . . . 140
5.4.1.4 Model Selection . . . 140
5.4.2 Experiments . . . 141
5.4.2.1 Single-year Interpolation Experiments . . . 142
5.4.2.2 Multi-year Extrapolation Experiments . . . 142
5.4.3 Analysis and Results . . . 143

5.4.3.1 Single-year Interpolation . . . 143
5.4.3.2 Multi-year Extrapolation . . . 151
5.4.4 Conclusion . . . 163
5.5 N_2O Emission Prediction Study: Using Temporal Cross-validation, Single-year Interpolation, and Distributed Machine Learning . . . 165
5.5.1 Materials and Methods . . . 166
5.5.1.1 Datasets . . . 166
5.5.1.2 Data Pre-processing . . . 166
5.5.1.3 Machine Learning Models . . . 167
5.5.1.4 Evaluation Methods . . . 168
5.5.2 Experiments . . . 168
5.5.2.1 Model Selection . . . 168
5.5.2.2 Federated Learning and Model Sharing . . . 169
5.5.3 Analysis and Results . . . 170
5.5.3.1 IID vs. non-IID Client Datasets . . . 171
5.5.3.2 Learning Method Comparison . . . 171
5.5.3.3 Model Comparison . . . 181
5.5.3.4 Resource Utilization . . . 181
5.5.4 Conclusion . . . 185

6 Conclusion **187**
6.1 Summary . . . 187
6.2 Limitations . . . 189
6.3 Future Work . . . 190

Appendices **192**

A Vegetation Index Equations **193**

B Comparison of Related Work **197**

C Model Hyperparameters **201**

D Supplementary Material **206**

References **218**

List of Tables

2.1 Comparison of the costs and benefits of the various N_2O emission measuring techniques (including the prediction-based approach), where pixel-level indicates 1 m × 1 m sampling areas are supported and field-level indicates sampling areas beyond 10 m × 10 m are supported. 31

4.0.1 System requirements for using the PIX4Dmapper software (suggested by PIX4D [205], such that raw image resolutions are assumed to be 14 MP) for generating orthomosaics from raw UAV imagery. 61
4.0.2 Yield prediction studies data source descriptions 63
4.0.3 Tillage technique, seeding rates, herbicide application rates, and fertilizer application rates for each field, where CT = conventional tilling (or broadcast), IST = innovative strip-tilling (or fertile stripping), LP = lime pellets, kS/ac = 1000 seeds per acre, 1 lbs ≈ 0.454 kg, 1 gal ≈ 3.785 L, 1 ha ≈ 2.471 ac, and the 7-32-23 notation indicates (7 % N, 32 % P, and 23 % K) fertilizer. 64
4.0.4 Field management activity dates. 64
4.0.5 UAV image acquisition details over the 2021 growing season 65
4.0.6 Mean (averaged over all the missions conducted in 2021) UAV imagery acquisition mission properties for a single mission, where results in brackets are the results multiplied by a factor of five to represent the inclusion of imagery from all five MS bands, where RGB = red-green-blue and MS = multispectral. 66
4.4.1 Estimated corn growth stages [190, 55] for the 2021 growing season using accumulated GDD [177, 188] and in-field crop camera imagery, where DoY = day of year, AGDD = accumulated growing degree days, and GS = growth stage. 72
4.4.2 Yield prediction experiments: five best-performing VIs on average over the growing season by LR model and LOFO-CV and rev-LOFO-CV evaluation methods, where VIs listed in cyan-, orange-, and black-colored font are NIR-, red-edge-, and RGB-based VIs, respectively; LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and the VIs are defined in Appendix A in Tables A.1, A.2, and A.3. 76

4.4.3 Yield prediction experiments: five worst-performing VIs on average over the growing season by LR model and LOFO-CV and rev-LOFO-CV evaluation methods, where VIs listed in cyan-, orange-, and black-colored font are NIR-, red-edge-, and RGB-based VIs, respectively; LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and the VIs are defined in Appendix A in Tables A.1, A.2, and A.3. 77
4.4.4 Yield prediction experiments: average R^2 performance of RF and LR over the entire season for LRe and rev-LOFO-CV, and for MS imagery (NIR-based and red-edge-based VIs) and RGB imagery, where RGB = red-green-blue, MS = multispectral, and DoY = day of year. 81
4.4.5 Yield prediction experiments: combined average LR and RF R^2 performance of each field over the entire season for SpKF-CV and rev-SpKF-CV, HRe and LRe, and for MS imagery (NIR-based and red-edge-based VIs) and RGB imagery, where SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, RGB = red-green-blue, and MS = multispectral. 83
4.4.6 Yield prediction experiments: average (sampled over three acquisition dates) image pre-processing execution time (min = minute and h = hour), where VI = vegetation index, RGB = red-greed-blue, MS = multispectral, and smaller fields = Fields 1 to 9. 89
4.4.7 Yield prediction experiments: average (sampled over three acquisition dates) execution time (min = minute) to split and create all datasets for all VIs and fields, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, RGB = red-greed-blue, and MS = multispectral. 89
4.4.8 Yield prediction experiments: average (sampled over three acquisition dates and five VIs) execution times (s = second and m = minute) for ML experiments, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, location-only = only location features used, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and N/A = not applicable. 90
4.5.1 Yield prediction experiments: average ZeroR baseline model performance, where the data shown are mean ± standard deviation. 97
4.5.2 Yield prediction experiments: average model R^2 performance for LRe-single-VI dataset experiments, where the data shown are mean ± standard deviation and experiments with zero R^2 were omitted. 97
4.5.3 Yield prediction experiments: location-only non-spatial/non-spatial experiment results, where the data shown are mean ± standard deviation. . . . 99
4.5.4 Average approximate execution times for the experiments (s = second, m = minute, and h = hour), where RF-no-tune = RF without hyperparameter tuning. 104

4.6.1 Yield prediction experiments: deep learning model hyperparameter choices for the main experiments detailed in Section 4.6.2.2, where FC = fully connected, rCNN = spectral reflectance CNN, and sCNN = spatial CNN. 110
4.6.2 Yield prediction experiments: centralized hyperparameter tuning experiment model performance, where results are shown as mean ± standard deviation, CNN results are averaged over all three input tile sizes (3×3, 7×7, and 11×11), and marked in bold are the best performing models (in terms of average performance) for a dataset; rCNN = spectral reflectance CNN, sCNN = spatial CNN, mono-temporal-DoY201 = the dataset with only DoY 201 features, mono-temporal-DoY216 = the dataset with only DoY 216 features, and multi-temporal = the dataset with both DoY 201 and 216 features. 115
4.6.3 Yield prediction experiments: model performance for the IID experiments, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) among the different types of learning methods; and L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. 117
4.6.4 Yield prediction experiments: model performance for the non-IID experiments, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) among the different types of learning methods; and L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. . . . 118
4.6.5 Yield prediction experiments: list of model performance average rankings from the Friedman test over all the IID and non-IID client dataset splits and learning methods, where a lower ranking number indicates better ranking performance, and marked in bold are the best performing models[1]; rCNN = spectral reflectance CNN, sCNN = spatial CNN, and N/A = not applicable. 120
4.6.6 Yield prediction experiments: list of learning method performance average rankings from the Friedman test considering only deep learning model results, where a lower ranking number indicates better ranking performance, and marked in bold are the best performing learning methods[1]; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. 123
4.6.7 Yield prediction experiments: total simulated network communication cost and execution time of the IID and non-IID experiments, where the network communication costs of the non-deep-learning models were not considered, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and N/A = not applicable. 125

5.0.1 The N_2O gas chambers by a) the notation used for identifying the chambers, b) tilling techniques applied, c) fertilizer information, and d) whether crops were grown in the row where the chamber was placed; ID = identification number, Conv. = Conventional, DoY = day of year, Ur = urea, PCUr = polymer-coated urea, and UrIn = urea coated with urease inhibitor. 134
5.4.1 N_2O prediction experiments: features used, including the window size and aggregation operator applied to derive the lagged features, in the multi-year extrapolation experiments, where Op. = operator, SSNC = synthetic soil nitrogen content, GDD = growing degree days, SHU = soil heating units, NHU = nitrification heating units, and N/A = not applicable. . . . 143
5.4.2 N_2O prediction experiments: single-year interpolation global model performance averaged over all chambers and all four dataset temporal resolutions, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) for a given year. 146
5.4.3 N_2O prediction experiments: single-year interpolation seasonal model performance averaged over all chambers and all four dataset temporal resolutions, where the data shown are mean ± standard deviation, N/A = no results available due to zero samples in that part of the season, and bold font results are the best (in terms of average performance) for a given year. 149
5.4.4 N_2O prediction experiments: list of model performance average rankings from the Friedman test for the single-year interpolation experiments, where a lower ranking number indicates better ranking performance and marked in bold are the best performing models[1]. 151
5.4.5 N_2O prediction experiments: multi-year extrapolation global model performance averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, CV = cross-validation, the data shown are mean ± standard deviation, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance. 154
5.4.6 N_2O prediction experiments: multi-year extrapolation seasonal model performance for the MY_{2021} experiments (experiments with models trained using 2021 data and tested using 2022 data) averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, the data shown are mean ± standard deviation, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance. . 158

5.4.7 N_2O prediction experiments: multi-year extrapolation seasonal model performance for the MY_{2022} (MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data) experiments averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, the data shown are mean ± standard deviation, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance. 159
5.4.8 N_2O prediction experiments: list of model performance average rankings from the Friedman test for the multi-year extrapolation experiments, where a lower ranking number indicates better ranking performance, marked in bold are the best performing non-baseline models[1], DMem = DSFAMem, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and Both = $MY_{2021} \cup MY_{2022}$. 162
5.4.9 N_2O prediction experiments: list of models significantly[1] affected by the choice of cross-validation (CV) type for hyperparameter tuning for the multi-year extrapolation experiments, where MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, Both = $MY_{2021} \cup MY_{2022}$, and Multi-model = every model's performance results were included in the significance testing analysis. 163
5.4.10 N_2O prediction experiments: list of models significantly[1] affected by the exclusion or inclusion of the SSNC feature for the multi-year extrapolation experiments, where SSNC = soil synthetic nitrogen content, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and Both = $MY_{2021} \cup MY_{2022}$. 164
5.5.1 N_2O prediction experiments: 2021 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods. . . . 172
5.5.2 N_2O prediction experiments: 2022 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods. . . . 173

5.5.3 N_2O prediction experiments: 2023 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods. . . . 174
5.5.4 N_2O prediction experiments: PTR_{RMSE} performance for IID and non-IID client datasets over all three years, where the data shown are mean ± standard deviation, AE = average ensemble learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and N/A = not applicable. 179
5.5.5 N_2O prediction experiments: learning method performance average ranking from the Friedman test, where a lower ranking number indicates better ranking performance, marked in bold are the best performing models[1], AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. . . . 181
5.5.6 N_2O prediction experiments: list of non-baseline models with the best performing evaluation metrics[1] for each client dataset split, learning method, and dataset, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and None = all models generally fail. 182
5.5.7 N_2O prediction experiments: model performance average ranking from the Friedman test, where a lower ranking number indicates better ranking performance, marked in bold are the best performing models[1], AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning, and N/A = not applicable. 183
5.5.8 N_2O prediction experiments: initial hyperparameter tuning process total execution time (min = minute) for each dataset. 183
5.5.9 N_2O prediction experiments: sequential and parallel total execution time and simulated network communication cost for each model and learning method, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, N/A = not applicable, and NIY = not implemented yet (since the network cost of sending ML models over the network was not considered). 184

A.1 Vegetation indices, derived using near-infrared (NIR) and red-green-blue (RGB) bands, involved in this thesis, where ρ_i = the reflectance of the spectral band i. 193
A.1 *Cont.* 194
A.2 Vegetation indices, derived using only red-green-blue (RGB) bands, involved in this thesis, where ρ_i = the reflectance of the spectral band i between 0 and 1. 195
A.3 Vegetation indices, derived using the red-edge band, involved in this thesis, where ρ_i = is the reflectance of the spectral band i. 196

B.1 Comparison of the related work, summarized in Section 2.9.2, that applied federated learning (FL) to agriculture settings, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, IMG = imagery, LL = local learning, CL = centralized learning, EL = distributed ensemble learning, and N/A = not applicable. 197
B.1 *Cont.* . . . 198
B.2 Comparison of the yield prediction related work summarized in Section 2.9.3, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, IMG = imagery, GEN = genotype information, RGB = red-green-blue, MS = multispectral, HS = hyperspectral, VI = vegetation index, TI = texture index, and $\sim$ indicates an approximate value. . . . 199
B.3 Comparison of the N_2O emission prediction related work summarized in Section 2.9.4, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, and $\sim$ indicates an approximate value. . . . 200

C.1 Hyperparameter search space ranges for the random forest (RF) model. . 201
C.2 Hyperparameter search space ranges and neural network structure for the multilayer perceptron (MLP) model used by all studies in this thesis (except the study described in Section 4.5), where i = the number of hidden layers, FC_i = fully connected dense layer i, and N/A = not applicable. . 202
C.3 Hyperparameter search space ranges and neural network structure for the multilayer perceptron (MLP) model used by only the study described in Section 4.5, where i = the number of hidden layers, FC_i = fully connected dense layer i, and N/A = not applicable. . . . 202
C.4 Hyperparameter search space ranges and neural network structure for the 1D convolutional neural network (CNN) model, where FC_i = fully connected dense layer i and N/A = not applicable. . . . 203
C.5 Hyperparameter search space ranges and neural network structure for the 2D spectral reflectance convolutional neural network (rCNN) model, where FC_i = fully connected dense layer i and N/A = not applicable. . . . 203
C.6 Hyperparameter search space ranges and neural network structure for the 2D spatial convolutional neural network (sCNN) model, where FC_i = fully connected dense layer i and N/A = not applicable. . . . 204
C.7 Hyperparameter search space ranges and neural network structure for the long short-term memory (LSTM) model, where FC_i = fully connected dense layer i, FC_1 only exists half the time (depending on the truth value of the hyperparameter f), and N/A = not applicable. . . . 204
C.8 Hyperparameter search space ranges for the support vector machine (SVM) model, where loguniform = log uniform random sampling. . . . 204

C.9 Hyperparameter search space ranges common to the deep learning models (excluding the multilayer perceptron model detailed in Table C.3) , where loguniform = log uniform random sampling. 205

D.1 Mean (averaged over all the missions conducted in 2021) size of the orthomosaic UAV imagery files used in the yield prediction experiments, where RGB = red-green-blue and MS = multispectral. 208

D.2 Yield dataset yield (in bushels per acre) statistical properties, where Std Dev = standard deviation, Max = maximum, Min = minimum, Q1 = quartile 1, Q3 = quartile 3, IQR = interquartile range, and n = the number of samples. 208

D.3 Summary statistics of the sensors' raw datasets (those used in the N_2O emission prediction experiments), where # nulls = number of missing readings, μ = mean, Med = median, Max = maximum, Min = minimum, σ = standard deviation, N/A = not applicable, and VPD = vapor pressure deficit; see Table D.4 for the units of the sensors readings. 213

D.3 *Cont.* 214

D.3 *Cont.* 215

D.4 A description of the data sources used in our N_2O emission prediction studies (described in Chapter 5) along with host device identifiers, sensor nodes, the available sensors, sensor units, sensor sampling rates, the placement of the sensors relative to the LI-COR gas chambers, and approximate elevation, where min = minute and ID = identification number. 216

List of Figures

3.1 Overview diagram of the proposed privacy-aware smart farming IoT system architecture, where WAN = wide-area-network, RN = root node, UART = universal asynchronous receiver / transmitter, GPIO = general purpose input/output, and WSN = wireless sensor network. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed. . . . 46

3.2 Privacy-aware predictive analytics smart farming system software module diagram when using federated learning (FL) as the distributed machine learning (ML) approach of choice, where we illustrate where encryption could be applied for solutions that use privacy preserving technologies; API = Application Programming Interface. 47

4.0.1 Overview diagram of the proposed privacy-aware smart farming IoT system architecture that was adapted to the imagery-based yield prediction use case, where WAN = wide-area-network, RN = root node, RGB = red-green-blue, and MS = multispectral. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed. 62

4.0.2 A map of the test site. The map illustrates the field numbers and field boundaries and includes a distance scale and coordinate grid that uses the coordinate reference system WGS 84 / UTM zone 18N. 63

4.0.3 Yield precision maps, for each field (see Figure 4.0.2 for field numbers), of the raw and pre-processed (cleaned and interpolated) yield, where units are in bu/ac (1 ac $\approx$ 0.405 ha). 67

4.4.1 RF average yield prediction CC performance comparing MS imagery ($NDVI_{RedEdge}$) to RGB imagery (ExR), for both LRe and HRe datasets, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, and rev-SpKF-CV = reverse spatial k-fold CV. 84

4.4.2 LR average yield prediction CC performance comparing MS imagery ($NDVI_{RedEdge}$) to RGB imagery (ExR), for both LRe and HRe datasets, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, and rev-SpKF-CV = reverse spatial k-fold CV. 85

4.4.3 Average yield prediction model CC performance for DoY 193, MS VI $NDVI_{RedEdge}$, RGB VI ExR, and each evaluation method, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b), KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, High = HRe, and Low = LRe. 86
4.4.4 Yield prediction experiments: location-only features 10-fold CV model performance (CC and RMSE), where the error bars represent 1 standard deviation and fi = Field i. 87
4.4.5 Yield prediction experiments: KF-CV average R^2 performance over the growing season for all fields and VIs, where MS = multispectral, RGB = red-green-blue, LRe = low spatial resolution dataset, and HRe = high spatial resolution dataset. 88
4.5.1 Yield prediction experiments: average RMSE performance for the LRe-single-VI experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b). 100
4.5.2 Yield prediction experiments: average R^2 performance for the HRe-single-VI experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b). 101
4.5.3 Yield prediction experiments: average R^2 performance for the LRe-all-bands experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b). 102
4.5.4 Yield prediction experiments: average R^2 performance for the HRe-all-bands experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b). 103
4.6.1 Yield prediction experiments: visual example of the data pre-processing steps involved with model evaluation for a 3-client system (where Fields 1, 2, and 3 are used as clients) using nested layered 4-3-fold block CV, where the outer training-testing fold pairs are further split into inner training-validation pairs within the inner CV loop to be used for hyperparameter tuning, and the inner CV loops for outer folds 2–4 are omitted for clarity. 108
4.6.2 Field 11 precision maps of actual yield (a) vs. model predictions (b–x), where the color legend is the same as the one used in Figure 4.0.3d, and the non-LocMem prediction maps use a '<client dataset split>-<learning method>-<model>' naming convention; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, sCNN = spatial CNN, and LocMem = location memorizer. 114
4.6.3 Smaller field precision maps of actual yield (a) vs. RF model predictions (b–i), where the color legend is the same as the one used in Figure 4.0.3f, and the prediction maps use a '<client dataset split>-<learning method>' naming convention; AE = average ensemble learning, CL = centralized learning, LL = local learning, and SE = stacked ensemble learning. . . . 115

4.6.4 Smaller field precision maps of actual yield (a) vs. sCNN (spatial CNN) model predictions (b–k), where the color legend is the same as the one used in Figure 4.0.3f, and the prediction maps use a '<client dataset split>-<learning method>' naming convention; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. 116

5.0.1 Overview diagram of the proposed privacy-aware smart farming IoT system architecture that was adapted to the N_2O emission prediction use case, where WAN = wide-area-network and RN = root node. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed. . . . 132

5.0.2 Line plots of N_2O emissions (scaled to a 6 h resolution) time series plot for each year, where Ci indicates chamber i for the given year. 133

5.0.3 N_2O prediction experiments: PN in-field setup in 2023, where the left image shows both a LI-COR gas chamber and in-field PN setup, the top-right image shows the PN's gateway components, and the bottom-right image shows the sensor probe placement in relation to a LI-COR gas chamber; where (a) = PN's gateway, (b) = the PN's soil sensor probes, (c) = LI-COR automated gas chamber, and (d) = the gateway's uninterruptible power supply. 135

5.0.4 N_2O prediction experiments: SysML [85] diagram illustrating the high-level hardware components involved with the prototype PN , where APC UPS = American Power Conversion uninterruptible power supply and NPK = Nitrogen-Phosphorus-Potassium. 136

5.0.5 N_2O prediction experiments: example of partitioning a 36-sample time series dataset into training, testing, and validation datasets using nested 3-2-fold block CV with a 1-day cluster size, where the temporal resolution of the dataset is 3 samples per day, the inner CV loops for outer folds 2 and 3 were omitted for clarity. 137

5.4.1 Line plots of actual N_2O emissions vs. single-year interpolation model predictions (of various models using 30 min temporal resolution datasets) for 2021, 2022, and 2023. 145

5.4.2 Line plots (at 180 min temporal resolution) of actual N_2O emissions vs. model predictions for the multi-year extrapolation non-SSNC experiments that used block CV for hyperparameter tuning, where SSNC = synthetic soil nitrogen content. 155

5.5.1 N_2O prediction experiments: bar plots of LSTM and RF R^2 seasonal performance averaged over 2021, 2022, and 2023 for each learning method, where the error bar represents 1 standard deviation, blue bars are early season, green bars are mid-season, red bars are late season, yellow bars are end of season performance, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. 178

5.5.2 Line plots (at 3 h resolution) of actual N_2O emissions vs. LSTM predictions using various learning methods for 2021, 2022, and 2023, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning. . . . 180

D.1 Field 1 June 22, 2021, NDVI and RGB orthomosaics such that the size of a 9 m diameter (4.5 m radius) filter is visualized in the field, where NDVI = normalized difference vegetation index and RGB = red-green-blue. . . 207

D.2 2021 UAV imagery-based yield prediction feature importance bar plots of the selected features in our study described in Section 4.6, where the importance is the normalized (via min-max scaling) absolute value of the PLSR coefficients of the features; the red, green, blue, orange, and purple bars are the red, green, blue, near-infrared (NIR), and red-edge spectral band features, respectively; the grey bars are for VI features; features have a '<imagery channel>_<aggregation filter>_<temporal feature type>' naming convention; and DoY = day of year. 209

D.3 Daily average air temperature time series plots for the 2021, 2022, and 2023 growing seasons. 210

D.4 Daily total precipitation time series plots for the 2021, 2022, and 2023 growing seasons. 211

D.5 N_2O prediction experiments: a sensor deployment map of the study area. The map illustrates the sensor node (SN) locations in Field 2 and includes a distance scale and coordinate grid that uses the coordinate reference system WGS 84 / UTM zone 18N. Sensors are identified on the map from 1 to 6, where 1 = the area where LI-COR gas chambers were deployed, 2 = the predictor node (PN), 3 = Pessl SN 01209E4F, 4 = Pessl SN 00209FC8, 5 = Pessl SN 01209E58, and 6 = Pessl SN 01209E52. 212

D.6 N_2O prediction experiments: component and wiring diagram of the Summer 2023 PN prototype described in Section 5.3.3, where APC UPS = American Power Conversion uninterruptible power supply, NPK = Nitrogen-Phosphorus-Potassium, TTL = Transistor-Transistor Logic, and SMA = SubMiniature version A; the Raspberry Pi and pin diagram was taken from [247]. 217

Nomenclature

List of Abbreviations

a.k.a. also known as
Adam Adaptive Moment Estimation
AE Average Ensemble Learning
agri-data agriculture data
agri-variable agriculture variable
API Application Programmable Interfaces
APSIM Agricultural Production Systems Simulator
ARIMA Autoregressive Integrated Moving Average
CC Pearson's Correlation Coefficient
CIR Color Infrared
CL Centralized Learning
CNN Convolutional Neural Network
CNN_L 1D CNN with long input tensor length
CNN_S 1D CNN with short input tensor length
CoAP Constrained Application Protocol
CV Cross-validation
DLEM Dynamic Land Ecosystem Model
DoY Day of Year
DSFA Days Since Fertilizer Application
DSFAMem DSFA Memorizer
DSSAT Decision Support System for Agrotechnology Transfer
EL Ensemble Learning
FedAvg FederatedAveraging
FL Federated Learning
GDD Growing Degree Day
GHG Greenhouse Gas
GPIO General Purpose Input/Output
GSM Global System for Mobile Communications
GWP Global Warming Potential
HRe High Spatial Resolution Dataset
HS Hyperspectral
IID Independent and Identically Distributed
IoT Internet of Things
IQR Interquartile Range
KF-CV k-fold Cross-validation
KNN k-nearest Neighbours
LiDAR Light Detection and Ranging
LL Local learning
LocMem Location Memorizer
LOFO-CV Leave-one-field-out Cross-validation
LoRaWAN Long-range Wide-area-network
LPWAN Low-power Wide-area-network
LR Linear Regression
LRe Low Spatial Resolution Dataset
LSTM Long Short-term Memory
LSTM_L LSTM with long input tensor length
LSTM_S LSTM with short input tensor length
MAE Mean Absolute Error
MAPE Mean Absolute Percentage Error
ML Machine Learning
MLP Multilayer Perceptron
MQTT Message Queuing Telemetry Transport
MS Multispectral
MSE Mean Squared Error

List of Abbreviations *Cont.*

MY_{2021} Experiments with models trained using 2021 data and tested using 2022 data
MY_{2022} Experiments with models trained using 2022 data and tested using 2021 data
N/A Not Applicable
NDVI Normalized Difference Vegetation Index (most of the other vegetation index abbreviations are defined in Appendix A)
NHU Nitrification Heating Units
NIR Near-infrared
NPK Nitrogen-Phosphorus-Potassium
NZ-DNDC New Zealand DeNitrification-DeComposition
P2P Peer-to-peer
PA Precision Agriculture
PLSR Partial Least Squares Regression
PN Predictor Node
PTR Positive Transferability Ratio
R^2 Coefficient of Determination
rCNN Spectral Reflectance CNN
RGB red-green-blue
rev-LOFO-CV Reverse Leave-one-field-out Cross-validation
rev-SpKF-CV Reverse Spatial-k-fold Cross-validation
RF Random Forest
RF-no-tune Random Forest without hyperparameter tuning
RMSE Root Mean Squared Error
RN Root Node
RNN Recurrent Neural Network
SA Spatial Autocorrelation
SCCCI Simplified Canopy Chlorophyll Content Index
sCNN Spatial CNN
SE Stacked Ensemble Learning
SFS Smart Farming System
SGD Stochastic Gradient Descent
SHU Soil Heating Units
SN Sensor Node
SpKF-CV Spatial-k-fold Cross-validation
SSD Solid State Drive
SSNC Synthetic Soil Nitrogen Content
SVM Support Vector Machine
TI Texture Index
UAN Urea Ammonium Nitrate
UART Universal Asynchronous Receiver / Transmitter
UAV Unmanned Aerial Vehicle
UTM Universal Transverse Mercator
VI Vegetation Index
WAN Wide-area-network
WGS World Geodetic System

List of Chemical Symbols

C Carbon
CH_4 Methane
CO_2 Carbon Dioxide
K Potassium
N Nitrogen
N_2 Dinitrogen
N_2O Nitrous Oxide
NO Nitric Oxide
NO_2 Nitrogen Dioxide
NO_2^- Nitrite
NO_3^- Nitrate
NH_4^+ Ammonium
P Phosphorus

Chapter 1

Introduction

This thesis analyzes spatiotemporal Internet of Things (IoT) sensor data and uses privacy-aware distributed ensemble learning (EL) and federated learning (FL), to avoid sharing raw data, in a fog computing environment for smart farming applications.

1.1 Motivation and Challenges

By 2050 the world population is estimated to reach 9.73 billion, and by 2100 it is expected to reach 11.2 billion [46]. Food production will be threatened by climate change, declining fresh water availability, and a reduction in the amount of available arable land [268], and by 2050, there will likely be a noticeable increase in food consumption as a result of population growth [229]. To address this increased stress on farmers' food production rates, input cost reductions and agriculture resource optimizations can be offered to farmers by using precision agriculture (PA). Smart farming can enable PA by gathering site-specific agriculture data (agri-data) from a) sensors using in-field gateways and/or b) other data sources (e.g., satellites). Smart farming is a management practice that makes use of a) the IoT, b) cloud computing, c) fog/edge computing, and d) artificial intelligence to optimize agriculture practices [66, 277]. Smart farming carries several benefits, including food security, information exchange, safety control and fraud prevention, and the creation of profitable business models by lowering costs [66].

IoT consists of many interconnected devices that have networking, computing, sensing, and storing capabilities. These devices tend to be resource-constrained, heterogeneous in nature, and deployed in large numbers [241]. Cloud computing performs central processing and storage in distant data centers and offers services such as big data analytics, decision making, and computational power [193]. PA models can be trained by sending all the data from the network's edge to the cloud [193] and then performing centralized learning (CL). CL involves combining all the data sources into a single centralized dataset before training a model. This approach unfortunately suffers from drawbacks. The cloud in its current state will have difficulty handling the growing amount of IoT data [60], mainly due to the sheer volume of IoT data and the low-latency requirements of IoT applications

[193]. Additional IoT challenges include a) heterogeneous device interoperability [66], b) connectivity, and c) bandwidth difficulties [193]. In the smart farming setting, large farms tend to not be fully covered by wireless network access, and energy efficient long-range wireless protocols tend to have low bandwidth [268], meaning energy efficiency and energy harvesting are important smart farming requirements [16]. The fog can be used to address these challenges by bringing the cloud processing power closer to the data sources, which enables local learning (LL) [195] and addresses some of the challenges of CL [285]. The fog is a resource-constrained cloud near the edge of the network and supports low-latency and lightweight computing [36] via IoT gateways deployed in local-area-networks [193]. IoT gateways bridge the gap between the cloud and the sensing devices in the edge, and should support short-range and long-range networking protocols in smart farming scenarios [46], and should store local sensor data. Gateways collectively create data islands that may be distributed around the globe, meaning laws and legislations may disrupt the sharing of this data [285]. Fog computing and LL can be combined such that many local models are trained using only local data [195], but unfortunately applying LL to agri-data may lead to loss of useful geographical trends that could have been useful to a centralized model [195, 229]. Furthermore, even though the data from each gateway may be collectively required to train a PA model, the data on the gateways may need to remain private, and this makes it particularly difficult when data from a single gateway is not sufficient for training a model [295]. This is relevant to smart farming, because farmers are skeptical of smart farming systems (SFSs) due to privacy and trust reasons [242], and for fear of misuse or helping competing farmers [119]. This skepticism leads to poor SFS adoption rates, which is especially the case for farmers that own small scale farms [242]. High system deployment costs is another reason that leads to low SFS adoption rates. Limiting a SFS's farmer user-base would result in less available training data, and this could in turn negatively impact model performance.

Some of these challenges can be addressed using distributed machine learning (ML), specifically by using model sharing combined with distributed EL [63, 199] and/or FL [293, 117]. In both cases models are trained at the edge of the network, on client devices, using only local agri-data. Private data is never shared. Only model updates are sent over the network, which reduces network communication by avoiding sending large datasets over the network. The result of the distributed ML process is a global PA model that can be used to benefit farmers with agriculture-related tasks. For the distributed EL (combined with model sharing) case, once the local models are trained, they are shared with each client over a single round of communication. The clients then each build an ensemble model (the global model) by combining local models together. For the FL case, multiple rounds of training and communication are performed. A coordination server initializes a global model and then broadcasts the model to each client. The clients then train their local model over many rounds of training. After each round of training, model updates are sent to the coordination server, and the server aggregates the model updates together to update the global model. The server then broadcasts the updated global model back to the clients to begin a new round of training. This process proceeds until some termination criteria occurs.

With access to predictions from a model trained via distributed ML, a farmer could

gain insight about the state of a field and could use this information to take corrective action [213] or to plan [71]. The cost-benefit ratio of a SFS should be accurate and understandable to make it worthwhile to the farmer [16]. This is particularly relevant to SFSs that make use of unmanned aerial vehicle (UAV)-mounted multispectral (MS) or hyperspectral (HS) cameras, since these setups can be quite expensive and have high technical complexity [296]. Farmers may not have the necessary technical expertise to use a complex SFS [213]. Furthermore, agricultural datasets tend to have a highly dynamic spatiotemporal nature. The analyst must consider the complex external environmental conditions that can affect the target variable being predicted when designing a PA model [229]. Furthermore, when analyzing spatial (or temporal) data, one must consider spatial (or temporal) structure within the data (e.g., spatial autocorrelation (SA)), since, according to Nikparvar et al. [189], the first law of geography states that "near things are more related than distant things". For example, phenotyping and imagery datasets gathered from a single field will likely be spatially autocorrelated [100]. This breaks the fundamental assumption that model evaluation techniques (such as k-fold cross-validation (KF-CV)) use to evaluate models; that is, the training and testing datasets are assumed to be independent and identically distributed (IID) [230, 100]. Models trained on spatially autocorrelated data will likely overfit and underestimate prediction errors [225]. Methods to address this include partitioning the training and testing datasets into spatially (or temporally) disjointed subsets when building the folds. This is known as block cross-validation (CV) [221]. However, such techniques tend to be pessimistic when evaluating extrapolation performance [230] involving data from new unseen locations, because trends may be lost when splitting the data into spatially disjointed folds [273, 230].

In summary, the formal problem being addressed in this thesis is as follows: some types of datasets (e.g., agri-data, which we focus on in this thesis) tend to be highly dynamic in space and time, making data analysis not trivial, and some sensing solutions may be too expensive to enable practical and affordable high spatiotemporal resolution sensing for gathering such datasets. Furthermore, without carefully considering the spatial and temporal structure of a spatiotemporal dataset before analyzing it, the results of the analysis may be over-optimistic, and this could lead an analyst or farmer into making incorrect business/management decisions. Furthermore, data sharing restrictions and the distributed nature of IoT sensor data can also add privacy, scalability, and network communication difficulties to spatiotemporal data analysis.

1.1.1 Yield Prediction

An application of smart farming is yield prediction. A yield prediction system could perform early-season yield forecasting, and this would help stakeholders (e.g., farmers, commercial suppliers, governments, and international organizations [17]) by enabling efficient crop management, food security evaluation, food trade planning, and policy design improvements [284]. Fields are heterogeneous by nature [229] despite most farming practices treating the fields as homogeneous. Yield precision maps can be created by using a ML model that was trained using agri-data, for example, imagery. Agriculture studies that apply remote sensing commonly use vegetation indices (VIs). VIs involve applying math-

ematical operations (e.g., linear combinations) to the imagery spectral channels/bands to derive an index that can be used to estimate vegetation/crop health conditions [296]. Imagery of a farm can be obtained using remote sensing via satellites, UAVs, or aircrafts, or hand-held/tractor-mounted imaging equipment [41]. Sending a tractor in the field too frequently can hinder crop yield [127], so remote sensing should be favored for imagery acquisition. A SFS designer may favor UAV imagery, since it offers higher spatiotemporal resolutions (and this benefits PA model performance [128]) than satellite imagery [296], although satellite imagery does have some advantages including it being more affordable [128]. However, UAV-based approaches that use MS/HS cameras have high monetary costs and high complexity [296]. Furthermore, autonomous UAV-based SFSs are not easy to deploy due to the laws and regulations around UAV flights [57], and these systems may also face the IoT system challenges [16]. To meet the requirements, the farm's network should be setup to enable the UAV to upload massive amounts of imagery to a local server and/or upload image aggregates to the cloud, but this is not trivial due to bandwidth and connectivity limitations [268]. Nevertheless, by only using imagery acquired from a UAV or satellite for crop analysis, a farmer could acquire on-demand yield precision maps by making use of an imagery-based yield prediction models without having to deploy many costly sensors in a field or investing into costly yield monitoring equipment [126, 112].

1.1.2 Nitrous Oxide (N_2O) Prediction

An application of smart farming is N_2O emission prediction. N_2O is a powerful greenhouse gas (GHG) that has approximately 273 times more global warming potential over a 100-year time horizon than carbon dioxide (CO_2) [73]. Since fertilizer application is necessary for healthy and productive crop growth, N_2O emission creation in agriculture is relatively unavoidable [75]. In fact, approximately 30 % of all GHG emissions are created by the agriculture sector [10]. Furthermore, nation states have put efforts into reducing GHG emissions, and this puts pressure on farmers to act. For example, by 2030, the current Canadian government expects farmers to reduce their synthetic fertilizer-based GHG emissions by 30 % [58]. However, mitigation actions taken to reduce N_2O emissions require the ability to measure emissions to be able to confirm that the emission reduction strategies were successful. Unfortunately, N_2O emissions are highly dynamic in space and time [216], and sensing continuous emissions is quite expensive due to sensing equipment costs, meaning that obtaining high spatiotemporal resolution N_2O emission datasets is expensive and not straightforward [115, 197, 210].

Continuous emission measuring techniques such as the automated gas chamber and flux tower (or micrometeorological) [48] approaches offer high temporal resolution, but offer low spatial resolutions due to their high monetary cost that makes it difficult to deploy many of such devices to achieve high spatial resolution sensing. The static gas chamber measuring technique offers higher spatial resolution sensing, since it is more affordable (allowing for the deployment of many such devices), but it has low temporal resolution due to the requirement of manual labor in the sensing process [115, 197, 210]. Therefore, one cannot expect farmers to purchase costly N_2O emission sensors to help meet emission mitigation reduction targets. As such, rather than measuring precise emission readings

using costly equipment, emissions can be predicted using data-driven models to generate inexpensive emission estimates, and in fact, data-driven models enable high resolution (in both the spatial and temporal dimensions) emission predictions [116]. N_2O emissions can be predicted using process-based (or mechanistic models) and/or data-driven models [216, 96]. Data-driven models a) can make higher spatiotemporal resolution predictions and b) suffer from fewer drawbacks than process-based models [48, 228, 116], making data-driven models appropriate candidates for replacing N_2O emission measuring technologies. However, the highly dynamic spatiotemporal nature of the agri-data and the requirement to make model predictions for data from new locations, pose challenges to data-driven models [122].

1.2 Solution Outline

In this thesis we proposed a fog computing-based methodology for analyzing spatiotemporal IoT sensor data using privacy-aware distributed ML to replace expensive sensing equipment with affordable sensors combined with models. We gathered agri-data from IoT sensors deployed in a smart farm. Since every farm is different, we emulated this in our analysis by using our single-farm dataset and evaluating PA models using a variety of a) homogeneous and heterogeneous dataset partitioning schemes and b) evaluation techniques for creating training and testing datasets.

Specifically, we proposed an IoT-based SFS architecture that uses privacy-aware distributed ML to train PA models without having to share farmers' private data. Only model updates are shared to collectively train a global model that can benefit multiple farmers. This satisfies a) network bandwidth requirements, since large datasets do not need to be sent over the network, only model updates are shared, and b) scalability and computational complexity requirements, since the training process is distributed across many client devices, which means the workload is shared via parallel processing.

Our system addresses farmers' privacy concerns by never sharing raw data, although model updates may still reveal sensitive information [195, 63, 117]); our proposed architecture does not provide any formal cybersecurity benefits.

Our proposed SFS architecture also satisfies the SFS affordability requirement to incentivize farmers to join the system as follows: expensive sensing equipment is first required to train the models using expensive-to-sense ground truth data and affordable sensors' data, but once the training is complete, new farmers can join the system to benefit from the models without needing any expensive equipment. The proposed architecture was not formally or empirically evaluated, it was simply proposed as a means to an end for showcasing our distributed ML experiments involving spatiotemporal IoT sensor data analysis.

With the help of Invest Ottawa, we had access to a Canadian smart farm named Area X.O located in Ottawa, Ontario, Canada (45°19'8.17"N, 75°45'22.40"W). By gathering data from Area X.O, we were able to conduct yield prediction and N_2O emission prediction studies to showcase our proposed architecture. Specifically,

- We conducted three imagery-based yield prediction studies in this thesis. UAV imagery was gathered from six fields at Area X.O, in 2021, by a drone service company. Yield was gathered using a harvester-mounted yield monitor in 2020 and 2021. Each of the three studies involved using a) features derived from UAV imagery to train and evaluate yield prediction models and b) various spatial block CV methods for addressing the spatial structure of the data and evaluating model spatial generalizability. The first two studies applied CL without any distributed ML, whereas the third study did involve applying distributed ML. The first study used a static choice of model hyperparameters. The second study applied hyperparameter tuning for selecting model hyperparameters. The third study applied hyperparameter tuning and feature selection.
- We conducted two N_2O emission prediction studies in this thesis. We gathered N_2O, soil, and weather sensor data from a single field at Area X.O, from 2021 to 2023. N_2O emissions were gathered using expensive automated gas chambers. Soil and weather data was gathered from in-field sensors. As part of the data collection process of both studies, in 2023 we built a prototype of the proposed privacy-aware smart farming architecture. Specifically, we built a predictor node (PN) prototype using a Raspberry Pi and RS485 soil sensors. This PN prototype was deployed in a field and used to gather soil readings near one of the automated gas chambers. Both the studies used a) features derived from soil and weather data to train and evaluate models, b) feature selection, and c) a temporal block CV method for addressing the temporal structure of the data and evaluating model temporal generalizability. The first study used CL without any distributed ML, and conducted gap-filling (interpolation to within the same year) and extrapolation (training a model using one year's data and predicting emissions for another year) experiments. The second study applied distributed ML, and conducted only gap-filling experiments (no extrapolation experiments were conducted).

1.3 Contributions

In this section we list the main contributions of this thesis in the form of computer science, crop yield prediction, and N_2O emission prediction contributions.

1.3.1 Computer Science

- We proposed a distributed ML methodology for analyzing spatiotemporal IoT sensor data. The proposed methodology replaces expensive monitoring equipment with affordable sensors combined with models deployed in the fog. It achieves high spatiotemporal resolution sensing in a a) privacy-aware, b) affordable, and c) scalable manner, at the cost of reduced sensing accuracy. Our results showed that, in some cases, models trained using distributed ML could achieve similar performance to those trained using CL without having to share raw data.

- We identified a knowledge gap in the literature; that is, most ML studies involving UAV imagery-based yield prediction that involved the use of spatial data ignored the data's spatial structure, meaning these works likely obtained over-optimistic model performance results. We bridge this knowledge gap by a) providing some background on spatiotemporal data and the challenges involved with analyzing such data using ML and b) evaluating the spatial (and/or temporal) generalizability of models in our yield prediction (and N_2O emission prediction) studies. This contribution is important, since it could help guide the yield prediction community towards more careful model evaluation when working with spatially autocorrelated agriculture data. This contribution is not limited to agriculture applications; it also applies to other domains involving spatiotemporal data analysis.
- We took into account the spatiotemporal heterogeneity of agri-data by making use of various block CV model evaluations techniques for partitioning datasets before evaluating models. These techniques included spatial and temporal CV, where the datasets were partitioned into spatially (or temporally) disjointed training and testing (and validation) datasets. We found that by using both homogeneous/IID and heterogeneous/non-IID dataset splitting techniques for deriving training, validation, and testing datasets, we were able to a) more carefully analyze and interpret model performance, b) avoid over-optimistic conclusions, and c) identify potential model performance biases caused by spatiotemporal structure found in the dataset.

1.3.2 Crop Yield Prediction

- We performed a yield prediction case study involving a) gathering high spatiotemporal resolution imagery using UAVs mounted with cameras, and gathering high spatial resolution corn yield data using a crop harvester equipped with yield monitoring equipment; b) pre-processing the datasets using commonly used techniques discussed in the imagery-based yield prediction literature, including yield dataset cleaning, yield dataset interpolation, stitching raw UAV imagery into orthomosaics, deriving VIs from the orthomosaics, and fusing the imagery and yield datasets into a tabular spatial dataset used for ML model training and evaluation; and c) performing imagery-based crop yield prediction experiments.
- We performed yield prediction studies using CL, high spatiotemporal resolution imagery, and high spatial resolution yield data, and found that reasonable model performance could be achieved using only imagery features. This is important, since it means future yield prediction studies can probably avoid deploying a large number of proximal sensors in a field by instead relying on non-intrusive imagery-based sensing solutions combined with PA models. This contribution is also significant, since high spatial resolution yield data is relatively expensive to obtain.
- We identified VIs that led to the best yield prediction model performance, which can help future yield prediction studies by speeding up their feature extraction process by limiting the number of VIs considered to only a select few of optimal (in the sense of benefiting yield prediction model performance) VIs.

- We identified what part of the growing season was best for gathering UAV imagery to maximize yield prediction model performance. This contribution is useful, since it can help reduce UAV mission flight costs of future UAV imagery-based yield prediction studies by limiting the number of imagery acquisition mission to only a select few of optimal (in the sense of benefiting yield prediction model performance) acquisition dates.

1.3.3 N_2O Emission Prediction

- We performed a N_2O emission prediction case study involving a) gathering high temporal resolution N_2O emission data using automatic gas chambers, and gathering high temporal resolution soil and weather sensor data from in-field sensors; b) preprocessing the datasets by cleaning the datasets, deriving and selecting features, and fusing the datasets to a tabular time series dataset used for ML model training and evaluation; and c) performing N_2O emission prediction experiments using soil and weather sensor data.
- We implemented a Raspberry Pi-based PN prototype and deployed it to Area X.O. This prototype was built for the N_2O emission prediction use case. The PN prototype gathered soil data near an N_2O automated gas chamber by using RS485 sensors. Sensor data was stored locally in a file on the Raspberry Pi. This file was accessible via a Linux serial console interface and a USB thumb drive.
- We performed N_2O emission prediction studies using CL, soil data, weather data, and high temporal resolution N_2O emission data, and we found that reasonable model performance could be achieved for both gap-filling and extrapolation tasks. This contribution is significant, since high temporal resolution N_2O emission data is expensive to obtain and N_2O datasets tend to have high spatiotemporal variability.

1.4 Outline of Thesis

This thesis is structured as follows: Chapter 2 provides some necessary background (including ML, FL, smart farming, and spatiotemporal data analysis) for explaining the studies reported in the next chapters. At the end of this chapter, we review the literature related to these studies; Chapter 3 proposes an IoT-based smart farming architecture that makes use of privacy-aware distributed ML technologies; Chapter 4 and 5 cover the crop yield prediction and N_2O emission prediction work portion of this thesis, respectively. In these chapters, we describe data sources, the methodologies used, and various studies that we conducted; and Chapter 6 concludes this thesis by summarizing the work done, listing limitations of this thesis, and discussing potential future work avenues. This thesis also includes appendices found at the end: Appendix A lists the equations and definitions of VIs used in this thesis; Appendix B presents the tables used to compare related work to the studies presented in this thesis; Appendix C presents the hyperparameter search spaces

used and other model hyperparameter details related to the models used in this thesis; and Appendix D provides additional supplementary material.

Chapter 2

Background and Related Work

2.1 Internet of Things

The IoT consists of many interconnected devices that have networking, computing, sensing, and storing capabilities. These devices tend to be resource-constrained, heterogeneous in nature, and deployed in large numbers [133]. It is expected that in the near future there will be 100 billion IoT devices that will be deployed and connected to the Internet [13]. There are five layers that are commonly used to describe an IoT architecture, namely the sensing, network, middleware, learning, and application layer [133]. A security and management layer (which intersect with the other layers) may also be included in an IoT architecture [269].

2.2 Cloud, Fog, and Edge Computing

Cloud computing performs central processing and storage in distant data centers and offers services such as big data analytics, decision making, and computational power [193]. The cloud may have difficulty handling the growing amount of IoT data [60] mainly due to the sheer volume of IoT data and the low-latency requirements of IoT applications [193]. Additional IoT challenges include a) heterogeneous device interoperability [66], b) connectivity, and c) bandwidth difficulties [193]. In the smart farming setting, large farms tend to not be fully covered by wireless network access, and energy efficient long-range wireless protocols tend to have low bandwidth [268, 135], meaning energy efficiency and energy harvesting are important smart farming requirements [16]. The fog can be used to address these challenges by bringing the cloud processing power closer to the data sources. The fog is a resource-constrained cloud near the edge of the network and supports low-latency and lightweight computing [36] via IoT gateways deployed in local-area-networks [193]. IoT gateways bridge the gap between the cloud and the sensing devices in the edge, should support short-range and long-range networking protocols in smart farming scenarios [46], and can store local sensor data. Gateways collectively create data islands that may be distributed around the globe, meaning laws and legislations may disrupt the sharing of this

data [285]. Although the terms fog and edge computing are often used interchangeably in the literature, the terms may be differentiated as follows: a) typically edge computing has less resources than fog computing, b) lightweight cloud servers can be thought to reside in the fog layer, and c) edge nodes reside closest to the data sources [193].

2.3 Machine Learning

ML involves automatically identifying patterns in a dataset between predictor variables (the features) and the target variable (the variable of interest to be predicted), and ML enables the use of trained models for predictive analytics applications [124, pp. 3–5]. ML can be categorize into 4 categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [65, 25][124, p. 3]. Supervised learning involves training a model using labeled data [72, p. 14][111, pp. 34–35]. Unsupervised learning involves trying to learn patterns without labeled data by using only predictor variables [72, p. 14][111, p. 34]. Semi-supervised learning can be used to train a model using a small amount of labeled data and then optimize the model using unlabeled data via automatic label generation [72, p. 18]. Active learning is a sub-category of semi-supervised learning, where less training data is required by allowing the machine to a) ask questions [189] and b) affect data generation [111, p. 34]. Reinforced learning involves an agent interacting with its environment and performing an action that affects the environment. Agents are rewarded or punished depending on the outcome of the action [189]. There are also other types of ML including EL, transfer learning, and FL. EL involves combining multiple models to collectively make predictions [239]. We discuss EL in more detail in Section 2.3.3. Transfer learning involves using a pre-trained model in another related domain with less training data [6]. FL is a form of distributed ML that involves multiple clients collaborating together to train a global model by only sending model updates over the network without sharing the clients' private data [117]. We discuss FL in more detail in Section 2.3.4. For readers that want to learn more about ML, we refer them to these three books [111, 124, 72], which cover the topic in detail.

2.3.1 Model Evaluation

When evaluating a model, training data is used to train the model and testing data is used to evaluate the model's generalization performance. Samples used to train the model should usually never be used to test the model's generalization performance[1], else the performance risks being optimistically biased [111, pp. 128–138]. There are many ways of sampling datasets to evaluate models and there are many types of evaluation metrics to choose from. We discuss these topics in the next sections.

[1] this is known as resubstitution [111, p. 138]

2.3.1.1 Dataset Resampling Methods

Typically, the input dataset is split randomly into training and testing subsets (e.g., 80 % training and 20 % testing, an 80-20 training-testing split). This is known as the holdout method [111, pp. 131–144]. The training dataset may further be split into a validation and training dataset to perform model selection (feature selection or hyperparameter tuning), where a model is trained on the training set and evaluated against the validation set (to evaluate its validation performance) to select the best model parameters [111, pp. 131–144]. The final model is then evaluated against the testing dataset to evaluate the model's generalization performance [101, pp. 171–175][111, pp. 131–144]. Another evaluation method is called KF-CV, which is also known as standard (or traditional) KF-CV [111, pp. 131–144]. KF-CV involves splitting a dataset into k subsets (or folds), and a model is trained for each fold. This is accomplished by combining $k-1$ subsets to form the training dataset and using the last subset as the testing set. This is repeated k times, where each subset is left out for testing one after another, meaning a sample is part of the testing dataset exactly once. Multiple iterations of this process can be run, and the final performance is averaged over the k folds and j iterations of the CV process [217][111, pp. 131–144]. Sometimes nested KF-CF is used when model selection is also desired, where an inner loop of KF-CV is applied to each outer CV loop's training dataset to produce a triplet of training-testing-validation datasets for each outer-inner fold pair. [111, p. 144]. In cases where data is extremely limited, leave-one-out CV can be used, where, similar to KF-CV, the dataset is split into folds, but instead the testing datasets only include a single sample [111, pp. 131–144].

2.3.1.2 Evaluation Metrics

Although models can be evaluated using many different types of evaluation metrics, we focus our discussion on metrics for evaluating regression models, given the nature of the experiments conducted in this thesis. Interested readers are referred to Japkowicz et al. [111, pp. 83-127] for a more in-depth look at evaluation metrics, including those for evaluating classification models. The evaluation metrics used in this thesis include the following: root mean squared error (RMSE), mean absolute error (MAE), Pearson's correlation coefficient (CC), coefficient of determination (R^2), and mean absolute percentage error (MAPE). These metrics are defined in Equations 2.1, 2.2, 2.3, 2.4, and 2.5, respectively, where x_i is the actual value of sample i, y_i is the predicted value for sample i, n is the number of samples, and $\bar{x}$ and $\bar{y}$ are the mean of the actual and predicted value, respectively.

2.3.1.2.1 Root Mean Squared Error

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{n}} \tag{2.1}$$

The RMSE, which is the square root of the mean squared error (MSE) [111, p. 67], describes how much model predictions can be expected to be off by on average, where

smaller values indicate better performance. Furthermore, RMSE shares the same units as the target variable [124, pp.443–444]. The values of RMSE lie in the range $[0, \infty)$ [42].

2.3.1.2.2 Mean Absolute Error

$$\text{MAE} = \sum_{i=1}^{n} \frac{|x_i - y_i|}{n} \tag{2.2}$$

The MAE is similar to RMSE, but the difference is that the MAE penalizes large errors to a slightly lesser degree by using the absolute value operator instead of the square term [124, p. 444].

2.3.1.2.3 Pearson's Correlation Coefficient – R

$$\text{CC} = \text{R} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{2.3}$$

The CC (sometimes referred to as R) ranges from –1 to 1 [72, p. 45]. It measures the linear association between the target variable and model predictions. The square of the CC can be treated as R^2 [179, pp.432–433] for reasons described next in Section 2.3.1.2.4.

2.3.1.2.4 Coefficient of Determination – R^2

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{n} (x_i - y_i)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{2.4}$$

R^2 is a measure that compares the model predictions to a naive baseline model that only predicts the average of the testing set's target variable [124, pp.443–447], and it explains how much the target variable can be explained by model predictions in terms of variance [124, pp. 443–447][42]. R^2 values can lie in the $(-\infty, 1]$ range, where larger values indicate better performance, $\text{R}^2 > 0$ is the square of multiple CCs, R^2=0 indicates the target variable and model predictions are independent, and $\text{R}^2 < 0$ indicates predictions are worse than the naive baseline model's predictions [42]. Any negative R^2 value can be treated as zero R^2 by replacing the model with the naive baseline model [42], so we apply this logic to the results of this thesis for result presentation clarity.

2.3.1.2.5 Mean Absolute Percentage Error

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - y_i}{\max(\epsilon, x_i)}\right| \tag{2.5}$$

MAPE is the average error percentage of the predictions vs. actual values. MAPE may be multiplied by 100 to convert it to percentage units. It is useful to determine the magnitude of the errors, since an analyst may find that using only RMSE can lead to difficulties in model performance interpretability. Note that MAPE assumes non-zero actual values [178, pp. 66–67], meaning that very small values may introduce outliers in MAPE performance. To address this, a small value (ϵ) can be used to prevent divisions by zero[2].

2.3.2 Data-driven Models

In this section we present data-driven models that are used in (or related to) the studies detailed in this thesis.

2.3.2.1 K-means

K-means is an unsupervised clustering algorithm that clusters a dataset into k[3] clusters. First, k centroids are randomly chosen in the dataset, where each centroid represents a cluster center. Each point is then compared to the k centroids and is assigned to the cluster of the nearest centroid. The centroids are recomputed as the mean Euclidean distance of all points to the cluster's centroid. The process is restarted until convergence occurs; that is, when the centroids do not change after an iteration [11].

2.3.2.2 Linear Regression

The most basic form of linear regression (LR) is the simple LR model that only has one predictor variable and is shown in Equation 2.6:

$$y = \alpha_1 x + \alpha_2 \tag{2.6}$$

where y is the response variable (the prediction), x is the predictor variable, and α_1 and α_2 are the regression coefficients. This type of linear model assumes x and y are linearly dependent [179, pp. 403–404].

[2] https://scikit-learn.org/stable/modules/model_evaluation.html, accessed on June 17 2024

[3] k is a user-defined parameter representing the number of desired clusters

2.3.2.3 Partial Least Squares Regression

Partial least squares regression (PLSR) is a model that can be used to predict multiple target variables (this is known as multivariate PLSR). PLSR works well in cases where there are more features than samples, especially when large random error variance is present. One of the hyperparameters of PLSR is the number of latent components. The components are formed by a linear combination of the features for target variable regression using ordinary least squares regression [77]. PLSR takes the following form:

$$Y = \beta_0 + \beta_1 T_1 + \beta_2 T_2 + ... + \beta_p T_p \tag{2.7}$$

where Y is the prediction, p is the number of latent components (typically much smaller than the number of features), T_i is a linear combination of the predictor variable for component i, and β_i is a regression coefficient. The sample correlation is 0 between any pair of latent components [77].

2.3.2.4 Random Forest

Random forest (RF) was introduced by Breiman [32] and is an ensemble model that comprises a collection of decision trees. The trees vote and the majority class (the average of the predictions, for regression tasks) is selected as the prediction. When the trees grow, a random selection of features is applied at each node. RF is one of the most popular ML models used in spatial predictions [176] and environmental mapping [173], and it is able to deal with missing data [204].

2.3.2.5 Support Vector Machine

The support vector machine (SVM) model is a linear classifier that attempts to maximize the margin (distance between the closest sample and the decision boundary) of a decision boundary [72, p. 22]. The decision boundary is defined as a linear combination of the support vectors, where the support vectors are the samples with the smallest distance to the decision boundary. Sometimes the input dataset is not linearly separable, so a slack variable can be introduced to instead have a soft margin, where soft margins allow samples to be within the margin or be on the wrong side of the decision boundary [72, pp. 211–217]. SVM models can be used for classification and regression problems [301].

2.3.2.6 ZeroR

ZeroR is a naive model that simply predicts the mean of the training data's target variable and ignores the input feature values [215]. It is a useful baseline model for identifying models that may be failing when evaluating model performance.

2.3.2.7 Neural Networks

Neural networks, sometimes referred to as artificial neural networks, consist of an input layer, a hidden layer, and an output layer. Deep learning models are neural network models that have multiple layers [185]. Neural networks can be trained by finding the optimal weights using the back propagation algorithm [271]. Fully connected networks are networks where nodes in a layer are connected to every node in the next (forward-direction) adjacent layer, while a partially connected network has missing connections between some nodes [101, p. 23]. Feed-forward networks do not have any form of feedback output where the forward layers act as input to previous layers, whereas recurrent neural networks (RNNs) do have such feedback loops [101, pp. 21–23]. Next, we discuss multilayer perceptron (MLP), long short-term memory (LSTM), and convolutional neural network (CNN) models. For readers that want to learn more about deep learning and neural networks, we refer them to this comprehensive book [101] on the subject.

2.3.2.7.1 Multilayer Perceptron A MLP model consists of an input layer, one or more hidden layers, and an output layer, each containing a collection of neurons. The model can be trained by finding the optimal weights using the back propagation algorithm [271], and can act as a complex non-linear mapping function to represent some phenomenon [291]. The model for neuron k can be represented algebraically (see Equation 2.8):

$$\begin{aligned} v_k &= b_k + \sum_{j=1}^{m} w_{kj} x_j \\ y_k &= f\left(v_k\right) \end{aligned} \tag{2.8}$$

where m is the number of input signals (features) and weights; $x_1, x_2, ..., x_m$ are the input signals; $w_{k1}, w_{k2}, ..., w_{km}$ are the inputs' respective synaptic weights; v_k demonstrates the resulting affine transformation of adding the bias b_k to the result of the input-weight summation; $f\left(\cdot\right)$ is a non-linear activation function (e.g., ReLu); and y_k is the output signal.

2.3.2.7.2 Long Short-term Memory LSTM was proposed by Hochreiter et al. [104] to handle the vanishing gradient problem [233]. LSTM is a type of RNN, where RNNs are typically used on sequences of data [233], since input from a previous state can be represented in the network, thanks to feedback neuron connections [233]. For every time step, LSTM keeps track of a hidden state vector that is affected by a previous state to be able to model temporal patterns. Parameters (e.g., weights) are shared across time steps, where fully connected layers are typically used [291].

Gated cells (namely the output gate, input gate, and forget gate) are used to store additional information for controlling memory. The input gate Γ_i (defined in Equation 2.9) controls the data fed into the LSTM cell. The output gate Γ_o (defined in Equation 2.10) controls the output of the LSTM cell. The forget gate Γ_f (defined in Equation 2.11) controls the cell's memory [233].

$$\Gamma_i = \text{sigmoid}\left(w_{hi}h^{<t-1>} + w_{xi}x^{<t>} + b_i\right) \tag{2.9}$$

$$\Gamma_o = \text{sigmoid}\left(w_{ho}h^{<t-1>} + w_{xo}x^{<t>} + b_o\right) \tag{2.10}$$

$$\Gamma_f = \text{sigmoid}\left(w_{hf}h^{<t-1>} + w_{xf}x^{<t>} + b_f\right) \tag{2.11}$$

where $x^{<t>}$ is the input vector at time t; w_{hi}, w_{xi}, w_{ho}, w_{xo}, w_{hf}, and w_{xf} are the weights; b_i, b_o, and b_f are biases; and $h^{<t-1>}$ is the hidden state of the LSTM cell at time $t-1$. An additional memory cell, named the candidate memory cell $c^{<t>}$, is also included in a LSTM cell. The idea is that LSTM can control when to update the additional memory cell with a new input and can decide by how much it forgets the old content. Equation 2.12 defines how $c^{<t>}$ is computed:

$$\begin{aligned} \tilde{c}^{<t>} &= \tanh\left(w_{hc}h^{<t-1>} + w_{xc}x^{<t>} + b_c\right) \\ c^{<t>} &= \Gamma_i \circ \tilde{c}^{<t>} + \Gamma_f \circ c^{<t-1>} \end{aligned} \tag{2.12}$$

where $\circ$ is the element-wise multiplication operator; w_{hc} and w_{xc} are weights; and b_c is the bias. The last computation that is part of an LSTM cell is the output of the hidden state $h^{<t>}$ and is computed as follows in Equation 2.13:

$$h^{<t>} = \Gamma_o \circ \tanh\left(c^{<t>}\right) \tag{2.13}$$

2.3.2.7.3 Convolutional Neural Networks CNN [146] models are a form of deep feed-forward neural networks [120] that use weight-sharing to limit the receptive field of neurons [101, p. 30]. They are commonly used in computer vision tasks, since they can learn spatial hierarchies of features [154]. Interested readers are referred to Nevavuori et al. [185] for a detailed summary and illustrations of CNN. CNN models have three main types of layers, namely, convolutional layers, pooling layers, and fully connected layers. Convolutional layers reduce the number of parameters by sharing weights across the first two dimensions of the input data, and convolutional layers consist of: a tenser $x \in \mathbb{R}^{h \times w \times d}$ as input, a non-linear activation function (typically ReLu), occasionally a pooling layer (typically maximum-pooling), and the output of a tensor $y \in \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{d}}$, where these layers are represented by Equation 2.14:

$$\boldsymbol{y} \quad = p\left(f\left(\boldsymbol{W} * \boldsymbol{x} + \boldsymbol{b}\right)\right) \tag{2.14}$$

where $p(\cdot)$ is a pooling function, $f(\cdot)$ is a non-linear activation function, $\boldsymbol{W} \in \mathbb{R}^{l \times l \times \hat{d}}$ is the weight matrix that defines the convolution filter of size $l \times l$ such that $\hat{d}$ filters (a.k.a. feature maps or kernels) are applied, * is a 2-dimensional convolution operator over dimensions h and w, $\boldsymbol{b} \in \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{d}}$ is the bias, and d is the number of features of the 2D ($w \times h$) input dataset [291]. For example, x could be a $w \times h$ image with d spectral bands.

2.3.3 Ensemble Learning

EL involves combining multiple models together to form a new model, called an ensemble model, that combines model predictions (instead of predictor variables) to make predictions. Typically, better model performance accuracy, and even lower performance variance [35, p. 369], can be achieved by using EL rather than using a single model [34]. However, using EL comes at the cost of increased model complexity [72, pp. 330–342]. One of the most common EL approaches is uniformly weighted ensemble averaging [34] defined in Equation 2.15:

$$\text{Pred}\left(y|x\right) = \sum_{k=1}^{M} w_k \text{Pred}\left(y|x;\theta_k\right) \tag{2.15}$$

where M is the number of models, w_k is the averaging weight for model k ($w_k = \frac{1}{M}\ \forall k$), y is the target variable being predicted, x is a multi-dimensional input sample, and θ_k are the hyperparameters for the pre-trained model k [63, 34].

Other ensemble approaches include bagging, boosting, and stacking (or stacked) ensembles [72, pp. 330–342]; we focus on the latter. Stacking involves using the predictions of many models to train a new meta-learner to improve predictions [35, pp. 482–484]. According to Ting et al. [257], stacking ensembles involve stacking models into two layers (many level-0 models and a level-1 model) such that the level-1 meta-model is trained using the predictions of level-0 models. It is also useful to think of two layers of datasets, where level-0 datasets are the original datasets used to train and tune the level-0 models, and the level-1 cross-validated datasets are created using level-0 model predictions, such that for every sample in the level-0 dataset, there is an associated sample in the level-1 dataset containing the predictions of every level-0 model.

2.3.4 Federated Learning

The term 'federated learning' was first created in 2016 by McMahan et al. [170]. FL is a form of distributed ML [271] that combines databases, cryptography, and ML [117]. The FL iterative training process can be generally broken down into the following steps [244]: a) hyperparameter initialization, b) client selection, c) global model broadcasting, d) local training, e) local model update sharing, f) global model aggregation, g) go back to step b) until a termination criteria occurs, and h) final global model update.

FL is straightforward to apply in the IoT setting, since by applying intelligence at the edge of the network near the data sources, network overhead is reduced and data-sharing privacy requirements are also respected [186], although model updates may still reveal sensitive information [195, 63, 117]. Devices that host ML logic are referred to as clients in the FL literature. Clients collectively hold the entire training dataset in the form of data islands [285]. Clients collaborate in coordinating with a central server (centralized FL) to train a global model using their locally stored raw data, with the goal of benefiting from one another's data without having to share their data. Clients only

share model updates. Once trained, the global model will later be used to benefit all clients [63], including new clients. The exact methodology taken to aggregate the model updates into a global model depend on the FL aggregator algorithm of choice. Examples of FL aggregator algorithms include FederatedAveraging (FedAvg) [170], FedProx [149], and Federated Matched Averaging [63]. In the present work we use the FedAvg algorithm (for its simplicity), which we discuss about next in Section 2.3.4.2. Furthermore, communication bottlenecks in centralized FL can be reduced by using peer-to-peer (P2P) systems [244] and hierarchical FL [195], such that fully distributed P2P systems remove the central server entirely, and hierarchical FL involves adding an intermediary node between the cloud and sensors. The presence of a central server in FL differentiates centralized FL from P2P (or decentralized) FL [293]. Hierarchical FL involves a) computing resource optimization, b) storing the data, and/or c) further aggregating model updates before sending the updates to the cloud. Hierarchical FL removes the necessity of sensor nodes (SNs) of performing all the local client data-processing. However, hierarchical FL comes at the cost of being more vulnerable to cyberattacks given the increased number of nodes [195]. To counteract the various attacks that target decentralized model training, privacy preserving technologies can be used, including secure multi-party computations, secure aggregation protocols, homomorphic encryption, differential privacy, and trusted execution environments [130], which we describe in more detail next in Section 2.3.4.1. FL can also be categorized into three different data partitioning classes depending on what features or instances are shared among client datasets, namely horizontal FL, vertical FL, and federated transfer learning [244]. Cross-device FL is the setting where client devices are resource-constrained, and when clients are heavy-duty nodes, this is known as cross-silo FL [244]. For readers that want to learn more about FL, we refer them to this book [117], which discusses many relevant topics.

2.3.4.1 Privacy Preserving Technologies

To counteract the various attacks that target model decentralized training, privacy mechanisms can be used, including: differential privacy, homomorphic encryption, secure multi-party computations, and trusted execution environments [130].

2.3.4.1.1 Differential Privacy Differential privacy quantifies and limits the amount of information that could be disclosed during distributed training [117]. This is achieved by adding noise to data or model outputs to a) limit the amount of information disclosed by clients [63] and b) make restoring the data impossible [285]. This comes at the cost of hindering model prediction accuracy, which creates a trade-off between privacy vs. accuracy [63, 285]. Differential privacy requires that removing a client's data from a dataset will not significantly change the result of aggregations applied to the dataset. This property guarantees that a client's privacy is not compromised if a data breach occurs [87].

2.3.4.1.2 Homomorphic Encryption Homomorphic encryption involves applying certain mathematical operations on ciphertext without needing to decrypt the data first

[63, 87, 117]. Homomorphic encryption enables secure multi-party computation [117]. A fully homomorphic scheme allows any operation to be applied to the ciphertext [63], but comes at the cost of increased computational complexity [117]. Partial homomorphic schemes are computationally less complex than fully homomorphic schemes, but partial homomorphic schemes only support a partial set of operations, such as addition (additive homomorphic scheme) or multiplication (multiplicative homomorphic scheme) [87].

2.3.4.1.3 Secure Multi-Party Computation Secure multi-party computation, a subfield of cryptography, involves multiple parties combining their private inputs to compute some output using an agreed-upon function without revealing their inputs to each other. Performing secure multi-party computations is resource-intensive [117]. A secure multi-party computation example follows: determining the wealthiest individual in a group without needing individuals to disclose their wealth to each other [87].

2.3.4.1.4 Trusted Execution Environments Trusted execution environments are used to execute code on machines such that the machine or machine's owner is not trusted. Trusted execution environments provide confidentiality in terms of hiding the code being executed, it provides integrity in the sense that the code being executed cannot be altered, and it provides measurement/attestation (proof that a specific binary is running) [117]. It also offers privacy, authenticity, and data access rights [244].

2.3.4.2 Federated Averaging Algorithm

In this section we present FedAvg [170], which is shown in Algorithm 1. The FedAvg algorithm computes the average model weights obtained from each client to compute a global model and repeats the process until model convergence, focusing on neural network models. The algorithm begins by having the coordination server randomly initialize a set of global model weights w_0 after which the rounds of training begin. A round of training involves selecting a random subset S_t of m clients to be involved in the training round t, which can be seen as bootstrap sampling over the clients that is done over several rounds of training to remove some bias from the client selection; where m is a C-fraction of the total number of clients K (e.g., $C = 0.5$ means half the clients would be selected). Once the clients have been selected, each client k in S_t is sent the up-to-date version of the global model weights w_t. Each client k proceeds by splitting its local dataset D_k into batches of size B, where β is the set containing these B batches. E epochs of local gradient descent is then performed by each client k. The model weights are updated locally using every batch in β via $w = w$ - $\eta \nabla \ell\,(w; b)$, where η is the learning rate, ∇ is the gradient operator, and ℓ is the loss function. Once client k finishes its round of training, it sends its local updated weights to the coordination server. The coordination server waits to receive the local updated weights from each client before updating the global model. Once all the updates are received, the server computes the average gradient over all clients using a weighted average based on the number of samples a client has (n_k) and the total number of samples (n). At this point a round of training is complete and the server begins a new

round of training. Once the model converges, the training process is complete and the server can broadcast the final model to each client. Note that with $B = \infty$ (using the entire local datasets in each client's training process) and $E = 1$ (one training epoch), FedAvg would be equivalent to Federated Stochastic Gradient Descent (Federated SGD).

Algorithm 1 `FederatedAveraging (FedAvg)` [170].

Data: clients are indexed by k, n_k = number of local examples for client k, D_k = training dataset partition for client k, K = number of clients, B = local minibatch size, E = number of local epochs (number of local training iterations per round), η = learning rate, $C \in (0, 1]$ = C-fraction of selected clients each round, and n = total number of examples.

Result: Trains a global model by iteratively training local models on the clients and aggregating the weights on the coordination server.

Function `startFederatedTraining(`K,B,E,η,C,n`):`

- `// this function is ran on the coordination server`
- initialize w_0 `// random weight initialization`
- `// training round`
- **for each** round $t = 0, 1, 2, ...$ **do**
  - $m =$ `max`$(C \cdot K, 1)$ `// number of clients to select`
  - $S_t =$ (random set of m clients) `// select the clients`
  - **for each** client $k \in S_t$ **in parallel do**
    - $w_{t+1}^k =$ `clientUpdate(`k,w_t`)` `// global model broadcast`
  - **end**
  - $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ `// client model update aggregation`
- **end**
- **return** w_{t+1}^k

Function `clientUpdate(`k,w`):`

- `// ran on client` k
- $\beta =$ (split D_k into batches of size B)
- **for each** local epoch $i = 1, 2, ..., E$ **do**
  - **for each** batch $b \in \beta$ **do**
    - w = w - $\eta \nabla \ell(w; b)$ `// local gradient descent`
  - **end**
- **end**
- **return** w to server `// sends local model update to coordination server`

2.3.4.2.1 Limitations The FedAvg algorithm has some limitations: FedAvg assumes IID client datasets [63], which is a strong assumption in the smart farming setting where agri-data is expected to be non-IID. In this thesis we chose FedAvg as the only FL aggregator algorithm of choice for simplicity reasons and for showcasing our proposed distributed ML methodology. Other FL aggregator algorithms have been proposed to better deal with non-IID client datasets, and these algorithms may have been more appropriate choices in

this thesis; such FL aggregator algorithms include FedProx [149], Federated Matched Averaging [63], and Personalized FL [244, 279, 147], which we discuss next in Section 2.3.4.3.

2.3.4.3 Other Federated Learning Aggregator Algorithms

FedProx [149] empirically improves FedAvg's performance by adding a proximal term to the local objective functions [117]. Federated Matched Averaging deals with the heterogeneity of the client resource requirements by assigning variable amounts of work to each client, and it has been found to outperform FedAvg and FedProx [244]. Furthermore, the global model may only be useful for a subset of clients. Personalized FL can be used to improve the clients' experience of using the global model by fine-tuning the model for each client. This can be achieved by a) global model personalization, by adding an additional final step and training the global model on each client using data-based approaches or model-based approaches, or b) learning personalized models, where the entire FL process is adapted to keep the client personalization in mind using either architecture-based approaches or similarity-based approaches [244]. An example of a personalized FL approach is Federated Deep Learning [98]. Federated Deep Learning has an additional client-side model aggregation step after the global model broadcast step. This additional step aggregates the global model with a client's model from the previous training round to enable learning local data patterns while retaining global data patterns learned.

2.4 Spatiotemporal Data Analysis

There are different types of data. For example, cross-section datasets are created by obtaining samples of data during some point in time (or some time period) without considering the effects of time on the variables [178, p. 109]. Spatial data, time series data, and spatiotemporal data are different from cross-section data in that they may vary in both space and time. In this section we focus on spatiotemporal data and describe important properties that should be taken into account when analyzing this type of data. Interested readers are referred to Shekar et al. [238] for an overview on spatial and spatiotemporal data analysis techniques.

2.4.1 Spatial Data

A spatial dataset, according to Nikparvar et al. [189], contains data samples that are tied to a location and has different properties compared to other types of datasets. These properties are: spatial dependence, spatial heterogeneity, and spatial scale. Geographic spatial data instances that are sampled close together tend to have values more similar to each other than samples that are sampled further apart from each other [230, 99]; this is called SA [99].

Methods for detecting and quantifying SA include Moran's I test statistic [100], semivariograms [189] (or variograms [23]), Geary's C [189], Getis [156], and Lagrange Multiplier tests [88].

To perform predictive analytics on multi-variate spatial data, it is important that the datasets share the same spatial resolution [88]. This can be achieved using data fusion. Examples of applying data fusion to spatial datasets include: a) mapping a higher resolution dataset and a lower resolution dataset to the same spatial grid by interpolating the high resolution data to a lower resolution grid via kriging and b) using a median or mean filter (taking the median/mean value of pixels in a neighbourhood) to lower the resolution of a spatial dataset [132].

2.4.2 Time Series Data

Time series datasets are chronologically ordered [105], and contain a sequence of samples of some variable of interest [178, p. 2]. Univariate time series data only have one variable at each timestamp, while multivariate time series data have multiple variables at each timestamp [7, p. 21]. Observations are typically sampled at regular intervals [178, p. 3], although some time series may have irregular interval duration lengths [79]. Applications of time series forecasting apply to many industries, including business, finance, economics, medicine, environmental science, social science, government, politics, and finance [178, p. 2]. Time series data may exhibit properties that non-time-series data do not have. These properties include: trends, seasonality, temporal autocorrelation, stationarity, atypical events, [178, pp. 6–20], multi-source time series [7, pp. 27–28], and spurious correlations [7, pp. 102–103].

A time series dataset with seasonality, periodicity, or trends is said to be non-stationary [158]. Temporal autocorrelation can be detected using the Durbin-Watson Test [178, p.178]. A partial autocorrelation function is useful when determining window sizes and temporal scales for further temporal data analysis, since it reveals the distance/lag after which the temporal autocorrelation structure ceases to be relevant. The autocorrelation function is similar to the partial autocorrelation function, but fails to remove redundant information [7, p.95].

To enable performing predictive analytics on time series data, models that directly capture the temporal structure of the raw data (e.g., LSTM [104], bidirectional LSTM [202], or autoregressive integrated moving average (ARIMA) [178, pp.327–427]) can be used, and/or data fusion can be applied. An example of temporal data fusion is extracting lagged features [70].

Methods for time series prediction can be categorized as classical forecasting methods, ML and deep learning methods, and hybrid methods [158].

- Among the classical methods, some models are designed for stationary time series forecasting, for example, autoregressive moving average models. N_2O emission datasets tend to be non-stationary, meaning autoregressive moving average models are a poor choice for predicting N_2O emissions. Differencing can be used to convert a non-stationary time series dataset into a stationary time series dataset, and this is the strategy employed by ARIMA models, to effectively model non-stationary time series datasets. ARIMA models assume linear relationships in the time series datasets, and, as a result, these models perform poorly when modeling non-linear relationships

[158]. The factors affecting the N_2O emission creation process are non-linear [228], making ARIMA unsuitable for N_2O emission prediction. There are classical time series forecasting methods that do well at performing non-linear time series forecasting, including constant conditional correlation models, threshold autoregressive models, and autoregressive conditional heteroscedasticity models [158]. Although in terms of applying these methods for N_2O emission prediction, it is worth noting that autoregressive models are not suited for time series analysis problems where the historical values of the target variable are not available at the time of making predictions. This makes autoregressive models not a viable option for prediction N_2O emissions using the methodologies applied in this thesis, since in this thesis we are assuming the expensive monitoring equipment used to gather the target variable will not be available after system deployment. Nevertheless, autoregressive models can still be useful for some PA applications, for example, in weather forecasting applications [202].

- ML and deep learning models are better suited for modeling complex non-linear relationships and have been found to perform well at non-linear time series forecasting [158], meaning ML and deep learning models should be reasonable candidates for making N_2O emission predictions, given the difficulty of the N_2O emission prediction problem (see Section 2.8.3 for more details on N_2O prediction). Examples of these types of models include deep learning models, fuzzy theory, and Gaussian process regression. Deep learning models (e.g., RNN, LSTM, and transformers) usually require large training datasets to perform well. Gaussian process regression may be favored over deep learning models, since it requires fewer parameters. Fuzzy theory forecasting methods have the benefit of having a more interpretable model [158].

- Hybrid approaches combine ML/deep-learning and/or classical time series forecasting methods. Using such methods have the benefit of combining the strength of multiple models; for example, for a CNN-LSTM hybrid model, the CNN part of the model could be used for identifying latent features, and the LSTM part of the model could be used for identifying short-term and long-term trends. Combining models for time series forecasting is also beneficial, since usually time series datasets will have both linear and non-linear relationships [158].

2.4.3 Machine Learning on Spatiotemporal Data

Doing regression or statistical operations on spatially (or temporally) autocorrelated data will lead to overfitting and underestimating prediction errors [225], since datasets that have SA violate the data independence assumption made by some ML methodologies [230] (such as KF-CV [100, 273]). To address this challenge, the following techniques could be used: a) applying block CV to create spatially (or temporally) disjointed testing and training datasets [234], but this may lead to poor performance when test samples are from new unseen locations (or time periods) [189, 230, 273], b) using spatial (or temporal) regression models [88], c) using spatiotemporal features, or d) using ML models that internally incorporate spatiotemporal structure [189]. Model selection methodologies, including feature

selection[100, 189] and hyperparameter tuning [234], should also take the spatial (and/or temporal) structure into account [234].

2.4.4 Block Cross-Validation

CV must be carefully applied to data that contains spatial (or temporal) structure [221]; otherwise, over-optimistic performance results may be obtained [234], and this could lead to drawing incorrect conclusions. In fact, most ML studies in the literature that involve the use of earth observation spatial data only apply KF-CV to evaluate their ML models [23]. We found that this trend was also observable in the UAV imagery-based yield prediction literature, which we elaborate on later in Section 2.9.3. Nevertheless, KF-CV can be appropriated for model evaluation if the model being evaluated is not expected to generalize to new spatial regions (or time periods) to make new causal inferences. On the other hand, for problems where data from new unseen spatial regions (or time periods) are expected to be fed to the model and reasonable extrapolation performance is expected, block CV can be used to evaluate the extrapolation performance. Unfortunately, geological trends or environmental gradients may be lost when splitting the training and testing data into spatially disjointed folds [230], making the performance evaluation of model spatial extrapolation pessimistic [230, 273]. Using both KF-CV and block CV can be insightful, since over-optimistic and over-pessimistic model performance estimations can be produced, respectively. Unfortunately, the challenge remains that estimating unbiased model performance, when using datasets that contain spatial structure, is not trivial [273].

2.5 Smart Farming

PA involves observing and analyzing agri-data to respond to the data's variability, whereas smart farming (otherwise known as Agriculture 4.0) extends the notion of PA by including real-time information technology and predictive analytics for decision making [229]. Smart farming is a management practice that makes use of a) IoT, b) cloud computing, c) fog/edge computing, and d) artificial intelligence to optimize agriculture practices [66]. Smart farming provides advantages, including information exchange, safety control, food security, fraud prevention, and business cost reductions [66]. The cost and benefits of a SFS should be clear to a farmer and affordable to make it worthwhile for these individuals [16]; a SFS should also be simple to use. Otherwise, a SFS may suffer from low adoption rates if a) it is too expensive [144], b) it was not built using a user-centric design [144], and c) it is too complex and requires too much technical knowledge from the farmer [246]. Furthermore, a SFS should support many communication technologies (e.g., Zigbee, Wi-Fi, and long-range wide-area-network (LoRaWAN)) [157]. A PA application should support low-latency requirements [242], and the enabling IoT system should support of heterogeneous network communication protocols and hardware [46]. A smart farming solution should have low-cost field nodes [242] to enable sensing node deployment over large areas, and the system should have self-organization and management [242]. Protocols that have

been found to be most appropriate for smart farming include: Message Queuing Telemetry Transport (MQTT) [82], NB-IoT, Zigbee, and LoRaWAN [242].

2.6 Remote Sensing

Remote sensing is a non-destructive [213] sensing method for gathering imagery. To obtain imagery using remote sensing, a red-green-blue (RGB) (or visible-light), MS, or HS camera can be used, where RGB and HS cameras tend to be the most affordable and expensive options, respectively [196]. Imagery of a farm can be obtained using remote sensing via satellites, UAVs, or aircrafts, or hand-held/tractor-mounted imaging equipment [41]. Using satellite imagery to enable yield prediction is a common approach [71] and comes with the advantages that a) there are many free publicly available satellite imagery datasets [128], b) satellite imagery tends to have high spectral resolution [196], and c) additional input data or specialized in-field sensing equipment is not required by the farmer [112]; however, satellite-based remote sensing suffers from low spatial and temporal resolutions [296] and is vulnerable to weather [296]. Performing remote sensing using UAVs is favorable due to the higher spatial and temporal resolutions [140] and the ability to estimate plant height using structure-from-motion processing [301]. However, UAV-based approaches that use MS/HS cameras have high monetary costs and high complexity [296]. Imagery and phenotyping[4] data tend to be spatially autocorrelated when gathered from a single field [100], meaning spatially aware analysis methods should be employed. Used frequently in agriculture studies are VIs. VIs are derived from the reflectance values of the raw imagery by using mathematical operations such as linear combinations or ratios and can be used to represent the state or condition of target vegetation [296]. For example, the normalized difference vegetation index (NDVI)[5] is a VI that can be used to represent the general rigour and health of plants [175]. Texture indices (TIs) are also sometimes used in remote sensing studies and are also derived from imagery [97].

2.7 Crop Yield Data

During a harvest, a yield monitor will periodically (typically at 1 Hz [251]) record its position (typically accurate to within 1 to 3 m [159]) and the measured yield. The yield dataset resolution is relatively coarse-grained, since a yield reading represents an estimation of the yield in a harvested area between two successive GPS time epochs [127].

2.7.1 Cleaning

Cleaning yield datasets is important before using them for analysis, since they tend to be noisy by nature. Variability in operating conditions can affect the yield sensor readings.

[4] plant trait information data

[5] We omit defining the full names of any other VI involved in the studies described in this thesis, since the names and abbreviations of these VIs can be found in Appendix A in Tables A.1, A.2, and A.3.

We identified common yield cleaning steps in our previous work [132] and listed them as follows: a) removing yield readings that are less than or equal to zero, b) removing yield readings that exceed biological limits (e.g., 250 bu/ac), c) removing extreme yield value outliers using inlier detection techniques that only consider samples in similar operating conditions to avoid bias, d) removing yield readings that fall within a short distance from the edge of the field boundary (or headlands), e) removing yield readings at the start and end of harvest passes, f) applying a moisture filter, g) removing points involved in a turn, h) removing overlapping points, i) removing short segments, j) removing samples with speed outliers, k) removing samples with positional errors, l) removing narrow finishes, and m) removing samples that are in an area that has already been harvested.

2.7.2 Post-processing

After cleaning, there are further post-processing steps required to prepare the yield dataset for analysis. Semivariograms are used to model geospatial data, and can be applied to yield data, since samples closer to each other are more correlated than samples that are farther apart. In particular, *interpolation* of yield data onto a grid is a common post-processing step after cleaning. Yield data points can be unevenly distributed in space, so separating the field into a grid can be useful for precision map generation, since a) crop locations with missing yield samples will be interpolated and b) it can be convenient to have only a single interpolated yield reading per grid cell/pixel instead of having a variable number of yield samples per cell. Interpolation can be done using kriging, and this can be done using the Vesper software. It is recommended in the literature [275, 274] to use local kriging and local variograms when datasets have a sufficient number of points. After interpolation, any yield cell with high kriging variance (high uncertainty in the interpolation prediction) can also be removed. Data fusion (previously described in Section 2.4.1) should also be applied to combine other data sources to a common grid.

2.7.3 Prediction

Yield prediction scale can be done at global-level [113], region-level/county-level, field-level/plot-level, or pixel-level/within-field-level [132]. The two most commonly used types of models for predicting yield are processed-based models (otherwise known as mechanistic crop growth models or crop simulation models) and data-driven models (e.g., ML or deep learning) [113, 17]. Processed-based models take as input weather, soil, and crop phenology data [112] and simulate the physiological process of crops, given input management practices and environmental conditions. They tend to have high complexity, long run-times, and complex calibration [17]. Examples of such processed-based models include Agricultural Production Systems sIMulator (APSIM) [71], Decision Support System for Agrotechnology Transfer (DSSAT) [71], and Hybrid-Maize [240]. Data-driven models are simpler than processed-based models, because they use statistical patterns found in the training data to model the relationship of the input factors affecting yield [113]. Data-driven models also have the advantage of being able to estimate yield at pixel-level scales

[17]. Furthermore, both types of models can be combined using a model-inversion approach [112]. A common yield prediction approach involves feeding VI features to ML models [212]. Some yield prediction studies also use TIs as features [282]. The most common VIs for yield prediction include: NDVI, $NDVI_{RedEdge}$, $NDVI_{Green}$, and WDRVI (where $\alpha = 0.1$) [37], which are defined in Appendix A in Tables A.1 and A.3. Regression models can be trained a) for each vegetation stage, instead of the entire growing season [37], b) using VIs from groups of vegetation stages [41], and/or c) using cumulative VIs (accumulated VI values over time) [38].

2.8 Nitrous Oxide (N_2O)

N_2O is one of the three primary long-lived GHGs (N_2O, CO_2, and methane (CH_4)) caused by human activities [208]. The lifetime of N_2O and CH_4 in the atmosphere is approximately 109 and 12 years, respectively [73], whereas for CO_2, parts of a CO_2 emission pulse can last from a couple hundred years to thousands of years in the atmosphere [245]. Given the difference in lifetime of these GHGs, the global warming potential (GWP) of GHGs should be compared over some time horizon [169]. Over a 20-year horizon, N_2O and CH_4 have approximately 273 and 80 times greater GWP than CO_2, respectively, and over a 100-year horizon, N_2O and CH_4 have approximately 273 and 30 times greater GWP than CO_2, respectively [73]. N_2O atmospheric concentrations have risen by approximately 20 % since the industrial revolution, from 270 ppb to 320 ppb [216], and are expected to keep rising [24]. N_2O is leading to ozone depletion in the stratosphere due to N_2O's reaction with oxygen, which produces nitric oxide (NO) [302] and nitrogen dioxide (NO_2), which are chemical molecules that react with ozone [216].

2.8.1 Creation

N_2O is primarily created by nitrogen (N) being transformed by bacteria in soil and oceans. The main bacterial processes are called nitrification and denitrification [216]. Nitrification occurs in aerobic[6] soil conditions [24], while the denitrification process requires organic matter and anaerobic[7] soil conditions. The absence of oxygen in soil may be due to wet soils or high concentrations of organic matter that consume oxygen rapidly [270]. Nitrification converts ammonium (NH_4^+) into nitrite (NO_2^-) and then converts nitrite into nitrate (NO_3^-). Measuring nitrification rates is difficult [198], and nitrification rates tend to increase with higher soil temperatures and when soil moisture reaches a field's capacity [256]. These two processes are affected by inorganic N, soil available water, organic carbon, soil temperature [254], and pH, where pH influences a) nitrate reductions (total denitrification) and b) N_2O to dinitrogen (N_2) reductions [270]. The amount of applied N has been found to be the most important driver for N_2O emissions [204].

[6] when there is access to free oxygen
[7] when there is a lack of oxygen

2.8.2 Measuring

Applying PA for estimating/measuring N_2O emissions is important for many reasons: a) food demand increase leads to an increase in N_2O emissions [75] and food demand is expected to rise significantly by 2050 [229], b) some countries are obliged to report annual N_2O emissions [216], c) modeling climate change's progression can be improved using emission predictions [210], and d) creating emission mitigation policies [115]. There are three types of emission estimation techniques: Tier I, Tier II, and Tier III. Tier I methods only consider applied N in their modeling [204], Tier II methods consider additional emission factors, and Tier III methods use models for estimation [204]. Analytical techniques for measuring N_2O can be classified into three categories, namely: chromatographic, optical, and amperometric [216]. Some challenges with measuring N_2O include a) N_2O emissions are episodic and have high spatiotemporal variability due to multiple processes and factors affecting emissions and b) the low atmospheric concentrations of N_2O make it difficult to measure emissions [216]. Requirements for an N_2O emission sensor include the sensor being affordable, being simple to deploy, having the ability to measure emissions over large areas, and ideally being accurate to within 0.1 ppb [216]. Measuring emissions can be done using gas chambers (static or automatic) or micrometeorological (or flux tower) techniques [28]. Static chambers have relatively low monetary costs and have low temporal resolution, while automatic chambers have high temporal resolution and are expensive [115].

2.8.3 Predicting

N_2O emissions can be predicted using process-based models, data-driven models [96], or a combination of both [155, 150]. Process-based models simulate the N_2O emission process by modeling known N_2O emission sources and sinks [216], but these models have limitations: a) they require agro-environmental expert knowledge [115, 96], b) model calibration is needed [96], c) they require a lot of computational execution time [204], d) they require the intensive availability of many site-specific parameters [96], e) they are not suited for making predictions at spatial resolutions higher than region-level [48], and f) they tend to produce low accuracy predictions at temporal resolutions higher than daily/monthly [228]. Examples of such process-based models include *ecosys* and Dynamic Land Ecosystem Model (DLEM) [150]. Data-driven models can be used instead (or combined with process-based models) to address the aforementioned limitations [96, 115] of process-based models. Data-driven models have been found to improve emission prediction accuracy compared to process-based models [198]. Data-driven models can require less input data, can be based on variables that are easy to measure, and can be tuned to handle various spatial and temporal resolutions [116]. Furthermore, data-driven models could be used by a) a farmer, to identify key drivers of emission events to help the farmer reduce emissions [22] or b) analysts, to fill (via interpolation) missing gaps of data in emission datasets [70]. However, data-driven models still have to deal with the high spatiotemporal variability of the emissions data, the low spatial resolution of N_2O emission datasets (since such high spatial resolution datasets are typically too expensive to obtain), and the non-linearity of the factors affecting the emission creation process [228]. Furthermore, N_2O emission time

series datasets tend to mostly contain static baseline emissions, whereas periodic bursts of high emission events occur infrequently [252]. This is an example of the common issue in ML known as the class imbalance problem [111, pp. 91–94].

By reviewing the literature [45, 160, 172, 115, 102], we created Table 2.1 to compare the costs and benefits of the various N_2O emission measuring techniques (including prediction-based approaches).

Table 2.1: Comparison of the costs and benefits of the various N_2O emission measuring techniques (including the prediction-based approach), where pixel-level indicates 1 m $\times$ 1 m sampling areas are supported and field-level indicates sampling areas beyond 10 m $\times$ 10 m are supported.

N_2O Emission Measuring Technique	**Economic Cost**	**Labor Cost**	**Sampling Area**	**Sampling Frequency**	**Sensing Accuracy**
Static Gas Chamber	Low	High	Pixel-level	Low	High
Automatic Gas Chamber	High[1]	Low	Pixel-level	High	High
Micrometeorological (or Flux Tower) Techniques	High[2]	Low	Field-level	High	High
Predicting N_2O Using Data-driven Models and Affordable Sensors	Low[3]	Low	Pixel-level	High	Low to Medium

[1] In our studies described in Chapter 5, a single LI-COR automatic gas chamber (approx. $20,000 CAD each), multiplexer, and spectrometer cost us approx. $100,000 CAD. Our entire LI-COR setup (including seven individual chambers) cost us approx. $250,000 CAD.

[2] Costs can exceed $100,000 CAD for flux tower setups that involve using high frequency (10 Hz) N_2O gas analyzers. For example, the price of an Aerodyne TILDAS Compact Single Laser N_2O Analyzer is approx. $200,000 CAD to $350,000 CAD.

[3] In our studies described in Chapter 5, building our Raspberry Pi-based predictor node prototype (described in Section 5.3.3) cost us approx. $1000 CAD.

2.9 Related Work

In this section we summarize the literature related to the studies conducted in this thesis.

Zhao et al. [300] proposed SpaceEye, a multimodal[8] and multi-temporal deep learning method for removing clouds from satellite images. SpaceEye makes use of radio frequency signals, MS imagery, and spatiotemporal analysis. They found SpaceEye was able to outperform other baseline approaches. They plan to continue their SpaceEye research by collaborating with insurance companies, agriculture companies, and governments to enable new PA applications. Khattab et al. [129] proposed an IoT architecture that has a back-end layer that is similar to the application layer of our proposed IoT architecture. Oliveira-Jr et al. [192] proposed an IoT solution based on off-the-shelf hardware for classifying soil types using soil, weather, and image data. Patrizi et al. [200] and Spisic et al. [248] proposed a similar system to our proposed PN, such that an expensive (or otherwise unavailable)

[8] multiple types of data are processed

sensor's readings were predicted using other sensor readings. Gunisetti et al. [91] performed a maize leaf disease detection study using deep learning models (using a deep quantum neural network model) that were ran in a fog computing node (a base station). Cruz et al. [49] proposed a LoRa-based solution for gathering agri-data and using fog analytics to make crop planting recommendations. They trained models using humidity and temperature sensor data and offered offline predictions. Onoufriou et al. [194] performed milk yield prediction, in the context of the agri-food supply chain, using encrypted deep learning to preserve privacy. They predicted milk yield with an accuracy of 87.6 %. There are air pollution monitoring studies that proposed a similar methodology to the one proposed in this thesis; that is, replacing expensive sensing equipment with a model and affordable sensors to enable more affordable and higher spatiotemporal resolution sensing solutions. Specifically, Kar et al. [121], Li et al. [148], and Yin et al. [288] performed air-quality monitoring using particulate matter data gathered from low-cost sensors and high-cost sensing stations, where model performance was validated against the more accurate (but more expensive) sensing station data.

We continue the discussion as follows: in the next sections, Section 2.9.1 summarizes studies (or works) that involved applying an IoT architecture to PA; Section 2.9.2 summarizes studies that involved using FL for PA; Section 2.9.3 summarizes studies that involved corn grain yield prediction using a) UAV-based and/or aircraft-based imagery and b) data-driven models; and Section 2.9.4 summarizes studies that involved N_2O emission prediction using data-driven models, such that a) we only considered studies that used terrestrial datasets, b) we did not consider studies that only used process-based models, and c) we did not consider studies that used datasets from aquatic ecosystems.

A detailed and qualitative comparison of most of the related works discussed later in this section can be found in Appendix B in the form of categorization criteria tables. These comparison tables also include studies described in this thesis; that is, Tables B.1, B.2, and B.3, compare the studies discussed in Sections 2.9.2, 2.9.3, and 2.9.4, respectively. We did not include the smart farming IoT architecture studies detailed in Section 2.9.1 in the comparison.

2.9.1 Smart Farming IoT Architectures

Many studies focused on precision irrigation. Premkumar et al. [209] included public weather data forecasts in their irrigation models. They trained cloud-based models using sensor data and public weather data forecasts, and deployed the trained models in the edge to predict soil moisture for automated soil moisture control. Patole et al. [199] proposed an IoT architecture for precision irrigation that made use of FL and EL. The roles of the aggregation nodes in their architecture are similar to the architecture proposed in this thesis. The difference between our proposed methodology and their work is that the authors created model ensembles using multiple global models that were trained using FL and various subsets of features from local datasets, whereas we created model ensembles using local models trained using LL. Dahane et al. [52] designed a cost-efficient platform for smallholder farmers. Goap et al. [84] proposed a system that makes use of

open-source technology, sensor data, and public weather data forecasts. Kamienski et al. [118] proposed a reference architecture for edge-enabled automated irrigation. Their cloud platform trained weather and soil moisture prediction models using multi-farm datasets. Manikandan et al. [163] proposed a system for automatic irrigation based on fuzzy logic and deep learning. They sent the data directly to the cloud and applied cloud analytics for deciding irrigation needs. Saad et al. [226] proposed a system such that their gateway was equipped with edge intelligence for predicting sensor values (temperature and humidity) using a local database of historical sensor values. Raghu et al. [214] proposed an IoT architecture that involved gathering sensor data, detecting outliers in the data, performing fog-based data filtering, and sending data aggregates to the cloud. Ting et al. [258] proposed a system that used blockchain and fuzzy logic. Their system sent all the sensor data directly to the cloud for further processing. Gueye et al. [89] proposed a cost-efficient IoT framework named eFarm-Lab that involved experts for gathering labeled data. They discussed crop disease detection and smart irrigation as system use cases. Other studies that also involved crop disease detection and precision irrigation include Rathore et al. [218] and Udutalapally et al. [265].

There are studies that only focused on crop disease detection. Cruz et al. [50] performed sensor data outlier detection. Their system supported both online and offline dashboards for visualizing the state of the farm. They ran disease detection models and outlier detection on a fog gateway. The crop disease detection was done using the Yolo v5 model and crop imagery, and outlier detection was conducted using an isolation forest model. Choudhury et al. [44] proposed an architecture that allows a farmer's smart phone to be used as a network bridge between the cloud and edge using a Wi-Fi hotspot. Imtiaz et al. [109] proposed an IoT platform designed to a) help connect farmers and stakeholders and b) incentivize farmers to use the system. They proposed a novel wireless sensor network protocol and performed CNN-based crop disease detection using imagery. Moreira et al. [180] proposed AgroLens, a low-cost green-friendly smart farming architecture that uses edge computing, and showcased their architecture for maize leaf disease prediction.

Some studies focused on PA domains other than precision irrigation and crop disease detection. Vasisht et al. [268] proposed an IoT platform called FarmBeats that enables weather-aware power cycling, supports UAV imagery streaming, supports local fog applications (e.g., UAV imagery orthomosaic[9] generation and soil precision map generation), and supports cloud analytics. They designed an IoT base station that acts as a gateway in the field to provide nearby sensors with Internet access. A gateway was deployed at the farmer's house where Internet connection was available, and the gateway performed more heavy-duty computations and hosted fog-based web services and other applications. Choudhury et al. [43] proposed a hand-held IoT device and showcased their proposed architecture in a crop recommendation use case. Colombo-Mendoza et al. [47] proposed an IoT system for yield prediction. Their models made use of climate data and crop management information. Guillén et al. [90] proposed an IoT architecture that supports both fog/edge and cloud intelligence for frost prediction. Lavanya et al. [145] proposed a three-tier smart farming architecture for nutrient sensing and fertilizer recommendation. They also proposed a novel light-based nitrogen-phosphorus-potassium (NPK) sensor. Rezk et

[9] a large image that is created by stitching many smaller images together

al. [220] proposed a high-level smart farming architecture and a ML methodology named WPART. Their study focused on crop productivity and drought prediction. Rodríguez et al. [222] proposed a smart farming solution named IoT-Agro involving edge-based sensor data outlier detection and cloud-based coffee production estimation. Zhang et al. [299] proposed an IoT architecture for smart farming to identify sensors that have high value data by making use of edge and cloud analytics.

2.9.2 Federated Learning in Agriculture

Many studies are related to disease detection using imagery. Aggarwal et al. [5, 4, 6] performed rice leaf disease classification studies by using transfer learning methods, feature extraction using pre-trained models, and FL. They evaluated cases with non-IID and IID client datasets. EfficientNetB3 was among their best performing models. Tripathy et al. [262] also performed rise leaf disease detection. They proposed a new model called Spotted Hyena Archimedes Optimizer (SHAO)-based LeNet (SHAO_LeNet). Some works performed apple disease detection [56, 260]. Other works performed tomato disease detection [153, 64]. Some works involved other types of crops in their disease detection studies. Antico et al. [14] used CNN models and found that AlexNet was the most appropriate model for FL, given its lower training time, high accuracy, and structure. Hari et al. [98] proposed a federated aggregation algorithm named Federated Deep Learning that was able to achieve 93 % accuracy. Mamba Kabala et al. [162] found ResNet50 generally produced the best performance. Muhammad et al. [181] found Mask R-CNN produced the best average performance of 93 % accuracy. Sharma et al. [237] found that the CL and FL setting obtained an accuracy of 89 % and an accuracy ranging from 82.19 % to 92 %, respectively. Thonglek et al. [253] proposed a resource-efficient FL methodology for decentralized FL that was able to reduce memory utilization of the MobileNet architecture by 67.58 %.

Some studies focused on pest detection and/or weed detection. Pastros et al. [201] proposed a system for cross-field weed detection in rural settings using FL and local HS imagery. They examined the trade-offs between LL, CL, and FL learning by evaluating the performance of a CNN model in those learning settings. They found CL achieved the best performance with 98 % accuracy, LL had the worst performance with 30 % to 70 % accuracy, and FL achieved up to 85 % accuracy. Jani et al. [110] proposed a FL algorithm that used a version of SGD that made use of differential privacy. Khan et al. [125] attempted to identify insects that harm crops by using imagery and FL. Their approach achieved 99.55 % classification accuracy. Seilov et al. [235] proposed preliminary planning steps for smart horticulture using FL for smart fruit storage and pest detection. Yoon et al. [290] found that their personalized FL approach outperformed standard non-personalized FL in terms of model performance, and slightly under-performed compared to CL model performance.

Some studies involved livestock. Huang et al. [106] performed a chicken age detection study. They gathered labeled chicken imagery from a private company's chicken coop. They designed lightweight versions of known CNN model architectures with the goal of deploying these models to resource-constrained edge devices. They implemented an application and deployed a prototype of their system in a chicken coop. They found that

their approach outperformed baseline model performance by 14.4 % to 96.1 % in terms of accuracy, and could reduce latency by 90.1 %. Mao et al. [165] proposed a FL aggregation algorithm for animal activity recognition named FedAAR that a) outperformed other FL aggregation algorithms and b) did only slightly worse than CL.

Some studies performed yield prediction and crop recommendation. Durrant et al. [63] performed a county-level yield prediction study using CL, model sharing via EL, and FL. Their experiments involved deep learning using satellite imagery and tabular data (soil, weather, and crop management data). They found that FL outperformed LL, and FL results were only slightly worse than CL. The EL approach generally performed better than LL, but worse than FL and CL model performance. Zhang et al. [298] proposed a federated RF approach for enabling maize crop breeding institutions. They found FL led to better performance than LL and similar performance to CL. Puri et al. [211] proposed a framework to improve some cybersecurity aspects of FL using blockchain and Ethereum. Li et al. [147] proposed a FL aggregation algorithm named FedPruning that led to 15.5 % to 20 % better yield prediction model performance than FedAvg. Major et al. [164] found FL ResNet-16 to be one of the better performing yield prediction models over the centralized ML models and centralized ResNet-16.

Some studies focused on intrusion detection. Friha et al. [74] proposed an intrusion detection system based on FL, called FELIDS. In their results, they found FELIDS did comparatively well to (and at times outperformed) CL. For example, for one of their datasets, the CNN model accuracy was 94.24 % and 94.09 % in the CL setting and FL setting (using FELIDS), respectively. Kumar et al. [142] proposed a 2-level privacy preserving framework for smart farming called PEFL and an intrusion detection system. Praharaj et al. [207] performed an intrusion detection study by using a digital twin architecture, personalized FL transfer learning, and a CNN-LSTM-based model.

There are also studies that focused on other application domains. Wu et al. [279] proposed a new personalized FL approach named FEDORA. Their goal was to make sure every client benefits from using FL and avoid having only some clients obtain better performance via LL. FEDORA used adaptive parameter propagation and selective regularization. They found FEDORA outperformed other baseline FL and personalized FL approaches. In two works, Akbari et al. [8, 9] proposed a system for optimizing UAV-based aerial networks. Their results demonstrated that their proposed approach could support real-time applications. Cui et al. [51] performed a groundwater level prediction study. They found their proposed ANFIS-IA-GWO model obtained better performance than other models. Idoje et al. [108] performed a crop classification study using FL. They compared the performance difference of using SGD vs. the Adaptive Moment Estimation (Adam) optimizer in the FL setting. They evaluated the performance of various models in a CL setting. Siniosoglou et al. [243] studied CO_2 and air humidity prediction. They found that FL was able to obtain generally better model performance than LL. Tiwari et al. [259] performed an imagery-based mango ripeness classification study using FL. They found that InceptionNet led to the best performance with 98.75 % accuracy. Vimalajeewa et al. [271] performed a milk quality prediction study using mid-infrared spectroscopy data and FL. They proposed a model by combining neural networks and PLSR, and found similar FL model performance as CL approaches. Yaseen [287] performed a heavy metal prediction

study in soil and water using FL, such that the available data sources were soil, weather, satellite imagery, and heavy metal indices datasets. Yu et al. [292] proposed a joint FL framework for edge computing-based smart farming. Their results showed that their proposed approach achieved increased model performance compared to horizontal FL and similar model performance compared to CL. Patole et al. [199] performed a precision irrigation study and proposed EnsembleFed, a methodology that combines fog computing, FL, EL (by combining multiple global models trained via FL), and multi-view dataset analysis. Their approach achieved up to 5.27 % better accuracy than other FL aggregator baseline algorithm methods in terms various metrics.

Some studies also involved IoT-based solutions. Ooi et al. [195] performed a precision spraying weed detection study using FL and proposed a hierarchical FL smart farming architecture. In their precision spraying study, they used a near-infrared (NIR) HS imagery dataset acquired from spraying robots. They found that using a FL-based model could achieve similar classification performance (96.3 % accuracy) compared to CL model performance (98.9 %), and found that FL led to better performance than LL (37.8 %, 60.1 %, and 69.7 % for the 3 clients, respectively). Abu-Khadrah et al. [2] proposed an amendable multi-function sensor control solution to improve agriculture sensor operation using the accumulation of historical sensor data and FL. Bera et al. [26] proposed a framework called E-CropReco that uses dew computing, edge computing, cloud computing, and FL for crop recommendation. Beras et al. [27] proposed a precision irrigation framework named FLAG that uses dew computing, edge computing, cloud computing, and FL. FLAG reduced latency and energy consumption, and led to improved model performance. Chen et al. [39] proposed an energy-aware FL scheme for sustainable smart farming that they called SusFL. Their system is a three-tier FL architecture. They compared SusFL to various FL aggregation schemes and found that their system outperformed these other methods in terms various metrics. Devaraj et al. [59] proposed a three-tier architecture named RuralAI that incorporates hierarchical FL for precision horticulture. They deployed a tomato monitoring prototype of their architecture at two sites and simulated two farms per site.

2.9.3 UAV Imagery-based Yield Prediction

Note that some of the literature reviewed in this section made use of corn growth stages to report their findings. We refer readers to Sections 4.3.3 and 4.4.1.2 for more details on corn growth stages.

Uno et al. [266] compared the prediction performance of MLPs and stepwise multiple LR models to baseline VI-based models. They found that the MLP and stepwise multiple LR model performances were superior to that of the VI-based approaches.

The following yield prediction works involved the optimal time to acquire imagery and the optimal VIs (or TIs) to use as features to maximize yield prediction model performance. Sunoj et al. [251] found that NDVI and EVI2 are the best VIs during the R3 and R4 growth stages, producing the most accurate results. Barzin et al. [21] used ML models that included multi-VI features and found that a) the Simplified Canopy Chlorophyll Content

Index (SCCCI), where SCCCI = $NDVI_{RedEdge}$/NDVI, was one of the best VIs for predicting yield at various growth stages, and b) the V10 and VT growth stages were the best growth stages for model performance. Saravia et al. [232] compared model performance when using multi-VI feature datasets vs. single-VI feature datasets. They found a high correlation between multiple VIs and yield during the reproductive growth stage of corn. Oglesby et al. [191] found that the period between VT and R1 was the best time to acquire imagery, and that the SCCCI VI was generally the best performing VI for those growth stages. Ramos et al. [215] found that the RF model was the better performing model, and that the NDVI, $NDVI_{RedEdge}$, and $NDVI_{Green}$ were the top ranked VIs for yield prediction performance. Zhang et al. [297] found that imagery (specifically, using the ExG VI) from crops closer to maturity led to better yield prediction model performance. Yang et al. [282] found that the use of multi-temporal features led to better performance compared to using only mono-temporal features. They found the R3 growth stage to be the best stage for the mono-temporal models, while combining imagery from VT, R1, R3, and R4 for the multi-temporal models, produced the best results. $NDVI_{Green}$ and $NDVI_{RedEdge}$ were found to be the best VIs for the R3 stage. Guo et al. [93] searched for optimal indices using stepwise regression models and fed the optimal indices to more complex ML models. They found that the RF model performed the best for yield prediction, and that the ML models generally performed better than the stepwise regression model. Chatterjee et al. [38] used temporally accumulated VIs as features fed into ML models and found that a) normalized-difference type VIs produced the best model performance and b) the flowering growth stage was the best time to acquire imagery.

Some yield prediction studies explored the prediction power of different types and combinations of features. Serele et al. [236] varied the types of features fed into a model, namely a) only VIs, b) only TIs, c) both VIs and TIs, and d) VIs, TIs, and topography features. They found neural network models generally performed better than the baseline multiple LR models. Fathipoor et al. [69] found that plant height was the most important feature for yield prediction, and when combined with VI features, slight model performance improvements could be achieved. Dilmurat et al. [61] found that combining VIs to Light Detection and Ranging (LiDAR)-based texture features improved yield prediction performance compared to using VIs and LiDAR texture features alone, although alone, these features still produced reasonable results. García-Martínez et al. [76] fed VIs, and imagery-derived canopy cover and plant density features, into a neural network model, and found that plant density and NDVI were the most important features for yield prediction. Baio et al. [18] varied the following features input into various ML models: a) irrigation management, b) irrigation management and imagery, and c) irrigation management, imagery, and temperature. They found the RF model to be the best for yield prediction, particularly for the b) and c) feature sets. Sapkota et al. [231] found that a UAV-based RGB VI, GRVI (which is referred to as VIg in the present work), led to better yield prediction performance than satellite imagery-derived NDVI (a MS VI).

Deep learning models were also used in the yield prediction literature. Baghdasaryan et al. [17] compared ML models with hand-crafted features (e.g., VIs) to deep learning models with automated feature extraction. They found that the deep learning models outperformed the ML models. Yang et al. [286] compared the prediction performance of

a) 2D CNN (which models spatial patterns), b) 1D CNN (which models spectral patterns), and c) 2-stream CNN (1D spectral CNN + 2D spatial CNN). They found the 2-stream CNN performed best. Bisht et al. [29] trained two types of CNN models using raw RGB imagery, namely a 2D spatial CNN and 3D spatiotemporal CNN. They found that the 3D CNN performed best. Kumar et al. [141] found that the SVM and k-nearest neighbours (KNN) models were the best performing models, while their MLP model generally performed worse due to the limited number of samples. Danilevicz et al. [53] combined multiple data sources into a learning task by using a multimodal deep learning model. They found that NDVI and $\text{NDVI}_{\text{Green}}$ were the most important VIs for model performance.

There are studies that, in addition to yield prediction, also predicted other crop characteristics. Khanal et al. [127] accessed the impact of field traffic induced soil compaction on yield by analyzing the yield precision maps generated by various ML models. Adak et al. [3] performed a flowering time prediction study using various regression models, various VIs, and imagery-derived plant height. They found that ridge regression was the better performing model for yield prediction. Khanal et al. [126] performed a soil property prediction study and evaluated multiple ML models using imagery, topography, and soil features. They found the RF model was the best model for yield prediction. Vong et al. [272] varied corn seed planting depth to analyze corn emergence spatial variability. The best performance was achieved using multi-temporal and multi-VI features. Herrmann et al. [103] attempted to identify the crop's growth stage using imagery. They found that a) imagery from the R2 stage produced the best prediction results, b) PLSR models led to the generally better performance compared to VI-based models, and c) red-edge-based VIs had the best performance. Guo et al. [92] performed a chlorophyll content estimation study and proposed a new VI called modified red blue VI (MRBVI). They found that their proposed VI relatively outperformed the other VIs and that the SVM model was the best model for yield prediction. Fan et al. [67] performed a flowering time prediction study and found a) multi-temporal imagery features led to better model performance than mono-temporal imagery features, b) ridge regression was the better performing model, and c) VT was the growth stage that produced the best yield prediction results.

Note that an interesting trend in the literature reviewed in this section can be seen by observing Table B.2 in Appendix B; that is, even though applying standard KF-CV for evaluating UAV imagery-based yield prediction model performance will likely lead to optimistic model performance, most of the studies still only applied KF-CV. Only one paper (Baghdasaryan et al. [17]) out of the 28 papers reviewed clearly performed spatial CV to evaluate model spatial generalizability. Since yield data and imagery from a field will likely be spatially autocorrelated [100], it means that the models evaluated in most of these works will likely have a) over-optimistic performance [221, 23], b) overfitted, and c) underestimated prediction errors [225]. Nevertheless, by expanding the scope of this review beyond only UAV imagery-based studies, there are imagery-based yield prediction studies that did consider the spatial structure of the data and/or model spatial generalizability evaluation: Barbosa et al. [20, 19], Danilevicz et al. [53], Davis et al. [54], Bose et al. [31], Maimaitijiang et al. [161], Qiao et al. [212], Yang et al. [284], Filippi et al. [71], and You et al. [291]. There are also those that considered the temporal structure of the data and/or model temporal generalizability evaluation: You et al. [291], Filippi et al. [71],

Yang et al. [284], Qiao et al. [212], Canata et al. [37], Bose et al. [31], Davis et al. [54], and Nevavuori et al. [184].

2.9.4 Nitrous Oxide (N_2O) Emission Prediction

Some studies collected continuous emission datasets using automatic gas chambers and/or flux towers. Saha et al. [228] collected emission data using both static and automatic chambers. They moved the chambers every 10 to 15 days to avoid introducing bias created from small-scale spatial variability. They found that combining ML models with process-based models outperformed process-based models by training ML models using the output of process-based models. Another work that combined data-driven models and process-based models is Lin et al. [150]. The authors proposed an agriculture-informed neural network (AINN) architecture that combines RNN and DLEM. Many of the inputs to the DLEM were implicitly learned as latent features by training the hybrid model using the back propagation algorithm. Various neural network models were combined with DLEM, and their performance was evaluated when combined with and without DLEM. They found that generalization error could be reduced by using their AINN approach. Joshi et al. [116] studied the effects that cover crop (rye) have on N_2O and CO_2 emissions for corn crop. They compared multiple ML models and found that a) RF was the best model for predicting N_2O emissions and b) cover crop-C was the most important variable affecting RF for N_2O emission predictions. Liu et al. [155] proposed a knowledge-guided ML model that integrated biophysical and chemical processes that control N_2O emissions. They compared their knowledge-guided model to other models and found that their proposed model outperformed traditional ML and process-based models. Maier et al. [160] performed a GHG (CO_2, N_2O, CH_4) emission study that analyzed drivers that affect emissions. They addressed missing flux readings by performing gap-filling using RF and investigated the importance of features that drive emissions using RF feature ranking. They found emissions were higher earlier in the season than later in the season. Feigenwinter et al. [70] performed an N_2O mitigation experiment and investigated CH_4 and N_2O drivers and important emission factors. They performed gap-filling with RF to predict missing emission values. They computed temporal features from soil and weather data by using various lags (average within a time window). They found lagged precipitation, water filled pore space, vegetation height, and leaf area index were important drivers for N_2O emissions. Taki et al. [252] investigated the gap-filling ability of MLP for N_2O emission annual time series by comparing MLP to linear estimation for filling in missing gaps of N_2O emission data. They also varied the size of the missing data gaps. They found MLP was better than linear interpolation for gap-filling. Biagaignon et al. [28] performed an N_2O emission gap-filling study by comparing the performance of linear interpolation, MLP, and a combination of both models. They found that MLP was the better performing model, although combining linear interpolation and MLP led to better performance during low emission periods. Goodrich et al. [86] performed a gap-filling study to estimate annual N_2O emissions to showcase that, using a single flux tower, two management units could be monitored using eddy covariance. They used partial least squares decomposition for feature selection and derived temporal features using moving windows. Cowan et al. [48] performed an N_2O

emission dataset interpolation study by performing gap-filling on an emission dataset created by using a flux tower using eddy covariance. Time-since-fertilization was one of the features included in the general additive model they used for gap-filling.

Some studies gathered emission data using static chambers and laboratory equipment. Hamrani et al. [96] performed a CO_2 and N_2O emission prediction study. They evaluated a variety of classical regression, shallow learning, and deep learning models and found that LSTM produced the best results, while RF was the second-best model. Glenn et al. [83] investigated the effects of precision fertilizer management on N_2O emissions. They found that a) high yield areas led to less N_2O emissions, b) low yield zones led to more emissions, and c) soil N and soil moisture content were among the most important predictor variables that influenced cumulative N_2O emissions. Mehmandoost Kotlar et al. [171] investigated the prediction performance of six ML models for predicting N_2O and CO_2 emissions. They ranked predictor variables by importance for predicting the emissions. They found that the Cubist and SVM models were the better performing models. Bastos et al. [22] examined N_2O emissions using a mixed-effect analysis of variance model and a conditional inference tree to analyze the factors that drive emission spikes. Among the soil and weather data samples, they sampled NH_4^+ and NO_3^- a few times per growing season. They found key emission spikes took place at approximately 80 days after fertilizer application when soil temperature was above 15 °C and water-filled pore space was above 57 %. Jiang et al. [115] performed an N_2O and CH_4 emission study using ML ensemble models. They found that their ensemble model performed best for N_2O emission prediction, while for the basic ML models, gradient boosting regression was the best and RF was the worst.

Instead of gathering emission data, some studies relied on datasets from the literature and/or simulated datasets for N_2O emission prediction. Philibert et al. [204] performed classification and regression N_2O emission prediction experiments. The classification experiments involved high and low emissions classes. The important predictor variables were found to include amount of N applied, crop type, and pH. Yin et al. [289] performed a global-scale study on background N_2O emissions by using RF and datasets obtained from many dozens of different soil and climate databases. They found the most important variable linked to background N_2O emissions was pH. Tian et al. [255] performed a global-scale N_2O emission prediction study using a) spatially explicit direct N_2O emissions and background emissions, b) RF, and c) various datasets from the literature. They explored what region exceeded emission limits and studied the main factors that affected emissions. They found that most croplands exceeded emission limits, meaning mitigation strategies need to be put in place. Kolasa-Więcek et al. [139] used MLP to examine the effects of livestock on N_2O emissions. They found that the amount of fertilizer used was the most significant predictor variable affecting N_2O emission prediction. Pan et al. [198] performed a global-scale N_2O emission mapping study. They used the Esri ArcGIS software to create their global emission map. They compared the stochastic gradient boosting ML model to various process-based models and found that the stochastic gradient boosting model outperformed the process-based models. Villa-Vialaneix et al. [270] compared the N_2O emission performance of various ML models trained using process-based model output, and they studied nitrogen leaching. They found splines approaches were the best models when there was a limited amount of data and that RF and SVM were the best models when more data was

available. They found that pH and soil organic content were the most important emission drivers. Giltrap et al. [81] also trained ML models using process-based model output. They simulated N_2O emission readings using the New Zealand DeNitrification-DeComposition (NZ-DNDC) process-based model and trained a LR meta-model using NZ-DNDC's output. They used the meta-model to create national emission maps and they considered the spatial correlation of residuals in their maps. They left spatial generalization evaluation as future work. Zhuang et al. [302] performed a global-scale N_2O emission study using MLP, and provided MLP prediction uncertainty bounds. They found that a large source of emissions was from warm and moist tropical soils. They also found that emissions tended to be most sensitive to precipitation. Benghzial et al. [24] performed a study on predicting direct and indirect N_2O emissions using ML models. They compared expectile regression and kernel expectile regression models to various ML models. They found the kernel expectile regression estimator produced the best results. Jiang et al. [114] performed a global-scale study to investigate the effects of using controlled release urea fertilizer on N loss (including N_2O emissions). They compared RF and MLP and found that RF did better than MLP at predicting N loss. Wu et al. [280] performed a GHG prediction study using datasets obtained from the literature. They predicted CH_4 and N_2O emissions using various ML and deep learning models using emission data from Chinese rice fields. They found Extreme Gradient Boosting was the best model.

Chapter 3

IoT-based Smart Farming Architecture Using Privacy-aware Distributed Machine Learning

Most of the content of this chapter was published in Killeen et al. [136].

In this chapter we propose an IoT-based smart farming architecture that makes use of privacy-aware distributed ML for enabling affordable PA applications. The architecture was proposed simply a means to an end for showcasing our distributed ML methodology. This thesis adapts our previous work's three-tier fleet management IoT architecture [133] to the smart farming setting for making PA predictions using IoT sensor data. The proposed architecture/SFS is separated into 3 layers that interact with the edge, fog, and cloud , namely, the perception, middleware, and application layer. The proposed SFS follows a two-tier architecture design involving two primary nodes, the PN and the root node (RN). Figure 3.1 illustrates a diagram of the proposed architecture. The predictive analytics ran in the architecture is achieved using distributed ML, specifically, using a) distributed EL combined with model sharing and b) FL.

The purpose of the architecture is to enable high spatiotemporal resolution model-based predictions of an "expensive-to-gather" variable of interest. Note that we refer to this variable of interest as the *target agri-variable* in this chapter. This target agri-variable could be useful to farmers but may otherwise be too labor intensive or expensive (due to high costs of specialized equipment) to expect farmers to gather using standard sensing techniques. By making use of the proposed IoT architecture, we propose a privacy-aware predictive analytics methodology that involves a) gathering agri-data via low-cost sensors and target agri-variable ground truth data via expensive equipment, b) transmitting and processing the data at the edge of the network, and c) training a collection of local ML models (via distributed ML), which are each deployed at different farm sites, without sharing private data. Only model updates are shared. Once the models are trained, they can offer farmers (including new farmers that join the system) with PA model target agri-variable predictions while a) somewhat respecting the farmer's privacy interests (although model updates may still reveal sensitive information [195, 63, 117]) and b) avoiding the

high costs of gathering target agri-variable datasets. The farmers would only need access to low-cost sensors and hardware to run the on-site PA models. At the expense of lower target agri-variable accuracy, the proposed predictive analytics methodology and the supporting IoT architecture enable affordability and somewhat enable privacy, which should encourage farmers to adopt the system. This should lead to improved model performance from the increased amount of available training data.

There are a few limitations with the proposed architecture and distributed ML methodology, namely a) our proposed architecture and methodology do not provide any formal cybersecurity benefits other than slight improvements to baseline privacy by avoiding sharing raw data; b) the proposed architecture was not formally or empirically evaluated; it was proposed simply as a means to an end for showcasing our distributed ML methodology for spatiotemporal IoT sensor data analysis experiments described later in this thesis; c) wireless network connectivity is assumed to be available, but in practice wireless network coverage is a known challenge on farm land, especially for large farms [268]; and d) the only FL aggregator algorithm showcased as part of our distributed ML methodology was FedAvg, which we chose for its simplicity as a proof of concept of our proposed privacy-aware distributed ML methodology, but FedAvg assumes IID data [63]. Agri-data tend to be heterogenous (non-IID) by nature, so FedAvg was not appropriate for this setting. More appropriate FL aggregator algorithm choices are discussed in Section 2.3.4.3.

The remainder of the chapter is organized as follows: Section 3.1 discusses the role of the three IoT layers involved with the architecture; Section 3.2 discusses the hardware and software components of the architecture; Section 3.3 discusses the data flow and the applied distributed ML methodology; Section 3.4 discusses the potential applications of the proposed SFS; Section 3.5 discusses the threat model considerations; and Section 3.6 discusses how our proposed SFS meets smart farming requirements.

3.1 Architecture Layers

The perception, middleware, and application layers are described in this section. The roles of these layers remain relatively unchanged compared to the architecture proposed in our old predictive maintenance work [133]. Below is a summary of the layers of our architecture, which includes a discussion of how we adapted the IoT layers mentioned in Section 2.1 to each layer of our architecture.

- The **perception layer** abstracts the edge. It combines the following IoT layers: the sensing layer and a lightweight version of the network layer. It performs sensing, actuation, and lightweight storage and networking. It provides the interface to the in-field resource-constrained nodes and virtually covers a farm's fields. What we did change from our old work [133] is that, for increased modularity, we removed the intelligence from the perception layer to make the layer's role more focused toward data acquisition and sensor network communication.

- The **middleware layer** combines the following IoT layers: the network, middleware, and learning layers. It abstracts the fog and the cloud, and generally performs more heavy-duty storage, networking, and ML in the cloud. Lightweight versions of these operations are performed in the fog. It performs resource optimization functionality, including power management and network optimization. It provides a high-level interface to the data streams and actuators of perception nodes. Furthermore, as recommended by Raza et al. [219], it also supports multiple communication protocols to enable heterogeneous perception nodes interoperability.
- The **application layer** combines the following IoT layers: the middleware and application layers. It provides the interface to the SFS by providing lower-layer information access and control, and this layer also hosts PA services.

3.2 Smart Farming System Components

In this section we describe the components of the proposed architecture. The proposed architecture is composed of multiple components. The main two components are the PN and the RN.

3.2.1 Predictor Node

The PN is a single logical node/module that is physically composed of many hardware sub-modules, namely, sensors, SNs, sensor networks, and a gateway. The PN combines technology from both the fog and the edge, and hosts the logic to perform local target agri-variable prediction via a data-driven model and proximal sensor data (e.g., soil data) ingestion. Providing power to the PN can be done by a) a battery (and/or solar panels) when the PN is deployed in a field and b) a more reliable power supply when deployed in (or nearby) a farmer's house. The PN is in both the perception and middleware layer. It is in the perception layer, since it contains sensors and sensor networks used to monitor its environment, and it is in the middleware layer, since its gateway performs lightweight analytics, processing, and storage. Depending on the use case, the PN may also be part of the application layer if local application services need to be hosted on its gateway (e.g., on-site orthomosaic generation). The architecture supports the ability to deploy multiple PNs per farm to cover a farm's regions of interest. For design simplicity, the predictive analytics methodology discussed later in this chapter assumes a farm only has a single PN. We proceed to discuss the PN's hardware sub-modules.

- **Sensors** are edge nodes that are the closest nodes to the field's data sources. Sensors are managed by a device (typically a microcontroller) called a SN.
- **Sensor nodes** are found in the edge. SNs are the devices responsible of making sensor data accessible to the gateway via a wired (e.g., RS485) or wireless (e.g., Bluetooth or Zigbee) connection. Instead of having sensors be directly connected to the gateway, we defined SNs as an intermediary node between sensors and the

gateway, because it is common for PA sensing devices to manage a collection of sensor probes. For example, soil nutrient sensing devices typically offer NPK readings. Furthermore, this design is flexible enough to treat SNs as sensor networks and allow for more complex devices to act as a sensing device (e.g., a UAV). SNs may be powered by a) the PN's gateway, when a wired SN-to-gateway connection is available, or b) their own power supply (typically via battery), when only a wireless SN-to-gateway connection is available.

- The **gateway** is a fog node that is the bridge between the edge and the cloud. Gateways are commonly used in smart farming architectures to connect data source nodes to the cloud. This is done by having the gateway support both short-range and long-range wireless protocols [46], which also improves energy efficiency [242]. The gateway reads sensor data from a collection of SNs, stores sensor data locally, runs local ML logic (including distributed ML client logic) using the sensor data, and communicates with the cloud via a long-range wireless communication protocol (e.g., cellular or LoRa) or a direct wide-area-network connection; where low-power wide-area-network (LPWAN) protocols should generally be favored as the wireless protocol of choice for cost and energy efficiency reasons [15]. In some architecture designs, gateways may be implemented using mobile edge computing devices (e.g., UAVs or a farm vehicle). In this setting, the gateways are treated as mobile devices that only exist temporarily in an area of interest to offer wireless connectivity to SNs [264].

3.2.2 Root Node

The RN is a cloud node responsible for managing the entire SFS. It is found in both the middleware and application layer. It is in the middleware layer, since it offers cloud-based ML (including the FL coordinator and EL model sharing logic), storage, and processing; it is found in the application layer, since it hosts application programmable interfaces (APIs) and can offer target agri-variable data ingestion services. It is the SFS's central access point, enables centralized FL and distributed EL, provides the interface for agri-service applications, connects geographically distributed farm nodes to the system, and enables multi-farm administration via an admin server (e.g., MeshCentral). The RN hosts a MQTT broker to enable efficient and lightweight client-side PN-to-RN communication [133]. We chose MQTT as the application protocol of choice, since a) it has been found to be one of the most appropriate protocols for smart farming [82] and b) it is lightweight on the client-side [133], although other application protocols may also have been appropriate choices, which we discuss about next in Section 3.2.3.

3.2.3 Application Protocols

The application network protocol should be chosen based on the use case. In the smart farming setting, most in-field agriculture sensors will have relatively low temporal resolution compared to traditional real-time IoT system requirements, although some applica-

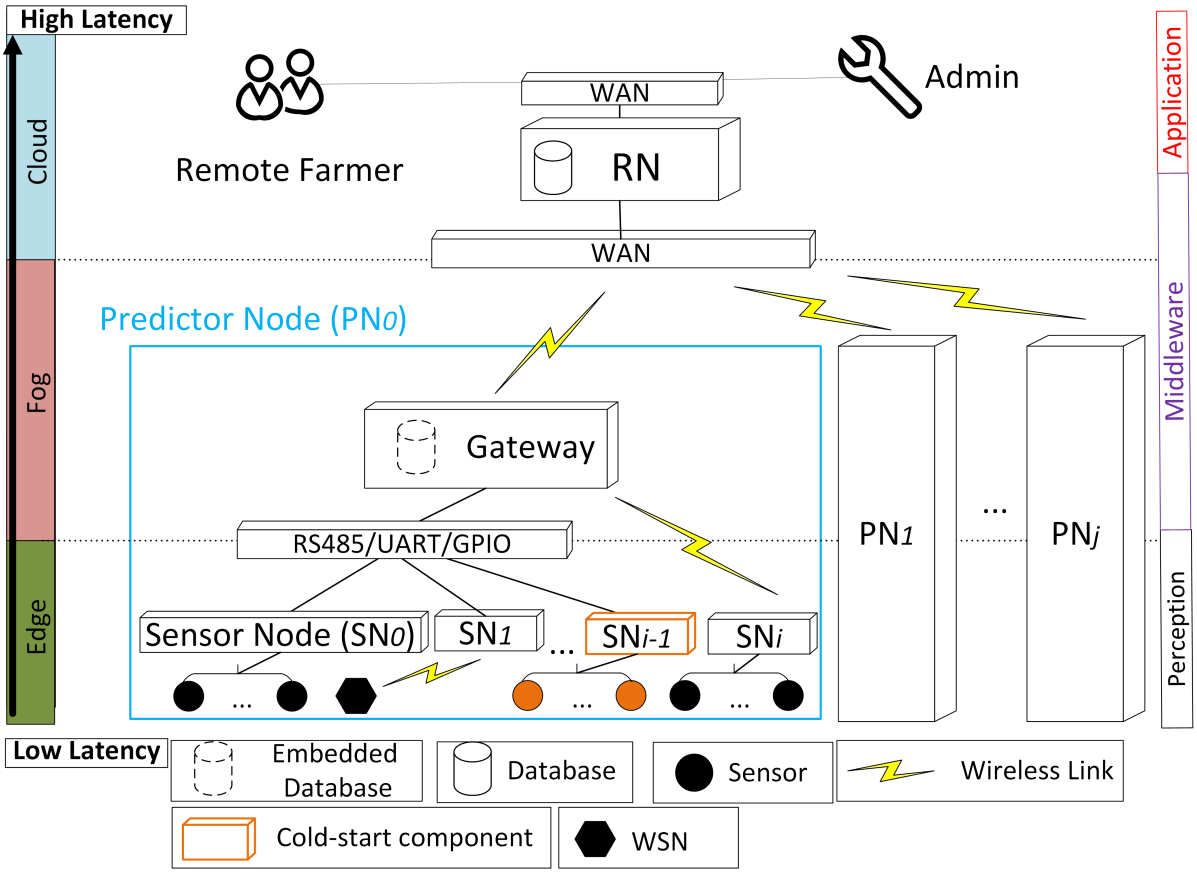


Figure 3.1: Overview diagram of the proposed privacy-aware smart farming IoT system architecture, where WAN = wide-area-network, RN = root node, UART = universal asynchronous receiver / transmitter, GPIO = general purpose input/output, and WSN = wireless sensor network. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed.

tions (e.g., imagery-based applications like precision map generation using UAV imagery) do have higher data generation rates. Application protocols commonly used in IoT include MQTT, Data Distribution Service, Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol, Hypertext Transfer Protocol [182, 183, 68], Extensible Messaging and Presence Protocol, and the WebSocket protocol [82]. Glaroudis et al. [82] provide a comprehensive review on a comparison of the different application layer protocols commonly used in IoT when applied to smart farming. The choice of the application layer protocol may be different at different layers of IoT systems [82]. We chose MQTT for simplicity reasons and given that our architecture was proposed only as a means to an end to showcase our proposed distributed ML methodology. Nevertheless, the choice of application protocol is important, since it will affect things such as reliability, energy consumption, and latency. Glaroudis et al. [82] found the best protocols for various key indicators to be the following:

- **Latency Over Local Area Networks:** CoAP

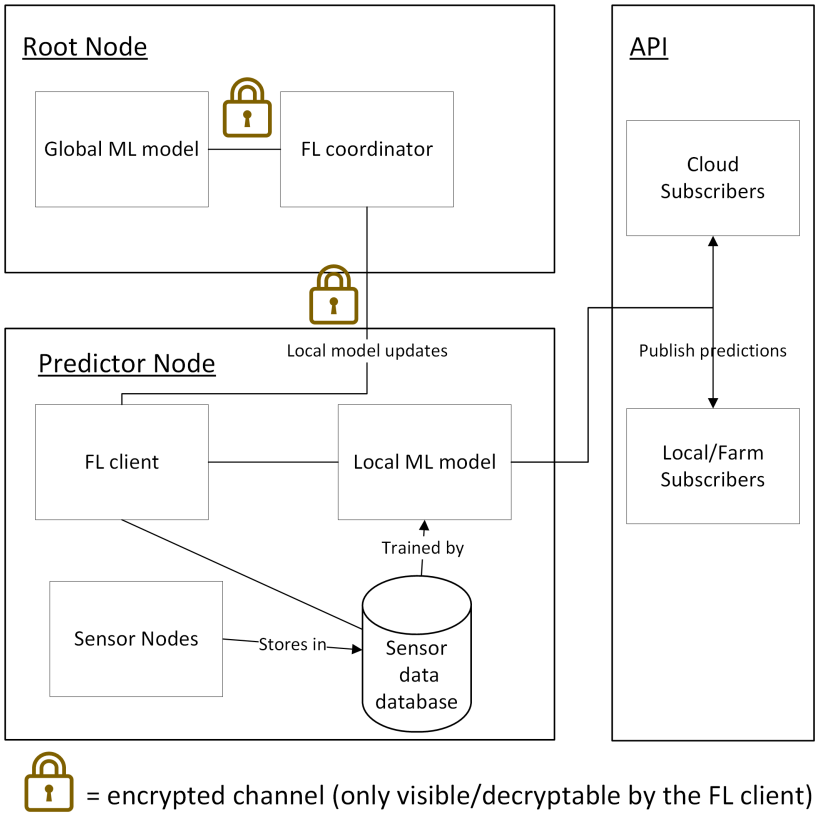


Figure 3.2: Privacy-aware predictive analytics smart farming system software module diagram when using federated learning (FL) as the distributed machine learning (ML) approach of choice, where we illustrate where encryption could be applied for solutions that use privacy preserving technologies; API = Application Programming Interface.

- **Latency Over Mobile Networks:** MQTT (with the quality of service value set to 0)
- **Bandwidth Consumption:** CoAP
- **Throughput:** MQTT
- **Reliability:** MQTT
- **Energy Consumption:** CoAP

3.2.4 Remote Administration

The architecture also supports remote administration. Since a SFS can be large, it becomes infeasible to manage all the nodes in the network manually, which is why a software administration tool is required. Each PN node has a software administration agent/client, allowing administrators to remotely update and configure the nodes' software, via an administration server installed on the RN. With this remote administration system in place, all other node access interfaces can be disabled (for example, secure shell or serial shells),

and as a result the only way to configure the SFS nodes is via the administration server. Note that redundancy and crash recovery are outside the scope of this thesis, but they should be considered; if the administration server (or RN, for that matter) fails, access to the SFS would be lost [131].

3.3 Predictive Analytics

This section discusses the proposed privacy-aware predictive analytics methodology that is enabled by the proposed IoT architecture. Our methodology involves the use of privacy-aware distributed ML for replacing expensive specialized sensing equipment with an affordable PA model-enabled PN. To deploy the proposed system, expensive ground truth data is first required to train the models. In this initial state, the proposed SFS is in the *cold-start state*. When in the cold-start state, the global model needs to be trained using sensor data, where the predictor variables are derived from low-cost sensors and the target agri-variable is derived from expensive (in terms of labor and/or monetary cost) sensing equipment. Once the global model is trained and deployed, the SFS will provide farmers (both new farmers and farmers that were involved with model training) with access to affordable target agri-variable estimates without requiring farmers to have access to expensive sensing equipment. We will refer to this expensive equipment as cold-start components/equipment in the remainder of this chapter. Farmers will only require affordable sensors along with the PN hardware to benefit from the system's global model predictions. In this post-training state, the SFS is in the *live state*; that is, when cold-start components are no longer required by the system. In the live state, the PN can be treated as a soft-sensor/smart-sensor. This design sacrifices target agri-variable accuracy for decreased monetary costs. This affordability also makes the gathering of higher spatiotemporal resolution target agri-variable datasets (via PN predictions) more feasible. Furthermore, both the affordability of the system and the fact raw data is never shared will encourage higher SFS adoption rates, which will lead to more available training data from the increased user-base, and as result, increased model performance accuracy can be expected from the increased amounts of training data.

We continue the discussion by providing three examples of target agri-variables in Examples 3.3.1 , 3.3.2, and 3.3.3. We provide these examples to help explain the notion of a target agri-variable introduced earlier in this chapter. In these examples we explain why certain types of agri-data are difficult or expensive to obtain, and why predicting the target agri-variable would be more feasible than sensing it.

Example 3.3.1. This example target agri-variable involves gathering non-destructive, on-demand, and high spatial resolution crop yield datasets. Although expensive, high spatial resolution yield data acquisition could be done using harvester-mounted yield monitoring equipment, but this sampling approach is destructive by nature, which practically limits data acquisition to only once per growing season [126]. Theoretically, this solution could meet the on-demand requirement (e.g., via a self-driving harvester), but this solution requires sending machinery to the field, which would negatively impact crops and soil [127].

Instead, yield could be predicted using UAV imagery. This would enable high spatial resolution, non-destructive, and on-demand yield data acquisition.

Example 3.3.2. This example target agri-variable involves gathering high spatiotemporal resolution N_2O emissions in an affordable manner in terms of both labor and sensing equipment costs. High spatial resolution emission monitoring could be achieved using many relatively inexpensive static gas chambers, but the associated labor costs render this solution less feasible, especially when high temporal resolution data is desired. High temporal resolution emission monitoring could be achieved using a single automated gas chamber, a less labor-intensive solution, but this type of sensing solution is expensive, especially if high spatial resolution data is desired. Instead, we could predict high spatiotemporal resolution emissions using data from many inexpensive soil and weather sensors (achieving high temporal resolution) that are deployed in a spatially dense area (achieving high spatial resolution).

Example 3.3.3. This example target agri-variable involves gathering non-intrusive high spatial resolution soil moisture data. We use the methodology applied by Vasisht et al. [268] as a case study for this example. The authors acquired UAV imagery and soil moisture readings from 16 soil moisture sensors deployed in a 2-acre field. Their goal was to generate soil moisture precision maps using UAV imagery and data-driven models to remove the need to deploy spatially dense soil moisture sensors. The data acquisition difficulty in this example is that it is not feasible to expect farmers to deploy spatially dense soil sensors in their field, since the sensors would get in the way of field management activities. By using UAV imagery to predict moisture, non-intrusive high spatial resolution soil moisture monitoring was achieved.

3.3.1 Distributed Machine Learning Methodology

In this section we explain the proposed predictive analytics methodology that makes use of distributed ML. The distributed ML approaches we chose in our methodology are FL and distributed EL via model sharing (which we will refer to as EL for simplicity in the remainder of this chapter). The methodology involves PNs (or client devices) using their private data to collaborate for training a global model using either FL or EL. Note that when using EL, the global model is an ensemble of local models. Once trained, the global model is broadcast to each client and can be used to make local target agri-variable predictions, and these predictions can be ingested by interested parties (or subscribers) via an API.

We proceed to discuss the details of our methodology. Figure 3.2 illustrates the main software modules involved in the process, where we chose FL as the distributed ML approach of choice. In the case of EL, the global model would not be found on the RN; a global model would be created in each PN. Also, the FL coordinator would be a service that simply broadcasts received local models to each client. Furthermore, to better explain the workflow and data processing logic of our proposed methodology, we included pseudo code; Algorithm 2 provides the server-side (RN-side) logic and Algorithm 3 provides client-

side (PN-side) logic of our methodology, where we make use of the FedAvg algorithm (see Algorithm 1) as a proof of concept of our proposed FL logic.

Deploying the system means a cold-start scenario occurs such that labeled data is unavailable and a model has not been trained yet. In this situation, expensive (and/or difficult-to-obtain) labeled data must be collected from cold-start equipment to train the global model. All the PNs at this point will have inexpensive sensors and expensive cold-start sensing equipment. To gather the target agri-variable sensor data, the PN gateways will rely on the cold-start component's sensing capabilities. Each PN k will store its sensor data locally in an embedded database D_k. When first deployed (in the cold start scenario), the RN will wait for join-requests of new PNs that wish to join the system. Upon receiving a join-request, the RN will add the new PN to the SFS. The RN will periodically check how many total sensor samples are available across all clients by a) polling each client for the number of sensor samples n_k on client k and b) counting the total number of sensor samples n. When n reaches some user-defined threshold value σ, the system has enough decentralized data to begin the distributed ML training process by using EL or FL (a system administrator's choice).

- When EL is applied, a LL process is applied such that each client trains a local model using only the client's local data. Note that the local models need not be the same type of model shared across all clients, they could be a mix of ML and deep learning models. Once trained, a single round of communication occurs. The local models (model updates) are sent, from each PN, to the RN, and the RN then broadcasts each local model to every PN. The clients each build an ensemble model using these local models. For each PN, the resulting ensemble model on the PN is treated as the PN's global model, where the global model may be unique to each client (this depends on the choice of EL method).
- When FL is applied, it has the following properties: a) it uses a horizontal FL data partitioning scheme, since the clients share the same type of data sources, and b) a cross-device FL scheme, since the client devices are expected to be resource-constrained in-field devices. For purposes of illustration, we use the FedAvg (Algorithm 1) algorithm to train the global model, where the FL clients are the software agents running on the PNs. In practice, any FL aggregator algorithm could be used, but we chose FedAvg for its simplicity. The training process begins by having the RN initialize a global model, using a pre-defined set of hyperparameters, and sends this global model to each client. This step ensures that each client's model has the same initial random weights and hyperparameters as every other client. The training process continues by running multiple rounds of training, such that for each round, the PNs with a sufficient amount of battery life are selected. The selected clients then update their local version of the global model by performing local training using their local data D_k. Once the round of training ends, the clients send their model weight updates to the RN. The RN aggregates the weights via weighted-averaging to update the global model. The RN then sends the updated global model to the PNs. These steps are repeated until model convergence occurs or until some other stopping criteria is met, at which point the global prediction model has been trained

and is sent to all the PNs.

Our methodology is not limited to using only FedAvg. We could use other FL aggregator algorithms that are more appropriate for non-IID agri-data (see Section 2.3.4.3 for examples of such algorithms). Modifying our proposed methodology would simply involve replacing the 'G = `Algorithm 1.startFederatedTraining(`ϕ`)`' line of pseudo code in the '`startDistributedTraining`' function of Algorithm 2 with a function call to the FL aggregator algorithm of choice (e.g., 'G = `FedProx(`ϕ`)`').

Once the distributed ML technique of choice completes the cold-start training phase, the SFS enters the live state. The PNs will download the global model upon connecting to the proposed SFS or will already have the global model if the PN was part of the cold-start training process. The target agri-variable can then be predicted by the global model deployed on each PN. A farmer can simply purchase (or rent) a PN to join the SFS to obtain non-intrusive and inexpensive target agri-variable estimates. Furthermore, online testing [111, p. 302] and on-going model validation should continue to be performed to prevent a model from going stale [124, pp. 447–456]. Although a thorough analysis of these concepts is outside the scope of this thesis, we present a brief summary of techniques to monitor a ML system after deployment. Online testing can be done by monitoring the response of users/clients, performing runtime monitoring, and tracking key performance indicators [111, p. 302]. Monitoring model performance can be done for concept drift detection, but this requires labeled data, meaning the cold-start components would need to remain in the system. In cases where cold-start components become inaccessible in the live state, concept drift may be detected by monitoring model prediction distributions and/or feature distributions. When concept drift is detected, a model should be re-trained and re-deployed [124, pp. 447–456], and smart farming services can remain uninterrupted by keeping the current models unchanged while training the new up-to-date models and having clients periodically download the most recent version of the global model [209].

Takeaway. We proceed to summarize our proposed SFS. Sensor data is never sent to the cloud and the farmers can still benefit from each other's data by collectively training a global model using distributed ML. The farmers can avoid investing into expensive cold-start sensing equipment, if lower accuracy of the target agri-variable monitoring is tolerable by the farmers. The only data that may be potentially shared are the global model predictions, which depends on how the subscriber services and users interact with the prediction API.

Algorithm 2 `Server-side Smart Farming Prediction System.`

Data: ϕ = distributed learning system hyperparameters, S = system state, K = number of connected clients, G = model(s) trained via distributed ML, G_k the local model of client k, n_k = number of sensor samples on client k, and σ = total number of sensor data samples required before training begins.

Result: The server-side RN logic that coordinates distributed ML using PNs' private sensor data.

S = `"cold start"`, $K = 0$, G = `null`

Function `startPredictionSystem()`:
- **when** a client k connects, call `onClientJoinRequest`(k)
- **while** S = `"cold start"` **do**
  - **if** $\sum_{k=1}^{K} n_k > \sigma$ **then**
    - S = `"training" // stop the data collection of each client`
    - G = `startDistributedTraining`(ϕ) `// clients perform distributed ML`
    - S = `"live" // ready to make target agri-variable predictions`
    - **for each** client $k \in K$ **in parallel do**
      - `Algorithm 3.deploy`(k,G) `// trained model(s) broadcast`
    - **end**
  - **end**
- **end**

Function `onClientJoinRequest`(k):
- $K = K + 1$ `// new client connected and joins system`
- **if** S = `"cold start"` **then**
  - `Algorithm 3.runClientDataCollection`(k) `// start client` k `local data collection`
- **else**
  - `Algorithm 3.deploy`(k,G) `// client` k `is sent the trained model(s)`
- **end**

Function `startDistributedTraining`(ϕ):
- **if** doing federated learning? **then**
  - G = `Algorithm 1.startFederatedTraining`(ϕ) `// train the global model using FL and FedAvg`
- **else**
  - **for each** client $k \in K$ **in parallel do**
    - G_k = `Algorithm 3.startLocalLearning`(k,ϕ) `// train local models on each client (for EL)`
  - **end**
- **end**
- **return** G

Algorithm 3 `Client-side Smart Farming Prediction System`.

Data: k = client id, G = model(s) trained via distributed ML, G_k is the local model of client k, S = system state controlled by server, s_k = sensor samples collected from a single sampling period from client k, D_k is the sensor reading dataset stored on client k, and c_k = cold-start equipment sensor samples collected by client k.

Result: The client-side PN logic that collects sensor data, trains local models, and sends model updates and target agri-variable predictions to subscribers.

Function `deploy(`k`,`G`):`
- **if** G **is** a set of local models? **then**
  - G = `buildModelEnsemble(`k`,`G_1`,`G_2`,...,`G_k`)` `// create model ensemble using each client's local model`
- **end**
- **while** S = `"live"` **do**
  - s_k = `pollSensorData(`k`)` `// read local sensor data`
  - p = G`.predict(`s_k`)` `// predict target agri-variable using sensor data`
  - publish target agri-variable predictions p to subscribers
- **end**

Function `runClientDataCollection(`k`):`
- **while** S = `"cold start"` **do**
  - s_k = `pollSensorData(`S`)` `// read sensor data`
  - store s_k in client k's local database D_k
- **end**

Function `pollSensorData(`k`):`
- s_k = read data from local sensor nodes `// inexpensive sensor data read`
- **if** S = `"cold start"` **then**
  - c_k = read target agri-variable from cold-start sensor node `// expensive sensor data read`
  - s_k = $s_k \cup c_k$`// merge target agri-variable and sensor samples`
- **end**
- **return** s_k

Function `startLocalLearning(`k`,`ϕ`):`
- G_k = `model.fit(`D_k`,`ϕ`)` `// train the local model`
- **return** G_k

3.4 Applications

Various applications could be enabled by applying the proposed SFS, including on-demand high spatiotemporal resolution yield prediction and N_2O emission prediction. Details on how farmers could benefit from our SFS and make use of yield and N_2O predictions are discussed in Sections 4.2 and 5.2, respectively. By applying our system for agriculture

applications, global food security and environmental sustainability could be improved by allowing farmers to optimize resource usage (e.g., fertilizer and pesticides) and reduce input costs. Since the system is affordable once deployed, farmers that own small scale farms could access such technologies. Furthermore, the proposed system is not limited to only PA applications. It is general enough to be applied to other use cases as well. In general, it could be used in any setting where an expensive sensor can be replaced by many more affordable (and probably less accurate) sensors combined with models to offer higher spatiotemporal resolution sensing at the cost of sensing accuracy; for example, like in air pollution monitoring.

3.5 Threat Model

Although our methodology improves the baseline privacy of the system by avoiding sharing raw data and instead sharing only model updates, our proposed SFS does not provide any formal cybersecurity benefits. Model updates (and local models) from distributed ML may still reveal sensitive information [195, 63, 117]. Privacy preserving technologies (some of which are described in Section 2.3.4.1) should be applied to better meet cybersecurity requirements [195, 6]. To limit the amount of data leakage that occurs from sharing model updates in our SFS, we could do the following:

- Apply differential privacy by adding noise to the client datasets before training or adding noise to the model updates [63], but this would come at the cost of reduced model accuracy [63, 285].
- Apply secure multi-party computations and run these computations in trusted execution environments to securely aggregate model updates on the FL coordination server, but this would come at the cost of increased computational complexity, which would make this solution less ideal for the setting where clients are run on resource-constrained PN devices [117] (e.g., like in the N_2O emission prediction use case).
- Apply homomorphic encryption to encrypt the model updates before sending them to the FL coordination server, and have the FL coordination server aggregate encrypted model updates (without decrypting them) to update the global model [63], but this would come at the cost of increased computational complexity [117].

A more detailed analysis of cybersecurity considerations for the proposed SFS is beyond the scope this thesis. Nevertheless, the cybersecurity of a system deployed to production should not be ignored, especially in the IoT setting. We refer readers that are interested in cybersecurity to these books [267, 249, 250, 168].

3.6 Meeting IoT-based Smart Farming Architecture Requirements

In this section, we discuss how our proposed architecture meets the requirements of an IoT-based smart farming architecture. IoT architecture system requirements are discussed by Killeen [131], and we identified SFS requirements by reviewing the literature [15, 46, 219, 68, 182].

We now discuss IoT requirements that are met by the proposed architecture. Scalability is achieved via the two-tier design of our proposed architecture and by using distributed ML. Instead of processing all the data in the cloud, we push computations to the edge of the networks, on PNs. Furthermore, the applied distributed ML methodology does not send raw data to the cloud, only model updates are sent to the cloud; this a) reduces network bandwidth usage and cloud processing for settings involving large datasets and b) addresses (only somewhat) privacy requirements by offering slight improvements over the baseline privacy of simply sharing raw data. Formal cybersecurity requirements are not met by the proposed SFS. The geographical distribution requirement is met by the two-tier hierarchical design of the proposed architecture. PNs can be deployed in a geographically distributed manner. Furthermore, if necessary, as the SFS grows, an administrator could add an additional computation layer between the RN and the PNs to partition the SFS into smaller geographical regions of PNs to reduce the work load on the RN, turning the proposed architecture into a three-tier IoT architecture similar to our previous work [133]. The edge/fog computing requirement is met by deploying sensors into a field (in the edge, although the proposed architecture is more of a fog solution than an edge solution) and installing a gateway either in a field or in the farmer's house (in the fog). This type of fog computing also meets the resource management requirement, since the fog is an ideal location for managing resources. For example, the network communication can be reduced by making sure only aggregates of sensor data are sent over the network. Furthermore, the gateway provides the ability to connect sensor networks together by supporting various heterogeneous protocols, which meets the requirement for connecting networks of sensors. Furthermore, the gateway also supports a broadband connection for PN-to-RN communication, meeting the multi-protocol support requirement. The data multi-tenancy requirement is met by having the RN manage the data of multiple different owners/tenants (farmers in this case) when applying FL via the RN's FL coordinator. To meet the efficient IoT communication requirement, the SNs and PNs can aggregate data before transmitting the data over the network. Furthermore, MQTT is chosen for the communication between PN and gateway, which also meets the efficient communication and resource management requirements. The cognitive computing requirement is met, since predictive analytics is performed in the fog and the cloud. The requirement for supporting a diverse set of distributed applications is met, since the architecture is distributed and can therefore support a variety of distributed applications.

We now discuss smart farming requirements that are met by the proposed architecture. The requirement for multidisciplinary technology support is met, because the proposed architecture supports many technologies of various disciplines. For example, in Sections 4.2

and 5.2, we showcase how the architecture can be adapted to the yield prediction and N_2O emission prediction settings, respectively. The low-latency PA application requirement is met by making predictions locally using local sensor data. The device heterogeneity support requirement is met by having a gateway that supports multiple network interfaces, and the architecture is general enough to support various hardware solutions to implement the gateway (e.g., an in-field Raspberry Pi or a desktop in a farmer's house could both act as a gateway) and SNs (e.g., a UAV or a NPK sensor could be a SN). The low-cost field nodes requirement is met by replacing expensive cold-start sensing equipment with an inexpensive computing node (the PN) that makes use of inexpensive off-the-shelf sensors and ML to estimate the expensive target agri-variable readings. The architecture supports the sensing over large areas requirement by supporting wireless SN-to-gateway communication, and this protocol could be over a wide area by using a LPWAN protocol (e.g., LoRa). This way, a single gateway could be deployed to manage many sensors deployed at long distances apart. The self-organization and management requirement is satisfied by using FL, where only PNs with sufficient battery life will be selected in the FL training rounds. We also only somewhat address common farmer privacy concerns by using FL and EL.

Furthermore, the architecture assumes that at least some form of wireless connectivity is available in the field. This is an arguably strong assumption, since one of the challenges of smart farming is network coverage. Nevertheless, we leave it to future work to a) address the case where network coverage is limited and b) explore the wireless protocols that enable smart farming. Interested readers are referred to a) Vasisht et al. [268] for a work that proposes a solution for limited network coverage on a farm and b) Farooq et al. [68] for a survey on smart farming and IoT that discusses communication protocols used in smart farming.

Chapter 4

Crop Yield Prediction

4.1 Introduction

Our population is fed mainly through the use of agriculture [8]. Climate change, loss of arable land, and declining fresh water availability are threatening the production of food [268]. By 2050 the world population is estimated to reach 9.73 billion, and by 2100 it is expected to reach 11.2 billion [46]. The increase of the world's population is expected to lead to more food consumption by 2050 [229]. This means farmers will need to produce more food while also conducting environmentally sustainable agricultural practices; currently the agriculture industry does not make use of sustainable practices [220].

A solution to these challenges to achieve better food security is by using PA and smart farming [10]. PA is the concept of analyzing agri-data and then taking action using the insights gained from observing the variability in the agri-data [224, 229]. Smart farming is a management technique that extends the idea of PA by taking advantage of various technologies to optimize agriculture input costs and resource usage via real-time information technology and predictive analytics. These technologies include a) cloud, fog, and edge computing, b) IoT, and c) ML and artificial intelligence [224, 66, 277]. One of the advantages of smart farming is business cost reductions [66]. A SFS should a) have clear costs and benefits to farmers [16], b) be straightforward to use, c) be affordable, and d) respect farmers' privacy concerns. A SFS that does not meet these requirements may suffer from reduced system adoption rates [144, 246, 152].

Yield prediction is one of the applications of smart farming. Data-driven models (e.g., ML) can be used to generate yield precision maps once the models are trained. One of the types of data that can be used to train these models is remote sensing imagery. It is important to note that agri-data (including crop imagery) should be treated as heterogeneous when planning agricultural management practices [229]. Imagery can be acquired via aircrafts, satellites, or UAVs [41]. A SFS designer may favor UAV imagery, since it offers higher spatiotemporal resolutions (and this benefits PA model performance [128]) than satellite imagery [296] and is non-intrusive [213]. The highly dynamic spatiotemporal nature of agricultural information and the many external environmental factors that can

affect yield must be taken into account by PA models [229]. SA will likely be present in a field's yield and imagery data [100], meaning overfitting and the underestimation of model prediction errors will likely occur [225]. Spatial CV can be used to address this [221], but this evaluation method tends to lead to pessimistic extrapolation performance results [230]. This idea is important, since using data from one farm to forecast yield for another farm tends to lead to erroneous results [112].

PA models can be trained by sending all the data from the network's edge to the cloud [193] and then performing CL. CL involves combining all the data sources into a single centralized dataset before training a model. This approach unfortunately suffers from drawbacks including: a) not being scalable [195], b) suffering from power and wireless network coverage limitations, which is a common limitation in agricultural fields [268], and c) having poor privacy, which may hinder SFS adoption rates [242, 119, 63]. Privacy is important, since low system adoption rates reduces the number of farmers that join SFSs. Limiting a SFS's farmer user-base would result in less available training data, and this could in turn negatively impact PA model performance. Fog computing can instead be applied to address the limitations of CL by running analytic processes at the edge of networks [285]. This enables LL. LL involves training many local models using only local data [195], but unfortunately, applying LL to agri-data may lead to loss of useful geographical trends that could have been useful to a centralized model [195, 229]. These limitations can be addressed by using model sharing combined with distributed EL [63, 199] and/or FL [293, 63, 271, 33], both of which are forms of distributed ML. In both cases, models are trained at the edge of the network (specifically, in the fog) on client devices, which reduces network communication and somewhat satisfies privacy requirements [285, 30, 63, 4], using only local data such that private data is never shared. Only model updates are shared. The result is a global PA model (in the case of EL, this would be an ensemble model created using a collection of local models) that can be used to benefit farmers for agriculture-related tasks by making use of model predictions [30, 63].

The remainder of the chapter is organized as follows: Section 4.2 adapts our smart farming IoT architecture proposed in Chapter 3 to the crop yield prediction setting; Section 4.3 describes the study site, the data sources, and some of the common methodologies applied by each of our UAV imagery-based yield prediction studies described in the remaining sections of this chapter; Section 4.4 describes a study that used CL and spatial CV; Section 4.5 describes a study that used CL, spatial CV, and hyperparameter tuning; and Section 4.6 describes a study that used distributed ML, hyperparameter tuning, feature selection, and spatial CV.

4.2 Adapted IoT Architecture

In this section we explain how we adapt our architecture proposed in Chapter 3 to the imagery-based yield prediction use case, how much deploying such a system would cost, how farmers can use and benefit from the system, and the adoption barriers of deploying such a system.

The adapted architecture can be seen in Figure 4.0.1. The components of the architecture that we adapted are a) the PN's internal components and b) the logical coverage of the three IoT layers. We left the RN as is, such that the RN still hosts the FL coordination server and still broadcasts local models to each client when using EL. We left the role of the PN's gateway unchanged. The gateway remains as a distributed ML client device that performs local predictive analytics. What we did change in the adapted architecture is as follows: the SNs are a) remote sensing platforms (e.g., UAV and satellite) equipped with cameras and b) crop harvester equipped with expensive yield monitoring equipment. Strictly speaking, even though the communication link between the satellites and the PN gateway would be presumably done via a satellite image cloud-service platform, without loss of generality, we treat the satellite as a SN from a logical point of view in terms of acting as a data source. We assume that the satellite image platform respects farmers' privacy concerns. For UAV imagery, given the volume of imagery data required to build an orthomosaic of a field, upon UAV mission completion, the adapted architecture assumes a reliable wireless network connection between the UAV and gateway is available (similar to Vasisht et al. [268]). A manual data dump (via SD card exchange) would also be sufficient for local data ingestion. Furthermore, the harvester and yield monitor are the cold-start equipment. The adapted architecture was designed with the idea that imagery and yield data never leave the farm-area-network. Furthermore, we defined the PN's gateway as a desktop machine deployed in a farmer's house instead of being an in-field resource-constrained device. This setup enables orthomosaic generation resulting from the stitching of UAVs' raw imagery. We were inspired by Vasisht et al. [268] in this design decision, since it should be reasonable to assume there is an available network connection and a power supply in a farmer's home. In terms of how we re-arranged the logical placement of the IoT layers in the adapted architecture, we did the following: we extended a) the application layer to include the PN's gateway, since we expect orthomosaic generation and yield precision map generation services to exist on the PN, and b) the middleware layer to include the SNs, since the SNs in this setting are complex sensing platforms with a lot of functionality. Note that the notion of having the gateway being part of the application layer is similar to the work done in Al Asif et al. [12] and Vasisht et al. [268].

4.2.1 System Deployment and Costs

In this section we discuss a) how a farmer would deploy the proposed SFS for predicting yield using UAV imagery and b) the associated costs.

- When the system is first deployed, a crop harvester equipped with a yield monitor would be needed to gather high spatial resolution crop yield datasets, where the estimated costs could range between $100,000 CAD to $500,000 CAD, depending on the condition, features, and age of the equipment. Instead, for lower initial costs, a harvester equipped with a yield monitor could be rented or acquired via contractors. In our studies, we acquired the equipment via contractors. This cost us $15 CAD per metric tonne of corn grain (for transporting the grain to a grain elevator, where the grain was dried, weighed, and recorded into final yield records). It also cost us

$65 CAD per acre of corn harvested. In our studies, we had access to a total of approximately 100 acres of corn fields, which cost us approximately $30,000 CAD (assuming an average yield of 2.3 tonnes per acre). Once the system goes live, this cost can be eliminated for farmers that join the system.

- When the system is first deployed and when the system goes live, an UAV equipped with at least one camera is required. In our studies, with the help of an UAV service company, a single UAV mission for gathering MS and RGB imagery covering approximately 100 acres of crops cost us approximately $1,500 CAD. In practice, we expect training accurate models would require imagery from multiple acquisition dates over multiple growing seasons. Therefore, assuming three image acquisition missions per year and three years of data are required for accurate model predictions, then the estimated costs for gathering imagery for training the models would be approximately $1,500 CAD $\times$ 3 $\times$ 3 = $13,500 CAD. Once the system goes live, imagery from multiple years will not be required to make a prediction. Therefore, assuming imagery from one to three image acquisition missions is required for making yield predictions, a single yield prediction map would cost approximately $1,500 CAD to $4,500 CAD. It is worth noting that for experiments not covered in this thesis, an UAV and MS camera were purchased for approximately $8,900 CAD.

- When the system is first deployed and when the system goes live, image stitching software would be required to build the orthomosaics from the UAV imagery. Assuming the PIX4Dmapper software is used, this software costs approximately $300 USD per month or $6000 USD for a one-time purchase. In our studies, the UAV service company stitched the imagery for us, and this was included in the $1,500 CAD per UAV mission fee. This software would be required for pre-processing the UAV imagery into an appropriate format before using imagery in further data pre-processing steps.

- When the system is first deployed and when the system goes live, a gateway will be needed to build the PN. The gateway would be a desktop machine with enough resources to support the PXI4Dmapper software. Table 4.0.1 provides suggested system requirements for a machine that PIX4Dmapper will be installed on, and the requirements depend on the number of images taken. For projects with more than 2000 raw images, PIX4D recommends using the PIX4Dmatic software instead of PIX4Dmapper [205]. In our studies, the smaller fields had approximately 150 to 500 images taken per mission, and the larger field had approximately 750 to 1650 images taken per mission. Therefore, we estimate the one-time cost for purchasing the gateway/desktop to be $1,500 CAD to $3,000 CAD.

4.2.2 Benefits of Using the System

To make use of our proposed SFS for yield prediction, yield prediction maps could be obtained via UAV imagery, and these yield precision maps could be ingested into a) automatic precision fertilizer or herbicide application systems and/or b) precision irrigation

Table 4.0.1: System requirements for using the PIX4Dmapper software (suggested by PIX4D [205], such that raw image resolutions are assumed to be 14 MP) for generating orthomosaics from raw UAV imagery.

Project Size	Number of Images	Suggested RAM (GB)	Suggested Free Disk Space (GB)
Small	<100	4	10
Medium	100 to 500	8	20
Large	500 to 1000	16	40
Very Large	1000 to 2000	32	80

systems. Low yield areas would be identified and UAVs could automatically fly to these areas and apply nutrients, pesticides, or water to address the problematic areas. The solution need not be automated. Addressing the low yield areas could simply involve manual intervention by the farmer. After addressing low yield areas, the farmer could then wait a couple weeks, send another UAV to produce yield predictions, and the farmer could verify if the low yield areas have been addressed. In other words, our proposed SFS could lead to direct increases in profit via increases in yield. Early season yield predictions could also enable early season crop pre-selling, such that earlier season business transactions could take place when expected yields would otherwise be usually too uncertain. This could help farmers get better deals when selling their crops and this could lead to more profit.

4.2.3 Adopting Barriers

Adoption barriers of the proposed SFS applied to yield prediction are a) the high cold-start equipment costs, b) the relatively high costs of UAV imaging solutions, and c) the high technical complexity of UAVs.

- High cold-start equipment costs could be addressed by reducing costs via subsidies. If purchasing or renting the harvester and yield monitoring equipment was subsidized by the government, the costs for having farmers join SFS in the cold-start state could be reduced.
- Instead of paying UAV service companies for gathering imagery each time a farmer wants to generate a yield prediction map, farmers could instead purchase an UAV and camera setup. This way, a farmer could generate many yield prediction maps without any service fees.
- UAVs may be too technically complex for certain farmers to be able to benefit from the SFS. The technical complexity and affordability of making yield predictions could be addressed by using satellite imagery instead of UAV imagery, since acquiring satellite imagery is more affordable [128] and less complex than gathering UAV imagery; however, satellite imaging is vulnerable to clouds, does not offer on-demand sensing, and has lower spatial resolution imagery. This means the farmer would need to be okay with a) lower spatial resolution yield prediction maps and b) the fact that yield prediction could not be done as frequently.

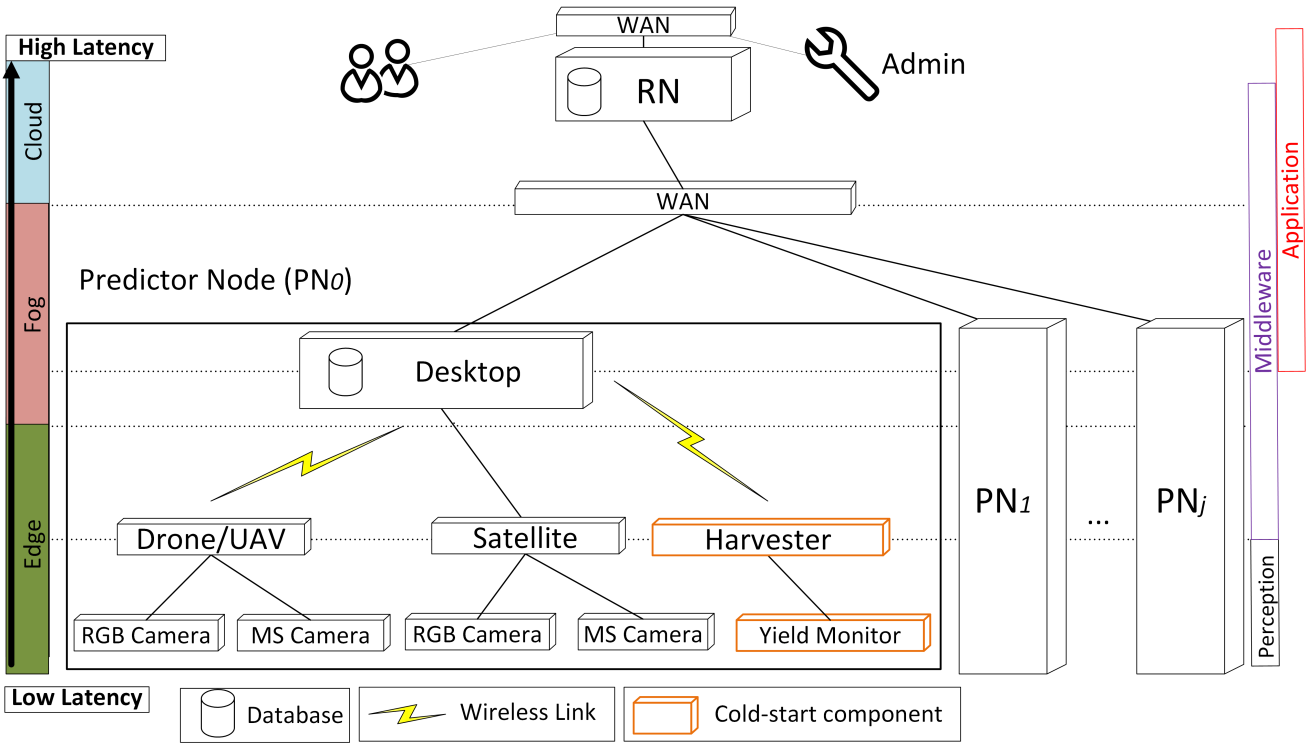


Figure 4.0.1: Overview diagram of the proposed privacy-aware smart farming IoT system architecture that was adapted to the imagery-based yield prediction use case, where WAN = wide-area-network, RN = root node, RGB = red-green-blue, and MS = multispectral. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed.

4.3 Materials and Methods

This section explains the methodology we applied for gathering sensor data and pre-processing the data, and we discuss evaluation techniques used.

4.3.1 Study Site

Corn was grown in the 2020 and 2021 growing season at a smart farm named Area X.O located in Ottawa, Ontario, Canada (45°19'8.17"N, 75°45'22.40"W). There were 6 fields (Fields 1, 2, 3, 4, 9, and 11) involved in the studies described in this chapter. The size of the fields was 3.38 ac (where 1 ac $\approx$ 0.405 ha), 4.67 ac, 3.55 ac, 2.07 ac, 1.43 ac, and 48.08 ac, respectively. Figure 4.0.2 illustrates a map of the study site.

4.3.2 Data Collection

In this section, we describe the available data sources at Area X.O that were used in our yield prediction studies described later in this chapter. A summary of these data sources

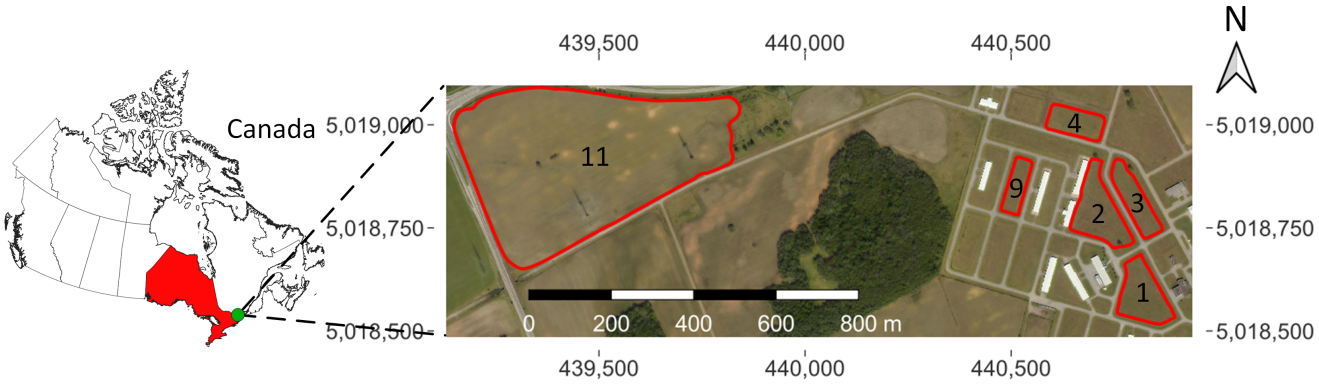


Figure 4.0.2: A map of the test site. The map illustrates the field numbers and field boundaries and includes a distance scale and coordinate grid that uses the coordinate reference system WGS 84 / UTM zone 18N.

Table 4.0.2: Yield prediction studies data source descriptions

Dataset Name	Temporal Resolution	Spatial Resolution	Growing Season	Data Source
Weather	daily	static	2021	Public Weather Station
UAV Imagery	∼weekly	0.7 cm to 3.9 cm	2021	InDro Robotics
In-field Camera	2/day	static	2021	Field Climate
Yield	1/year	∼2.5 m	2020, 2021	Trimble

can be found in Table 4.0.2.

4.3.2.1 Weather

We gathered daily weather data from a local weather station named *OTTAWA INTL A* that was located near the study site. The dataset was publicly available and was obtained from the Government of Canada's Weather website[1].

4.3.2.2 Management

Planting was conducted using a white 6606 planter that had 6 rows (15 ft) 30 inches apart (where 1 ft ≈ 0.3048 m and 1 inch ≈ 2.54 cm). The crops were rainfed. For the 2020 growing season, the following field-average management information was applied: seeding rate of 37.4 kS/ac (kS/ac = 1000 seeds per acre), 12.6 gal/ac of herbicide (a Halex 1.7 L + Atrazine 0.25 L mix), 119.4 lbs/ac of 8-17-27 (8 % N, 17 % P, and 27 % K) fertilizer, 100.2 lbs/ac of 40-0-0 5.5 UAS, 19.1 gal/ac of 10-10-10 50 % Poly, and 28.4 gal/ac of Urea Ammonium Nitrate (UAN). For the 2021 growing season, Table 4.0.3 presents the tilling techniques, seeding rates, herbicide rates, and fertilizer rates applied to each field. For both

[1] https://climate.weather.gc.ca/historical_data/search_historic_data_e.html, accessed on February 9 2024

2020 and 2021 growing seasons, Table 4.0.4 presents the dates of these field activities. The management information presented in this section was obtained from the Trimble platform.

Table 4.0.3: Tillage technique, seeding rates, herbicide application rates, and fertilizer application rates for each field, where CT = conventional tilling (or broadcast), IST = innovative strip-tilling (or fertile stripping), LP = lime pellets, kS/ac = 1000 seeds per acre, 1 lbs ≈ 0.454 kg, 1 gal ≈ 3.785 L, 1 ha ≈ 2.471 ac, and the 7-32-23 notation indicates (7 % N, 32 % P, and 23 % K) fertilizer.

			Herbicide Rates	**Fertilizer Rates**					
Field	**Tilling Technique**	**Seeding Rates (kS/ac)**	**Acuron 1.5 L + Crush 0.8 L (gal/ac)**	**7-32-23 (lbs/ac)**	**40-0-0 5.5 UAS (lbs/ac)**	**Urea 60/40 with LP (lbs/ac)**	**3-18-18 (gal/ac)**	**UAN 32 % (gal/ac) (Side-Dressing)**	**LP (lbs/ac)**
1	CT	33.7	12.3	150	300	0	0	20.1	0
2	IST	33.7	12.4	125	0	300	10.4	20.3	4409
3	CT	33.8	12.1	150	300	0	0	20.3	4409
4[1]	CT	33.6	12.2	125	0	0	0	20.0	4409
9	IST	33.8	12.4	128	0	327	10.6	20.0	4409
11[2]	IST	31.8	12.1	125	0	362	10.2	20.0	4409

[1] Field 4 also had 25 gal/ac of UAN 32 % applied during its conventional tilling.

[2] Field 11 also had rates of approximately 150 lbs/ac of 5-26-30 and 9-23-31 fertilizer applied to the southern and northern parts of the field, respectively, during its Fall 2020 strip tilling.

Table 4.0.4: Field management activity dates.

Field Activity	**Date/Period (2020)**	**Date/Period (2021)**
Tilling[1]	May 13	April 28 to May 6
Planting	May 14	May 14
Herbicide Spraying	June 4	May 27 to May 28
Side-dressing	June 25	June 23 to July 7
Harvesting	November 10	November 5 to November 6

[1] Field 11 also had strip tilling conducted November 13 2020.

4.3.2.3 Imagery

Two sources of imagery were available in 2021, namely from UAV cameras and an in-field camera. UAV imagery was captured by InDro Robotics from May 26 2021 to October 1 2021. The company created orthomosaics using the PIX4Dmapper software version 4.6.4, performed geometric calibration, and performed radiometric calibration. Initially an Autel EVO 2 drone was used and on June 22 2021 we upgraded to a dual-payload DJI M210 drone. Table 4.0.5 provides image acquisition and flight details. The MS camera supported the red (668 nm ± 5 nm), green (560 nm ± 10 nm), blue (475 nm ± 10 nm), NIR (840 nm ± 20 nm), and red-edge (717 nm ± 5 nm) bands [174]. Images were acquired daily from May 26 to May 31 and weekly from June 8 to October 1, generally between 11 am and 3 pm. The image resolution was between 12.0 megapixels and 33.2 megapixels for the RGB imagery and was 1.2 megapixels for the MS imagery. The image spatial resolution was between 0.7

Table 4.0.5: UAV image acquisition details over the 2021 growing season

Camera	Image Type	Acquisition Period	Flight Height (m)
EVO 2 Gimbal	RGB	May 26 to June 8	50
Zenmuse X4S	RGB	June 22 to October 1	40
Mica Sense Red-Edge M	MS	June 22 to October 1	40

cm and 1.3 cm for the RGB imagery and between 2.8 cm and 3.9 cm for the MS imagery. Table 4.0.6 provides a summary of the UAV imagery dataset. The total size of the raw imagery dataset, which contained approximately 256,000 raw images, was approximately 916 GB. For each field, the raw images were then stitched together into a single large image, called an orthomosaic. Orthomosaics were generated for each field, a single RGB orthomosaic and five MS orthomosaics (one for each MS band). Figure D.1 in Appendix D shows a visual example of a NDVI and RGB orthomosaic. The PIX4Dmapper software was used for orthomosaic generation. The total size of all the MS and RGB orthomosaics (there were approximately 330 orthomosaics) was approximately 130 GB and 89 GB, respectively. Table D.1 in Appendix D provides the mean size of the resulting orthomosaic files. From the orthomosaics, we computed VIs (which resulted in additional orthomosaics) in our experiments described next in this chapter. A Pessl CropVIEW® camera was installed in Field 11 and captured two daily RGB images throughout the 2021 growing season, which were accessible via the Field Climate platform.

4.3.2.3.1 UAV Imagery Quality The imagery data contained some imperfections. Some raw imagery had artifacts in the form of parts of the UAV being visible in the image. Some orthomosaics were quite noisy (e.g., missing parts of the field or parts of the field being stitched to incorrect locations). For the MS imagery, approximately 20 % of the orthomosaics contained a bit of noise, and there were no orthomosaics with a lot of noise. For the RGB imagery, approximately 22 % of the orthomosaics contained a bit of noise, and approximately 6 % of the orthomosaics contained a lot of noise. Furthermore, sometimes a mission was split into two missions (for the same field) due to battery issues. This resulted in two orthomosaics being generated for a single field, and, as a result, we chose the orthomosaic that contained the most imagery and the least amount of noise before proceeding to the next data processing step. Also, at times the NIR orthomosaics from a mission were very noisy, meaning we had to exclude all of the MS imagery from those missions (since VIs requiring the NIR band could not be computed for these missions). More details of the imagery data processing steps are discussed later in Section 4.4.3.7.

4.3.2.4 Yield

Yield data from the 2020 and 2021 growing seasons were used in the studies described in this chapter. Shelled corn was harvested November 10 2020 and November 5–6 2021 using a John Deere S660 combine equipped with a yield monitor and GPS equipment. Yield readings were sampled at 1 Hz. The harvester had an 8-row combine harvest width of

Table 4.0.6: Mean (averaged over all the missions conducted in 2021) UAV imagery acquisition mission properties for a single mission, where results in brackets are the results multiplied by a factor of five to represent the inclusion of imagery from all five MS bands, where RGB = red-green-blue and MS = multispectral.

Field	Imagery Type	Total Raw Image Size (MB) per Mission	Number of Raw Images per Mission	Raw Image Size (MB)	Imagery Data Generation Rates (Mbps)
1	RGB	2,620	263	11	41.6
2	RGB	2,775	292	11	39.7
3	RGB	2,180	220	11	41.4
4	RGB	1,507	150	11	41.8
9	RGB	1,413	145	11	40.7
11	RGB	7,698	769	10	41.8
1	MS	1,138 (5,692)	462 (2,308)	2 (12)	73.0
2	MS	1,264 (6,318)	512 (2,562)	2 (12)	73.0
3	MS	1,021 (5,107)	414 (2,071)	2 (12)	73.0
4	MS	734 (3,672)	298 (1,489)	2 (12)	73.0
9	MS	760 (3,798)	308 (1,540)	2 (12)	73.0
11	MS	4,080 (20,401)	1,655 (8,273)	2 (12)	73.0

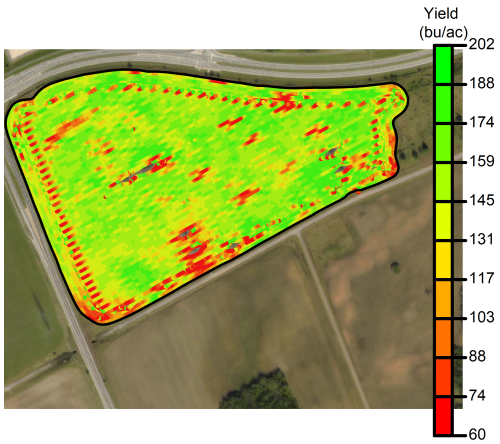
20 ft that automatically adjusted its width to avoid harvesting previously harvested crop rows. The average moisture content was approximately 21 % in 2020 and 23 % in 2021. The harvester's yield data were calibrated to compensate for the yield sensor lag time delay. Figure 4.0.3 shows the precision maps of the actual yield in a) raw form and b) pre-processed form after applying cleaning and interpolation to a 2.5 m $\times$ 2.5 m grid. Table D.2 in Appendix D presents statistical properties of the raw, cleaned, and interpolated yield datasets for each field. In 2020, the raw yield dataset from Field 11 contained 13,140 samples and had an average yield of 155.98 (in bu/ac). In 2021, the raw yield datasets from Fields 1, 2, 3, 4, 9, and 11 contained 1038, 1577, 840, 873, 514, and 13 264 samples, respectively, and had an average yield (in bu/ac) of 112.58, 146.34, 113.16, 74.48, 118.88, and 149.35, respectively.

4.3.3 Corn Growth Stages

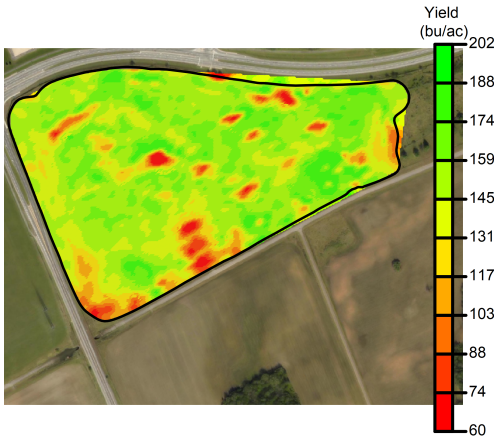
Some of our experiments involved estimating the crop growth stage of corn. Growth stage estimation is important, because it allows our findings to be compared to other related works that present results in terms of growth stage. Corn growth stages can be split into vegetative (V) and reproductive (R) stages. For example, corn at the V8 vegetative stage has 8 collars, and R1 is the silking reproductive stage [190, 55].

4.3.4 Vegetation Indices

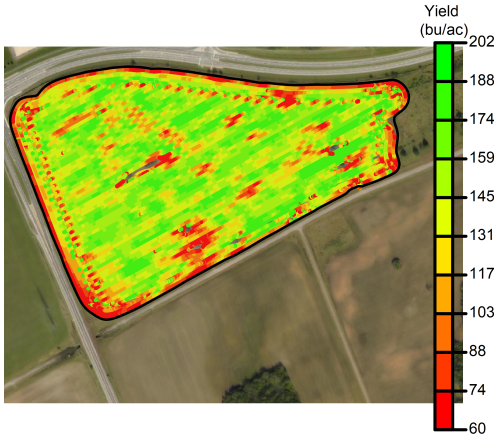
The VIs used in the studies described in this chapter are listed in the following tables found in Appendix A: NIR+RGB band (NIR-based) VIs in Table A.1, RGB band VIs in Table A.2, and red-edge-based VIs in Table A.3. The RGB imagery had digital pixel values



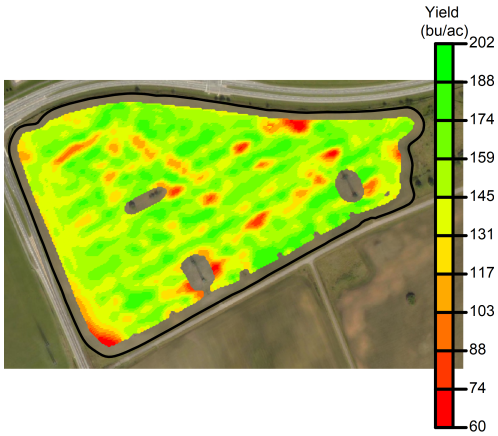
(a) 2020 Field 11 (raw)



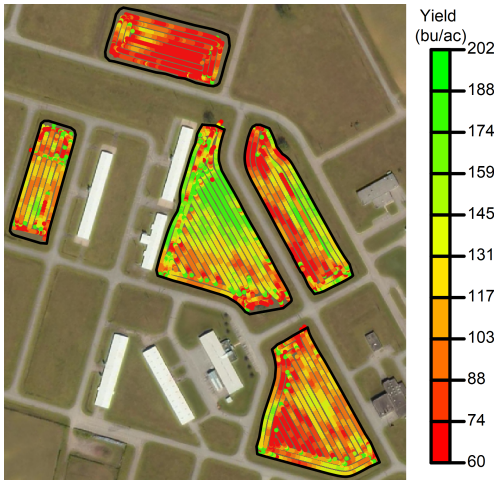
(b) 2020 Field 11 (pre-processed)



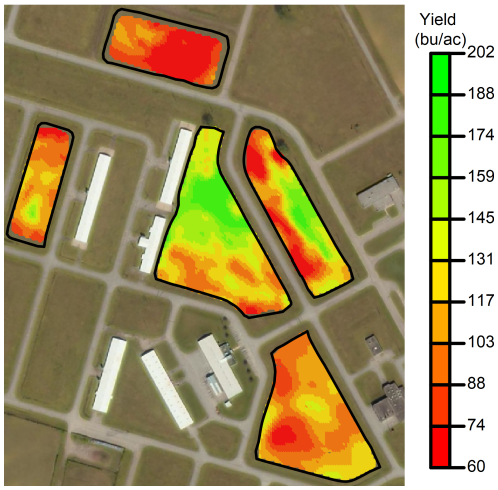
(c) 2021 Field 11 (raw)



(d) 2021 Field 11 (pre-processed)



(e) 2021 Small fields (raw)



(f) 2021 Small fields (pre-processed)

Figure 4.0.3: Yield precision maps, for each field (see Figure 4.0.2 for field numbers), of the raw and pre-processed (cleaned and interpolated) yield, where units are in bu/ac (1 ac $\approx$ 0.405 ha).

between 0 and 255, so we normalized the values between 0 and 1 and kept both versions, because some RGB VIs (e.g., ExG) expect band values between 0 and 1, while others (e.g., CIVE) expect band values between 0 and 255.

We used the QGIS software version 3.22.6 to crop the orthomosaics into smaller orthomosaics for each respective field. We used the R programming language version 4.1.3 *raster* library to compute VI rasters from the cropped orthomosaics.

4.3.5 Model Selection

In some of our studies discussed in this chapter we applied feature selection and hyperparameter tuning to perform model selection. We discuss the applied methodology in this section.

4.3.5.1 Features Selection

For some of our studies, since there were many features in the derived datasets and we wanted to avoid the *curse of dimensionality*[2], we performed feature selection, using partial least squares decomposition. We chose this to perform feature selection, because it addresses the multicollinearity problem (predictor variables are correlated), which is a common property of agri-data, and it also considers the predictor variables' relationship to the target variable when selecting the features [86]. We used Python's *sklearn* package for the PLSR implementation[3]. As recommended by Geladi et al. [78], we scaled the predictor variables by mean-centering them and dividing them by their respective standard deviation before applying PLSR feature selection. We iteratively trained and evaluated PLSR models using block CV. Specifically, for each fold, we applied a greedy feature selection approach, where we a) trained a PLSR model and b) iteratively removed the feature with the smallest PLSR coefficient in absolute value (the least important feature) and re-trained a PLSR model with the remaining features. We applied this entire process a number of times, equal to the number of trials involved with using grid search, to vary the number of PLSR latent components. Over all trials and feature sets, we selected the feature set that led to the best PLSR model performance in terms of both RMSE and R^2.

4.3.5.2 Hyperparameter Tuning

For some of our studies, hyperparameter tuning was conducted. We explain the details of this hyperparameter tuning in this section. Hyperparameter tuning was conducted using i iterations of nested k-j-fold CV (standard and/or block CV) using random search with t search trials and e executions per trial. The input dataset was first split into k folds to create the training-testing dataset pairs. A second inner loop of CV was then applied to

[2] A large number of features may lead to a very sparse dataset and this can hinder model performance [72, p. 243], especially when there are more features than samples.

[3] Our implementation was inspired by an online tutorial [203].

the training data of each of the k folds. Each inner CV loop split the training dataset from the outer loop's fold into j training-validation inner folds. Each training-validation fold had t sets of randomly selected hyperparameters that were produced by randomly sampling (with replacement) from a search space. Let H denote the set (or superset) that contained these t sets of hyperparameters for a single inner fold. Each hyperparameter set in H was used to train and evaluate e models in terms of their validation performance. That is, for each hyperparameter set $h \in H$, we initialized, trained, and evaluated a model e times using h as the model's hyperparameters to minimize the performance impacts resulting from a model's initial random state (e.g., random weight values for deep learning models). The model that produced the best RMSE and R^2 validation performance was chosen as the best performing model for the inner fold, and model generalization performance was then assessed by training the model (from scratch) using the entire outer fold's training dataset and evaluating the model against the testing set. We repeated this process for k outer loops and j inner loops of CV to produce model generalization performance for each outer-inner fold pair, and we performed this nested CV over i iterations. We treated the overall performance of a model as the average performance across all folds and iterations. In summary, to estimate a model's generalization performance using this methodology, the total number of models that were trained and evaluated was $i \times k \times j \times t \times e$ models.

4.3.6 Significance Testing

This section describes the significance testing methodology applied by some of the studies discussed in this chapter. We combined the Friedman test [72, pp. 351–357] with a confidence level of 5 % ($\alpha = 0.05$) and the Wilcoxon's signed-rank test [72, pp. 351–357] to perform statistical significance testing to compare groups of results. The Friedman test by itself is insufficient for identifying the best-performing result set, so, as advised by Trawiński et al. [261], we combined the Friedman test with the Wilcoxon's signed-rank test as follows: we applied the Friedman test to the entire set of results (the collection of result sets) to identify whether all result sets were similar (when the hypothesis is not rejected) or not (when the hypothesis is rejected). When the Friedman test null hypothesis was rejected, we then applied the Wilcoxon's signed-rank test to each pair of result set. For experiments that only had two groups of result sets, we only applied the Wilcoxon's signed-rank test. To identify the result sets that had significantly superior performance than the other result sets, we first identified the best ranked result set. Let B denote this best ranked result set for notation simplicity. Once identified, B was used to find the other result sets that were not significantly different than B (i.e., loosely speaking, were similar and among the best result sets) in terms of the Wilcoxon's signed-rank test. These result sets were also considered as significantly superior in terms of performance compared to the other result sets that were significantly different from B.

4.4 Yield Prediction Study: Using UAV Imagery and Spatial Cross-validation

Most of the content of this study was published in Killeen et al. [135] as a conference extension of Killeen et al. [132].

The present work is an UAV imagery-based corn yield prediction study that analyzed the effects of image spatial and spectral resolution, image acquisition date, and model evaluation scheme on model performance. CL was used to train and evaluate models. We used various spatial generalization evaluation methods, including spatial CV, to a) identify over-optimistic models that overfit to the spatial structure found inside datasets and b) estimate true model generalization performance. We compared and ranked the prediction power of 55 VIs and five spectral bands over a growing season.

The objectives of the present work were to: a) bridge the knowledge gap between the UAV imagery-based yield prediction literature and spatial data analysis to reveal and avoid over-optimistic model performance; b) determine the best time during the growing season (or the best phenological growth stage) to capture imagery to optimize yield prediction results and minimize the number of UAV flight missions; c) determine the best-performing VIs; d) determine whether an inexpensive RGB camera can be used instead of a costly MS camera; e) determine whether the VI calculation step can be skipped in the prediction process by comparing the prediction performance of raw-bands vs. VIs; and f) determine whether satellite imagery can be used instead of UAV-based imagery to achieve comparable performance at a reduced price. We compared the effectiveness of cameras by examining the difference in yield prediction model performance between: 1) NIR+RGB band VIs, 2) RGB band VIs, and 3) red-edge-based VIs.

In terms of how the present work relates to this thesis: the spatial analysis methodologies applied in the present work tie into the spatiotemporal data analysis aspect of this thesis. Furthermore, the experiments performed in the present work were a starting point for our yield prediction studies. The results of the present work helped us design later yield prediction studies presented next in this chapter. Specifically, acquisition dates and VIs that led to the best model performance allowed us to more efficiently limit the features and datasets considered for analysis in our yield prediction studies presented next in this chapter in Sections 4.5 and 4.6.

The following is a list of limitations of the present work: a) the data gathered was from only a single study site, b) only mono-temporal datasets were used to evaluate models, c) a limited number of models (two ML models) were evaluated, d) no deep learning models were evaluated, e) no hyperparameter tuning was performed, f) no feature selection was performed, and g) UAV imagery and yield datasets were both available for only a single year, so cross-year extrapolation experiments could not be conducted.

4.4.1 Materials and Methods

4.4.1.1 Datasets

The datasets used in the present work were from 2021 data sources; that is, we used the daily weather station data, the yield data, the Pessl in-field camera daily imagery, and all of the UAV MS and RGB imagery datasets.

4.4.1.2 Corn Growth Stages

For the purposes of analyzing the effects of growth stage on crop yield model performance, we assumed that all the crops from each field were in the same growth stage, since we only have one CropVIEW camera that captured Field 11 imagery.

4.4.1.2.1 Growing Degree Days In the present work we estimated the growth stage of the corn by examining the images from the in-field CropVIEW camera, counting the number of plant collars on each plant, and using the accumulated growing degree days (GDDs) method [187] (otherwise known as Growing Degree Units [17]). GDDs can be used to estimate crop growth by modeling the number of days that have ideal/sufficient temperature for crop growth [17]. GDD can be calculated as follows [55] in Equation 4.1:

$$\text{GDD} = \frac{T_{max} + T_{min}}{2} - T_{base} \tag{4.1}$$

where T_{max} is the maximum daily temperature in °F, T_{min} is the minimum daily temperature in °F, and T_{base} ($T_{base} = 50$ °F in this study [17]) is the base temperature for the corresponding crop (corn). Any maximum daily temperature above 86 °F is set to 86 (the optimum temperature for corn [80]) and any minimum daily temperature below 50 °F is set to 50 in the GDD calculation [188, 80, 1]. The GDD is accumulated over the season to estimate the growth stage, where a new collar appears approximately every 82 GDD from VE to V10 and every 50 GDD from V11 to Vn [187]. The reproductive development after silking (R1) can also similarly be predicted via GDDs accumulation. For stages after R1, we use the accumulated GDD provided in Monsanto [177].

4.4.1.2.2 Estimation By analyzing the CropVIEW imagery, the VE stage started on day of year (DoY) 145 (May 25 2021) when the crop emerged. Similar to Oglesby et al. [191], from V1 to V13 we roughly counted the number of collars on each plant from the CropVIEW images. We estimated the VT stage when most of the plants had visible tassels forming in the CropVIEW images. R1 was determined when observable silks were found in the CropVIEW images [191]. From R2 to R6, we used the accumulated GDD suggestions from Monsanto [177], which are approximately 1660, 1859 (interpolated via the days after silking), 1925, 2320, and 2700 accumulated GDD for stages R2, R3, R4, R5, and R6,

Table 4.4.1: Estimated corn growth stages [190, 55] for the 2021 growing season using accumulated GDD [177, 188] and in-field crop camera imagery, where DoY = day of year, AGDD = accumulated growing degree days, and GS = growth stage.

DoY	145	152	154	157	159	164–170	179–186	190–197	201	204	226	230	237	264
AGDD	191	264	294	360	415	506–596	769–897	957–1108	1195	1249	1668	1739	1923	2315
GS	VE	V1	V2	V3	V4	V5–7	V8–10	V11–13	VT	R1	R2	R3	R4	R5

respectively. This should be a reasonable estimate given that the accumulated GDD for the VT stage was suggested by Monsanto [177] to be 1135 , while in the present work the corn reached the VT stage at DoY 201 with 1195 accumulated GDD (both are relatively similar). Furthermore, Monsanto [177] states that, during the R3 stage, the corn ears become brown and dry. We confirmed via the CropVIEW imagery that the ears achieved a peak dry brown color on DoY 230 with accumulated GDD 1739. This is relatively close to the interpolated accumulated GDD 1859 suggested by Monsanto [177], providing further evidence that the growth stage estimates performed in the present work were reasonable. The present work's growth stage estimates are listed in Table 4.4.1.

4.4.1.3 Feature Extraction

Since the yield and imagery datasets did not share the same spatial and temporal resolution, data fusion was required to perform feature extraction.

4.4.1.3.1 Yield We applied most of the yield cleaning steps mentioned in Section 2.7.1 by implementing the steps in Java version 18.0.1 (the project is open-source and can be found on GitHub[4]). We did not remove samples from headlands[5] due to the small size of the fields. For harvester speed and yield inlier removal, and for turn removal, we applied the forward-backward pass method proposed by Lyle et al. [159]. Note that there was a parameter configuration error in our implementation of the cleaning process, and as a result, the forward-backward pass method was effectively not applied correctly, meaning the yield datasets used in the experiments may have had a few more outliers. We used the Vesper software version 1.6 to perform yield semivariogram and interpolation using the block kriging method with a block size 10 m × 10 m, an interpolation grid of 2.5 m × 2.5 m, and a local variogram with 30 lags, 50 % lag tolerance, and a maximum distance of 55 m. We removed readings with high kriging variance. We used the R programming language version 4.1.3 *sf* and *sp* libraries to remove readings from Field 11 that were inside no-yield areas (e.g., below a power tower). The mean yield (in bu/ac) for each field after cleaning and interpolation is as follows: Field 1 = 107.81, Field 2 = 144.18, Field 3 = 118.18, Field 4 = 77.46, Field 9 = 111.31, and Field 11 = 154.67. Table D.2 in Appendix D shows raw, cleaned, and interpolated yield dataset's statistical properties for each field.

[4] https://github.com/patkilleen/geospatial, accessed on January 23 2024

[5] the border of fields where there is increased agriculture machinery traffic

4.4.1.3.2 Imagery and Vegetation Indices In total 55 VIs were chosen and 5 bands (red, green, blue, NIR, and red-edge) were used to derive features for the prediction models. A few VIs were defined by the present work using standard VI operations (a ratio or difference, for example) to add additional RGB and red-edge VIs. All the VIs used are listed in Appendix A in Tables A.1, A.2, and A.3.

4.4.1.3.3 Data Fusion We performed data fusion using a Java program we implemented and applied a mean-, maximum-, and minimum-filter over the imagery data using a circular neighbourhood with a 4.5 m radius around a yield cell's center, p, where p is a 2D coordinate/position value. Figure D.1 in Appendix D provides a visual reference of this neighborhood relative to a NDVI and RGB orthomosaic. We denote this neighbourhood around p as $N(p)$ for notation simplicity, where elements in $N(p)$ are pixel values (raw-band reflectance or VI). A radius of 4.5 m was chosen to compensate for the fact that the orthomosaics' extent may be offset by approximately 2 m due to GPS accuracy limitations. Two types of datasets resulted from the fusion, namely high spatial resolution (HRe) datasets and low spatial resolution (LRe) datasets. HRe datasets represent the availability of high spatial resolution imagery. Its variables are interpolated yield at location p, $\texttt{mean}(N(p))$, $\texttt{max}(N(p))$, and $\texttt{min}(N(p))$. It captures more fine-grained imagery details by additionally including the maximum and minimum aggregates. LRe datasets represent lower spatial resolution imagery (e.g., satellite-imagery). Its variables are interpolated yield at location p and $\texttt{mean}(N(p))$. It fails to capture the heterogeneity of fine-grained image details due to the coarse-grained nature of only using the mean aggregation.

4.4.1.4 Model Hyperparameters

Models used Weka's default hyperparameter configuration, that is, for LR[6], the M5 attribute/feature selection method was chosen; ridge parameter = 1×10^{-8} (or 1E–8); and number of decimal places = 4. For RF[7], bag size = 100%; number of trees = 100; number of candidate attributes = $\lfloor \log_2(\textit{number of input features}) + 1 \rfloor$; leaf minimum number of instances = 1; minimum variance for a split = 0.001; unlimited tree depth; number of decimal places = 2; and random number generation seed = 1.

4.4.2 Experiments

In this section we describe the experiments conducted in the present work. We used the output of the data fusion step to train and evaluate the mono-temporal ML models using various forms of CV. Our goal was to evaluate the accuracy of the predictions made by the models and the execution time of the models. The evaluation metrics used were RMSE, R^2, and CC. The models used were RF and LR. RF and LR models were chosen, since

[6] https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/LinearRegression.html, accessed on February 26 2025

[7] https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html, accessed on February 25 2025

RF [37] and LR [297] have commonly been shown to perform well for yield prediction. We used Weka version 3.8.5 to train and evaluate the models.

There were four types of CV experiments that we ran, namely two standard KF-CV-based experiments (discussed in Sections 4.4.2.1 and 4.4.2.2, respectively) and two spatial CV experiments (discussed in Sections 4.4.2.3.1 and 4.4.2.3.2, respectively), where 10 iterations of each type of CV experiment were performed.

4.4.2.1 Location-only Standard K-Fold Cross-validation

To explore the extent of the effects SA may have on the yield prediction experiments in the present work, as suggested by Ploton et al. [206], 10-fold CV experiments were conducted using only location features to train field-level models (each model only involved data from a single field) to predict yield for each of the fields.

4.4.2.2 Standard K-Fold Cross-validation

In this type of experiment, field-level models were evaluated using 10-fold CV, where folds were created via random sampling, ignoring any spatial structure in the data. Models were trained and evaluated for each of the 6 fields, imagery acquisition dates, VI/raw-band, and both the HRe and LRe dataset types (e.g., for some DoY, $6 \times (55 + 5) \times 2$ datasets would be used to train and evaluate the RF and LR models). We will refer to this type of experiment as a KF-CV experiment.

4.4.2.3 Spatial Cross-validation

We applied two types of spatial CV, to address the spatial structure in the datasets, by strategically creating the folds to reduce SA between the training and testing data.

4.4.2.3.1 Leave-one-field-out Cross-validation In this type of spatial CV experiment, farm-level models were trained and evaluated, and the datasets used were the same as the KF-CV experiments (detailed in Section 4.4.2.2), but the folds were defined differently by sampling 500 samples from each individual field's dataset to define a fold. Meaning, instead of 10 folds, 6 folds were created (or 5 folds when a field's imagery was missing for a day), one fold for each field. This sampling scheme was designed to avoid having larger fields' data be assigned more weight during model training. In this type of experiment, days with imagery available only from a single field were ignored. This type of experiment had two versions, namely leave-one-field-out CV (LOFO-CV) and reverse LOFO-CV (rev-LOFO-CV). LOFO-CV involved training the model using every field but one and testing a model using the remaining field, while rev-LOFO-CV involved training a model using only a single field and testing the model using the remaining fields.

4.4.2.3.2 K-means-based Cross-validation This type of experiment, which we refer to as a spatial-k-fold CV (SpKF-CV) experiment, is similar to the LOFO-CV experiments, but here the experiments were field-level, and instead of defining a fold as an entire field's dataset, the folds were defined as samples from clusters resulting from applying the k-means clustering algorithm on location data to create 10 spatially disjointed folds inside a single field. Since the number of samples per cluster varied slightly, the fold sizes were defined using the size of the smallest cluster to make every fold equally sized. This type of experiment had two versions, namely SpKF-CV and reverse SpKF-CV (rev-SpKF-CV). SpKF-CV involved training a model using 9 folds and testing the model using the remaining fold, while rev-SpKF-CV trained a model with one fold and tested the model using the remaining 9 folds.

4.4.3 Analysis and Results

In this section, we analyze the results of our experiments and present key findings. Figures 4.4.1 and 4.4.2 illustrate the results of an analysis that involved examining average CC performance of RF and LR over the entire growing season, to compare the difference between a) the evaluation method, b) RGB imagery, c) MS imagery, and d) the image acquisition dates. Two VIs were chosen: ExR (a RGB VI that represents RGB imagery) and $\text{NDVI}_{\text{RedEdge}}$ (a MS VI that represents MS imagery). These VIs were chosen, since $\text{NDVI}_{\text{RedEdge}}$ was one of the better performing MS VIs and ExR was one of the better performing RGB VIs. The results for each DoY were averaged over both types of datasets (HRe and LRe). Focusing less on the effects of image acquisition date, Figure 4.4.3 illustrateS the effects that image spatial resolution (HRe vs. LRe) and ML model choice (RF vs. LR) have on yield prediction performance results for a single image acquisition date (DoY 193). Figure 4.4.3 also enableS the comparison of RGB imagery vs. MS imagery. ExR was chosen as the VI to represent the RGB imagery, since it was one of the better performing RGB VIs and it did well for DoY 193. Similarly, $\text{NDVI}_{\text{RedEdge}}$ was one of the better performing MS VIs, so it was chosen in this analysis. DoY 193 was chosen, since ExR and $\text{NDVI}_{\text{RedEdge}}$ did similarly well on that day, which enables analysis of the effects that evaluation method, ML model choice, and imagery spatial resolution have on performance. Figure 4.4.4 shows the results of performing yield prediction using KF-CV and using only location data as features.

It is worth noting that although we compared the performance results of LOFO-CV and rev-LOFO-CV experiments to the results of the SpKF-CV, rev-SpKF-CV, and KF-CV experiments, strictly speaking, it may not be necessarily correct to compare these results. The SpKF-CV, rev-SpKF-CV, and KF-CV experiments differed in their sampling scheme, but virtually they shared the same input datasets (input datasets were field-level datasets, i.e., only from a single field), whereas the LOFO-CV and rev-LOFO-CV experiments' input datasets (input datasets were farm-level datasets, i.e., they included data from multiple fields) were mostly different from those of SpKF-CV, rev-SpKF-CV, and KF-CV. Nevertheless, we compared their results to gather insights on the effects the various evaluation methods had on model performance.

4.4.3.1 Imagery Type Analysis

4.4.3.1.1 Vegetation Index Ranking In this analysis the spectral channels (VIs and raw-bands) were ranked by average performance for each field over different stages in the growing season, ignoring the scale of performance differences between VIs. For notation simplicity, we may refer to raw-bands as VIs in this analysis.

In our previous work [132] (the conference version of the present work), the CC was incorrectly used for VI ranking (large negative correlations may have been ignored in the ranking analysis). In the present work we chose the R^2 measure because, a) it is more adequate for ranking (larger R^2 values represent strictly better performance) and b) large outliers in the RMSE results existed, likely caused by VI divisions by nearly zero, which skewed the RMSE average results. Furthermore, we chose the LOFO-CV and rev-LOFO-CV experiment results for performing this VI ranking analysis to avoid having over-optimistic performance affect rankings. The LR model results were chosen, since LR was the better performing model for these experiments. Table 4.4.2 illustrates the top five best ranked VIs for each of the defined stages of the growing season, namely the early, middle (mid), late, and entire season. These seasons were defined as follows: early = {DoY = 173, DoY = 179}, mid = {DoY = 193, DoY = 201, DoY = 211, DoY = 216, DoY = 223, DoY = 231}, late = {DoY = 237, DoY = 245, DoY = 252, DoY = 257, DoY = 273}, and entire = {early ∪ mid ∪ late}. In terms of growth stages, by observing Table 4.4.1, early season includes stages V7 and V8 (V10 is not included, for example, since early season includes up to DoY 179 and does not include DoY 186), mid season includes stages from V11 to R3, and late season includes stages R4 and R5. A source of bias is that the mid season contained more growth stages than the early and late seasons. The RGB VIs included in this analysis were only from imagery gathered by the RGB camera, and the MS VIs were from the MS camera. Note that the early season only included two acquisition dates because no MS imagery was gathered before that.

Table 4.4.2: Yield prediction experiments: five best-performing VIs on average over the growing season by LR model and LOFO-CV and rev-LOFO-CV evaluation methods, where VIs listed in cyan-, orange-, and black-colored font are NIR-, red-edge-, and RGB-based VIs, respectively; LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and the VIs are defined in Appendix A in Tables A.1, A.2, and A.3.

	Early Season		Mid Season		Late Season		Entire Season	
	HRe	**LRe**	**HRe**	**LRe**	**HRe**	**LRe**	**HRe**	**LRe**
LOFO-CV	OSAVI	$NDVI_{RedEdge}$	$NDVI_{RedEdge}$	red-edge (raw)	$NDVI_{Green}$	$SRI_{NIR,RedEdge}$	$NDVI_{Green}$	$SRI_{NIR,RedEdge}$
	RDVI	$SRI_{NIR,RedEdge}$	$SRI_{NIR,RedEdge}$	$SRI_{NIR,RedEdge}$	$SRI_{NIR,Green}$	$SRI_{NIR,Green}$	$NDVI_{RedEdge}$	$NDVI_{RedEdge}$
	SAVI	NDVI	red-edge (raw)	$NDVI_{RedEdge}$	GCVI	NG	$NDVI_{Blue}$	NG
	MCARI2	$DVI_{Green,Red}$	NG	GCVI	$NDVI_{Blue}$	GCVI	$SRI_{NIR,RedEdge}$	GCVI
	MTVI2	$NDVI_{Green}$	$NDVI_{Green}$	NG	red-edge (raw)	$NDVI_{RedEdge}$	NG	$SRI_{NIR,Green}$
reverse LOFO-CV	MSAVI	NDVI	red-edge (raw)	$NDVI_{RedEdge}$	MCARI	MCARI	$SRI_{NIR,RedEdge}$	$NDVI_{RedEdge}$
	$DVI_{NIR,RedEdge}$	OSAVI	$NDVI_{RedEdge}$	$SRI_{NIR,RedEdge}$	TCARI	$DVI_{RedEdge,Red}$	MCARI	$SRI_{NIR,RedEdge}$
	NDVI	$NDVI_{Blue}$	$SRI_{NIR,RedEdge}$	red-edge (raw)	$SRI_{RedEdge,Red}$	$SRI_{RedEdge,Red}$	$NDVI_{RedEdge}$	MCARI
	$NDVI_{Blue}$	RDVI	NG	NG	IKAW4	TCI	red-edge (raw)	$DVI_{RedEdge,Red}$
	MCARI2	SAVI	$SRI_{NIR,Green}$	GCVI	OSAVI	TCARI	NG	NG

By analyzing the best performing VIs in Table 4.4.2, we can see that

Table 4.4.3: Yield prediction experiments: five worst-performing VIs on average over the growing season by LR model and LOFO-CV and rev-LOFO-CV evaluation methods, where VIs listed in cyan-, orange-, and black-colored font are NIR-, red-edge-, and RGB-based VIs, respectively; LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and the VIs are defined in Appendix A in Tables A.1, A.2, and A.3.

	Early Season		Mid Season		Late Season		Entire Season	
	HRe	**LRe**	**HRe**	**LRe**	**HRe**	**LRe**	**HRe**	**LRe**
LOFO-CV	MCARI	$SRI_{Green,Blue}$	TCARI	IKAW2	$MSR_{Green,Red}$	IKAW2	$DVI_{g,b}$	ExB
	red (raw)	IKAW2	$SRI_{Green,Blue}$	CIVE	NGRDI	blue (raw)	IKAW5	IKAW2
	IKAW2	blue (raw)	IKAW2	$SRI_{Green,Blue}$	IPCA	$SRI_{Green,Blue}$	ExB	blue (raw)
	blue (raw)	green (raw)	IKAW5	blue (raw)	$SRI_{Green,Red}$	green (raw)	IKAW2	$SRI_{Green,Blue}$
	green (raw)	*b*	ExB	IPCA	$SRI_{Green,Blue}$	IPCA	$SRI_{Green,Blue}$	IPCA
reverse LOFO-CV	$SRI_{Red,Blue}$	IKAW1	red (raw)	green (raw)	ExB	blue (raw)	$SRI_{Green,Blue}$	$SRI_{Green,Blue}$
	green (raw)	$SRI_{Red,Blue}$	green (raw)	*b*	$DVI_{g,b}$	WDRVI	ExB	IPCA
	blue (raw)	green (raw)	$MSR_{Green,Red}$	IKAW1	IKAW5	$SRI_{Green,Blue}$	green (raw)	green (raw)
	b	*b*	blue (raw)	IPCA	red (raw)	NDVI	blue (raw)	blue (raw)
	IPCA	IPCA	$SRI_{Green,Red}$	$SRI_{Red,Blue}$	$SRI_{Green,Blue}$	red (raw)	red (raw)	red (raw)

- In general, MS imagery led to better performance than RGB imagery. We can see two RGB VIs that were among the top five best ranked VIs: $DVI_{Green,Red}$ in early season for LOFO-CV-LRe and IKAW4 in the late season for rev-LOFO-CV-HRe.
- For rev-LOFO-CV, we can see red-edge-based VIs did better from middle to late season.
- NIR-based VIs did especially well earlier in the season, which makes sense, since the NIR reflectance decreases around the middle of the growing season [166]. NDVI was also among the top-ranked VIs in early season.
- Another noteworthy VI is the $NDVI_{Green}$, who was relatively high ranked for the LOFO-CV experiments using HRe data.
- We can also see that the red-edge raw-band was frequently among the top five highest ranked VIs, suggesting that we could save computational costs and skip the VI calculation step by using the red-edge band directly.
- Over the entire growing season, the three VIs among the top five best ranking performance for each of HRe, LRe, LOFO-CV, and rev-LOFO-CV, were $SRI_{NIR,RedEdge}$, $NDVI_{RedEdge}$, and NG.

By analyzing the worst performing VIs in Table 4.4.3, we can see that

- Approximately 70 % of these VIs included the blue band in their definition, while the top five best ranked VIs rarely included the blue band in their definition. In fact, the middle of the season had no VIs derived from blue that were ranked among the top five. Interestingly, $NDVI_{Blue}$ was ranked among the top five best VIs for LOFO-CV-HRe, suggesting that the blue band still has prediction power when combined with other MS bands.
- Approximately 30 % of the raw-bands, all of which were RGB, were among the worst performing VIs.

- Nearly all the worst performing VIs (95 %) were RGB-based. There were no NIR-based VIs in the early and middle season that were among the five worst VIs. Only in late season for rev-LOFO-LRe were there two NIR-based VIs among the worst VIs. On the other hand, there were no red-edge-based VIs among the worst during late season. Only in the early and middle season for LOFO-CV-HRe was there a red-edge-based VI among the five worst.

4.4.3.1.2 MS Imagery vs. RGB Imagery We can see in Figure 4.4.5 that in the over-optimistic case, when using KF-CV, on average, MS imagery consistently outperformed RGB imagery for both LR and RF, and for both HRe and LRe datasets.

We can see in Figures 4.4.1 and 4.4.2 that for DoY 193, LR-rev-LOFO-ExR (an over-pessimistic evaluation method) did better than LR-KF-CV-$\text{NDVI}_{\text{RedEdge}}$ (an over-optimistic evaluation method), suggesting that RGB imagery can outperform MS imagery. It should be kept in mind that the performance trends in Figures 4.4.1 and 4.4.2 are averaged out over both the HRe and LRe dataset types, meaning performance trends such as the excellent results of RF-KF-CV-HRe are masked in this chart. Furthermore, rev-LOFO-ExR also outperformed every other over-pessimistic spatial CV approach for DoY 193 for both RF and LR. This suggests a RGB camera could be used to obtain reasonable results in early-to-mid season, instead of using an expensive MS camera. For almost every other DoY other than 193, the $\text{NDVI}_{\text{RedEdge}}$ experimental results consistently did better than the ExR. For LR, even the four more pessimistic spatial CV techniques for $\text{NDVI}_{\text{RedEdge}}$ (rev-LOFO-CV, LOFO-CV, rev-SpKF-CV, and SpKF-CV) outperformed the over-optimistic KF-CV-ExR on multiple occasions during the middle of the season. This was not observable for RF performance, although RF overfitting to the spatial structure of the data may explain these trends. Nevertheless, this suggests MS imagery is generally better for yield prediction than RGB imagery, which is consistent with the findings in other literature [93].

We can see in Figure 4.4.3 that in general, $\text{NDVI}_{\text{RedEdge}}$ led to more standard deviation in the CC performance than ExR, except for the SpKF-CV approach. We can also see that for the LOFO-CV experiments (and most other types of experiments), $\text{NDVI}_{\text{RedEdge}}$ outperformed ExR in average CC performance, while for the rev-LOFO-CV experiments, the opposite was the case, ExR outperformed $\text{NDVI}_{\text{RedEdge}}$. The rev-LOFO-CV experiments were designed to be more pessimistic than LOFO-CV and act as a baseline to evaluate the true generalizability of a model that is attempting to perform extrapolation from one field's imagery to a new field's unseen imagery. In other words, this type of experiment simulates the situation where a cold-start SFS is being deployed, and only data from one farm is available. These results suggest that ExR generalizes better than $\text{NDVI}_{\text{RedEdge}}$ in a cold-start situation where limited field imagery is available (for DoY 193). This begs the question: for MS imagery, is there a VI that has better average performance for rev-LOFO-CV experiments than $\text{NDVI}_{\text{RedEdge}}$? If such a VI exists, it would suggest that such a VI is better than $\text{NDVI}_{\text{RedEdge}}$ at generalizing when limited field data is available, meaning the choice of VI could be made based on the amount of data available.

Source of bias. Note that for DoY 193, there was an issue with the NIR band for

Field 11, so no MS imagery for Field 11 was considered, while there was RGB imagery for Field 11. In addition, the SpKF-CV results using Field 11 data had better performance than the two smallest fields (Fields 4 and 9). This suggests that the LOFO and rev-LOFO results for DoY 193 might have a bias that favors RGB imagery results, due to MS imagery missing for Field 11 that same day. Furthermore, some days only had imagery from one field (DoY 211 for both RGB imagery and MS imagery, and DoY 231 for MS imagery), so the LOFO-CV and rev-LOFO-CV evaluation methods could not be applied, meaning the data had to be interpolated in Figures 4.4.1 and 4.4.2 for plot line continuity. Similar line continuity interpolation for DoY 252 was performed for MS imagery plotted in Figure 4.4.5. Furthermore, for DoY 252, we only had RGB imagery available, so the MS performance trends were also interpolated for that day. In particular, if we observe Figures 4.4.1 and 4.4.2, for DoY 211 and 231 there were spikes in performance, which may be attributed to only having imagery from a single field (Field 3 for DoY 211 and Field 11 for DoY 231 for the MS imagery). This could also be attributed to a previously discussed observation that red-edge-based VIs performed generally better during the middle to late season.

Takeaway. MS imagery, especially imagery containing the red-edge band, led to the best yield prediction results and should be favored over RGB imagery if MS imagery is available. One could save computational costs and skip the VI calculation step by using the red-edge band directly. The $\text{SRI}_{\text{NIR,RedEdge}}$, $\text{NDVI}_{\text{RedEdge}}$, and NG VIs were found to be among the VIs with the best yield prediction power for the entire season.

4.4.3.2 Effects of Spatial Autocorrelation on Performance

The effects of SA on yield prediction ML model performance are examined in this section by analyzing the results of the location-only feature CV experiments. Figure 4.4.4 shows the results of performing yield prediction using KF-CV and using only location data as features. We can see in Figure 4.4.4 that the RF model learned each field's spatial patterns well, performing virtually perfectly, while LR did more poorly and had trouble doing well on every field.

RF obtained 0.996 CC and 2.281 RMSE, on average, and LR obtained 0.490 CC and 21.260 RMSE, on average. Furthermore, Figure 4.4.5 compares the results of KF-CV experiments for LR vs. RF, HRe vs. LRe, and MS imagery (the average over all NIR-based and red-edge-based VIs) vs. RGB imagery (the average over all RGB-based VIs from the RGB camera); we can see that the yield prediction performance results obtained from RF when using HRe datasets were quite good even early in the season, when images mostly contained soil with little vegetation. One explanation could be that RF was overfitting to the spatial structure in the HRe datasets. Another explanation could be that RF is sensitive to the number of features in the dataset (which we discuss briefly in Section 4.4.3.4) and did better with an increased number of features, although Figure 4.4.3 does not support this as the only explanation, since only when HRe and KF-CV were combined did RF do quite well. In all the other spatial CV cases with HRe, RF did not do as well. These observations suggest that the results of applying KF-CV to RF with HRe (RF-KF-CV-HRe) were over-optimistic and were due to RF overfitting to the spatial structure found in HRe datasets instead of learning the reflectance trends in relation to yield. Intuitively,

good yield prediction performance should be difficult to obtain in the early season when imagery mostly contains bare soil reflectance.

Takeaway These results illustrate that one has to be careful with how one designs ML experiments and sampling schemes using crop imagery and yield data for yield prediction, since assuming independence between the training and testing datasets should be avoided when a spatial dependence structure exists; otherwise over-optimistic results may be obtained and this could lead to misinformed decision making by stakeholders.

4.4.3.3 Model Comparison

We analyze the differences in performance between RF and LR in this section. We can see in Figures 4.4.3 and 4.4.5 that RF overfitted to the spatial structure when RF-KF-CV-HRe was used, since for no other spatial CV method did RF outperform RF-KF-CV-HRe.

The observations we made about RF may be explained by using the location-only feature experiment findings discussed in Section 4.4.3.2; that is, RF could make use of the underlying spatial structure in the data to make yield predictions using the HRe dataset type instead of learning reflectance trends, since the HRe dataset by design contained more spatial structure information than LRe. When using LRe datasets, RF appeared to overfit less to the spatial structure, since a) RF's performance was lower than LR's for the entire season (shown in Figure 4.4.5) and b) in both the RF-rev-LOFO-CV-LRe and RF-rev-LOFO-CV-HRe experiments, ExR did better than RF-KF-CV-LRe (shown in Figure 4.4.3).

In general, LR appears to be better at generalizing, which is consistent with claims made by Zhang et al. [297], since a) over the entire season for KF-CV experiments, the difference in performance of LR for the LRe and HRe datasets was not visibly significant (shown in Figure 4.4.5), although it is worth pointing out that HRe generally did lead to slightly better performance than LRe, and b) the over-optimistic KF-CV results were similar to the over-pessimistic spatial CV results. In fact, LR did generally better than the over-optimistic KF-CV when using LOFO-CV and rev-LOFO-CV, providing further evidence that LR generalizes better than RF when a dataset has a spatial structure (shown in Figure 4.4.3).

Furthermore, we can see in Figure 4.4.3 that in general, RF tended to have less standard deviation than LR when using ExR (except in the case of rev-LOFO-CV), suggesting that RF produced more consistent prediction results than LR for RGB imagery for DoY 193.

Moreover, by observing Table 4.4.4, we found that for rev-LOFO-CV, LRe datasets, and MS imagery, LR did much better than RF, especially at the start of the season. As the season progressed, the performance difference between RF and LR generally decreased. For RGB imagery, the performance difference between LR and RF was larger than MS imagery at the end of the season.

Takeaway. LR had better generalizability than RF when used on imagery and yield data with spatial structure, suggesting that complex models may also overfit to spatial structure in datasets if the spatial dependence is not addressed via spatial CV. LR also

Table 4.4.4: Yield prediction experiments: average R^2 performance of RF and LR over the entire season for LRe and rev-LOFO-CV, and for MS imagery (NIR-based and red-edge-based VIs) and RGB imagery, where RGB = red-green-blue, MS = multispectral, and DoY = day of year.

		DoY																		
Imagery Type	**Model**	**146**	**147**	**148**	**149**	**150**	**151**	**159**	**173**	**179**	**193**	**201**	**216**	**223**	**231**	**237**	**245**	**252**	**257**	**273**
RGB	LR	0.08	0.14	0.19	0.15	0.15	0.14	0.01	0.12	0.20	0.37	0.20	0.07	0.07	0.04	0.02	0.03	0.08	0.12	0.23
RGB	RF	0.02	0.08	0.07	0.08	0.06	0.07	0.01	0.03	0.03	0.15	0.06	0.01	0.03	0.02	0.01	0.02	0.03	0.04	0.08
MS	LR	-	-	-	-	-	-	-	0.31	0.41	0.32	0.34	0.23	0.15	-	0.14	0.11	-	0.09	0.25
MS	RF	-	-	-	-	-	-	-	0.08	0.14	0.11	0.14	0.10	0.07	-	0.06	0.06	-	0.05	0.08

generally did better than RF with LRe datasets, suggesting that LR should be chosen over RF when only satellite imagery is available.

4.4.3.4 High vs. Low Spatial Resolution Imagery Analysis

We examine the performance differences between the HRe vs. LRe datasets in this section for DoY 193 (one of the acquisition dates that led to the best performance for RGB imagery), $NDVI_{RedEdge}$, and ExR, for the various evaluation methods. We can see in Figure 4.4.3 that changes in the imagery's spatial resolution had the most impact on RF. RF did better with HRe than with LRe (especially for KF-CV), suggesting that the good performance of RF-KF-CV-HRe may not have been exclusively the result of overfitting to spatial structure; RF may also have been taking advantage of the higher resolution imagery via the additional features. LR appears to have done slightly better on average with LRe data, except in the case of KF-CV. In particular, the configuration that achieved the best results for DoY 193 using LR involved the LRe data for the rev-LOFO-CV method, suggesting that affordable satellite imagery could be used instead of more expensive UAV imagery.

Takeaway. High spatial resolution imagery obtained from expensive UAV missions is not necessarily required to obtain reasonable yield prediction results. The less expensive and more simple approach of using satellite imagery (RGB or MS imagery) instead of UAV imagery could potentially be applied to obtain reasonable results if a proper VI is chosen and the image acquisition is well timed.

4.4.3.5 Evaluation Method Comparison

In this section the performance difference between standard KF-CV and the spatial CV evaluation methods is examined. We can see in Figure 4.4.3 that RF generally overfitted, while LR was better at generalizing. In fact, for two of the pessimistic evaluation techniques, LOFO-CV and rev-LOFO-CV, LR did comparably well compared to the over-optimistic KF-CV method. This was particularly the case for LRe datasets, where both LOFO-CV and rev-LOFO-CV outperformed KF-CV. However, the higher LR-LOFO-CV and LR-rev-LOFO-CV performance compared to LR-KF-CV could be attributed to the

increased number of training samples per fold compared to the fold size of KF-CV for the smaller fields, since in all cases, both versions of the LR-SpKF-CV did worse than LR-KF-CV, and in these experiments the dataset sizes were similar. This begs the question: to what extent does fold size affect performance?

4.4.3.5.1 LOFO-CV vs. rev-LOFO-CV

- We can see that earlier in the season for the RGB VI (ExR) for LOFO-CV and rev-LOFO-CV in Figures 4.4.1 and 4.4.2, there was no large distinction between both evaluation methods, other than the peak performance achieved for DoY 193 by rev-LOFO-CV; although LOFO-CV did generally better than rev-LOFO-CV later in the season. An observable difference between LR and RF is that negative CC performance was achieved by LR-LOFO-CV later in the season, while RF had positive CC performance.
- For both the $\text{NDVI}_{\text{RedEdge}}$ and ExR, the peak performance achieved was by rev-LOFO-CV.
- There was also some bias that could be introduced in the two types of LOFO-CV experiments, since imagery missions were occasionally conducted 1 to 3 days apart (delayed) from the other fields for some weeks, especially at the end of the season. In fact, LOFO-CV and rev-LOFO-CV did most poorly at the end of the season for both ExR and $\text{NDVI}_{\text{RedEdge}}$, which may be attributed to these delays in field imagery acquisition missions.
- There were also days when one field was missing from the dataset, meaning the two LOFO CV methods (rev-LOFO-CV and LOFO-CV) may have had bias in the results involving experiments with missing fields due to the reduced number of folds.

4.4.3.5.2 LOFO-CV vs. SpKF-CV When observing the LOFO-CV and SpKF-CV methods for ExR, we can see that earlier in the season there was no large difference between the two; that is, LOFO-CV, rev-LOFO-CV, SpKF-CV, and rev-SpKF-CV were relatively similar (although SpKF-CV did do slightly better). For both ExR and $\text{NDVI}_{\text{RedEdge}}$, in the middle of the season, LOFO-CV and rev-LOFO-CV generally did slightly better than SpKF-CV and rev-SpKF-CV, and late in the season, LOFO-CV and rev-LOFO-CV did generally worse than the SpKF-CV and rev-SpKF-CV approaches (especially for ExR), further suggesting the inter-week delays in field imagery acquisition missions negatively impacted the LOFO-CV and rev-LOFO-CV performance. Since LOFO-CV is a farm-level evaluation method and the SpKF-CV is a field-level evaluation method, another possible reason for LOFO-CV doing better than SpKF-CV earlier in the season and doing more poorly than SpKF-CV later in the season is that the early-to-middle and middle-to-late season, for ExR and $\text{NDVI}_{\text{RedEdge}}$, respectively, held spectral information patterns that were strongly tied to potential yield and were present in each of the fields' imagery, while later in the season these yield-reflectance relationships weakened and became field-dependent (e.g., depending on the management practices applied to the field), and had trouble being used by models to be generalized to each field. Note that the comparison between LOFO-CV

and SpKF-CV is not necessarily fair because the fold sizes were not the same. SpKF-CV was more pessimistic because of the smaller fold sizes.

The two types of SpKF-CV methods had larger standard deviation than LOFO-CV and rev-LOFO-CV, probably due to the smaller training dataset size.

4.4.3.5.3 SpKF-CV vs. rev-SpKF-CV From Figures 4.4.1 and 4.4.2, we can see that, for ExR and $NDVI_{RedEdge}$, SpKF-CV generally did better than rev-SpKF-CV for the entire season, except for DoY 211 (and DoY 257 for LR) for $NDVI_{RedEdge}$. This suggests increasing the amount of available training data from a field will increase model performance.

Table 4.4.5: Yield prediction experiments: combined average LR and RF R^2 performance of each field over the entire season for SpKF-CV and rev-SpKF-CV, HRe and LRe, and for MS imagery (NIR-based and red-edge-based VIs) and RGB imagery, where SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, RGB = red-green-blue, and MS = multispectral.

			Field					
Imagery Type	**Imagery Resolution**	**Evaluation Type**	**1**	**2**	**3**	**4**	**9**	**11**
MS	HRe	SpKF-CV	0.24	0.13	0.27	0.17	0.18	0.32
MS	LRe	SpKF-CV	0.19	0.11	0.22	0.12	0.14	0.25
RGB	HRe	SpKF-CV	0.17	0.10	0.17	0.12	0.14	0.19
RGB	LRe	SpKF-CV	0.13	0.09	0.12	0.10	0.10	0.16
MS	HRe	rev-SpKF-CV	0.13	0.13	0.25	0.18	0.05	0.26
MS	LRe	rev-SpKF-CV	0.12	0.14	0.23	0.17	0.03	0.23
RGB	HRe	rev-SpKF-CV	0.08	0.09	0.11	0.09	0.04	0.12
RGB	LRe	rev-SpKF-CV	0.07	0.10	0.09	0.09	0.02	0.12

Table 4.4.5 presents the combined average R^2 performance results of LR and RF for the two types of SpKF-CV experiments for each field. We can see that generally Field 11 imagery led to better performance. This might be because the spatially-wide clusters in Field 11 were wide enough to capture the underlying imagery yield trends, while the smaller fields did not have wide enough clusters to do the same. For SpKF-CV experiments, imagery of Fields 2 and 4 had the worst performance, which may have been due to yield spatial variability reasons; that is, the clusters of low yield areas were mostly found in a single sub-region of each field instead of multiple sub-regions. For rev-SpKF-CV experiments, Field 9 imagery had the worst performance, probably because Field 9 was the smallest field.

Takeaway. Results suggest that using LOFO-CV and rev-LOFO-CV has the advantage of evaluating the extrapolation ability of a model when sampling regions (different fields) are relatively similar, but when the sampling regions start to differ (field imagery that do not share the same acquisition date) these two methods become overly pessimistic and the SpKF-CV should be favored, since the imagery from a single field was typically always taken on the same day (in rare circumstances a field mission may have been split into

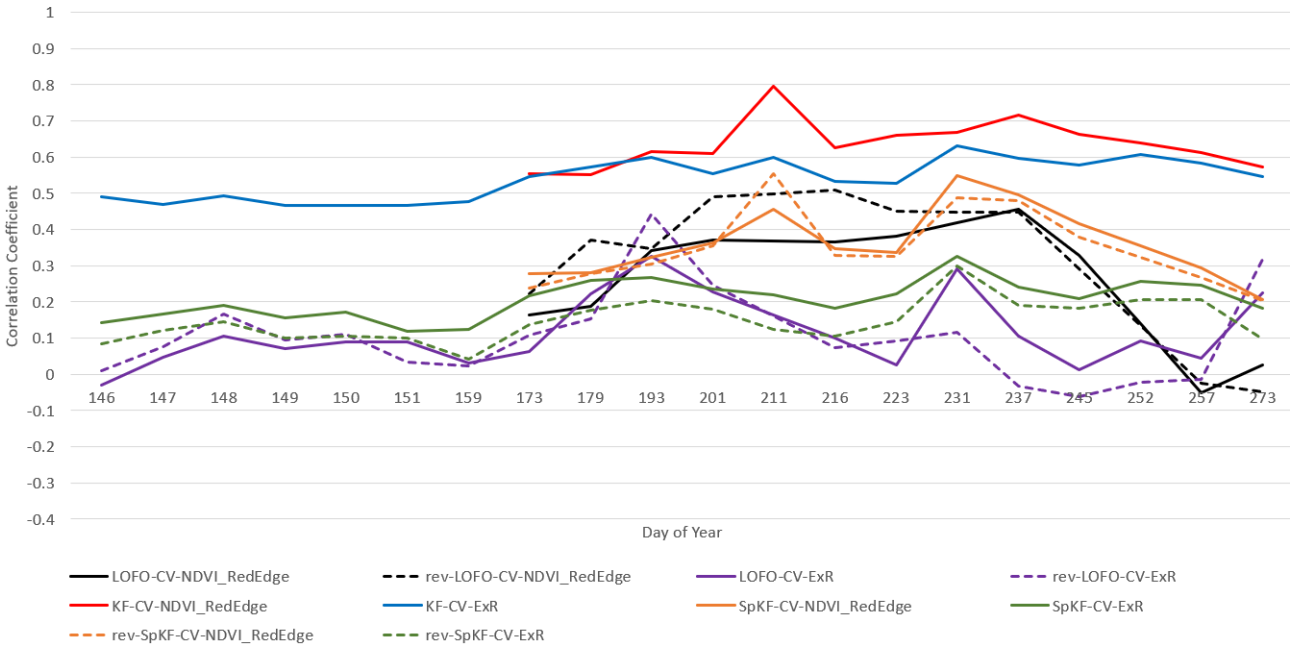


Figure 4.4.1: RF average yield prediction CC performance comparing MS imagery ($\text{NDVI}_{\text{RedEdge}}$) to RGB imagery (ExR), for both LRe and HRe datasets, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, and rev-SpKF-CV = reverse spatial k-fold CV.

two consecutive days due to drone battery issues). However, the SpKF-CV struggled due to having lower training dataset sizes compared to LOFO-CV, especially for the smaller fields, and as a result, SpKF-CV was also over-pessimistic. Therefore, the SpKF-CV may be appropriate when field sizes are sufficiently large, while LOFO-CV would be more appropriate when imagery from multiple smaller fields is available. Using KF-CV alone as an evaluation method is not sufficient to fairly assess the generalizability and extrapolation ability of a model; spatial CV and location-only feature CV evaluation methods should also be used. One should keep in mind that there was bias in the assessment of the extrapolation ability of the models evaluated in the present work using any of the spatial CV methods, since the fields were all from the same farm and shared the same weather conditions.

4.4.3.6 Imagery Acquisition Date Analysis

We can see in Figures 4.4.1 and 4.4.2 that the middle of the season was generally the best time to capture imagery for maximizing yield prediction performance results, whereas early season led to poor performance and late season had generally lower performance.

Takeaway. The middle of the season was the best time to acquire imagery to maximize yield prediction results.

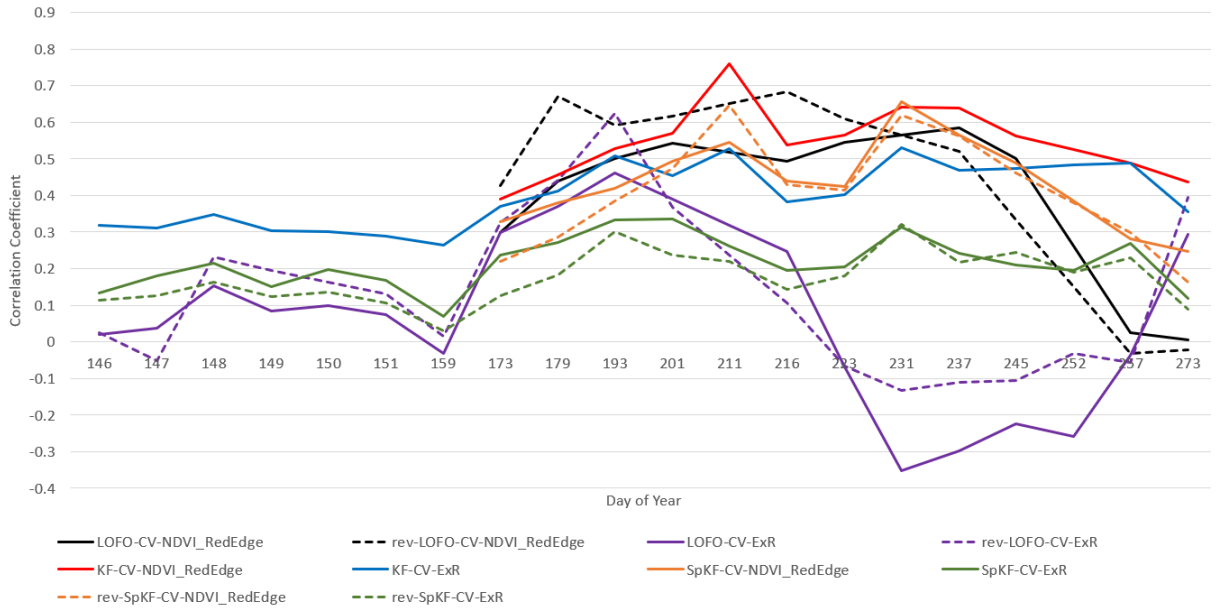


Figure 4.4.2: LR average yield prediction CC performance comparing MS imagery ($\text{NDVI}_{\text{RedEdge}}$) to RGB imagery (ExR), for both LRe and HRe datasets, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, and rev-SpKF-CV = reverse spatial k-fold CV.

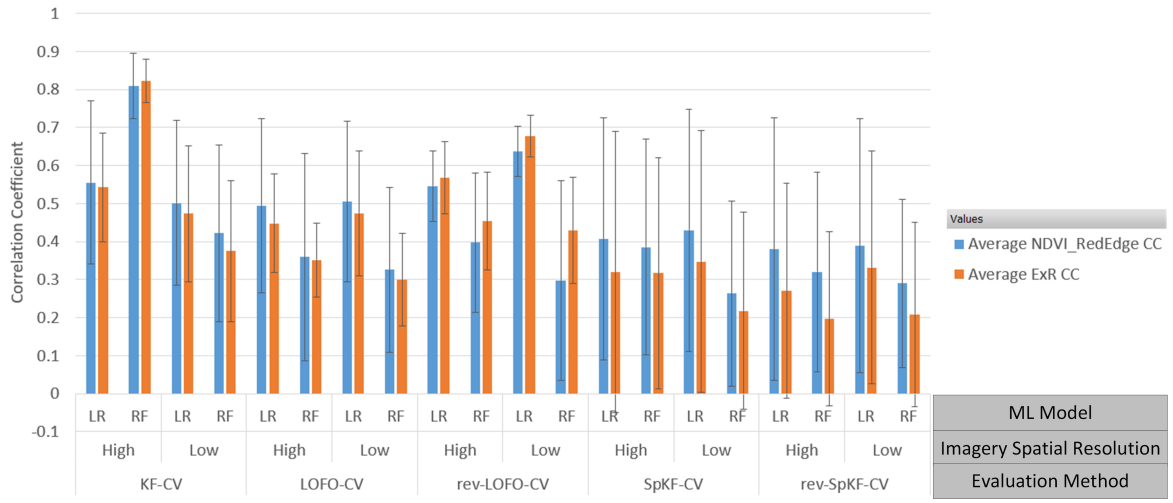
4.4.3.7 Data Processing Time Analysis

An execution time analysis of the image processing and ML models tested in the present work was performed and is explained in this section to illustrate the feasibility of deploying the applied methodology to a live setting. Table 4.4.6 presents the average execution times for the image pre-processing steps. Table 4.4.7 shows the average execution times to create all the ML input datasets for a single DoY. Table 4.4.8 shows the average execution times of the ML experiments.

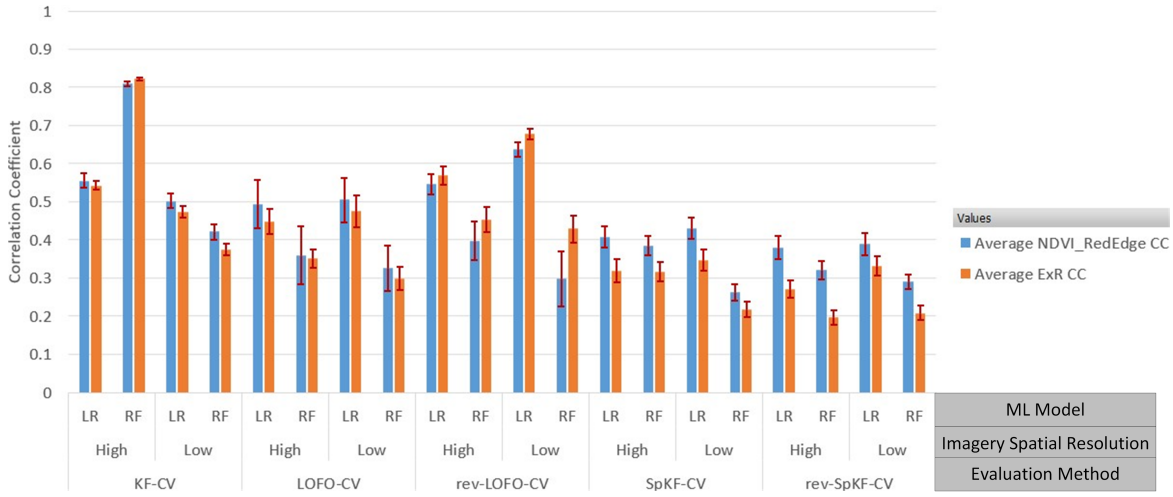
4.4.3.7.1 Imagery Acquisition During an image capture mission, the UAV captured MS images (five images, one for each band) and RGB images (a single image) approximately every 1 s and 2 s, respectively. On average 114.14 Mbps of raw imagery was created during a mission. The average UAV flight duration was 474.6 s (7.9 min) for the smaller fields and 1855.5 s (30.9 min) for Field 11.

4.4.3.7.2 Imagery and Yield Data Pre-processing From the imagery stored on the UAV's SD card, InDro Robotics generated orthomosaics using PIX4Dmapper on a machine with the following specifications: a 64-bit Windows 10 machine with 64 GB of RAM, a 24-core AMD Ryzen Threadripper 3960X CPU @ 3.8 GHz, and a NVIDIA GeForce RTX 3080 Ti GPU. For a single UAV mission:

- For RGB imagery, six orthomosaics were typically created (one for each field). Some-

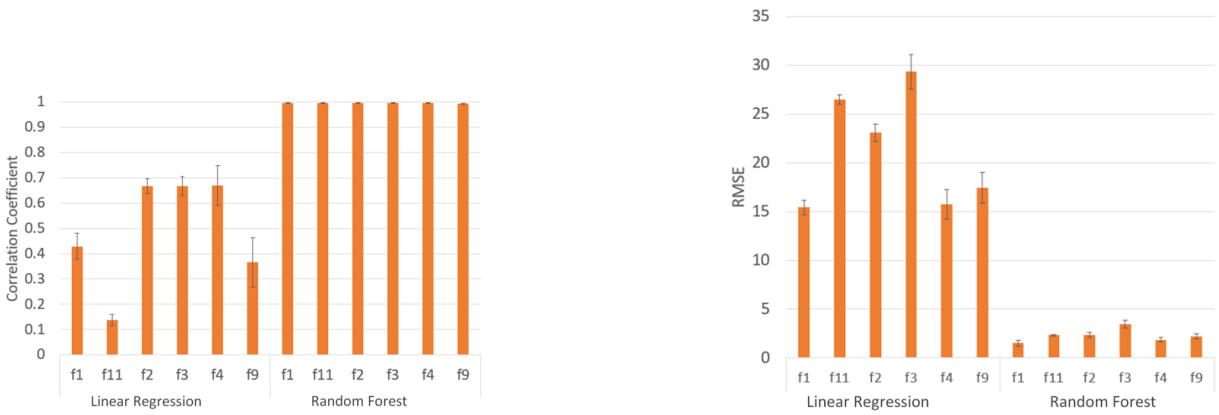


(a) with 1 standard deviation error bars



(b) with 95 % confidence interval error bars

Figure 4.4.3: Average yield prediction model CC performance for DoY 193, MS VI $\text{NDVI}_{\text{RedEdge}}$, RGB VI ExR, and each evaluation method, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b), KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, High = HRe, and Low = LRe.



(a) CC

(b) RMSE

Figure 4.4.4: Yield prediction experiments: location-only features 10-fold CV model performance (CC and RMSE), where the error bars represent 1 standard deviation and fi = Field i.

times Field 11 was split into two orthomosaics for a single UAV mission (this was done for five missions from August onward), so the orthomosaic with the most imagery and the least amount of noise was chosen for further processing for that DoY. On average, it took 2.4 h and 3.1 h to generate orthomosaics for the smaller fields and Field 11, respectively.

- For MS imagery, twelve ($2\times(5+1)$) orthomosaics were typically created, one for each of the five bands and one for NDVI , and two sets of six orthomosaics (one set for Field 11 and another set for the smaller fields) were generated. On average (ignoring a 74.2 h outlier), it took 6.8 h and 4.8 h to generate the six single-band orthomosaics for the smaller fields and Field 11, respectively.

The orthomosaic creation process also included 3D surface model creation and a PIX4Dmapper output report. After being delivered via an external hard drive at the end of the 2021 growing season, the orthomosaics were then processed using a machine with the following specifications: a 64-bit Windows 10 desktop with 32 GB of RAM, an 8-core Intel Core i7 CPU @ 3.50 GHz, and a 1 TB solid state drive (SSD). For the execution time analysis discussed next in this section, three DoYs were randomly sampled for MS imagery and another three DoYs were randomly sampled for RGB imagery; the execution times were approximated by taking the average execution time over those samples. Table 4.4.6 shows these results, where the smaller fields execution time is the average of the smaller fields. The cropping of the orthomosaics was done using QGIS, where fields were cropped into multiple overlapping tiles to avoid reading entire orthomosaics into memory (this was especially problematic with Field 11) when applying a mean-, maximum-, and minimum-filter to fuse the imagery to the yield data. Fields 1, 2, 3, 4, 9, and 11 were vertically partitioned into 5, 6, 4, 6, 5, and 21 tiles, respectively. Some RGB orthomosaics were too big for further processing, so their spatial resolution was lowered to 1.149 cm. The cropped orthomosaics tiles were then processed by the R programming language version 4.1.3 to create VI rasters for each tile. The *Rcpp* package version 1.0.9 was used to enable the use of C++ to calculate VI rasters in a memory-efficient manner.

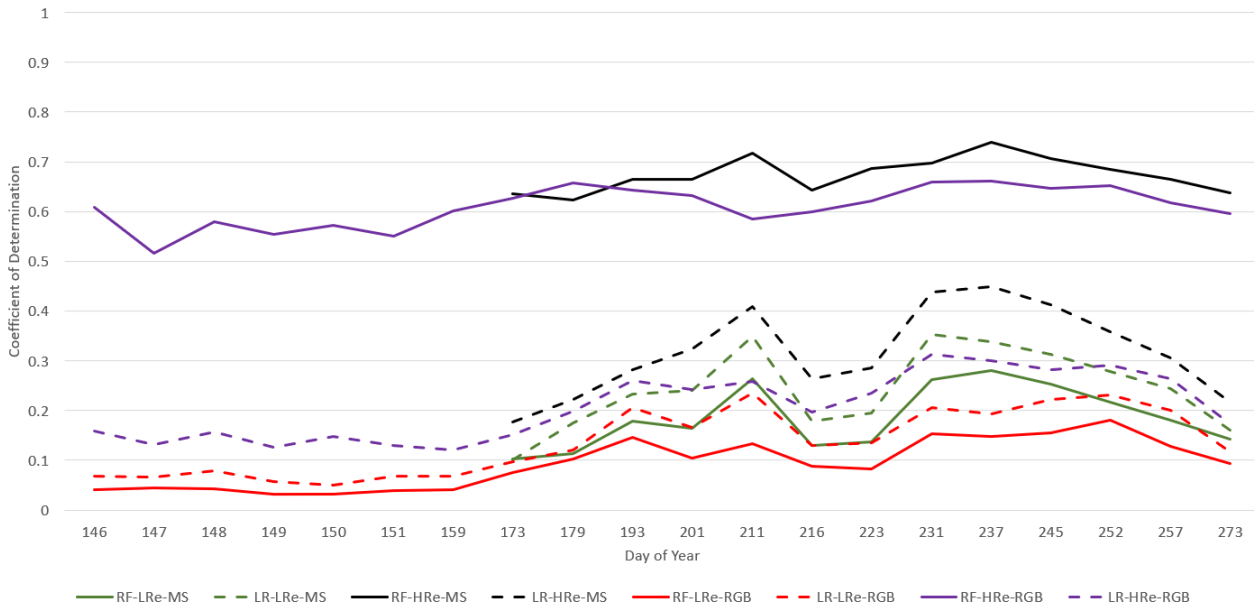


Figure 4.4.5: Yield prediction experiments: KF-CV average R^2 performance over the growing season for all fields and VIs, where MS = multispectral, RGB = red-green-blue, LRe = low spatial resolution dataset, and HRe = high spatial resolution dataset.

The data fusion of the imagery and yield was done using Java version 18.0.1, where the cropped orthomosaic tiles were progressively read to fuse each yield sample to its corresponding pixel neighbourhood, where pixels outside field boundaries were ignored. Once fused, the resulting dataset contained every feature and had to be split into smaller HRe and LRe single-VI feature datasets for each fold and for each iteration of the CV experiments using a combination of Java and R programming languages. The average total execution time for all VIs for each evaluation method type (averaged over both HRe and LRe) of this final dataset splitting step (before feeding the datasets to the ML models) can be found in Table 4.4.7. For location-only CV experiments, there was no dataset splitting required: we just used the interpolated yield dataset and fed it to Weka.

Takeaway. For orthomosaic generation, on average, orthomosaics with the largest area (greatest number of input images to be stitched together) took the longest execution time. For VI creation and data fusion, Field 11 took the longest execution time compared to the smaller fields. For splitting the datasets, the HRe datasets generally took longer than LRe, probably because HRe had two additional features. The MS imagery took longer than the RGB imagery when splitting the data. This may be explained by the fact that the MS imagery was generally less noisy than the RGB imagery, meaning more samples were found in the MS datasets. The dataset splitting involving clustering took the longest, which was likely due to the added computational complexity of the clustering step.

4.4.3.7.3 Machine Learning For the execution time analysis discussed in this section, three DoYs were randomly sampled for MS imagery and another three DoYs were randomly sampled for RGB imagery, and the execution times were approximated by taking the

Table 4.4.6: Yield prediction experiments: average (sampled over three acquisition dates) image pre-processing execution time (min = minute and h = hour), where VI = vegetation index, RGB = red-greed-blue, MS = multispectral, and smaller fields = Fields 1 to 9.

Imagery Processing Step	**Imagery Type**	**Smaller Fields Execution Time (min)**	**Field 11 Execution Time (h)**
Orthomosaic Cropping	RGB	2	0.1
Orthomosaic Cropping	MS	2	0.3
VI Raster Creation	RGB	6	5.6
VI Raster Creation	MS	12	1.1
Yield + Imagery Data Fusion	RGB	72	5.5
Yield + Imagery Data Fusion	MS	30	2.1

Table 4.4.7: Yield prediction experiments: average (sampled over three acquisition dates) execution time (min = minute) to split and create all datasets for all VIs and fields, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, RGB = red-greed-blue, and MS = multispectral.

Evaluation Method	**Imagery Type**	**LRe Execution Time (min)**	**HRe Execution Time (min)**
KF-CV	RGB	0.1	0.1
KF-CV	MS	0.1	0.1
LOFO-CV and rev-LOFO-CV	RGB	1.0	1.5
LOFO-CV and rev-LOFO-CV	MS	1.2	2.0
SpKF-CV and rev-SpKF-CV	RGB	5.6	8.8
SpKF-CV and rev-SpKF-CV	MS	11.9	20.2

Table 4.4.8: Yield prediction experiments: average (sampled over three acquisition dates and five VIs) execution times (s = second and m = minute) for ML experiments, where KF-CV = k-fold CV, LOFO-CV = leave-one-field-out CV, rev-LOFO-CV = leave-all-but-one-field-out CV, SpKF-CV = spatial k-fold CV, rev-SpKF-CV = reverse spatial k-fold CV, location-only = only location features used, LRe = low spatial resolution dataset, HRe = high spatial resolution dataset, and N/A = not applicable.

CV Type	Resolution	Field Size	LR Execution Time (s)	RF Execution Time (min)
KF-CV	HRe	big	1	6.0
KF-CV	HRe	small	1	1.3
KF-CV	LRe	big	1	6.5
KF-CV	LRe	small	<1	1.2
location-only	N/A	big	<1	1.4
location-only	N/A	small	<1	0.2
LOFO-CV	HRe	all	<1	0.3
LOFO-CV	LRe	all	<1	0.4
rev-LOFO-CV	HRe	all	<1	0.1
rev-LOFO-CV	LRe	all	<1	0.1
rev-SpKF-CV	HRe	big	5	1.6
rev-SpKF-CV	HRe	small	2	0.2
rev-SpKF-CV	LRe	big	2	1.3
rev-SpKF-CV	LRe	small	1	0.2
SpKF-CV	HRe	big	3	5.0
SpKF-CV	HRe	small	2	1.1
SpKF-CV	LRe	big	2	5.1
SpKF-CV	LRe	small	1	1.2

average execution time over those samples. In this analysis, the models were trained and evaluated using a desktop machine with the following specifications: a 64-bit Windows 10 desktop with 48 GB of RAM, and a 12-core Intel Core i7-12700 CPU @ 2.10 GHz.

The average ML execution time, over a sample of five VIs, is shown in Table 4.4.8, where an experiment in this case consists of all 10 iterations and folds.

Takeaway. Computations involving Field 11 datasets took longer than those involving smaller field datasets. LR experiments were much faster than RF experiments. rev-LOFO-CV and rev-SpKF-CV were faster than LOFO-CV and SpKF-CV, respectively, probably because training took longer than testing and there were less training data in the reverse CV methods. The LOFO-CV and rev-LOFO-CV methods were among the fastest methods, probably because they had fewer folds than the other methods.

4.4.4 Discussion

In this section we compare our results to the results of similar yield prediction studies. Sapkota et al. [231] found that using UAV-based imagery led to better corn yield prediction results than using satellite-based imagery, which is similar to our findings that suggest that higher spatial resolution imagery may lead to better model performance. In similar corn

yield prediction studies that used KF-CV, Guo et al. [93], Baio et al. [18], and Ramos et al. [215] also found that RF was the best-performing model. There are some studies where RF was not found to be the best model: Fan et al. [67] found that ridge regression outperformed the RF model, where multi-temporal VI features and multiple VIs were fed as input to their ML models, and Guo et al. [92] found SVM to be one of the better-performing models for corn yield prediction. Our hypothesis discussed in Section 4.4.3.4 that RF may have obtained better performance in the RF-KF-CV-HRe experiments due to the additional number of features is supported by the results of two studies [103, 141] that found that additional imagery features led to better model performance. Herrmann et al. [103] found that VI-based models performed generally worse than PLSR models that included all MS imagery bands. Kumar et al. [141] found that typically one to two VI features (those most correlated with yield) led to the best performance and that adding more VIs as features generally did not improve yield prediction model performance.

We proceed to discuss the works that found similar VI performance results as those discussed in the present work. Sunoj et al. [251] also found that MS VIs performed better than RGB VIs for corn yield prediction. Our findings showed that OSAVI and SAVI were among the best-performing VIs in early season. This is consistent with other studies that found that OSAVI does well earlier in the season [21]. This performance trend aligns with the fact that SAVI and OSAVI [223] were designed to reduce the unwanted influence of soil-background reflectance (which is more prevalent in early season) on VIs [21] such as $SR_{NIR,Red}$ and NDVI [107], although there are studies that found that OSAVI does not perform well in early growth stages [53]. Furthermore, the good NDVI performance (in early season) we observed in the present work is consistent with the literature: many studies [251, 76, 215, 53] have found NDVI to be among the most important VI for yield prediction. The good performance we found with $NDVI_{Green}$ is also consistent with other yield prediction literature that found $NDVI_{Green}$ to be among their top-performing VIs for corn yield prediction [215, 282, 141, 53]. Barzin et al. [21] found the $NDVI_{Green}$ to best perform during the V6–7 stages of corn growth. Our findings that the red-edge imagery tended to lead to good model performance are consistent with other studies. Herrmann et al. [103] found that the bands and VIs in the red-edge spectral region led to the best yield prediction model performance. Kumar et al. [141] also found the red-edge band to enable good corn yield prediction model performance. Furthermore, our observation that red-edge-based VIs did better from middle to late season aligns with claims made in the literature [175, 140]. $NDVI_{RedEdge}$ was found to be the best-performing VI, in Canata et al. [37], for sugarcane yield prediction, and was among the best-performing VIs in other corn yield prediction studies [282, 103, 215]; Barzin et al. [21] found the $NDVI_{RedEdge}$ to best perform during the V6–7 stages of corn growth.

We will now discuss the effects of image acquisition date on model performance. Bose et al. [31] found that imagery from the middle of the growing season led to better yield prediction results for winter wheat. Yang et al. [282] found that mid-season imagery (in particular, during the milking growth stage) produced the best corn yield prediction results. Sunoj et al. [251] found the R4 growth stage (which would be considered late season in the present work) to be the best time to acquire imagery to maximize performance, although a reliable performance was still obtained for most of the mid season, especially when using

imagery acquired after the R1 growth stage. Guo et al. [93] found that mid-season imagery, especially after the tasselling stage (VT), was the acquisition period that led to the best corn yield model prediction performance. Fan et al. [67] found the best performance to be during the VT stage. Oglesby et al. [191] found the VT and R1 growth stages to produce the best results, although there are some studies that found that imagery from late season produced the best results [297]. Poor early season performance was also found in other corn yield prediction studies [278, 93, 67, 92]. Saravia et al. [232] found that imagery from the reproductive growth stage (mid season) had the most correlation with yield. Sunoj et al. [251] also found that, at the very end of the season (R5), yield prediction performance was considerably lower. The trend that the middle and sometimes late season imagery leads to better model prediction performance compared to early season imagery may be explained by the fact that although corn can be stressed by drought in both early and reproductive growth stages, stress in the reproductive stages of corn can reduce yield, whereas early-stage stress may have less of an impact on yield [103].

4.4.5 Conclusion

In the present work we performed a yield prediction study using UAV-based RGB and MS imagery and using yield data. Most UAV imagery-based yield prediction studies only apply KF-CV for model evaluation, which tends to lead to over-optimistic results. We applied various spatial generalizability evaluation techniques to showcase that over-optimistic yield prediction model performance results may be obtained and that the underestimation of errors can be avoided if a proper spatial data analysis methodology is applied. We cleaned and interpolated yield data, computed VIs from the imagery, and fused the yield and imagery (55 VIs + 5 raw-bands) data together to produce two types of datasets: LRe and HRe. LRe datasets only had mean reflectance values, whereas HRe datasets had the mean, maximum, and minimum reflectance values around a yield point. The fused data were used as input to train and evaluate RF and LR regression models using 10 iterations of standard 10-fold CV and spatial CV methods. In addition, a dataset with only location features was used to train and evaluate the two models via standard 10-fold CV. This was performed to have a baseline for spatial analysis and to examine the effects of SA on model yield prediction performance. We found that the middle of the season was the best time to acquire imagery. MS imagery provided better results than RGB imagery, especially when the red-edge band was available. We found that the best-performing VIs were $SRI_{NIR,RedEdge}$, $NDVI_{RedEdge}$, and NG. In fact, the red-edge band's raw reflectance alone was able to produce reasonable prediction results compared to the other VIs, so computational resources could be saved by skipping the VI calculation step and using the red-edge raw-band instead of a VI. Although MS imagery generally did better than RGB imagery, if the image acquisition date was well timed (e.g., DoY 193) and a good RGB VI choice was made (e.g., ExR), the RGB imagery's yield prediction performance was able to compete with an MS camera. In terms of the difference between UAV imagery and satellite imagery (simulated by comparing the model performance when using high spatial resolution vs. low spatial resolution datasets), the choice of the imagery's spatial resolution most impacted RF, who did best with high spatial resolution imagery, whereas LR was less

impacted by the spatial resolution of the imagery and did best with low spatial resolution imagery. This suggests that the choice of the model could be made based on the spatial resolution of the available imagery, and input costs could be reduced by favoring satellite imagery and choosing LR, since LR had better generalizability than RF. The effects of spatial structure (for example, SA) on performance depended on the chosen model; that is, RF did not generalize well and was overfitting to the spatial structure in the location-only and high spatial resolution imagery datasets, whereas LR generalized better in the presence of spatial structure in the data. The spatial generalizability experiments performed in the present work are important because they revealed that over-optimistic model performance may be obtained if an analyst is not careful when performing ML on yield and UAV imagery data. Improper management or economic decisions may be made when one relies on an over-optimistic model, and these decisions could lead to economic loss or other damages (e.g., reduced crop yield). Therefore, identifying over-optimistic models using spatial CV and estimating their true extrapolation performance should be considered when predicting yield using UAV imagery. We hope these findings can help guide the yield prediction community towards careful model evaluation when working with spatially autocorrelated agri-data. A use case of the present work involves deploying models, which are trained on imagery and yield data for each week from previous growing seasons, to a SFS such that a farmer could a) view weekly yield prediction maps by feeding the system with weekly crop imagery, b) address the low yield areas, and c) examine the precision maps generated in the upcoming weeks to confirm that the issue causing low yield has been addressed by actions conducted in step b).

Future work involves: a) reproducing the present work's results using additional UAV imagery from Area X.O and actual satellite imagery instead of simulated satellite imagery, b) including ZeroR as additional benchmark to give more insight into how poorly models are performing[8], c) evaluating additional ML models[8], especially spatially aware models such as generalized least squares [100], d) evaluating and examining the effects on performance of applying various spatial sampling schemes (systematic random, simple random, and clustered random [273]) before applying CV, e) performing hyperparameter tuning[8], f) combining features from multiple DoYs in the datasets to add a temporal dimension to the study[9], g) performing a temporal generalizability analysis[9], h) investigating the effects of the number of features on RF performance[10], i) expanding on the VI ranking analysis by splitting the mid season into two parts, j) exploring multi-band/VI feature models for yield prediction[8], and k) plotting and analyzing charts similar to Figures 4.4.1 and 4.4.2 that only involve Field 11 data.

[8] We explored this in the studies described in Sections 4.5 and 4.6.

[9] We explored this in the study described in Section 4.6.

[10] We explored this in the study described in Section 4.5.

4.5 Yield Prediction Study: Using UAV Imagery, Spatial Cross-validation, and Model Selection

Most of the content of this study was published in Killeen et al. [134].

The present work is an UAV imagery-based corn yield prediction study that analyzed the effects of image spatial and spectral resolution on model performance, and compared the prediction power of two VIs and 5 spectral bands for a single image acquisition date. CL was used to train and evaluate models. In this study we included deep learning models and hyperparameter tuning. We also explored various spatial CV techniques and their effects on hyperparameter selection and model performance.

The objective of the present work was to determine: a) what is the best performing imagery-based yield prediction model; b) whether satellite imagery can be used instead of UAV-based imagery to achieve comparable performance at a reduced price; c) whether an inexpensive RGB camera can be used instead of a costly MS camera; and d) whether single-VI feature datasets can achieve comparable performance to multi-band feature datasets.

In terms of how the present work relates to this thesis: the spatial analysis methodologies applied in the present work tie into the spatiotemporal data analysis aspect of this thesis. The hyperparameter tuning and spatial CV techniques applied allowed us to more efficiently decide on how hyperparameters would be tuned and what spatial CV techniques to apply in our yield prediction study presented next in this chapter in Section 4.6.

The following is a list of limitations of the present work: a) the data gathered was from only a single study site, b) data from only a single field was included in the study, c) only mono-temporal datasets were used to evaluate models, d) we did not evaluate CNN models, which are an appropriate model choice for analyzing spatial data, e) no feature selection was explicitly performed, and f) UAV imagery and yield datasets were both available for only a single year, so cross-year extrapolation experiments could not be conducted.

4.5.1 Materials and Methods

4.5.1.1 Datasets

The 2021 UAV imagery dataset was used in the present work such that only imagery gathered from a) the MS camera, b) a single field (Field 11), and c) a single imagery acquisition date (DoY 201) was used. Note that since we used UAV imagery only from the MS camera in the present work, when we refer to RGB imagery, we are referring to imagery features derived from only the red, green, and blue bands of the MS camera; we refer to MS imagery when the imagery features were derived using the NIR and/or red-edge bands.

4.5.1.2 Feature Extraction

In total two VIs, $\text{NDVI}_{\text{RedEdge}}$ and ExR (which are listed in Appendix A in Tables A.3 and A.2, respectively), were chosen, and 5 raw-bands (red, green, blue, NIR, and red-edge)

were chosen as the spectral channels to use in this study. We used the same steps as described in Section 4.4.1.3 to a) clean and interpolate Field 11 yield data to a 2.5 m × 2.5 m grid and b) fuse the yield and imagery into HRe and LRe datasets. We then derived an additional nine datasets from the results of the data fusion step. We created eight datasets by varying each combination of imagery spatial resolution (HRe vs. LRe), imagery type (MS vs. RGB), and multi-channel ({red, green, blue, NIR, red-edge} for MS imagery and {red, green, blue} for RGB imagery) vs. single-channel VI ($NDVI_{RedEdge}$ for MS imagery and ExR for RGB imagery) features. We will refer to these datasets using the following notation: (imagery resolution)-(imagery type)-(number of channels); for example, a) the HRe-MS-all-bands dataset has mean, maximum, and minimum reflectance features from all five bands (totaling 15 features) and b) the LRe-RGB-single-VI dataset has the mean reflectance feature from the ExR VI (totaling 1 feature). The ninth dataset, denoted as location-only, was created where the yield dataset's coordinates were used as features, for a total of 2 features. This dataset was used to explore the extent of the effects SA may have on the conducted yield prediction experiments, as suggested by Ploton et al. [206].

4.5.2 Experiments

We used the output of the data fusion and data pre-processing steps to train and evaluate the ML models using various forms of CV for each of the nine datasets. Our goal was to evaluate the accuracy of the predictions made by the models and the execution time of the models. The evaluation metrics used are RMSE and R^2. The ML models used are ZeroR, MLP, RF, RF without hyperparameter tuning (RF-no-tune), and LR. RF and LR models were chosen for the same reasons described in Section 4.4.2 (i.e., since RF and LR have been found to perform well for yield prediction). We chose MLP, since a) we wanted to include a deep learning model in our experiments (we did not include deep learning models in our previous study described in Section 4.4) and b) it is one of the classic deep learning models. We chose ZeroR as an additional benchmark to give us more insight into how poorly models were performing. For the MLP model, the yield and features were normalized between 0 and 1 using min-max scaling.

The models were implemented using Python, where the MLP model was implemented using *TensorFlow* version 2.13.0, *CUDA* version 11.8, and *cuDNN* version 8.6, while LR and RF were implemented using *sklearn* version 1.2.2. The models were trained and evaluated using a machine that had the following specifications: a 64-bit Ubuntu 20.04 desktop with 48 GB of RAM, a 12-core Intel Core i7-12700 CPU @ 2.10 GHz, an NVIDIA GeForce RTX 3060 graphics card with 12 GB of RAM, and a 2 TB SSD.

We ran three types of nested CV experiments (inspired from Schratz et al. [234]):

- **non-spatial/non-spatial**: both the training-testing set and training-validation set splits were done via random sampling (i.e., nested KF-CV).
- **spatial/non-spatial**: the training-testing set split was done using k-means clustering to create spatially disjointed folds (i.e., block CV) and the training-validation set split was done via random sampling (i.e., KF-CV).

- **spatial/spatial**: both the training-testing set and the training-validation set splits were done using k-means clustering to create spatially disjointed folds (i.e., nested block CV).

Note that we refer to both the spatial/non-spatial and spatial/spatial experiments as the spatial CV experiments.

4.5.2.1 Model Selection

In this section we discuss the model selection approaches we applied in the present work. For all three types of nested CV experiments, we used a single iteration of nested 10-5-fold CV (see Section 4.3.5.2 for more details) to perform model selection. We used 50 trials and 15 trials of random search for RF and MLP, respectively, using a single execution per trial. We used a smaller number of trials for MLP for time constraint reasons, given that MLP took much longer to run than RF. This should not have introduced much bias, since we found in the present work that RF could provide consistent performance regardless of hyperparameter choices. The hyperparameter search spaces for RF and MLP (including the MLP neural network architecture) are described in Appendix C in Tables C.1 and C.3, respectively. For LR, due to a lack of hyperparameters that can be tuned, we instead performed feature selection by using an exhaustive search approach that considered all possible subsets of feature choices[11]. The RF-no-tune model had the following hyperparameters: number of trees = 100; unlimited maximum depth; minimum samples in leaf = 1; number of candidate attributes = $\texttt{max}\left(0, \lfloor \log_2 \left(\textit{number of attributes}\right) \rfloor\right) + 1$; and no bootstrap sampling.

4.5.3 Analysis and Results

In this section, we analyze the results of our experiments and present key findings.

Table 4.5.1 displays the performance results of ZeroR, where there were only three different result sets, since regardless of imagery type, imagery resolution, or number of features, ZeroR predicted the same thing. Figure 4.5.1 and Table 4.5.2 show the results of only including a single feature (LRe datasets) from a single VI, where for presentation clarity reasons, spatial/non-spatial and spatial/spatial experiment performance results were averaged, because there was no large visual distinction between the results of both types of experiments. Table 4.5.3 shows the results of applying non-spatial/non-spatial experiments using the location-only dataset. Figure 4.5.2 shows the results of only including the HRe features from a single VI. Figures 4.5.3 and 4.5.4 show the results of including multiple raw-band features and can be compared to show the differences between using LRe and HRe datasets.

[11] This was computationally feasible, since datasets only had up to 15 features.

Table 4.5.1: Yield prediction experiments: average ZeroR baseline model performance, where the data shown are mean ± standard deviation.

Evaluation Method	**R^2**	**RMSE**
non-spatial/non-spatial	0.00±0.00	22.43±0.31
spatial/non-spatial	0.00±0.00	22.00±3.56
spatial/spatial	0.00±0.00	22.02±3.56

Table 4.5.2: Yield prediction experiments: average model R^2 performance for LRe-single-VI dataset experiments, where the data shown are mean ± standard deviation and experiments with zero R^2 were omitted.

Algorithm	**Training-Testing Split**	**Imagery Type**	**R^2**
RF	non-spatial	MS	0.41±0.02
RF	non-spatial	RGB	0.37±0.02
RF	spatial	MS	0.31±0.14
RF	spatial	RGB	0.27±0.16
RF-no-tune	spatial	MS	0.03±0.06
RF-no-tune	spatial	RGB	0.02±0.06

4.5.3.1 Small vs. Large Number of Features

In this section we examine the effects on performance from varying the number of features. By comparing Figures 4.5.1 and 4.5.2, and Table 4.5.2, we can see that with two additional features (minimum and maximum reflectance), RF-no-tune, MLP, and LR recover from their poor performance. Furthermore, the spatial CV experiment results show that HRe features led to better performance than LRe features, although Figure 4.5.3 shows that reasonable results could still be obtained from lower spatial resolution imagery, if multiple bands were included as features of a model. Herrmann et al. [103] also found that VI-based models performed generally worse than models that included many MS imagery bands. Nevertheless, by comparing Figures 4.5.3 and 4.5.4, we can observe that increasing the spatial resolution of the imagery for multi-band datasets led to better model performance for all the models. This suggests that satellite imagery can be used to obtain reasonable results, but UAV imagery should be favored to maximize performance. By observing Figures 4.5.3 and 4.5.4, we can see that for RF, RF-no-tune, and LR, performance was worse for LRe-all-bands datasets than HRe-all-bands datasets for the spatial CV experiments, while the opposite was the case for MLP. This difference may be explained as follows: a) MLP has the potential to automatically derive new VIs via its ability to perform feature extraction, while LR and RF do not have this ability, and b) having less spatial structure in the features (by removing minimum and maximum reflectance) reduces the opportunity for the model to overfit to the spatial structure, forcing spectral patterns to be learned instead.

Takeaway. Higher spatial resolution imagery led to generally better model performance than lower spatial resolution imagery, suggesting that UAV imagery should be favored over satellite imagery; although reasonable performance should still be possible us-

ing satellite imagery. Furthermore, multi-band feature datasets outperformed the single-VI feature datasets.

4.5.3.2 RGB vs. MS Imagery

In this section, we compare the performance differences between MS and RGB imagery. By observing Figure 4.5.1 and Table 4.5.2, we can see that for RF, the MS imagery ($NDVI_{RedEdge}$) slightly outperformed the RGB imagery (ExR), while in Figure 4.5.2, RGB imagery did slightly better than MS imagery for most of the experiments. These observations suggest that with low spectral resolution imagery, we can expect MS imagery to outperform RGB imagery when the imagery's spatial resolution is low, but when the imagery's spatial resolution is higher, we can expect RGB imagery to outperform MS imagery. When multiple bands were available, by observing Figures 4.5.1 and 4.5.3, and Table 4.5.2, we can see that RGB imagery generally did worse than MS imagery and that higher spectral resolution imagery led to considerably better performance for all the models.

Takeaway. MS imagery generally led to better model performance than RGB imagery. Datasets with multi-band features will likely achieve better performance than single-VI feature datasets. Nevertheless, RGB imagery could still be used to obtain reasonable performance.

4.5.3.3 Model Spatial Generalizability

In this section we evaluate the spatial generalizability of the ML models. By examining Figures 4.5.1–4.5.4, and Table 4.5.2, we can see that the non-spatial/non-spatial experiments tended to do better and tended to have a smaller standard deviation than the spatial CV experiments, which is expected, since non-spatial/non-spatial experiments were over-optimistic in the present work. In Table 4.5.3 we can see that RF performed nearly perfectly and was clearly taking advantage of the spatial structure in the location-only dataset. Even without hyperparameter tuning, RF did very well. MLP did not do as well as RF and RF-no-tune, but it still did reasonably well. On the other hand, LR performed very poorly, and in Table 4.5.1, we can see that LR did almost as poorly as the baseline model ZeroR. Figure 4.5.2 shows that RF and RF-no-tune did well in the over-optimistic non-spatial/non-spatial case, but had considerably lower performance in the spatial CV cases. This observation combined with the fact that a) RF and RF-no-tune obtained excellent performance for the non-spatial/non-spatial location-only dataset experiment and b) HRe datasets had more underlying spatial structure by design than LRe, suggests that RF and RF-no-tune were overfitting to the spatial structure found in the input datasets. Furthermore, Figures 4.5.1–4.5.4 and Tables 4.5.2 and 4.5.3 for the non-spatial/non-spatial experiments (other than those using the LRe-single-VI datasets), show that hyperparameter tuning had little to no impact on RF's performance. This suggests that in the presence of spatial structure in the input dataset, RF hyperparameter tuning is not necessary to obtain good performance. Figure 4.5.3 shows the case where generally the difference between the pessimistic spatial CV results and the over-optimistic non-spatial/non-spatial results

Table 4.5.3: Yield prediction experiments: location-only non-spatial/non-spatial experiment results, where the data shown are mean ± standard deviation.

Algorithm	**R^2**	**RMSE**
LR	0.02±0.01	22.19±0.27
MLP	0.82±0.05	9.48±1.20
RF	0.98±0.01	3.31±0.55
RF-no-tune	0.96±0.00	4.30±0.15

was the smallest, suggesting that the LRe-all-bands datasets offered the most insight into what the true model extrapolation (making predictions for new locations) performance was. This also suggests favoring the addition of new raw-bands to a dataset instead of deriving additional areal[12]-based features (such as maximum- and minimum-filters) from the same raw-bands.

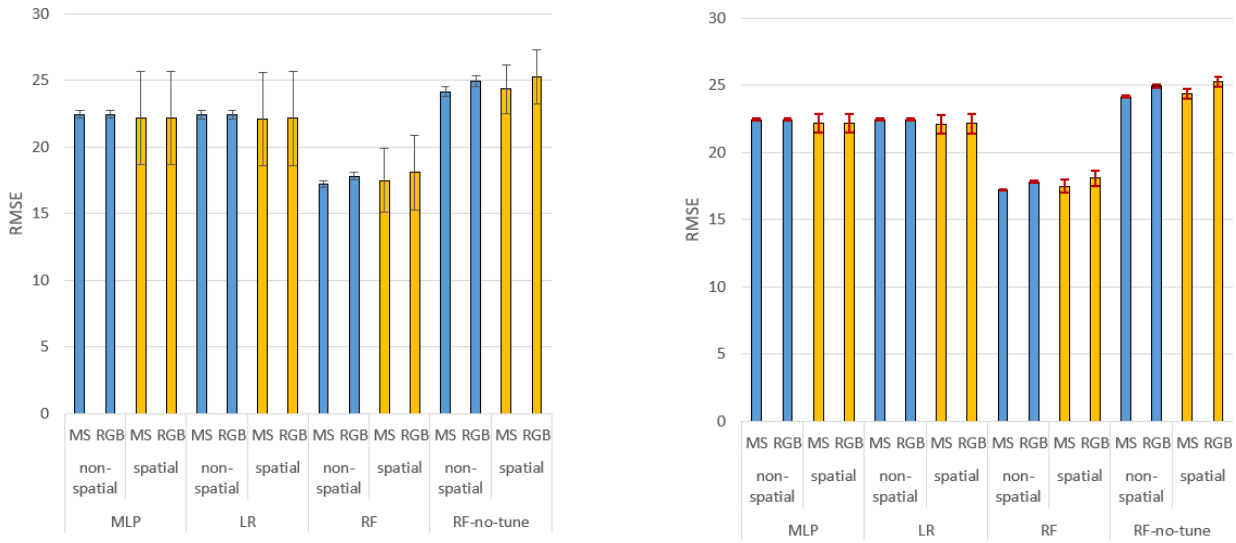
The following are the cases with the most noticeable performance differences for spatial/non-spatial vs. spatial/spatial experiments: a) RF-HRe-single-VI shown in Figure 4.5.2, where spatial/spatial outperformed spatial/non-spatial, b) MLP-HRe-all-bands shown in Figure 4.5.4, where spatial/spatial outperformed spatial/non-spatial, c) LR-HRe-all-bands shown in Figure 4.5.4, where spatial/non-spatial outperformed spatial/spatial, and d) LR-LRe-all-bands shown in Figure 4.5.3, where spatial/non-spatial outperformed spatial/spatial. These above cases suggest that for the MLP and RF models, hyperparameter optimization performed on a spatially disjointed validation and training datasets reduced overfitting to the spatial structure and increased model extrapolation ability to new unseen locations (represented by the testing dataset); while for LR, LR may have done better on spatial/non-spatial, because the model already resisted the bias effects invoked by spatial structure in the data, given LR's poor performance in Table 4.5.3; so, the spatial split over the training-validation datasets may be unnecessary and over-pessimistic in the spatial/spatial case.

Takeaway. Spatial CV should be used in addition to non-spatial/non-spatial CV to get a better understanding of model performance and extrapolation performance to new locations, and spatial CV should be considered when performing feature selection and hyperparameter tuning. Furthermore, the choice of RF's hyperparameters had less of an impact on performance when spatial structure existed in the input dataset.

4.5.3.4 Model Comparison

This section compares the performance of the ML models. Tables 4.5.1 and 4.5.2, and Figure 4.5.1 show that LR, MLP, and RF-no-tune performed no better than the ZeroR baseline when a single-VI feature was used. In fact, RF-no-tune had worse RMSE performance than ZeroR. On the other hand, RF did better than the other models even in the more pessimistic cases that used spatial CV. In Figures 4.5.3 and 4.5.4, we can see that LR generally outperformed MLP using the HRe-MS-all-bands dataset for spatial CV

[12]involving (or relative to) an area or region of interest



(a) with 1 standard deviation error bars

(b) with 95 % confidence interval error bars

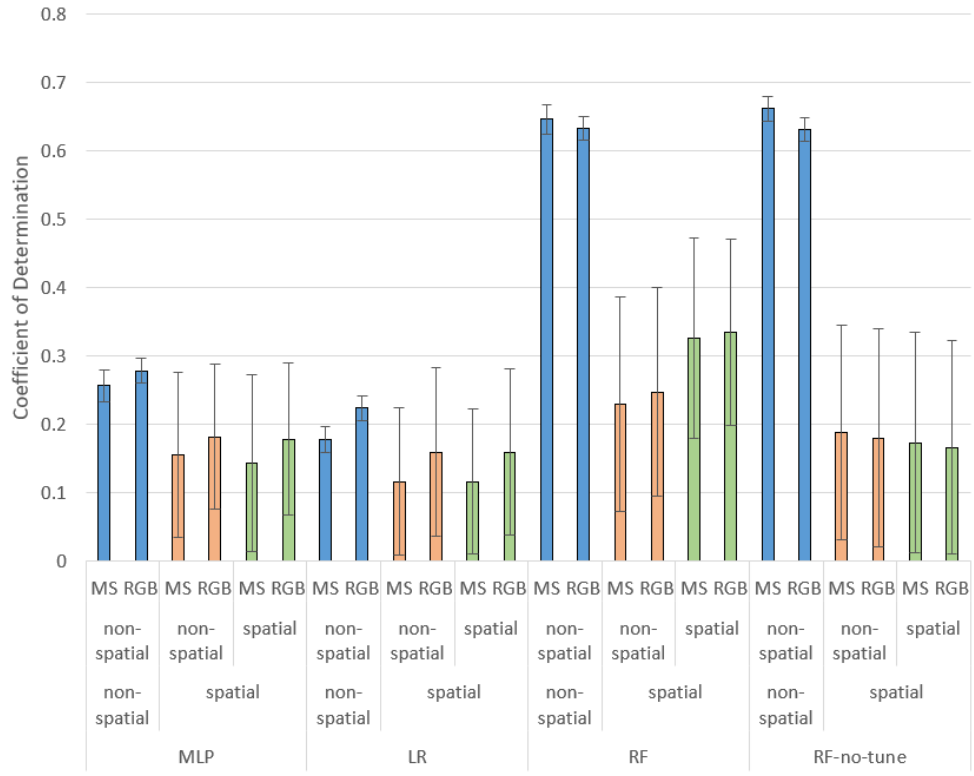
Figure 4.5.1: Yield prediction experiments: average RMSE performance for the LRe-single-VI experiments, where the error bars represent 1 standard deviation in plot (a) and 95 % confidence intervals in plot (b).

experiments; however, since a) MLP-LRe-all-bands outperformed LR-HRe-all-bands and b) an analyst could lower the spatial resolution of imagery, this means that MLP can be considered as the better performing model over LR. Figures 4.5.1–4.5.4 and Table 4.5.2 show that among RF, MLP, and LR, RF was the better performing model and LR was generally the worst model. In our study described in Section 4.4, LR did better than RF. This difference in findings may be due to a) the present work did not perform reverse CV where a single fold was used for training and the remaining folds were used for testing, b) our Section 4.4 study did not conduct RF hyperparameter tuning, and c) only a small number of imagery features (one and three features) were used in our Section 4.4 study. Furthermore, by comparing RF and RF-no-tune spatial CV results from all the experiments, we can see that the impact hyperparameter tuning had on RF's performance decreased as the number of features increased. It is worth noting that in the spatial CV experiments shown in Figure 4.5.4, MLP's performance was not much lower than that of RF. In fact, given the observable trend that increasing the spectral resolution (number of channels) of the datasets led to generally better results, it would be interesting to see which model performed best using HS imagery with 100 or so raw-bands.

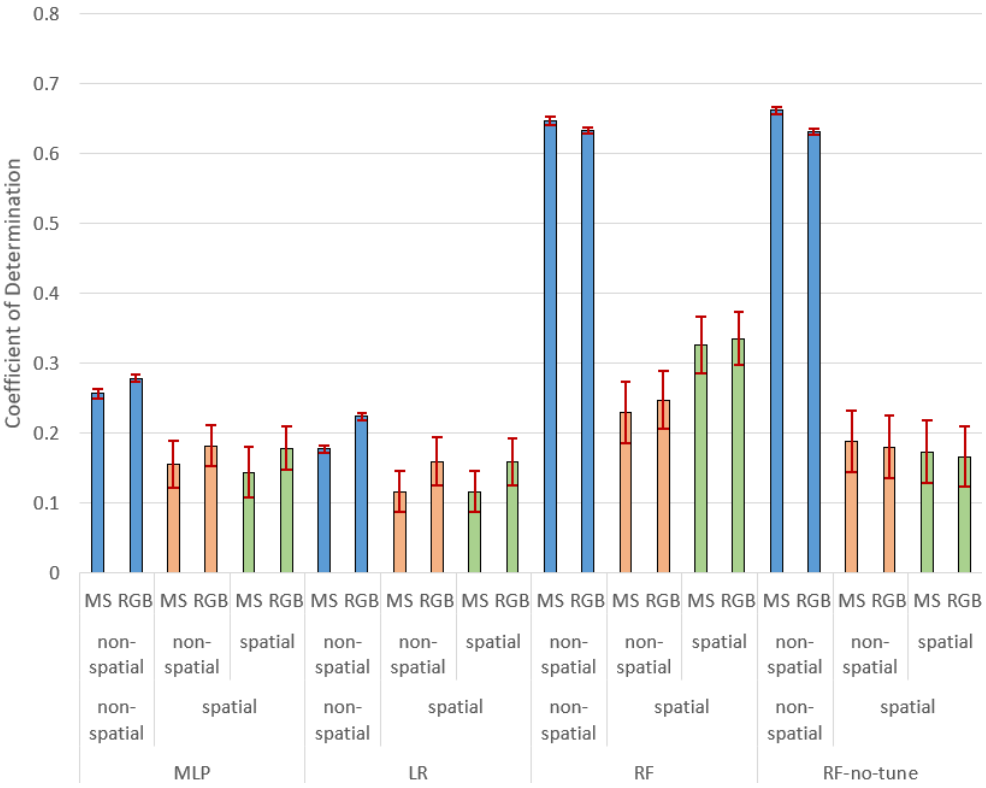
Takeaway. RF was the best performing model and LR was the worst. The choice of RF's hyperparameters had more of an impact on performance when the number of features was smaller and had less of an impact when the number of features was larger.

4.5.3.5 Data Processing Time Analysis

An execution time analysis of the ML models tested in the present work was performed and is explained in this section to illustrate the feasibility of deploying the applied methodology

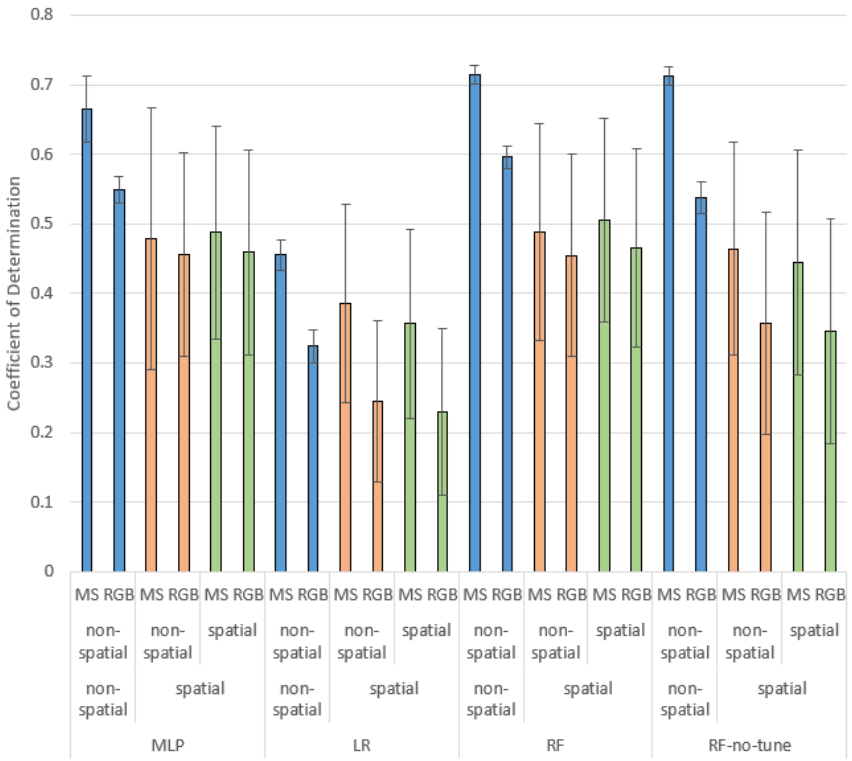


(a) with 1 standard deviation error bars

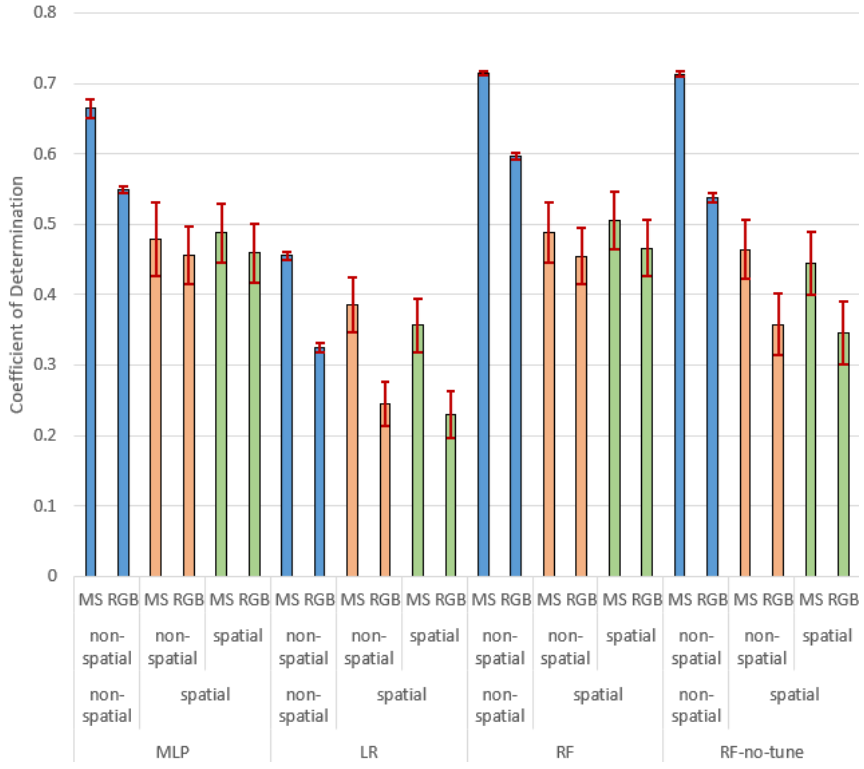


(b) with 95 % confidence interval error bars

Figure 4.5.2: Yield prediction experiments: average R^2 performance for the HRe-single-VI experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b).

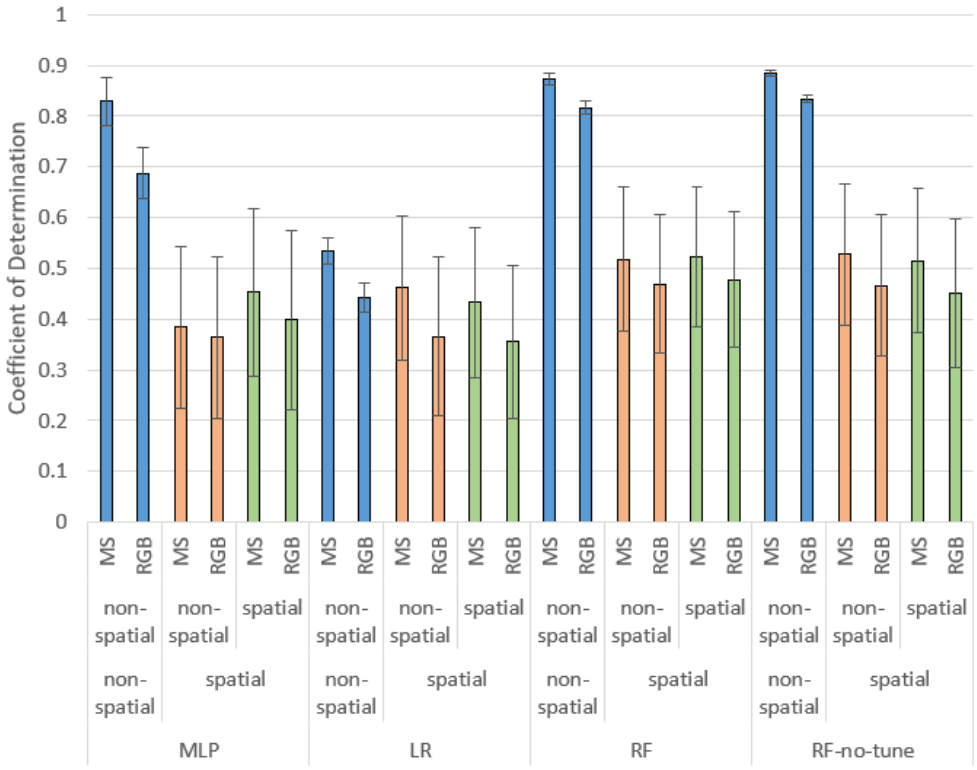


(a) with 1 standard deviation error bars

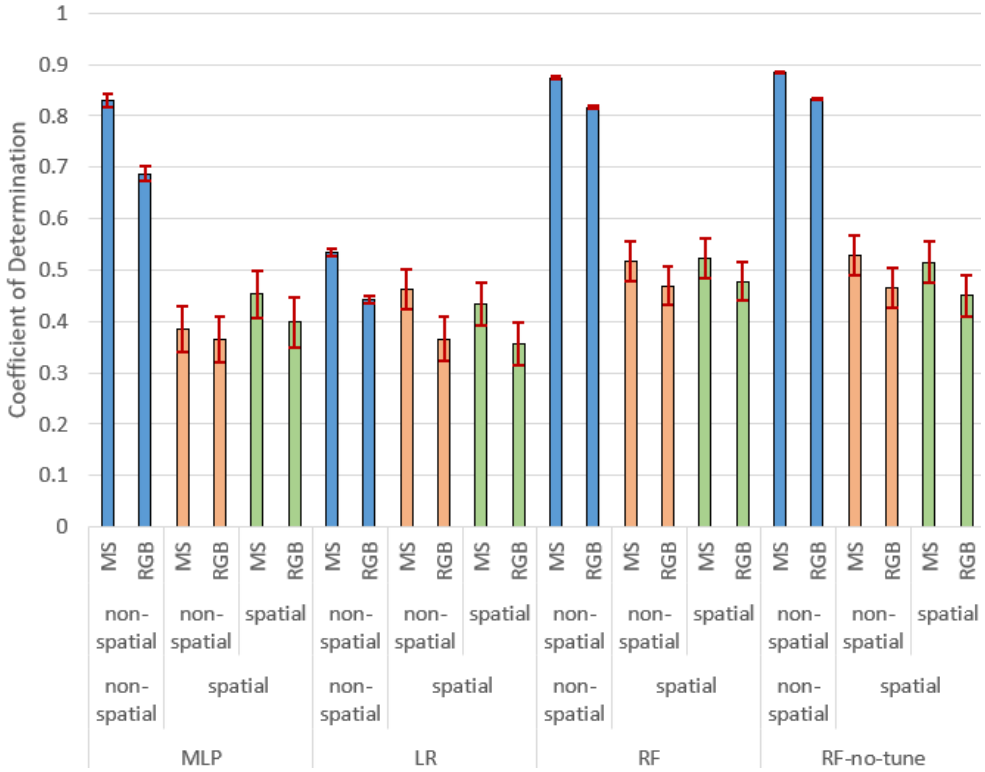


(b) with 95 % confidence interval error bars

Figure 4.5.3: Yield prediction experiments: average R^2 performance for the LRe-all-bands experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b).



(a) with 1 standard deviation error bars



(b) with 95 % confidence interval error bars

Figure 4.5.4: Yield prediction experiments: average R^2 performance for the HRe-all-bands experiments, where the error bars represent 1 standard deviation in plot (a) and a 95 % confidence interval in plot (b).

Table 4.5.4: Average approximate execution times for the experiments (s = second, m = minute, and h = hour), where RF-no-tune = RF without hyperparameter tuning.

Experiment Type	RF (h)	RF-no-tune (m)	MLP (h)	LR (s)
All Imagery Experiments	3.2	2.8	11.8	360
HRe-MS-all-bands	8.2	4.2	20.0	2826
HRe-RGB-all-bands	5.4	4.3	15.1	35
LRe-MS-all-bands	3.2	3.4	16.3	4
LRe-RGB-all-bands	2.0	2.5	12.5	2
HRe-(MS&RGB)-single-VI	1.8	2.4	9.1	2
LRe-(MS&RGB)-single-VI	1.4	1.7	6.3	2
Location-only	0.8	1.2	36.8	0

to a live setting. Since the present work applied many of the same pre-processing steps as those applied in our study discussed in Section 4.4, we omit including imagery acquisition and data processing in the analysis of this section and instead refer readers to Sections 4.4.3.7.1 and 4.4.3.7.2 for more details.

Table 4.5.4 presents the average approximate execution times of the ML models for all the experiments.

Takeaway. LR had generally the shortest execution time and MLP had the longest execution time.

4.5.4 Conclusion

In the present work, we performed a yield prediction study using UAV-based imagery from a single image acquisition date and from a single field. We cleaned and interpolated yield data, computed VIs from the imagery, and fused the yield and imagery data together using a mean-, maximum-, and minimum-filter to produce two types of datasets: LRe and HRe. LRe datasets only had mean reflectance, while HRe datasets had the mean, maximum, and minimum reflectance. These two types of datasets were used to create 8 different datasets by varying each combination of HRe vs. LRe, MS vs. RGB imagery, and multi-band vs. single-VI features. A ninth dataset that only contained location features was also used. These 9 datasets were used to train and evaluate ZeroR, MLP, RF, RF-no-tune, and LR models using three types of nested CV, namely, non-spatial/non-spatial, spatial/non-spatial, and spatial/spatial. We found that multi-band datasets led to better performance than single-VI datasets. MS imagery led to generally better performance than RGB imagery. HRe datasets led to better performance than LRe datasets. RF was the best performing model and LR was the worst. The choice of RF's hyperparameters had more of an impact on performance when the number of features was smaller, and had less of an impact when the number of features was larger or when the input dataset had a lot of spatial structure.

Future work includes: a) considering multi-VI and single-band datasets[13], b) performing similar experiments over multiple image acquisition dates, c) performing reverse CV similar

[13]We explored the use of multi-VI datasets in the study described in Section 4.6.

to those performed in the study described in Section 4.4, d) create RGB datasets using imagery from a RGB camera instead of from a MS camera, and e) include a MLP model that has a static set of hyperparameters similar to RF-no-tune.

4.6 Yield Prediction Study: Using UAV Imagery, Spatial Cross-validation, Model Selection, and Distributed Machine Learning

At the time of writing, this study has not been published, although it has been submitted to a journal for publication [138].

The present work is an UAV imagery-based corn yield prediction study that made use of existing distributed ML techniques to avoid sharing private data. Specifically, we evaluated models using CL, LL, and three distributed ML methods, FL using FedAvg and two EL methods. We had data from only a single study site despite us wanting to explore the difficulties of extrapolating yield predictions to other regions (training a model using one farm's data and evaluating the model using a different farm's data), so we partitioned the data among the clients using data splits with varying degree of heterogeneity.

Our goal was to investigate whether accurate yield precision maps could be produced, without needing to share farmers' private data, by training models using existing distributed ML methods (including FL and EL approaches). Among the yield prediction literature that made use of FL, we are among the few studies that:

a) included SE as a distributed ML method in their model evaluation methodology;

b) predicted crop yield at pixel-level scale;

c) and clearly compared models evaluated using IID datasets vs. non-IID datasets.

In terms of how the present work relates to this thesis: the spatial analysis methodologies applied in the present work tie into the spatiotemporal data analysis aspect of this thesis, and the distributed ML experiments applied are directly tied to the distributed ML aspects of this thesis.

The following is a list of limitations of the present work: a) the data gathered was from only a single study site, b) only FedAvg was considered, which is not an appropriate choice for non-IID client data, and we did not consider other FL aggregator algorithms, c) we did not explore the effects on model performance of varying the number of clients (the number of clients were fixed), d) the networking logic and nodes for the distributed ML experiments were simulated on a single machine, and e) UAV imagery and yield datasets were both available for only a single year, so cross-year extrapolation experiments could not be conducted.

4.6.1 Materials and Methods

4.6.1.1 Datasets

We used 2021 UAV imagery only from the MS camera and from two imagery acquisition dates. We chose imagery from DoY 201 and DoY 216, since those imagery datasets had imagery from all six fields and those acquisition dates led to some of the best yield prediction model performance in our study discussed in Section 4.4.

4.6.1.2 Feature Extraction

The VIs we chose were some of the VIs that led to the best mid-season yield prediction model performance in our study described in Section 4.4, namely, $\text{NDVI}_{\text{RedEdge}}$, $\text{NDVI}_{\text{Green}}$, $\text{SRI}_{(\text{NIR,RedEdge})}$, NG, GCI, and $\text{SRI}_{(\text{NIR,Green})}$, and we also chose a few VIs that are commonly used in the yield prediction literature, namely, NDVI, ExG, VIg, and OSAVI. In addition to the VIs, we also included the five spectral bands (or raw-bands) as the spectral channels used for feature derivation. We used the same steps described in Section 4.4.1.3 to a) clean and interpolate yield data to a 2.5 m $\times$ 2.5 m grid and b) fuse the yield and imagery into HRe datasets using a mean-, maximum-, and minimum-filter. After fusing the imagery to the yield, the resulting multi-temporal datasets had 2016, 2436, 1360, 975, 759, and 19 634 samples for Fields 1, 2, 3, 4, 9, and 11, respectively.

We proceeded by deriving an additional type of temporal feature, a delta feature, for each of the 15 spectral channels (10 VIs and 5 spectral bands) by taking the reflection difference between both DoY 201 and DoY 216 features of the respective spectral channel; that is, we subtracted DoY 216 pixel reflectance from DoY 201 pixel reflectance for each pair of features[14]. This process resulted in 135 features = (15 spectral channels $\times$ 3 filters) $\times$ (2 imagery acquisition dates + 1 delta feature). We then partitioned the dataset into a) two mono-temporal datasets, one with only DoY 201 features (mono-temporal-DoY201) and the other with only DoY 216 features (mono-temporal-DoY216), and b) a dataset with every feature included (multi-temporal). All three of the datasets had a total of 27,180 samples.

4.6.1.3 Machine Learning Models

We used various models in the present work, including tabular models, tensor-based models, and baseline models. The tabular models were MLP, RF, and LR. The tensor-based models were two types of 2D CNNs that we named spectral reflectance CNN (rCNN) and spatial CNN (sCNN). The neural network architectures of MLP, rCNN, and sCNN are described in Appendix C in Tables C.2, C.5, and C.6, respectively. Furthermore, the design of the rCNN and sCNN network architectures was inspired by the idea of the "Network In Network" [151] design principle and other related literature [291, 143, 212]; that is, rCNN

[14]For example, delta maximum NDVI reflectance = DoY 201 maximum NDVI reflectance − DoY 216 maximum NDVI reflectance.

and sCNN were designed to automatically extract spectral and spatial reflectance features, respectively, from spectral bands. We used two types of baseline models, ZeroR and a model we named Location Memorizer (LocMem). LocMem was a simple KNN model with one neighbour ($k = 1$) and was given only location features. It memorized the prior year's yield (2020 yield) data by location and predicted 2021 yield using only location features.

We chose RF, LR, MLP, and ZeroR for the same reasons explained in Section 4.5.2. We included LocMem, since it enabled the analysis of yield variability between years. We chose CNN, since it is a model that has been found to perform well for image analysis.

4.6.1.4 Layered Block Cross-validation

We performed a form of block/spatial CV using clustered sampling similar to how a) Nevavuori et al. [185], b) our study described in Section 4.4, and c) our study described in Section 4.5 partitioned data for evaluating models. We performed two layers of clustered sampling such that an input spatial dataset was split into spatially disjointed client datasets and then was further segmented into spatially disjointed local blocks for each client. We did this to avoid favoring the inclusion of data from any one client in particular during the fold creation process. Data segmentation was done via k-means clustering or used an already pre-defined partitioning scheme (for example, assigning each client to a field). The first layer of clustered sampling clustered samples into j outer blocks (one block per client), and the second layer of clustered sampling was applied to each outer block to form k inner blocks per client. We then randomly shuffled samples in each inner block and merged the inner blocks into folds (via random sampling without replacement) such that a fold consisted of samples from j different inner blocks where each inner block was from a different client. This meant that the folds were forced to contain data from each client. These folds were then treated the same way KF-CV treats folds to iteratively partition the datasets into k training-testing dataset pairs such that every sample of the original dataset was part of the testing set exactly once. In some of our experiments we applied a second inner loop of layered block CV (nested layered block CV) to perform hyperparameter tuning. Figure 4.6.1 provides an illustrative example of how we performed layered block CV in the present work. Note that to further reduce the impacts of SA on model performance, some works that used block CV introduced a buffered layer between training data and testing data such that training samples near test samples (in the buffered layer) were removed, where the size of the buffered layer was set to be equal to the spatial extend of the dataset's SA [273]. We did not apply such a methodology, which is a limitation of the present work.

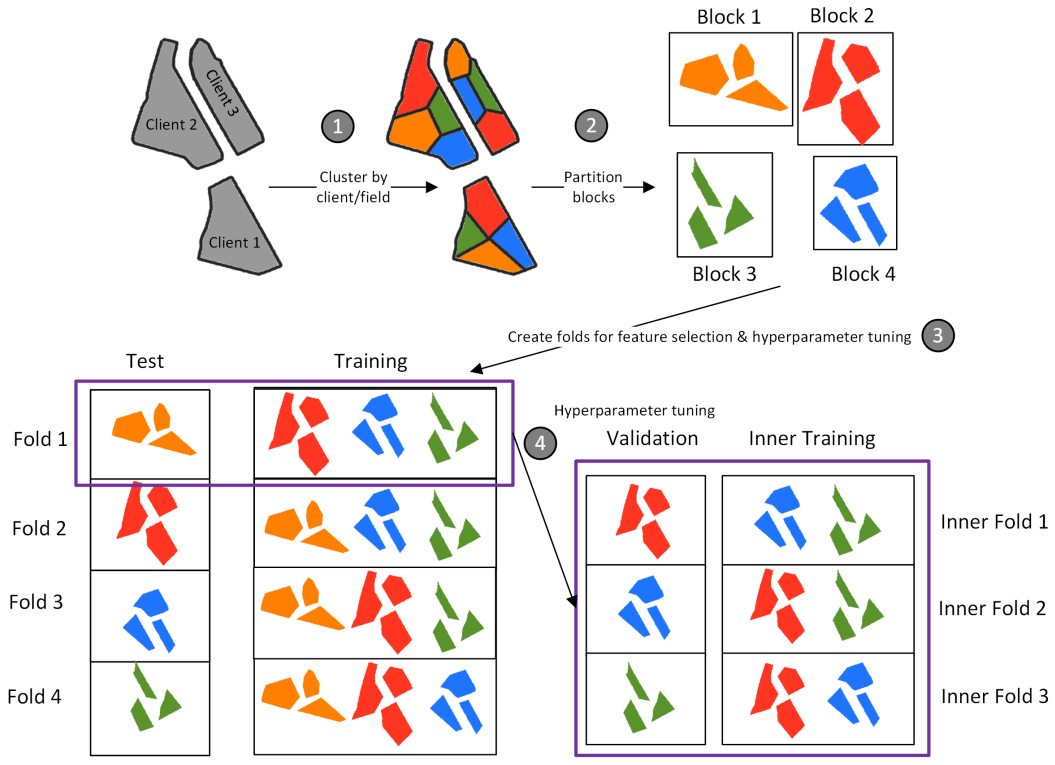


Figure 4.6.1: Yield prediction experiments: visual example of the data pre-processing steps involved with model evaluation for a 3-client system (where Fields 1, 2, and 3 are used as clients) using nested layered 4-3-fold block CV, where the outer training-testing fold pairs are further split into inner training-validation pairs within the inner CV loop to be used for hyperparameter tuning, and the inner CV loops for outer folds 2–4 are omitted for clarity.

4.6.2 Experiments

In this section we describe our pixel-level imagery-based crop yield prediction regression experiments. Our goal was to evaluate a) the accuracy of the predictions made by the models when trained using various learning methods and b) the resource utilization (execution time and network usage) of the models. We evaluated the models using the following evaluation metrics: R^2, RMSE, and MAPE. We implemented deep learning and ML models using Python version 3.8.19, where *sklearn* version 1.3.2. was used to implement the ML models, and *TensorFlow* version 2.12.0, *cuDNN* version 8.6, and *CUDA* version 11.8 were used to implemented the deep learning models. Other Python packages used in our study included *pandas* version 2.0.3, *numpy* version 1.24.3, and *geopandas* version 0.13.2. To enable the reproducibility of our experiments, we fixed the random seed's value to 123. We trained and evaluated models using the same machine as the one described in Section 4.5.2. We made our software scripts and datasets open source, and they are available on GitHub[15].

[15] https://github.com/patkilleen/fed_learn_yield_prediction_2025, accessed on February 25 2025

4.6.2.1 Model Selection

In this section we explain how we performed model selection. This included feature selection and choosing hyperparameters. Since our experiments involved FL and model sharing combined with distributed EL (we will simply refer to this as EL in the present work) such that data was never shared, this means the clients must have previously agreed on the features to use. We also assumed clients agreed before-hand on which hyperparameters their models would use. As such, we initially performed model selection (i.e., feature selection and hyperparameter tuning) via CL to satisfy these assumptions before running the main experiments (described next in Section 4.6.2.2) involving CL, FL, EL, and LL. We used layered block CV (described in Section 4.6.1.4) for model selection. Furthermore, during the model selection process, for each iteration of layered block CV, we randomly sampled 500 instances from each client to avoid class imbalance issues arising from the heterogeneity of the dataset (size and yield distribution differences of the fields). Model selection was performed for all three datasets (the mono-temporal-DoY201, mono-temporal-DoY216, and multi-temporal dataset).

4.6.2.1.1 Feature Selection We performed feature selection using the PLSR-based methodology described in Section 4.3.5.1 using layered 10-fold block CV and *sklearn* version 1.3.2. The number of grid search trials was 100.

The feature selection process resulted in 10 selected features from the mono-temporal-DoY201 dataset (resulting in 11.66 RMSE and 0.58 R^2), 16 selected features from the mono-temporal-DoY216 dataset (resulting in 11.54 RMSE and 0.62 R^2), and 47 selected features from the multi-temporal dataset (resulting in 11.52 RMSE and 0.64 R^2). The selected features can be seen in Figure D.2 in Appendix D along with their importance, such that the importance was defined as the normalized absolute value of a feature's PLSR coefficient. The importance represents the relative scale at which a feature affects PLSR yield predictions.

Note that for experiments involving CNN models, instead of using the selected features, to evaluate the ability of CNN to perform automated feature extraction, we designed the CNN experiments such that only mean reflectance spectral band features were selected (e.g., mean red reflectance). This simulated the situation where 2.5 m spatial resolution raw imagery was fed to the CNN models. As a result, both mono-temporal datasets had five selected features and the multi-temporal dataset had 10 selected features (no delta features were selected from the multi-temporal datasets) for the CNN experiments.

4.6.2.1.2 Hyperparameter Tuning We performed hyperparameter tuning using the methodology described in Section 4.3.5.2 using layered block CV. The features used were those resulting from the feature selection step. Specifically, we applied min-max scaling to scale values between 0 and 1. Then we used a single iteration of nested layered 3-3-fold block CV, 50 trials of random search, and 2 executions per trials. The hyperparameter search spaces (and the neural network architectures for the deep learning models) for RF, MLP, rCNN, and sCNN are described in Appendix C in Tables C.1, C.2, C.5, and C.6,

respectively. Both types of CNN models (rCNN and sCNN) were evaluated using various input tile sizes (3×3, 7×7, and 11×11) to treat the input tile size as a hyperparameter to be tuned.

We proceed to discuss the selected hyperparameters. Table 4.6.1 lists the hyperparameter value choices of hyperparameters that were common to the deep learning models. RF had 375 trees, 2 minimum samples per leaf, bootstrap sampling was applied using 65 % of the samples, the number of features used for a split was 5, and the maximum depth of the trees was 20. The deep learning model network structures were fixed as follows: a) MLP had two hidden layers such that the first and second layer had 40 and 205 neurons, respectively, and b) rCNN and sCNN had 215 and 130 feature maps in their 2D convolution layer, respectively, and had 244 and 228 neurons in their FC_1 layer, respectively. Furthermore, as we later discuss in Section 4.6.3.1, we found that there was no statistically significant difference in CNN model performance between the three different input tile sizes. Therefore, we chose 3×3 (the smallest tile size) as the input tile size for the CNN models for the remaining experiments to minimize processing overhead.

Table 4.6.1: Yield prediction experiments: deep learning model hyperparameter choices for the main experiments detailed in Section 4.6.2.2, where FC = fully connected, rCNN = spectral reflectance CNN, and sCNN = spatial CNN.

Hyperparameter	**MLP**	**rCNN**	**sCNN**
Early Stopping Patience	12	12	9
Number of Epochs	650	500	600
FC Layers' Activation Function	ReLu	ReLu	Sigmoid
Batch Size	495	510	436
Learning rate	0.00103	0.00413	0.00007
Optimizer	Adam	Adam	Adam

4.6.2.2 Federated Learning and Model Sharing

We performed two iterations of layered 5-fold block CV. A single machine was used to simulate the network of client devices, such that six clients were part of the network. Ideally the clients would have had data from different study sites, but due to time and data availability constraints, clients in the present work were assigned data from the same study site. Training and testing data was created over many CV loops using layered block CV over each client's dataset.

We used the model selection results analysis discussed in Section 4.6.3.1 to guide the design of the experiments discussed in this section; that is, we found from the results of the hyperparameter tuning process that there was no statistically significant difference in model performance between the three datasets (mono-temporal-DoY201, mono-temporal-DoY216, and multi-temporal), so we decided to only use the multi-temporal dataset. We applied min-max scaling to scale dataset feature values between 0 and 1 before partitioning the dataset into client datasets. This assumed that the global maximum and minimum yield was known by all clients, which should be reasonable, since such a maximum could

be defined by biological yield limits (e.g., 250 bu/ac) and the minimum could be set to 0 bu/ac. Our main crop yield prediction regression experiments involved comparing the model performance of models evaluated using various learning methods, namely CL, LL, FL (using FedAvg [170] as the FL aggregator), and EL. Furthermore, given that a) our yield dataset was spatially heterogeneous, b) the FedAvg algorithm assumes IID client datasets [63], and c) we had data from only a single study site despite us wanting to explore the difficulties of extrapolating yield predictions to other regions (training a model using one farm's data and evaluating the model using a different farm's data), we partitioned the data among the clients using three types of splits of varying heterogeneity:

- **IID**: only Field 11 (the largest field) data was used for this data split, such that the field was clustered into six spatially disjointed blocks (via k-means clustering) and each block was assigned to a client. The goal was to increase the similarity between client dataset distributions by using data from a single field for creating a more optimistic baseline. This simulated nearby farms joining the SFS. This dataset splitting methodology may be slightly over-optimistic, because SA will more likely affect testing and training dataset similarities, given that the geographical extent of the clusters used to build the folds was smaller.
- **non-IID1**: client datasets were assigned by treating each field as a client, and clients shared the same number of samples by randomly sampling (without replacement) 500 samples for each iteration of the evaluation process. This simulated similar-sized farms from different regions joining the SFS.
- **non-IID2**: client datasets were assigned by treating each field as a client, and no random sampling was applied. The number of samples per client differed. This simulated a more realistic setting, where smaller farms and larger farms from different regions joined the SFS.

Informally, the heterogeneity (or 'IID-ness') of the splits was quantified by a) the degree to which the spatial autocorrelation could lead to over-optimistic model performance as a result of having training and testing readings being sampled nearby in time and b) the size (number of samples) of the client datasets. More formally, in terms of spatial autocorrelation, the heterogeneity of the splits was quantified by the average spatial distance of the samples between clients; that is, for random samples s_i and s_j taken from the datasets of clients c_0 and c_1, respectively, the Euclidean distance between the spatial coordinates of s_i and s_j would more likely be shorter for IID datasets than for non-IID datasets. It is important to note that fields generally had different yield data distributions, and we took this into account when defining the heterogeneity of the splits. For each learning method we evaluated each model against these three different client dataset splits (IID, non-IID1, and non-IID2). We proceed to explaining how we applied CL, LL, FL, and EL for model evaluation. In the CL case, all the clients sent their training and testing datasets to the simulated cloud for centralized model evaluation. In the LL case, clients each trained and evaluated a local model using only their private local data. In the FL case, we implemented FL by using the FedAvg algorithm. All six clients were selected every round (to simulate the cross-silo setting where clients are always online, since the PN in the yield prediction

setting would be a desktop at the farmer's house) and a total of 64 communication rounds were conducted. In the EL case, using the notions described in Section 2.3.3, we evaluated two types of EL, SE and AE. The AE and SE methods involved clients first applying LL. Clients then shared their local model with each other client. AE involved having clients feed their testing data to each local model and treating the average of the local model predictions as the final prediction. SE, instead of averaging local model predictions, involved combining these predictions with the expected yield (from the level-0 testing dataset) to create a level-1 dataset that contained yield ground truth data and local model predictions. Standard 5-fold CV was then used to split this level-1 dataset into multiple level-1 training-testing dataset pairs for training and evaluating a LR meta-model. Note that we used standard 5-fold CV instead of block CV, since a) it was more straightforward to implement and b) we found in our study described in Section 4.4 that LR can be somewhat resistant to the effects of spatial autocorrelation. Note that for AE, FL, LL, and SE, for a given fold, we took the average model performance across all clients and treated this performance as the learning method's performance for that fold. It is important to note that the SE method requires new clients to have labeled data to build the meta-model, whereas FL and AE do not have this requirement. This is a limitation of the SE approach methodology applied in the present work and in our study described in Section 5.5.

4.6.3 Analysis and Results

The results of our yield prediction experiments are discussed in this section. In our analysis we performed statistical significance testing on results using the Wilcoxon's signed-rank test and Friedman test [72, pp. 351–357][261] using a confidence level of 5 % ($\alpha = 0.05$) (see Section 4.3.6 for more details). We proceed by listing the tables and figures that present experimental results. Table 4.6.2 contains the centralized hyperparameter tuning model performance results. Tables 4.6.3 and 4.6.4 present the model performance results for each model and learning method for the IID and non-IID client dataset splits, respectively. Tables 4.6.5 and 4.6.6 present the performance ranking of the models and learning methods, respectively. Table 4.6.7 presents the total resource utilization (including simulated network communication cost and execution time). Figure 4.6.2 presents yield precision maps of the actual and predicted yield for Field 11, and Figures 4.6.3 and 4.6.4, for RF and sCNN, respectively, present yield precision maps of the actual and predicted yield for the smaller fields. The maps were created by taking the average yield of a model's predictions over all iterations and folds for a location/pixel. Note that for the non-IID1 experiments, predictions over multiple iterations were merged, since random sampling was applied, meaning the same pixel was not necessarily present in all iterations. Furthermore, we took for granted the ability to generate precision map diagrams of model predictions: this methodology requires the predictions and actual yield data to be centrally available. In a live FL system, an analyst may not have access to this type of information, meaning the analyst would have to rely only on performance metrics to conduct a model performance analysis.

4.6.3.1 Centralized Learning with Hyperparameter Tuning

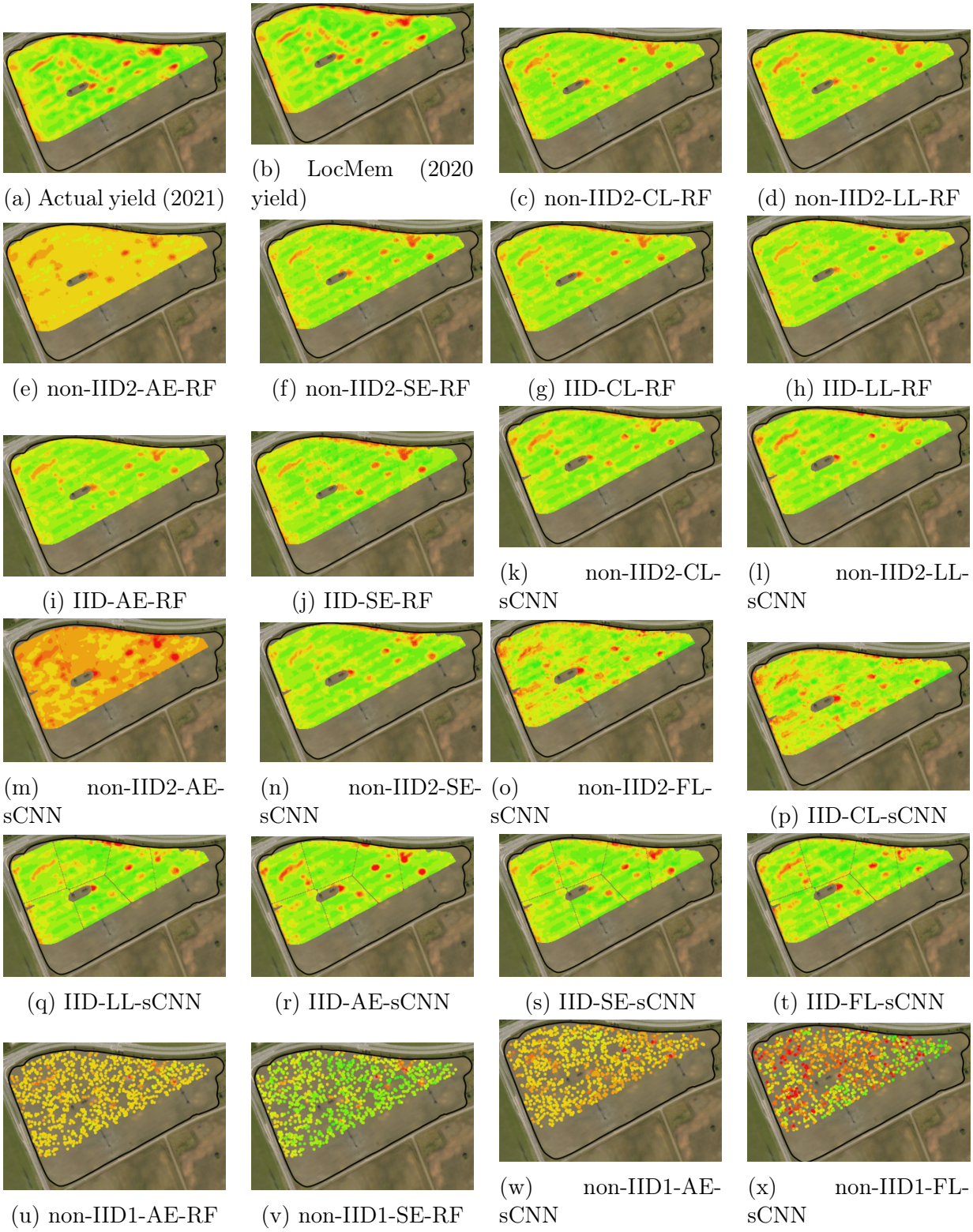
In this section the results of the model selection hyperparameter tuning process are discussed. The model performance results are found in Table 4.6.2. We found no statistically significant difference in performance between input tile sizes for the CNN models, so we decided to combine the results of all tile sizes in the Table 4.6.2. There was no statistically significant difference in performance between all the models, meaning the models were all similar. When only comparing rCNN to sCNN performance, results were similar in terms of R^2 and RMSE performance, although there was significantly better performance demonstrated by rCNN in terms of MAPE. Given that the CNN models used raw imagery and did similarly well to the models that used VI features and other aggregated reflectance features (e.g., maximum red reflectance), this suggests that deriving VIs is not required to obtain reasonable centralized model performance, since CNN could be applied to raw imagery, which enables reductions in data pre-processing complexity. These findings also suggest that to achieve reasonable performance, lower spatial resolution imagery could be used in combination with CNN models instead of using high spatial resolution imagery combined with tabular models; this is the case, since mean reflectance features were sufficient for reasonable performance and so maximum/minimum reflectance features (obtained from higher spatial resolution imagery) would not be required. Furthermore, since models did similarly well, basic ML models (like RF) could be used, instead of more complex deep learning models, to reduce the execution time and computational complexity of the learning process. Furthermore, no dataset led to significantly better performance than others when considering every model. This suggests that a single imagery acquisition mission is sufficient to obtain reasonable centralized yield prediction performance.

Takeaway. All the models involved in the hyperparameter tuning process did similarly well. There was no significant performance difference found that was induced by the CNN input tile size choice. Each of the three datasets (mono-temporal-DoY201, mono-temporal-DoY216, and multi-temporal) led to relatively similar model performance.

4.6.3.2 Baseline Model Performance

In this section we discuss the performance of the baseline models LocMem and ZeroR. By observing Tables 4.6.3 and 4.6.4 we can see that there were some experiments that led to worse performance than ZeroR; for example, for the AE experiments (for all three client dataset splits), LR did worse than ZeroR in terms of RMSE and MAPE. Furthermore, for experiments that used the IID dataset split and LL (IID-LL experiments), LR also did worse than ZeroR.

Now we focus on LocMem. LocMem achieved 0.45 R^2, 16.67 RMSE, and 9 % MAPE. By observing the precision maps of the actual yield and the LocMem predictions in Figures 4.6.2a and 4.6.2b, respectively, we can see that the variability in Field 11's yield between 2020 and 2021 did not change much. There were low and high yield sub-regions that were present in both years' datasets. Assuming that this trend applied to other fields, models would need to have performed reasonably well (that is, achieving at least 0.45 R^2) to bring predictive value to a farmer. Otherwise, if a model cannot outperform LocMem, then a

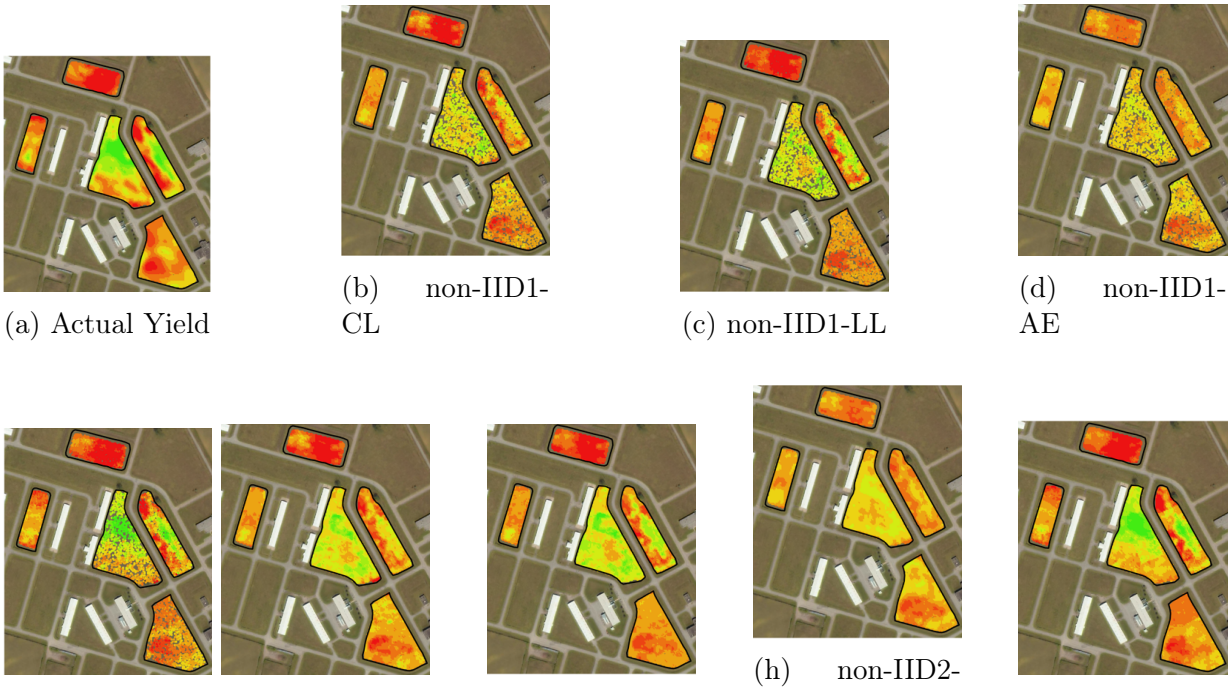


(a) Actual yield (2021)
(b) LocMem (2020 yield)
(c) non-IID2-CL-RF
(d) non-IID2-LL-RF
(e) non-IID2-AE-RF
(f) non-IID2-SE-RF
(g) IID-CL-RF
(h) IID-LL-RF
(i) IID-AE-RF
(j) IID-SE-RF
(k) non-IID2-CL-sCNN
(l) non-IID2-LL-sCNN
(m) non-IID2-AE-sCNN
(n) non-IID2-SE-sCNN
(o) non-IID2-FL-sCNN
(p) IID-CL-sCNN
(q) IID-LL-sCNN
(r) IID-AE-sCNN
(s) IID-SE-sCNN
(t) IID-FL-sCNN
(u) non-IID1-AE-RF
(v) non-IID1-SE-RF
(w) non-IID1-AE-sCNN
(x) non-IID1-FL-sCNN

Figure 4.6.2: Field 11 precision maps of actual yield (a) vs. model predictions (b–x), where the color legend is the same as the one used in Figure 4.0.3d, and the non-LocMem prediction maps use a '<client dataset split>-<learning method>-<model>' naming convention; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, sCNN = spatial CNN, and LocMem = the location memorizer baseline model.

Table 4.6.2: Yield prediction experiments: centralized hyperparameter tuning experiment model performance, where results are shown as mean ± standard deviation, CNN results are averaged over all three input tile sizes (3×3, 7×7, and 11×11), and marked in bold are the best performing models (in terms of average performance) for a dataset; rCNN = spectral reflectance CNN, sCNN = spatial CNN, mono-temporal-DoY201 = the dataset with only DoY 201 features, mono-temporal-DoY216 = the dataset with only DoY 216 features, and multi-temporal = the dataset with both DoY 201 and 216 features.

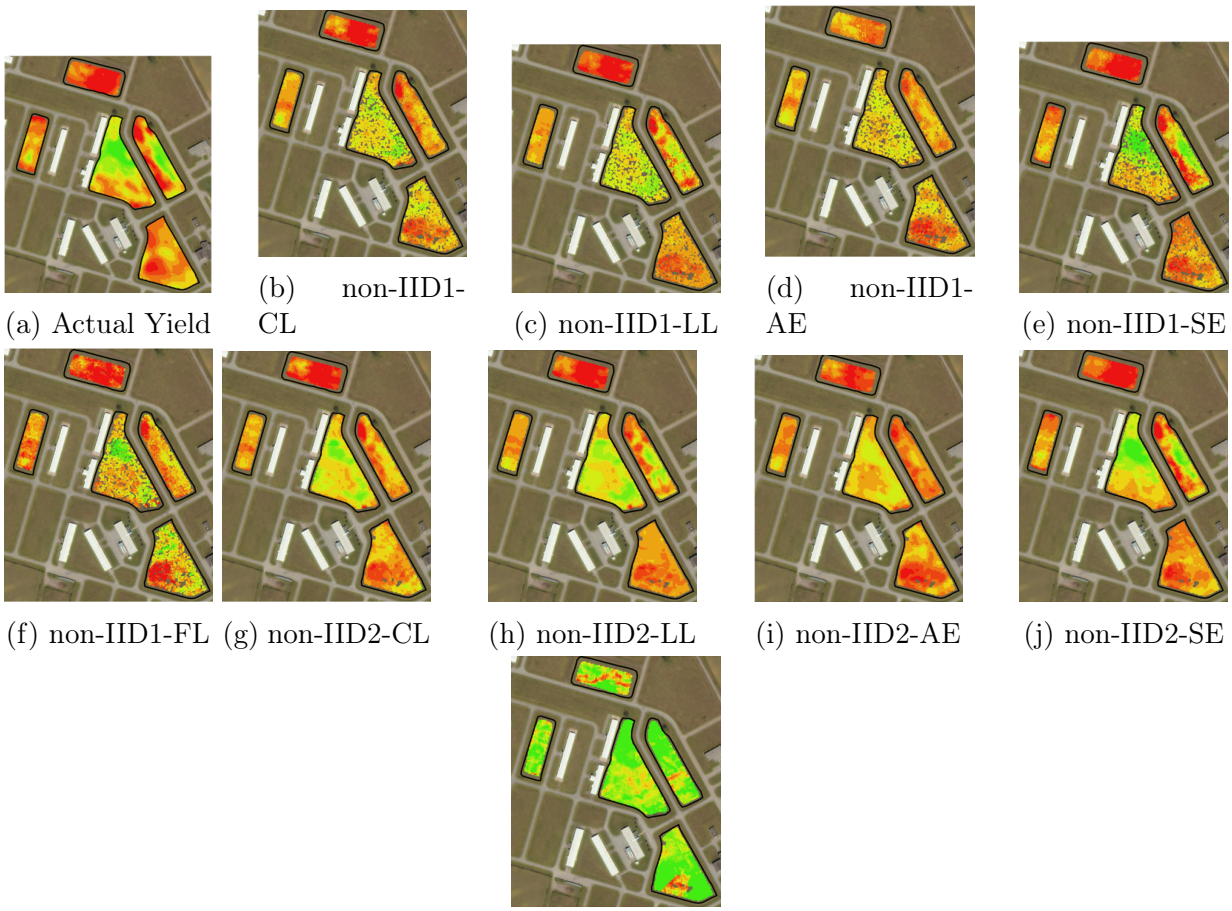
Dataset	**Model**	**R^2**	**RMSE**	**MAPE (%)**
mono-temporal-DoY201	MLP	0.49±0.11	26.11±1.59	19±1
mono-temporal-DoY201	rCNN	0.39±0.24	28.01±4.62	19±4
mono-temporal-DoY201	sCNN	0.38±0.20	29.41±4.02	20±4
mono-temporal-DoY201	RF	**0.50**±0.19	**24.71**±2.14	**18**±4
mono-temporal-DoY216	MLP	0.47±0.22	25.94±1.89	19±2
mono-temporal-DoY216	rCNN	**0.53**±0.17	**24.13**±3.34	**17**±3
mono-temporal-DoY216	sCNN	0.49±0.22	25.22±3.48	**17**±2
mono-temporal-DoY216	RF	0.42±0.12	27.54±3.00	20±3
multi-temporal	MLP	0.50±0.11	26.23±4.16	19±3
multi-temporal	rCNN	0.40±0.23	29.13±9.57	22±11
multi-temporal	sCNN	0.43±0.18	27.36±4.59	19±4
multi-temporal	RF	**0.56**±0.10	**23.78**±3.49	**17**±2



(a) Actual Yield (b) non-IID1-CL (c) non-IID1-LL (d) non-IID1-AE

(e) non-IID1-SE (f) non-IID2-CL (g) non-IID2-LL (h) non-IID2-AE (i) non-IID2-SE

Figure 4.6.3: Smaller field precision maps of actual yield (a) vs. RF model predictions (b–i), where the color legend is the same as the one used in Figure 4.0.3f, and the prediction maps use a '<client dataset split>-<learning method>' naming convention; AE = average ensemble learning, CL = centralized learning, LL = local learning, and SE = stacked ensemble learning.



(a) Actual Yield (b) non-IID1-CL (c) non-IID1-LL (d) non-IID1-AE (e) non-IID1-SE

(f) non-IID1-FL (g) non-IID2-CL (h) non-IID2-LL (i) non-IID2-AE (j) non-IID2-SE

(k) non-IID2-FL

Figure 4.6.4: Smaller field precision maps of actual yield (a) vs. sCNN (spatial CNN) model predictions (b–k), where the color legend is the same as the one used in Figure 4.0.3f, and the prediction maps use a '<client dataset split>-<learning method>' naming convention; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

Table 4.6.3: Yield prediction experiments: model performance for the IID experiments, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) among the different types of learning methods; and L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

L.Method	**Model**	**R^2**	**RMSE**	**MAPE (%)**
AE	MLP	0.41±0.24	14.92±4.21	**8**±3
AE	LR	0.19±0.26	619.88±4124.77	142±920
AE	rCNN	0.41±0.24	**14.09**±4.41	**8**±3
AE	RF	**0.44**±0.22	14.56±3.98	**8**±3
AE	sCNN	0.38±0.24	14.82±5.68	**8**±3
AE	ZeroR	0.00±0.00	21.79±6.95	12±6
CL	MLP	0.47±0.08	16.02±0.78	8±0
CL	LR	0.53±0.07	15.06±1.64	8±1
CL	rCNN	0.40±0.13	16.14±1.79	8±1
CL	RF	**0.56**±0.10	**14.18**±1.30	**7**±1
CL	sCNN	0.54±0.08	14.22±1.96	**7**±1
CL	ZeroR	0.00±0.00	22.27±3.16	12±2
FL	MLP	**0.31**±0.21	16.43±4.27	9±3
FL	rCNN	0.27±0.24	**15.84**±4.52	**8**±3
FL	sCNN	0.27±0.24	16.45±6.12	9±5
LL	MLP	0.33±0.25	16.43±5.62	9±4
LL	LR	0.33±0.26	25.09±63.44	9±5
LL	rCNN	0.29±0.26	17.20±7.72	9±5
LL	RF	**0.48**±0.22	14.18±4.20	**7**±3
LL	sCNN	0.44±0.26	**14.05**±5.37	8±4
LL	ZeroR	0.00±0.00	22.01±7.14	12±6
SE	MLP	0.66±0.13	10.99±3.54	6±2
SE	LR	0.66±0.12	12.68±12.53	6±3
SE	rCNN	0.66±0.13	10.49±3.62	6±2
SE	RF	0.66±0.14	11.06±3.51	6±2
SE	sCNN	**0.71**±0.12	**9.68**±3.25	**5**±2
SE	ZeroR	0.00±0.00	1.08E+4± 1.72E+4	7E+3 ± 1E+4

Table 4.6.4: Yield prediction experiments: model performance for the non-IID experiments, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) among the different types of learning methods; and L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

		R^2		RMSE		MAPE (%)	
L.Method	**Model**	**non-IID1**	**non-IID2**	**non-IID1**	**non-IID2**	**non-IID1**	**non-IID2**
AE	MLP	**0.07**±0.14	0.03±0.08	**26.54**±11.94	28.74±12.64	**20**±9	22±11
AE	LR	0.05±0.14	0.03±0.12	3306.90±16389.17	813.86±3472.68	546±2493	143±540
AE	rCNN	0.03±0.08	0.04±0.11	29.63±14.90	32.31±14.61	24±14	27±18
AE	RF	0.05±0.11	**0.05**±0.12	26.69±10.72	**26.45**±9.68	23±17	23±14
AE	sCNN	0.05±0.12	0.05±0.13	28.63±12.04	27.30±12.67	25±19	**21**±9
AE	ZeroR	0.00±0.00	0.00±0.00	34.02±16.04	33.41±15.62	31±26	30±25
CL	MLP	0.60±0.17	0.61±0.07	21.96±2.35	19.67±1.50	16±3	11±2
CL	LR	0.59±0.16	0.61±0.08	22.56±2.70	19.64±2.11	16±2	11±1
CL	rCNN	**0.64**±0.17	0.60±0.10	**20.70**±1.52	19.50±1.86	**15**±2	11±1
CL	RF	0.60±0.11	**0.68**±0.07	22.10±2.26	17.74±1.96	16±4	**10**±2
CL	sCNN	0.56±0.15	0.67±0.09	23.25±4.00	**17.39**±2.29	16±3	**10**±1
CL	ZeroR	0.00±0.00	0.00±0.00	36.75±5.46	32.25±3.20	30±8	22±4
FL	MLP	0.05±0.14	0.00±0.01	**28.23**±10.18	**48.76**±26.65	22±12	**41**±34
FL	rCNN	0.03±0.08	**0.01**±0.03	33.14±14.50	90.26±68.22	27±16	82±82
FL	sCNN	**0.06**±0.16	**0.01**±0.06	29.09±13.01	72.18±49.83	**21**±9	72±75
LL	MLP	0.13±0.20	0.14±0.21	21.07±7.29	21.18±8.10	16±8	17±11
LL	LR	0.16±0.22	0.18±0.23	21.28±8.44	20.08±7.37	17±11	16±10
LL	rCNN	**0.19**±0.25	0.16±0.26	**19.29**±7.42	**19.48**±6.32	**15**±11	**14**±6
LL	RF	0.18±0.24	**0.19**±0.25	20.39±8.56	19.84±7.66	16±10	16±9
LL	sCNN	0.17±0.24	0.17±0.24	21.46±10.20	20.70±9.09	16±9	16±9
LL	ZeroR	0.00±0.00	0.00±0.00	26.42±11.56	26.20±11.26	22±17	22±16
SE	MLP	0.40±0.21	0.48±0.20	13.46±4.43	12.73±3.98	10±4	9±3
SE	LR	0.44±0.22	0.51±0.21	28.89±91.78	12.30±4.16	12±13	9±3
SE	rCNN	**0.53**±0.21	0.56±0.21	**11.29**±4.21	**11.04**±3.93	**8**±3	**8**±3
SE	RF	0.42±0.18	0.51±0.20	13.09±4.88	12.53±4.22	10±4	9±3
SE	sCNN	0.52±0.23	**0.58**±0.19	11.53±4.45	11.07±4.09	**8**±3	**8**±3
SE	ZeroR	0.00±0.00	0.00±0.00	3.23E+4± 1.36E+5	3.89E+4± 6.70E+4	3E+4± 2E+5	4E+4± 7E+4

farmer's educated guess based on the farmer's history of managing his or her field would likely be more accurate than the precision map of a model that had worse performance than LocMem. Note that the fields had varying yield distributions, so the scale of RMSE and MAPE was different between the fields. We, therefore, focus on the R^2 performance for identifying models that outperformed LocMem in the next part of this analysis. Every CL experiment involving non-baseline models led to better R^2 than LocMem except for IID-CL-rCNN. There was also a LL experiment, IID-LL-RF, that was able to achieve better R^2 (0.48) than LocMem. Furthermore, many of the SE experiments also led to better performance than LocMem.

Takeaway. A farmer that estimates yield using only his or her intuition, based on the management history of a field, can make reasonable yield prediction estimates. Nevertheless, data-driven models can be used to outperform a farmer's intuition, and that can be particularly useful when a farmer purchases new farm land for growing crop. Specifically, we found that CL and SE, and in one case LL (IID-LL-RF), led to better R^2 performance than the LocMem baseline model for predicting yield.

4.6.3.3 Model Comparison

In this section we examine the performance differences between models. We performed significance testing over all the results for each fold (and client), with and without FL results, to fairly include the LR and RF models in the analysis. The results of the significance testing are found in Table 4.6.5, where the average ranking performance of the models is presented. When excluding the FL results, for all three performance metrics, RF and sCNN were the highest ranked models, and in terms of MAPE, rCNN was also one of the best performing models. When including the FL results and only considering deep learning model results, sCNN was the best performing deep learning model for all three metrics, and in terms of MAPE, MLP was also among the best performing models.

This significance test has highlighted the overall best performing models. We now proceed to examining the best average model performance over various experiments. By observing Tables 4.6.3 and 4.6.4, we can see that MLP, RF, sCNN, and rCNN were the models that achieved the best average performance over all the IID and non-IID experiments. For the non-IID1 experiments, the best performance was achieved by CL-rCNN with 0.64 R^2, SE-rCNN with 11.29 RMSE, and SE-rCNN with 8 % MAPE. For the non-IID2 experiments, the best performance was achieved by CL-RF with 0.68 R^2, SE-rCNN with 11.04 RMSE, and both SE-rCNN and SE-sCNN with 8 % MAPE. For the IID experiments, the best performance was achieved by SE-sCNN with 0.71 R^2, SE-sCNN with 9.68 RMSE, and SE-sCNN with 5 % MAPE.

Takeaway. For the non-FL experiments, RF and CNN (especially sCNN) demonstrated superior performance over all the other models. When including FL experiment results and only considering deep learning models, sCNN and MLP were the best performing models.

Table 4.6.5: Yield prediction experiments: list of model performance average rankings from the Friedman test over all the IID and non-IID client dataset splits and learning methods, where a lower ranking number indicates better ranking performance, and marked in bold are the best performing models[1]; rCNN = spectral reflectance CNN, sCNN = spatial CNN, and N/A = not applicable.

			Average Performance Ranking for Each Model					
Includes FL Results	**Metric**	**Friedman Test**	**MLP**	**LR**	**rCNN**	**RF**	**sCNN**	**ZeroR**
No	R^2	Rejected	2.64	2.71	2.50	**2.41**	**2.29**	N/A
No	RMSE	Rejected	3.36	3.64	3.18	**3.01**	**2.84**	4.96
No	MAPE	Rejected	3.35	3.64	**3.18**	**3.10**	**2.88**	4.86
Yes	R^2	Rejected	1.74	N/A	1.70	N/A	**1.57**	N/A
Yes	RMSE	Rejected	2.06	N/A	2.05	N/A	**1.88**	N/A
Yes	MAPE	Rejected	**2.07**	N/A	2.03	N/A	**1.89**	N/A

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

4.6.3.4 Learning Method Comparison

In this section we compare the performance differences of the various learning methods. We begin by performing significance testing, then comparing model performance for each learning method, and we finish the analysis by analyzing precision maps of model predictions.

We begin by performing a significance test to rank the learning methods over all experiments. Since FL was only applied to deep learning models, we only included deep learning model results in this significance test analysis. We can see in Table 4.6.6 that SE is the method that led to significantly better model performance compared to all the other learning methods (in most cases). In one case another method did better than SE; that is, in terms of R^2 for the non-IID experiment results (results from both non-IID1 and non-IID2 experiments), CL was the only other method that led to the best performance. LL had worse average performance ranking than CL and SE in all cases. LL had better average performance ranking than FL in most (all but the IID-MAPE case) cases. FL also had worse average ranking performance than AE in all cases. For the IID experiments, AE had better ranking performance than LL, except for the non-IID experiments.

If observing learning method ranking alone, our analysis is limited. Therefore, we proceed to analyze learning method performance result trends by observing Tables 4.6.3 and 4.6.4. In terms of CL performance, it is interesting that non-IID-CL performance was better than IID-CL performance. The IID experiments only included Field 11 data, whereas non-IID experiments included every field's data. This means that even though the fields were heterogeneous, they likely shared common trends that collectively could be used to improve model performance. This supports the narrative of the present work that a collection of datasets from various farms should be more useful to a model than a dataset from a single farm. This is evidence that increasing SFS adoption rates will lead to increased model performance. Furthermore, as expected, we can see that SE and

CL led to some of the best performance. Unfortunately, as previously mentioned, SE, by design, requires labeled data from clients that join the proposed SFS even when the system is in the live state; therefore, even though SE obtained the best performance, in practice the deployability of SE will be limited by this requirement. Furthermore, CL also has limitations in terms of lacking privacy and scalability. FL and AE are the next best candidates to offer privacy-aware distributed learning without needing labeled data from new clients, so we analyze the results of those methods next. We can see that for the IID experiments, FL led to worse R^2 performance than LL, although FL was able to achieve comparable (and even better) RMSE and MAPE performance than LL (e.g., IID-LL-rCNN had 17.20 RMSE and IID-FL-rCNN had 15.84 RMSE). Furthermore, AE performance was better than LL performance. However, for the non-IID experiments, AE and FL had worse performance than LL in terms of all three performance metrics. At first glance this is concerning, since it goes against the proposed methodology's objective. Agri-data tend to be heterogeneous (non-IID), meaning farmers could achieve better model performance without doing distributed learning and simply performing LL without sharing their data. Nevertheless, this conclusion may not be entirely correct, which we elaborate on later in this section when we analyze prediction maps. In summary, FL and AE did poorly in non-IID experiments and had better performance for IID experiments. This performance trend was similar to our findings reported in our study described in Section 5.5; that is, both AE and FL apply a form of averaging in their learning process, and client datasets were heterogeneous in the present work. This means the average operations likely removed the learned local trends from the global model and resulted in global trends that did not generalize well to any client. For the IID experiments, client datasets were purposely more similar, meaning these local trends were not removed by the learning process from applying averaging. Unfortunately, it is not reasonable to assume yield datasets from multiple farms to be IID, since a) agri-data tend to be heterogeneous and b) even the datasets from a single study site used in the present work were heterogeneous. Nevertheless, our findings are interesting, since they suggest that there is potential to use EL and FL to achieve reasonable yield prediction performance without needing to share data. Such models would need to be tuned to avoid removing local data trends in the resulting global model. Personalized FL could be a potential next step to for this idea [244, 279].

Furthermore, given that the yield dataset in the present work is a spatial dataset, to get a better understanding of the spatial variability of a model's predictions, we created yield precision maps. We proceed by examining Field 11 maps in Figure 4.6.2. We can see that in many cases the predictions maps are visually similar to the actual yield map in Figures 4.0.3d and 4.6.2a. In particular, many of the experiments that had poor performance metrics still did reasonably well in terms of creating a useful yield precision. For example, non-IID2-AE-RF had 0.05 R^2 performance, which suggests the model was completely failing, but when we look at the precision map (Figure 4.6.2e), we can see that low yield areas were still properly identified even though the scale of the yield was otherwise underestimated for the entire field. In fact, although non-IID AE and FL experiments had poor performance, in terms of their ability to generate useful yield precision maps, FL did quite well compared to AE. The non-IID2-AE-RF, non-IID2-AE-sCNN, non-IID1-AE-RF, and non-IID1-AE-sCNN maps (shown in Figures 4.6.2e, 4.6.2m, 4.6.2u, and 4.6.2w, re-

spectively) illustrate that the AE method led to yield being underestimated in high yield areas, but AE could correctly identify low yield areas, especially in the non-IID2-AE-sCNN case. These AE spatial prediction patterns can also be observed in the precision maps of the smaller fields in Figures 4.6.3 and 4.6.4, where we can see that the high yield areas were generally underestimated. Continuing our analysis of Field 11 map spatial patterns, what can also be observed in Figures 4.6.2i and 4.6.2r is that for IID-AE experiments, the maps produced were quite representative of the areas of high and low yield in the actual yield dataset. Even FL was also able to produce a reasonable map when using IID data, specifically, for the IID-FL-sCNN experiment (see Figure 4.6.2t). Even with non-IID data, specifically in the non-IID2-FL-sCNN case, a useful map that contained identifiable yield spatial patterns (present in the actual yield map) was still achievable such that high and low yield areas were represented reasonably well (see Figure 4.6.2o). What is interesting from observing the non-IID1-FL-sCNN map in Figure 4.6.2x is that we can see the map generally fails to capture the yield trends of the actual dataset (ignoring the fact that the map is more coarse grained due to random sampling being applied). This is probably because the non-IID2-FL-sCNN experiment used the entire Field 11 dataset to train models, whereas the non-IID1-FL-sCNN case only used 500 samples per field. This means the smaller fields, which had generally lower yield, were introducing a bias in the model that favored predicting low yield. This hypothesis is supported by the yield prediction patterns of the smaller fields shown in Figure 4.6.4. In the case of non-IID1-FL-sCNN (see Figure 4.6.4f), the maps captured reasonably well the spatial trends of the actual yield of the smaller fields, whereas for Field 11 in the same case, the yield spatial trends were not well represented in the non-IID1-FL-sCNN map (see Figure 4.6.2x). Further evidence that supports the idea that the client with the largest dataset will hinder performance of the other clients that have different dataset distributions, can be observed in the Field 11 and smaller field maps of the non-IID2-FL-sCNN experiments in Figures 4.6.2o and 4.6.4k, respectively. The yield was overestimated in the smaller fields, and this resulted in a less useful map. On the other hand, the map of Field 11 captured the spatial variability of the actual yield reasonably well. Note that in the FL-sCNN maps shown in Figures 4.6.2 and 4.6.4, in none of the non-IID experiments did every client have a reasonably accurate precision map. Either Field 11 had a poor map and the smaller fields had a reasonable map, or vise-versa, the smaller fields had a poor map and Field 11 had a reasonable map. This finding is important, since it furthers the narrative that performing distributed learning (via EL or FL) on agri-data (that are typically heterogeneous) poses many challenges. Deploying a PA model that can benefit multiple farmers equally is not straightforward, especially when farms of varying size co-exist in the SFS.

In terms of the ability of CL, LL, and SE to properly produce maps, in general, the maps were reasonably accurate. However, although the LL maps were (and LL's performance was) reasonably accurate, in practice LL will likely fail to extrapolate to new regions of interest. It is more interesting that CL and SE produced accurate maps, since they included data from various fields to build the maps and the models were not limited to a single region of interest. Note that we made a point earlier in this section about why CL and SE applied in this setting poses challenges, and that these challenges render CL and SE as poor choices compared to distributed ML.

Table 4.6.6: Yield prediction experiments: list of learning method performance average rankings from the Friedman test considering only deep learning model results, where a lower ranking number indicates better ranking performance, and marked in bold are the best performing learning methods[1]; AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

			Average Performance Ranking for Each Learning Method				
Client Dataset Split	**Metric**	**Friedman Test**	**AE**	**CL**	**FL**	**LL**	**SE**
IID	R^2	Rejected	3.13	2.50	4.57	3.77	**1.03**
IID	RMSE	Rejected	2.83	3.67	3.83	3.63	**1.03**
IID	MAPE	Rejected	3.03	3.27	3.80	3.83	**1.07**
non-IID	R^2	Rejected	4.23	**1.23**	4.53	3.10	1.78
non-IID	RMSE	Rejected	4.10	2.58	4.75	2.57	**1.00**
non-IID	MAPE	Rejected	4.12	2.27	4.78	2.77	**1.07**

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

Takeaway. Using a multi-farm dataset will likely lead to better model performance than using only a single-farm dataset. CL and SE were the methods that produced the best model performance. Unfortunately, a) CL does not satisfy privacy concerns and is not scalable, since all the data must be sent to the cloud to train a centralized model, and b) SE requires every client to have labeled data to train a local meta-learning model, even new clients that join the live system after models were initially trained. By using IID data, both AE and FL produced reasonably accurate yield precision maps, but assuming agri-data are IID is not reasonable. By using non-IID data, the AE and FL methods led to poor model performance and they generally under-performed compared to the other learning methods. However, the precision maps generated by using AE and FL were reasonably accurate despite the poor model performance: AE maps identified low yield areas; and for clients that shared similar yield dataset distributions, the maps generated by FL could be used to reasonably identify high and low yield areas.

4.6.3.5 Resource Utilization

In this section we discuss the resource utilization of the experiments. Table 4.6.7 presents the total simulated network communication cost and execution time of the models, where we only focus on the deep learning model communication cost and leave it as future work to explore network bandwidth usage of the ML models. Note that the execution times discussed in this section are the results of serial execution of models. There was no parallel client model execution performed. Furthermore, reading and pre-processing the datasets took dozens of minutes for each experiment. We did not include this data pre-processing time in the execution time results discussed in this section.

In terms of execution time, AE and SE experiments took at least as much time as LL experiments to run, since LL was a sub-process of AE and SE (local models produced from LL were required to build ensemble models). Furthermore, we can see that the ML

models took less time to run than the deep learning models. We can also see that non-IID2 experiments took the longest to run, IID experiments took the next longest to run, and non-IID1 experiments took the least amount of time to run, which makes sense, since for a larger number of samples (non-IID2 had the most samples) ML experiments should take longer.

In terms of network communication, we can see that non-IID1-CL experiments had less network communication than non-IID2-CL and IID-CL experiments. This make sense, since non-IID1 experiments involved randomly sampling 500 instances from each client's dataset, so less data was sent to the cloud. Furthermore, CL-CNN experiments had more network communication than CL-MLP experiments. This was because the pre-processed data was sent over the network, and since the CNN models used 3×3 input tiles, their samples were larger than MLP samples. In practice the pre-processing of the data could be done in the cloud, meaning MLP and CNN models would share the same communication costs in the CL setting. We can also observe that FL experiments had approximately 9 to 25 times more network communication than CL experiments. This is because the dataset used in the present work's experiments was relatively small. If the dataset was much larger, then sending model updates over the network over multiple rounds of training would result in less bandwidth usage than sending a very large dataset over the network. What is important to note is that the AE and SE experiments required approximately 20 times less bandwidth than the FL experiments, and in one case (for MLP, since the model was smaller than the CNN models) the AE and SE experiments required less bandwidth than CL experiments.

Takeaway. Deep learning and ML models took the most and least amount of time to run, respectively. Experiments that had a large number of samples in their datasets took more time to run than experiments that had smaller dataset sizes. FL experiments used the most network bandwidth compared to all other learning methods, and LL experiments used the least amount of network bandwidth. SE and AE experiments were similar to (and at times better than) CL experiments in terms of network bandwidth usage. Even though FL experiments used the most bandwidth, in practice, larger datasets will be used, and that is when FL would require less bandwidth than CL.

4.6.4 Conclusion

In the present work we performed yield prediction regression experiments using UAV imagery and yield data gathered from a single study site (a Canadian smart farm named Area X.O). We fused the imagery and the yield datasets together using a mean-, maximum-, and minimum-filter. We created three types of datasets, two mono-temporal datasets and a multi-temporal dataset. We performed model selection using feature selection and hyperparameter tuning via random search. Features were selected using PLSR feature selection, and the selected features were used in the hyperparameter tuning process. There was no significant difference in performance between the mono-temporal and multi-temporal datasets, so we chose to use the multi-temporal dataset for the main experiments of the study. We simulated a network of six clients to run distributed ML experiments. Specifically, we conducted CL, LL, SE, AE, and FL (using FedAvg) experiments using the models

Table 4.6.7: Yield prediction experiments: total simulated network communication cost and execution time of the IID and non-IID experiments, where the network communication costs of the non-deep-learning models were not considered, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and N/A = not applicable.

		Network Communication Cost (MB)			**Execution Time (min)**		
Model	**L.Method**	**IID**	**non-IID1**	**non-IID2**	**IID**	**non-IID1**	**non-IID2**
MLP	AE	14.8	14.8	14.8	12.3	8.8	15.6
MLP	CL	36.8	5.6	51.0	6.7	2.0	9.9
MLP	FL	318.4	318.4	318.4	748.1	535.7	949.1
MLP	LL	0.0	0.0	0.0	12.1	8.7	15.4
MLP	SE	14.8	14.8	14.8	12.3	8.8	15.7
LR	AE	N/A	N/A	N/A	0.0	0.0	0.0
LR	CL	N/A	N/A	N/A	0.0	0.0	0.0
LR	LL	N/A	N/A	N/A	0.0	0.0	0.0
LR	SE	N/A	N/A	N/A	0.0	0.0	0.0
rCNN	AE	77.8	77.8	77.8	10.7	7.2	15.0
rCNN	CL	64.0	10.8	90.4	7.3	1.7	11.5
rCNN	FL	1672.4	1672.4	1672.4	640.8	422.7	906.1
rCNN	LL	0.0	0.0	0.0	10.5	7.1	14.8
rCNN	SE	77.8	77.8	77.8	10.7	7.2	15.1
RF	AE	N/A	N/A	N/A	1.6	0.3	2.5
RF	CL	N/A	N/A	N/A	1.9	0.2	2.7
RF	LL	N/A	N/A	N/A	1.5	0.3	2.4
RF	SE	N/A	N/A	N/A	1.6	0.3	2.6
sCNN	AE	49.8	49.8	49.8	13.5	8.6	19.5
sCNN	CL	64.0	10.8	90.4	10.1	2.2	15.9
sCNN	FL	1071.1	1071.1	1071.1	811.7	514.9	1209.5
sCNN	LL	0.0	0.0	0.0	13.3	8.5	19.3
sCNN	SE	49.8	49.8	49.8	13.5	8.6	19.5
ZeroR	AE	N/A	N/A	N/A	0.0	0.0	0.0
ZeroR	CL	N/A	N/A	N/A	0.0	0.0	0.0
ZeroR	LL	N/A	N/A	N/A	0.0	0.0	0.0
ZeroR	SE	N/A	N/A	N/A	0.0	0.0	0.0

that resulted from the model selection step. We also varied the heterogeneity of the datasets for each learning method by considering three types of client dataset partitioning schemes, namely an IID split and two types of non-IID splits (non-IID1 and non-IID2).

We proceed to discuss our findings. By including the LocMem baseline model in our analysis, we found that data-driven models can be used to outperform a farmer's intuition and knowledge of his or her field for estimating yield. RF, CNN, and MLP were the best performing models. Using multi-field datasets can lead to better performance than using single-field datasets. CL and SE were the best performing learning methods, although CL does not respect farmers' privacy concerns and does not scale well, and SE requires new clients (clients that join after the distributed training is complete and the system goes live) to have labeled data. AE and FL somewhat respect farmers' privacy concerns and new clients do not need labeled data to benefit from a deployed PA model. Although LL outperformed FL, and in some cases outperformed AE, our analysis only involved data from a single study site, so in practice we expect LL to fail to generalize to new locations. Therefore, we focus on AE and FL performance. In the optimistic case where client datasets were more homogeneous (in the IID experiments), AE and FL produced reasonable performance, but in the more realistic scenario that involved heterogeneous client datasets (in the non-IID experiments), AE and FL generally failed in terms of performance metrics. Nevertheless, despite the poor performance for the non-IID experiments, AE and FL were still able to produce relatively useful yield prediction maps. AE was able to identify low yield areas and FL could identify both high and low areas for most clients. This illustrates the importance of including precision maps (in addition to evaluation metrics) to an analysis when evaluating model performance involving spatial datasets. Our results showcase that our proposed privacy-aware smart farming architecture can be applied to yield prediction. It is important to keep in mind that even though FL and EL do not share private data, these approaches still have privacy issues that our proposed architecture does not explicitly address. Model updates (and local models) may still reveal sensitive information [195, 63]. Privacy preserving technologies should be applied to better meet cybersecurity requirements [195, 6].

We discuss future work as follows: a) including yield and imagery from different farms/regions, b) investigating the use of Personalized FL instead of FedAvg to tune the global model to the unique distribution of each client's local dataset [244, 279, 147] so that all clients can benefit from the global model, c) including a leave-one-client-out CV evaluation methodology to the study, to evaluate the ability of models to deal with new clients joining the system, d) evaluating a variation of AE that uses the maximum of local model predictions instead of the average. This could be interesting, since AE was able to reasonably identify low yield areas, so maybe the maximum operation could help identify high yield areas as well; e) evaluating a variation of AE, k-nearest-clients, that involves a new client i forming an ensemble of models using local models only from the k nearest clients in geographical space. The idea would be that the models of these k clients would more likely capture local weather and other environmental trends than models from clients further away; f) evaluating a variation of AE that uses weighted averaging by defining the weights based on the average yield of the local models' associated field. This could improve performance, although it does not meet the privacy requirements of the setting, so privacy

preserving technologies like homomorphic encryption would need to be applied to safeguard the shared yield information; g) evaluating 3D CNNs (similar to the methodology applied in Bisht et al. [29]) to better capture the multi-temporal trends in the features, instead of only using 2D CNNs, h) investigating whether the local model predictions used to build the meta-model in SE were spatially autocorrelated to determine if the SE performance in the present work was over-optimistic, and i) evaluating precision map accuracy using a map evaluation technique that involves "probability sampling and design-based inference" [273] for a more appropriate and unbiased map accuracy evaluation.

Chapter 5

Nitrous Oxide (N_2O) Emission Prediction

5.1 Introduction

Agriculture is the main source of food for our population [8]. Food production is expected to be threatened by climate change, declining fresh water availability, and a reduction in the amount of arable land available [268], and by 2050, there will likely be a noticeable increase in food consumption as a result of population growth [229]. By 2050 the world population is estimated to reach 9.73 billion, and by 2100 it is expected to reach 11.2 billion [46]. Dealing with this rising food demand will require an increase in food production from farmers in an environmentally sustainable manner, but this is not feasible using traditional agricultural practices [220].

PA and smart farming can be used to address these challenges for increased food security [10]. PA involves observing and analyzing agri-data to respond to the data's variability, whereas smart farming (otherwise known as Agriculture 4.0) extends the notion of PA by including real-time information technology and predictive analytics for decision making [224, 229]. Smart farming is a management practice that makes use of a) IoT, b) cloud computing, c) fog/edge computing, and d) artificial intelligence to optimize agricultural practices [66, 277]. Smart farming provides advantages, including information exchange, safety control, food security, fraud prevention, and business cost reductions [66]. The cost and benefits of a SFS should be clear to a farmer and affordable to make it worthwhile for the farmer [16]; a SFS should also be simple to use. Otherwise, a SFS may suffer from low adoption rates if a) it is too expensive [144], b) it was not built using a user-centric design [152, 144], and c) it is too complex and requires too much technical knowledge from the farmer [246].

One of the applications of smart farming is N_2O emission prediction. N_2O is a powerful GHG that has approximately 273 times more GWP over a 100-year time horizon than CO_2 [73]. The agriculture industry is responsible for about a third of all GHG emissions [10]. With GHG reduction strategies put in place by nation states, this puts pressure on

farmers to take action, since N_2O creation is an unfortunate consequence for effectively growing crops using fertilizers [75]. An example of such a reduction strategy is the one put in place by the Canadian government that expects Canadian farmers, by 2030, to reduce their GHG emissions from synthetic fertilizers by 30 % [58]. However, we must first be able to measure emissions before identifying and evaluating the feasibility of an emission mitigation strategy. Unfortunately, measuring N_2O emissions is not a trivial task, since a) N_2O emissions have high spatiotemporal variability [216] and b) it is difficult to obtain high resolution N_2O emission datasets in both the temporal and spatial dimensions due to the high cost of continuous emission measuring equipment [115, 197, 210]. It is not feasible to expect farmers to monitor accurate N_2O emissions using expensive sensing equipment, but it is possible to instead predict emissions using PA models to produce affordable rough emission estimates. Data-driven models can make high spatiotemporal resolution predictions using inexpensive proximal sensor measurements [121, 116].

PA models can be trained by sending all the data from the network's edge to the cloud [193] and then performing CL. CL involves combining all the data sources into a single centralized dataset before training a model. This approach unfortunately suffers from drawbacks including: a) not being scalable [195]; b) suffering from power and wireless network coverage limitations, which is a common limitation in agricultural fields [268]; and c) having poor privacy, which may hinder SFS adoption rates [242, 119, 63]. Privacy is important, since low system adoption rates reduces the number of farmers that join SFSs. Limiting a SFS's farmer user-base would result in less available training data, and this could in turn negatively impact PA model performance. Fog computing can instead be applied to address the limitations of CL by running analytic processes at the edge of networks [285]. This enables LL. LL involves training many local models using only local data [195], but unfortunately, applying LL to agri-data may lead to loss of useful geographical trends that could have been useful to a centralized model [195, 229]. These limitations can be addressed by using model sharing combined with distributed EL [63, 199] and/or FL [293, 63, 271, 33], both of which are forms of distributed ML. In both cases, models are trained at the edge of the network (specifically, in the fog) on client devices, which reduces network communication and somewhat satisfies privacy requirements [285, 30, 63, 4], using only local data such that private data is never shared. Only model updates are shared. The result is a global PA model (in the FL case) or a collection of local PA models (in the EL case) that can be used to benefit farmers for agriculture-related tasks by making use of model predictions [30, 63].

The remainder of the chapter is organized as follows: Section 5.2 adapts our smart farming IoT architecture proposed in Chapter 3 to the N_2O emission prediction setting; Section 5.3 describes the study site, the data sources, and some of the common methodologies applied by each of our N_2O emission prediction studies described in the remaining sections of this chapter; Section 5.4 discusses a study involving single-year interpolation (gap-filling) and multi-year extrapolation experiments using CL, temporal CV, soil sensor data, and weather sensor data; and Section 5.5 discusses a study involving single-year interpolation (gap-filling) experiments using distributed ML, temporal CV, soil sensor data, and weather sensor data.

5.2 Adapted IoT Architecture

In this section we explain how we adapt our architecture proposed in Chapter 3 to the N_2O emission prediction use case, how much deploying such a system would cost, how farmers can use and benefit from the system, and the adoption barriers of deploying such a system. We present a prototype PN later in this chapter in Section 5.3.3.

The adapted architecture can be seen in Figure 5.0.1. The components of the architecture that we adapted are the PN's internal components. We left the RN as is, such that the RN still hosts the FL coordination server and still broadcasts local models to each client when using EL. We left the logical placement of the IoT layers unchanged. We left the role of the PN's gateway unchanged. The gateway remains as a distributed ML client device that performs local predictive analytics. What we did change in the adapted architecture is as follows: the PN's gateway in this use case is a Raspberry Pi (or other type of microprocessor/microcontroller) with lightweight computational resources. The gateway is deployed in a field, is powered by battery (and possibly solar panels), and has Internet connection access via a long-range wireless connection (e.g., via Global System for Mobile Communications (GSM) or LoRa). The gateway has up to three types of SNs, namely soil SNs, weather SNs, and a LI-COR automated gas chamber for N_2O emission sensing. The soil and weather SNs may be connected to the gateway via a wireless or wired (via RS485) connection. The LI-COR equipment is the cold-start sensing equipment in this use case, where the gateway polls N_2O emission data from the LI-COR SN via MQTT over Wi-Fi. This is a straightforward design, since the LI-COR SN hosts a Wi-Fi network and a MQTT broker as an interface for data polling. The gateway does not host any applications. Once in the live state, the system can then act as an inexpensive solution for mapping regional N_2O emissions at high temporal resolutions, since many additional PNs can then be deployed to cover large spatial regions. The N_2O predictions will be uploaded directly to the RN, which should be feasible from a scalability perspective, given the relatively slow data generation speeds (e.g., predictions made every 30 minutes) of this setting. The emissions will be stored in a database on the RN and will be made available by a cloud webservice that allows farmers and government officials to access N_2O emission readings.

5.2.1 System Deployment and Costs

In this section we discuss a) how to deploy the proposed SFS for predicting N_2O emissions using soil and weather sensor data and b) the associated costs.

- When the system is first deployed, a sensing solution based on an automatic gas chamber or flux tower would be needed to gather high temporal resolution N_2O emission datasets. By observing Table 2.1, estimated costs of such sensing solutions range from $100,000 CAD to $350,000 CAD. Once the system goes live, this cost can be eliminated for farmers that join the system.
- When the system is first deployed and when the system goes live, a gateway equipped with sensors for gathering weather and soil data will be required to create the PN.

The Pessl SNs used in our case studies cost approximately $750 CAD, so a rough estimate of the cost of buying a SN that offers weather and soil data could range from $500 CAD to $1,500 CAD. In our 2023 case study, our PN prototype described in Section 5.3.3 cost us approximately $1,000 CAD to build.

- There may be cellular data service fees associated to the PN-to-RN network connection, given that the PN is deployed in the field. For reference, during our studies at Area X.O, there were at one point 44 Pessl SNs deployed across the entire farm. They were each connected to the internet via 4G, and the annual cellular data service fees for all the Pessl SNs cost approximately $2,000 CAD ($45 CAD per device).

5.2.2 Benefits of Using the System

The proposed SFS applied to N_2O emission predictions enables applications such as regional mapping, climate change modeling, legislation enforcement auditing, carbon tax rebates, and the potential to reduce emissions by allowing farmers to observe if their emission counter-measure actions have succeeded in reducing emissions. If emissions failed to be reduced, the farmer could try applying another type of fertilizer application technique until the predictions reported satisfy emission sustainability targets. Achieving reduced emission targets means farmers could be credited carbon tax rebates by the government, leading to direct profits for the farmers by adopting our proposed system. The proposed system could also be used by agriculture scientists to test different fertilizer application techniques for designing techniques that optimize N_2O emission reductions. Another benefit is including N_2O monitoring in precision fertilizer application design. A fertilizer application technique that maximizes yield, reduces fertilizer usage, and minimizes N_2O emissions (e.g., seed-placed fertilizer banding) could lead to direct profits for farmers via reduced fertilizer costs (less fertilizer used) and increased crop yields, while respecting GHG reduction sustainability targets. The proposed system will come at the cost of being less accurate than the LI-COR automated gas chamber measuring technique. Nevertheless, with reasonable ML prediction accuracy, the system should still be useful for gathering emission trend estimates and identifying high emission events (or hot-spots).

5.2.3 Adopting Barriers

Adoption barriers of the proposed SFS applied to N_2O emission prediction are a) the high cold-start equipment costs and b) the lack of financial incentives for farmers.

- High cold-start equipment costs could be addressed by reducing costs via subsidies or renting the N_2O monitoring equipment. Farmers could rent this sophisticated sensing equipment from the government for a reasonable price or could purchase the equipment at a discount via subsidies.

- It may not be clear to farmers how monitoring their N_2O emissions via our SFS's predictions will benefit them financially, meaning they will not want to use our system.

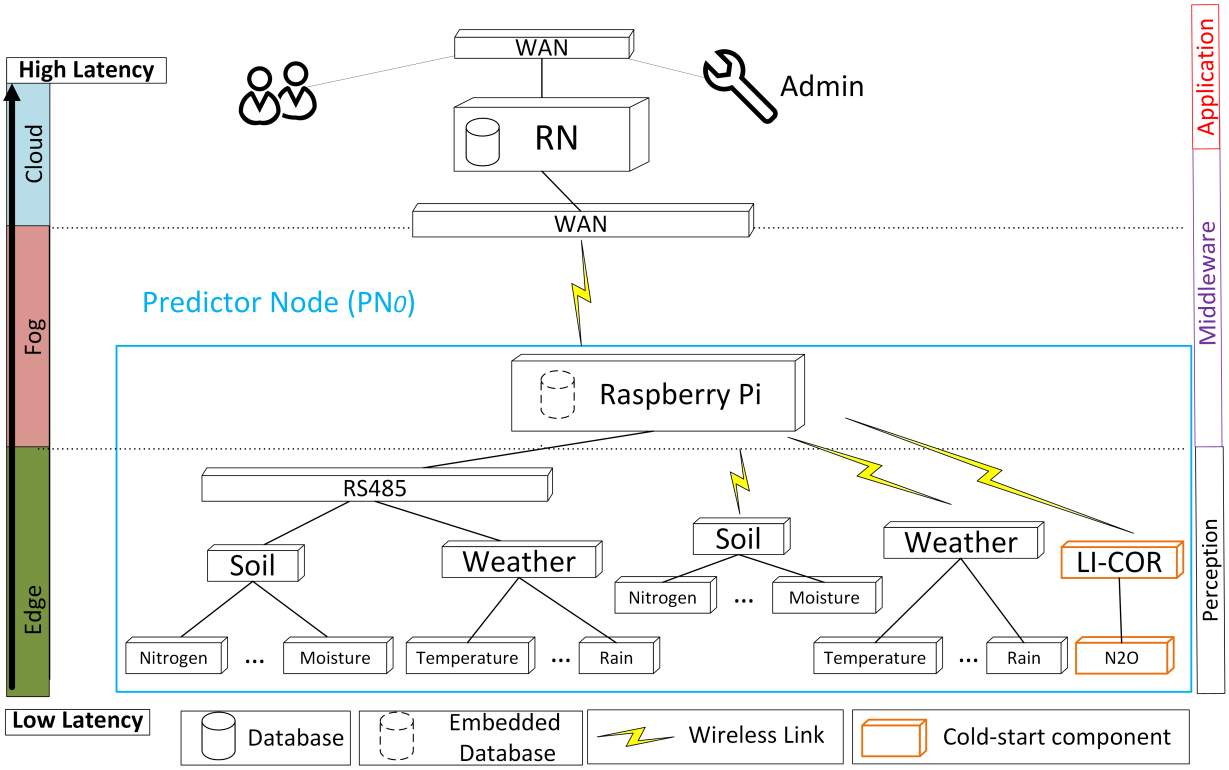
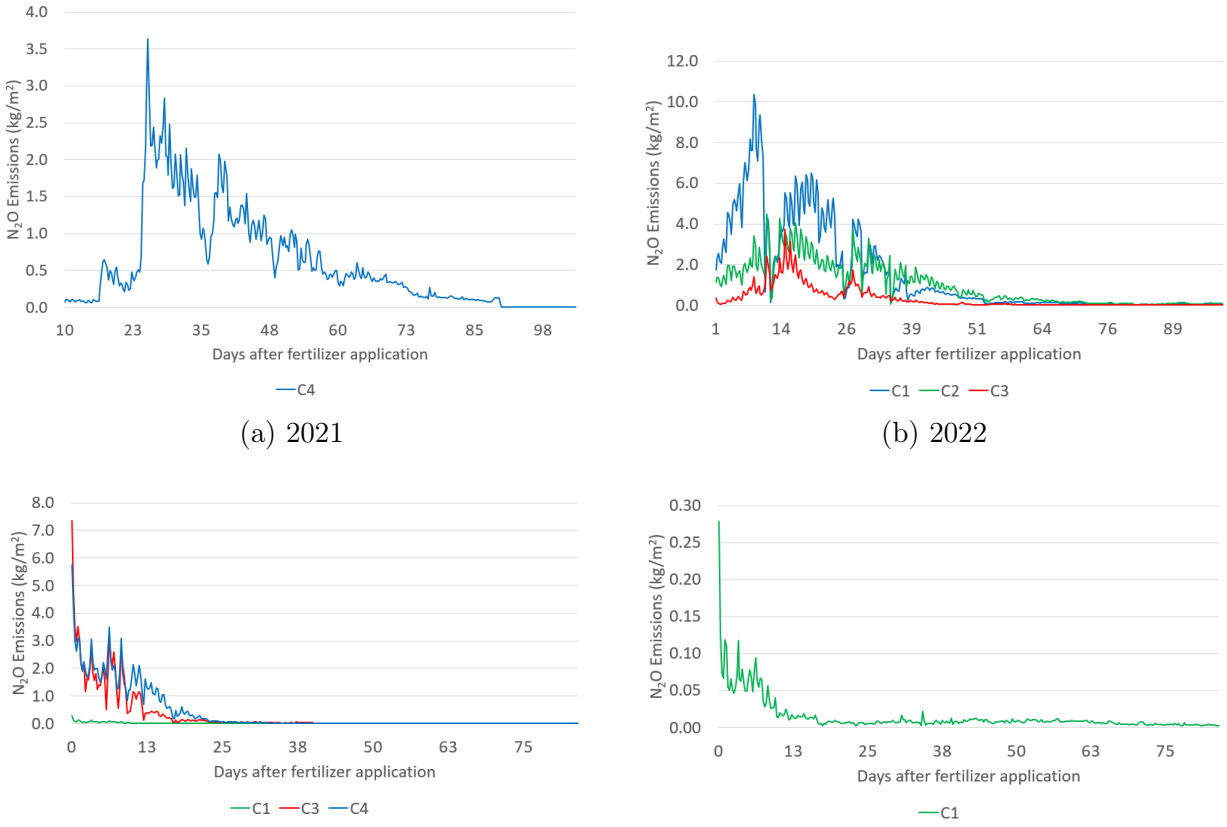


Figure 5.0.1: Overview diagram of the proposed privacy-aware smart farming IoT system architecture that was adapted to the N_2O emission prediction use case, where WAN = wide-area-network and RN = root node. To monitor a target agri-variable, the cold-start components (the orange components) are only required to train models when the system is first deployed.

Without external funding or incentive systems, predicting N_2O emissions may not directly help farmers produce more from their fields and may not help farmers lower their input costs. Incentive systems would need to be put in place to a) help incentive farmers to monitor their N_2O emissions (via our SFS's predictions), b) have the farmers reduce their emissions to meet GHG reduction targets (via using a sustainable fertilizer application practice, for example), and c) then financially compensate the farmers who meet the GHG reduction targets.

5.3 Materials and Methods

This section explains the methodology we applied for gathering sensor data and preprocessing the data, and we discuss evaluation techniques used.



(a) 2021

(b) 2022

(c) 2023

(d) 2023 (zoomed-in)

Figure 5.0.2: Line plots of N_2O emissions (scaled to a 6 h resolution) time series plot for each year, where Ci indicates chamber i for the given year.

5.3.1 Study Site

Corn was grown in a test plot during the 2021, 2022, and 2023 growing seasons at a smart farm named Area X.O located in Ottawa, Ontario, Canada (45°19'8.17"N, 75°45'22.40"W). The test plot was approximately 20 m × 20 m and was located in the northern part of Field 2. Table 5.0.1 describes the management practices applied in this study. For the 2021, 2022, and 2023 growing seasons, the average air temperature was 20.7 °C, 14.9 °C, and 19.4 °C, respectively, and the total rainfall was 267.5 mm (a dry year), 536.3 mm, and 535.5 mm, respectively. Figures D.3 and D.4 in Appendix D illustrate time series plots of temperature and precipitation, respectively, over each of the seasons. Figure 5.0.2 illustrates the N_2O emissions for each year plotted relative to the time since fertilizer was applied.

5.3.2 Data Collection

Field data was collected during 2021, 2022, and 2023 growing seasons from in-field SNs, namely N_2O emission sensors, soil sensors, and weather sensors. The sensors were deployed

Table 5.0.1: The N_2O gas chambers by a) the notation used for identifying the chambers, b) tilling techniques applied, c) fertilizer information, and d) whether crops were grown in the row where the chamber was placed; ID = identification number, Conv. = Conventional, DoY = day of year, Ur = urea, PCUr = polymer-coated urea, and UrIn = urea coated with urease inhibitor.

N_2O Target Variable Name	Year	Gas Chamber ID	Tilling Technique	Fertilizer Applied	Fertilizer Application Date (DoY)	Fertilizer Type	Has Crops
$C_{2021,4}$	2021	4	Conv. Broadcast	225 kg/ha	153	Ur & PCUr	no
$C_{2022,1}$	2022	1	Conv. Broadcast	225 kg/ha	209	Ur & PCUr	no
$C_{2022,2}$	2022	2	Conv. Broadcast	225 kg/ha	209	Ur & PCUr	yes
$C_{2022,3}$	2022	3	Conv. Strip Till	225 kg/ha	209	Ur & PCUr	yes
$C_{2023,1}$	2023	1	Conv. Broadcast by PN	150 kg/ha	185	Ur & PCUr	yes
$C_{2023,3}$	2023	3	Conv. Broadcast	150 kg/ha	185	UrIn	yes
$C_{2023,4}$	2023	4	Conv. Broadcast	150 kg/ha	185	UrIn	no

in Field 2, mostly around the 20 m × 20 m test plot area. Figure D.5 in Appendix D illustrates a map of the field and the sensor deployment locations.

- We collected N_2O emission (or flux) data using LI-COR automated gas chambers. The equipment consisted of a LI-8250 Multiplexer, a LI-780 N2O/H2O Trace Gas Analyzer, and multiple LI-8200-104 Opaque Long Term Gas Chamber. Emission samples from up to seven chambers were taken every 30 min during each year's growing season.
- Soil temperature, soil moisture, and weather data were collected using nearby (within approximately 15 m to 200 m) cloud-connected Pessl SNs. Samples were taken at sub-hourly intervals. The soil readings were available at various depths (10 cm, 20 cm, 30 cm, 40 cm, 50 cm, and 60 cm).
- During the 2023 growing season, we collected additional soil data properties (which were not offered by the Pessl sensors) via an IoT soil sensor gateway PN prototype that we describe next in Section 5.3.3. The available soil sensors were moisture, temperature, NPK, and pH, and samples were taken every 15 min. We refer to this gateway as the PN in the remainder of the chapter.

The LI-COR and PN data were stored locally, and periodically, data were manually extracted via on-site visits. The LI-COR equipment and PN were powered by a persistent 120 V AC power supply enabled by a long extension cable, and the Pessl SNs were powered by battery and solar panels. Readers interested in summary statistics of the sensors' raw datasets are referred to Table D.3 in Appendix D, and those interested in sampling rates, SN models, and specific soil and weather parameters monitored, are referred to Table D.4 in Appendix D.

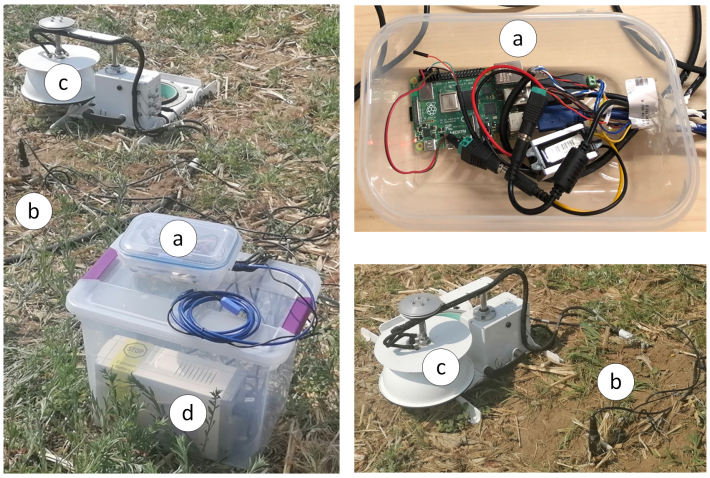


Figure 5.0.3: N_2O prediction experiments: PN in-field setup in 2023, where the left image shows both a LI-COR gas chamber and in-field PN setup, the top-right image shows the PN's gateway components, and the bottom-right image shows the sensor probe placement in relation to a LI-COR gas chamber; where (a) = PN's gateway, (b) = the PN's soil sensor probes, (c) = LI-COR automated gas chamber, and (d) = the gateway's uninterruptible power supply.

5.3.3 Predictor Node Prototype

During the 2023 growing season, we collected soil data via a PN prototype that we built to showcase our proposed architecture in Chapter 3. The data gathered from the PN enabled the studies presented later in this chapter. The PN was build using a Raspberry Pi 4 Model B with 4 GB of RAM and a 32 GB SD card, an American Power Conversion uninterruptible power supply Model BK350, an Adafruit Ultimate GPS sensor for clock system synchronization, and four RS485-based SNs connected to the Raspberry Pi via an RS485-to-USB adapter. The total component cost for building the gateway was approximately 1000 CAD $, which was a fraction of the cost of the LI-COR equipment[1]. The PN SNs formed an RS485 sensor network and were individually addressed for data polling, by querying each sensor using the sensor's unique RS485 address. The GPS sensor was used for system clock synchronization. Clock synchronization was required, because the Raspberry Pi's internal clock would desynchronize relatively quickly (e.g., the clock was offset by a few hours after only a week elapsed), and an accurate timestamp for soil samples was necessary for properly fusing the soil data with a) the N_2O emission data and b) weather data from nearby Pessl SNs. The prototype was not used to run any predictive analytics logic. It was a minimal prototype that was deployed in a field in 2023 and was only used to gather soil sensor data. Figure 5.0.3 shows a diagram of the PN and gas chamber $C_{2023,1}$ setup in 2023. Figure 5.0.4 shows a SysML diagram of the prototype PN's hardware configuration, and Figure D.6 in Appendix D shows the complete diagram of all the components and wiring of the PN.

[1]Each automated LI-COR gas chamber cost us approximately 20,000 CAD $.

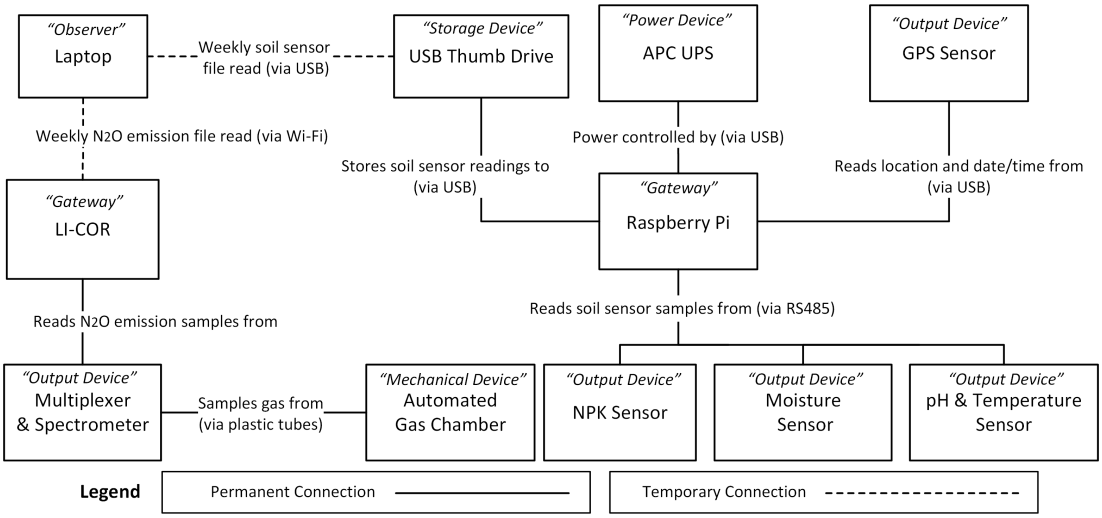


Figure 5.0.4: N_2O prediction experiments: SysML [85] diagram illustrating the high-level hardware components involved with the prototype PN , where APC UPS = American Power Conversion uninterruptible power supply and NPK = Nitrogen-Phosphorus-Potassium.

5.3.4 Evaluation Methods

In this section we discuss the applied evaluation methods. In the studies discussed later in this chapter, we used the same significance testing methodology as the one discussed in Section 4.3.6. Furthermore, note that since, in our case, N_2O emissions near 0 ppb/m^2 were common, to avoid large MAPE errors due to small emission values, we swapped small emission readings (for samples with values ranging between –1 and 1 ppb/m^2) to –1 or 1 ppb/m^2 (whichever was nearest in value) before computing the MAPE. This should be fine, since a prediction between –1 and 1 ppb/m^2 can effectively be treated as correctly predicting low baseline emissions for the datasets used in this study.

5.3.4.1 Block Cross-validation

In some of the studies discussed in this chapter we used a temporal block CV evaluation method that we describe in this section. The temporal block CV methodology (which we will refer to as block CV in the remainder of this chapter) involved first clustering (using a user-defined cluster size) the dataset into temporal blocks of consecutive samples (with respect to timestamps), and then treating the blocks as samples for the sake of randomly sampling to populate the folds. The folds were then treated the same way KF-CV treats folds to iteratively partition a dataset into k training-testing dataset pairs, such that every sample of the original dataset was part of the testing set exactly once. In some of our experiments we applied a second inner loop of block CV (nested block CV) to perform model selection. Figure 5.0.5 provides a visual example of fold creation using nested 3-2-fold block CV with a 1-day cluster size.

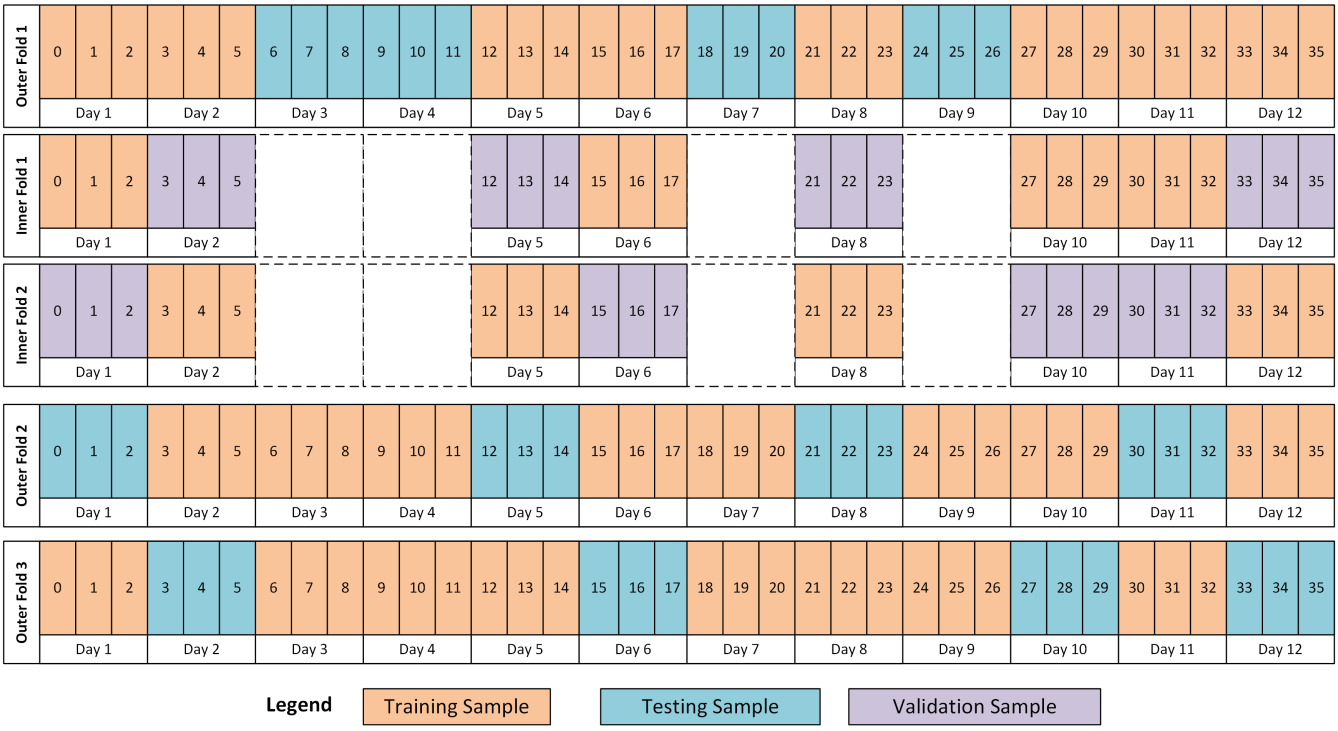


Figure 5.0.5: N_2O prediction experiments: example of partitioning a 36-sample time series dataset into training, testing, and validation datasets using nested 3-2-fold block CV with a 1-day cluster size, where the temporal resolution of the dataset is 3 samples per day, the inner CV loops for outer folds 2 and 3 were omitted for clarity.

This methodology forced training and testing datasets to have a user-defined time gap (or cluster size) between most[2] of their samples. This methodology effectively evaluated a model in terms of its ability to make predictions for gap-filling tasks. Using smaller time gaps would be more appropriate for evaluating model interpolation performance (or gap-filling ability), and using larger time gaps would be more appropriate for evaluating model extrapolation performance.

Source of Error. There was a design issue with part of the applied sampling methodology. Samples within the blocks were not shuffled before creating folds; the samples kept their order from the original dataset. We should have shuffled the samples in each block before assigning the samples to a fold. We addressed this in later studies (the study described in Section 4.6) after identifying this source of error.

5.3.4.2 Seasonal Performance

In some parts of our analysis, we focused on model performance at different parts of the season. To perform this seasonal analysis, we broke the season into four parts, namely

[2]Some training data samples at the boundary of the blocks were chronologically adjacent to some testing dataset samples.

early season, middle season (mid-season), late season, and end of season. These parts of the season were defined using a 25-day sliding window with an offset of 25 days, from the start of a dataset's first timestamp t_0 until the end of the dataset, where the period the seasons cover is as follows:

- **Early season**: $[t_0, t_0 + 25 \text{ days})$
- **Mid-season**: $[t_0 + 25 \text{ days}, t_0 + 50 \text{ days})$
- **Late season**: $[t_0 + 50 \text{ days}, t_0 + 75 \text{ days})$
- **End of season**: any remaining samples from $t_0 + 75$ days onward

For predictions made at a given timestamp, we averaged out a model's predictions over all folds and iterations and treated this average as the model's prediction for the timestamp. This was done for all timestamps. For each part of the season, only predictions made for timestamps that fell within the desired part of a season were considered in the seasonal analysis.

5.4 N_2O Emission Prediction Study: Using Temporal Cross-validation, Single-year Interpolation, and Multi-year Extrapolation

Most of the content of this study was recently accepted for publications in Killeen et al. [137].

The present work is a study that performed N_2O emission prediction using soil and weather sensor data. CL was used to train and evaluate models. The present work attempts to address the challenge of measuring high spatiotemporal resolution N_2O emissions by making emission estimations at a fraction of the cost of using expensive automated gas chamber solutions. The present work differs from our N_2O emission prediction work described in Section 5.5 in that we included additional automated gas chamber N_2O emission datasets (obtained from the same study site) in our experiments, we included temporal features, and we performed extrapolation experiments where we trained models using one year's data to predict emissions of another year.

In terms of how the present work relates to this thesis: the temporal analysis methodologies applied in the present work tie into the spatiotemporal data analysis aspect of this thesis. The hyperparameter tuning, temporal CV techniques, and other experiments applied in the present work allowed us to more efficiently decide on what features to consider, how hyperparameters would be tuned, and what temporal CV techniques to apply in our N_2O emission prediction study presented next in this chapter in Section 5.5.

The following is a list of limitations of the present work: a) the data gathered was from only a single study site and b) we ran ML experiments on a desktop machine instead of evaluating and training models on a resource-constrained IoT device (the type of device we want to deploy the models to in a production setting).

5.4.1 Materials and Methods

5.4.1.1 Datasets

The datasets used in the present work included all available N_2O, soil, and weather data sources described in Section 5.3.2.

5.4.1.2 Data Pre-processing

Before feeding the data to the models, we analyzed the raw datasets to help us decide on the necessary data pre-processing steps to apply. Table D.3 in Appendix D presents summary statistics of the raw sensors' datasets used in the present work. After understanding our datasets, we a) cleaned them to deal with noisy or missing sensor data and b) derived additional features. We used hot deck imputation [178, pp. 18–19] to replace missing N_2O readings. Missing readings of other sensors were addressed by a) removing features with too many missing values, b) applying linear interpolation to fill in small missing blocks of data[3], and c) replacing missing values of sensors that were highly correlated with other sensors via RF regression imputation[4]. For example, we removed wind sensor features due to a large number of missing readings. We also addressed missing precipitation readings by taking advantage of the redundant precipitation sensors. After applying these cleaning steps, the datasets' number of samples ranged from 3148 to 4653. We derived three additional lower temporal resolution datasets for each chamber's 30 min temporal resolution dataset via averaging, namely 60 min, 180 min, and 360 min resolutions. This resulted in four total datasets for each of the seven chambers (a total of $4 \times 7 = 28$ datasets).

5.4.1.2.1 Feature Extraction In this section we discuss how we extracted and derived features. For each dataset, we fused soil and weather sensor data to N_2O emission data by deriving lagged features using the methodology applied by Feigenwinter et al. [70]; that is, for each N_2O sample, we applied aggregation operations (average, maximum, minimum, and/or summation) over windows of size 1 h, 3 h, 6 h, 9 h, and 12 h, such that the windows included predictor variable samples within a short history (controlled by the window size) of an N_2O sample's timestamp. For example, features derived this way included average soil moisture (at 20 cm depth) within the last 3 h and total rainfall within the last 9 h of an N_2O sample's timestamp. This resulted in datasets having 314 to 419 different features. Note that there were redundancies in some of the features in that there were redundant SNs deployed in different parts of the field[5]. In experiments not covered in this thesis, to address noisy sensors and missing readings, we left out some of the features from the nearby (15 m away) sensors and replaced the features with those derived from the further

[3]Only blocks of consecutive missing values of lengths of up to 6 h were considered for interpolation.

[4]2023 datasets were the only datasets that we used RF regression for imputation, and we obtained at least 0.85 R^2 KF-CV performance.

[5]For example, soil moisture sensors approximately 15 m away (from Pessl SN 01209E4F) and soil moisture sensors approximately 205 m away (from Pessl SN 01209E58) from our N_2O emission gas chambers

away (205 m away) redundant sensors, and we did not observe any significant impact on N_2O emission prediction model performance.

We also derived four temporal features, namely three accumulated heating unit features and an exponentially scaled days-since-fertilizer-application (DSFA) feature. Inspired from Glenn et al. [83], the three accumulated heating unit features were GDDs (see Section 4.4.1.2.1), soil heating units (SHU), and nitrification heating units (NHU) [256]. The exponentially scaled version of the DSFA feature, which we used in one of our previous works [150], was designed to simulate exponentially decreasing synthetic soil nitrogen content (SSNC).

5.4.1.3 Machine Learning Models

In this section we list the models used in the present work. The models evaluated include tabular models (both ML and deep learning models), tensor-based models, and baseline models. The tabular models were LR, RF, MLP, and SVM. The tensor-based models were LSTM and 1D CNN. The two baseline models were a) a model we named DSFA memorizer (DSFAMem) and b) ZeroR. DSFAMem assumed to have access to the DSFA feature. DSFAMem mapped the DSFA and associated emissions from the training dataset and then simply outputted the emissions from its DSFA-emission lookup-map when given a DSFA value. DSFAMem was effectively a KNN model, with $k = 1$, that only used the DSFA as a feature.

In addition to the reasons explained in Section 2.4.2 (that ML and deep learning models should be reasonable choices for N_2O emission prediction), the chosen models were included in our experiments for the following reasons:

- for implementation simplicity reasons (it was straight forward to re-use some of the models used in our yield prediction experiments described in Chapter 4);
- we chose SVM, since a) SVM was commonly used in the N_2O emission prediction literature discussed in Section 2.9.4 and b) we wanted to include another ML model in our experiments in addition to LR and RF;
- we included LSTM, since it has been found to do well at modeling time series data;
- we included 1D CNN to include another tensor-based model in our experiments in addition to LSTM;
- and we included ZeroR and DSFAMem to provide us with additional benchmarks to give us more insight into how poorly models were performing.

5.4.1.4 Model Selection

5.4.1.4.1 Feature Selection Since there were many features in the datasets, we performed feature selection for each of the 28 datasets. We performed feature selection using

the PLSR-based methodology described in Section 4.3.5.1 using 5-fold block CV with a 1-day cluster size (see Section 5.3.4.1 for details on the block CV methodology applied), and *sklearn* version 1.3.2. The number of grid search trials was 100. Note that there was a source of error such that there was a bug in the feature selection code used to implement the feature selection in the present work; the last feature of each dataset never had a chance to be selected, but given the large number of source features, this bug was likely not that impactful. The resulting number of features that were selected ranged from 48 to 153 for each dataset.

5.4.1.4.2 Hyperparameter Tuning For tuning hyperparameters, we used the methodology described in Section 4.3.5.2 using nested block CV and/or KF-CV. The hyperparameter search spaces (and the neural network architectures for the deep learning models) for RF, MLP, 1D CNN, LSTM and SVM are described in Appendix C in Tables C.1, C.2, C.4, C.7, and C.8 respectively.

Both types of tensor-based models were evaluated using various tensor lengths (number of time steps for an input sample) to treat the tensor length as a hyperparameter being tuned. We selected two tensor lengths (small and long lengths) that led to the best model performance, and as a result, we included two versions of each tensor-based model, one with a small number of tensors (LSTM_S and CNN_S) and one with a larger number of tensors (LSTM_L and CNN_L). Since the datasets in this study had varying temporal resolution, the tensor length used for shorter vs. longer tensor lengths depended on the temporal resolution of the dataset. We list the tensor lengths as follows: for models using the longer tensor lengths, 56 (28 h), 32 (32 h), 14 (42 h), and 8 (48 h) tensors were used for datasets of 30 min, 60 min, 180 min, and 360 min temporal resolutions, respectively. For models using the shorter tensor lengths, 16 (8 h), 8 (8 h), 10 (30 h), and 5 (30 h) tensors were used for datasets of 30 min, 60 min, 180 min, and 360 min temporal resolutions, respectively.

5.4.2 Experiments

In this section we discuss the details of our experiments. Our goal was to evaluate the accuracy of the predictions made by the models. The evaluation metrics used were RMSE, R^2, and MAPE. The ML and deep learning models were implemented using the same execution environment as the one discussed in Section 4.6.2, except that we did not use the *geopandas* package in the present work. We trained and evaluated models using the same machine as the one described in Section 4.5.2. The datasets and scripts used to run the experiments discussed in the present work are open source and can be found on GitHub[6].

Two types of experiments were conducted, interpolation experiments and extrapolation experiments, and they are discussed in Sections 5.4.2.1 and 5.4.2.2, respectively.

[6] https://github.com/patkilleen/extrap_n2o_prediction_2024, accessed on February 25 2025

5.4.2.1 Single-year Interpolation Experiments

In these experiments we evaluated the gap-filling ability of various models using datasets that only included data from a single year, chamber, and temporal resolution. In these experiments we performed one iteration of nested 5-3-fold block CV with a 1-day cluster size. We used 50 trials of random search using a single execution per trial for hyperparameter tuning. The features we used were those resulting from the PLSR feature selection step. We avoided adding the temporal features to these experiments to avoid having the strong temporal structure of these features introducing over-optimistic interpolation performance. Each of the 28 datasets was included in these experiments. We used min-max scaling to normalize the data between 0 and 1 before splitting the data into training-testing pairs. All models described in Section 5.4.1.3 were included in these experiments except for DSFAMem.

5.4.2.2 Multi-year Extrapolation Experiments

In these experiments we evaluated the cross-year extrapolation performance of models by training models using one year's data and testing models using another year's data. Furthermore, we only considered $C_{2021,4}$ and $C_{2022,1}$ datasets in this analysis, since they were the only cross-year chamber pairs that shared the same fertilizer management practices. For notation simplicity, we will refer to experiments that a) trained models using 2021 data and tested models using 2022 data and b) trained models using 2022 data and tested models using 2021 data, as MY_{2021} and MY_{2022} experiments, respectively. We only used 180 min resolution datasets in order to speed up the experiments. Furthermore, we also examined the effects of excluding vs. including the SSNC feature on model performance. We refer to experiments that used datasets that included the SSNC feature as SSNC experiments, and we refer to experiments that used datasets that excluded the SSNC feature as non-SSNC experiments.

For each dataset we performed two types of hyperparameter tuning, one using standard CV and the other using block CV. We used five iterations of 3-fold CV, such that the cluster size of the block CV was equal to 4 days. We used 100 trials of random search using three executions per trial for hyperparameter tuning. Note that every fold had the same testing dataset (2022 data for MY_{2021} experiments and 2021 data for MY_{2022} experiment). We included all the models (including DSFAMem), described in Section 5.4.1.3, in these experiments except for SVM. We did not include SVM, since the execution time of some SVM experiments exceeded 24 h, which forced us to manually terminate these SVM experiments. In terms of dataset normalization, we used min-max scaling over both the training and testing datasets, but we used the training dataset's minimum and maximum values to emulate a realistic scenario.

We initially performed preliminary extrapolation experiments (not covered in the present work), which led to poor performance. These preliminary experiments included all the features selected from the feature selection step. Therefore, we tried various subsets of the selected features (by trial and error) until reasonable extrapolation performance was achieved. By applying this methodology, we found a set of 17 features that we chose to

Table 5.4.1: N_2O prediction experiments: features used, including the window size and aggregation operator applied to derive the lagged features, in the multi-year extrapolation experiments, where Op. = operator, SSNC = synthetic soil nitrogen content, GDD = growing degree days, SHU = soil heating units, NHU = nitrification heating units, and N/A = not applicable.

Feature/Sensor	**Lag Window Size**	**Aggregation Op.**
Accumulated GDD	Entire history	Summation
Accumulated SHU	Entire history	Summation
Accumulated NHU	Entire history	Summation
Air temperature	1 h	Average
Precipitation	12 h	Summation
Relative humidity	1 h	Average
Solar radiation	1 h	Average
Soil moisture @ 20 cm	1 h	Average
Soil moisture @ 20 cm	3 h	Average
Soil moisture @ 20 cm	12 h	Average
Soil temperature @ 30 cm	1 h	Average
Soil temperature @ 30 cm	3 h	Average
Soil temperature @ 30 cm	12 h	Maximum
Soil temperature @ 30 cm	12 h	Minimum
SSNC	N/A	N/A
Vapor pressure deficit	1 h	Average
Vapor pressure deficit	12 h	Average

use for the extrapolation experiments covered in the present work. These 17 features are explained in Table 5.4.1.

5.4.3 Analysis and Results

In this section, we discuss the results of the N_2O emission prediction interpolation and extrapolation experiments in Sections 5.4.3.1 and 5.4.3.2, respectively.

5.4.3.1 Single-year Interpolation

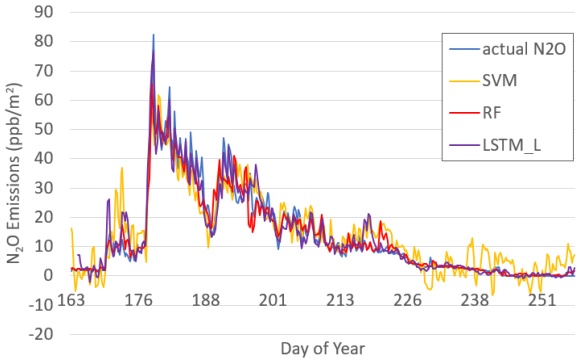
In this section we analyze the results of the single-year interpolation experiments explained in Section 5.4.2.1. The analysis includes examining model performance over an entire year (global performance) and an analysis of model performance for different parts of the season (seasonal performance). We also compared the models to one another to identify the best performing model. Tables 5.4.2 and 5.4.3 show the global average performance and seasonal average performance, respectively, for each model and year averaged over each chamber and the four dataset temporal resolutions. Note that we analyzed the average R^2, RMSE, and MAPE performance over various datasets, and strictly speaking, analyzing the average MAPE and RMSE over different chambers is not appropriate, because the scale of the emissions may have been different for different chambers within the same year,

but nevertheless, we included the average of both metrics in some parts of our analysis for presentation purposes. Figure 5.4.1 illustrates time series charts that plot the actual emissions vs. model predictions of some of the models, where predictions are the average model predictions over all folds and iterations.

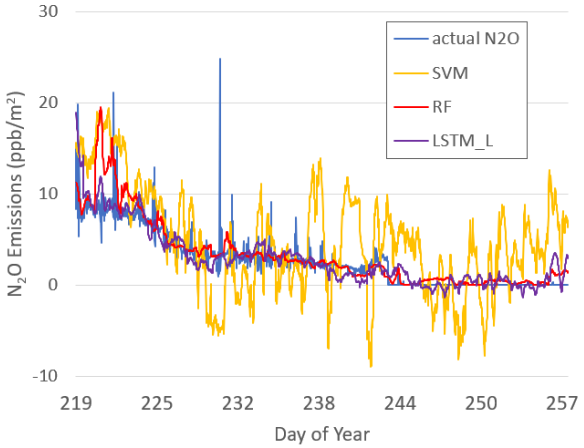
5.4.3.1.1 Global Performance In this section we analyze the average model performance over the entire season. By observing Figure 5.4.1, we can see that the models generally did well at filling the gaps (of at least 24 h in size for the testing data) of missing emissions. The emission trends were accurately captured by the models and only in a few cases were large emissions spikes not predicted properly. By observing Table 5.4.2, we can see that RF had the highest average R^2 in 2021 and the lowest R^2 standard deviation for all three years. RF had the lowest average MAPE for all three years, which is likely because a) we found later in the present work that RF did best at the end of the season and b) of how MAPE may have exaggerated prediction errors at the small scales of the low baseline emissions. LSTM_L had the highest R^2 in 2022, the lowest RMSE for all three years, and the lowest RMSE standard deviation in 2022 and 2023. These observations suggest LSTM_L and RF were among the best performing models over the entire season. LR and SVM were the models with the worst average performance for all three years. Furthermore, 2023 had generally worse model performance than 2021 and 2022. This may also have been because we had to manually water crops in 2023 due to it being a dry year, meaning chambers without adjacent soil moisture sensors did not capture this manual watering event. We also suspect this poor 2023 average model performance may have been due to chamber C1 in 2023 having very low emissions compared to all other chambers (see Figure 5.0.2), although we have no evidence to support this hypothesis. One can also notice that SVM had noticeably large average RMSE in 2023, which was larger than the ZeroR RMSE performance, suggesting the presence of prediction outliers in 2023 and/or the complete failure of SVM. In general, other than SVM in 2023, the models performed better than the naive baseline ZeroR.

Observing only global annual model performance was not sufficient to gain proper insight into whether temporal autocorrelation or other temporal patterns existed within the model residuals. Furthermore, the distribution of N_2O emissions for each chamber for each year changed throughout the seasons (emissions were non-stationary), so we suspect there may have been performance patterns unique to different parts of the growing season. Therefore, before drawing any conclusions, we proceed to perform a seasonal performance analysis in the next section.

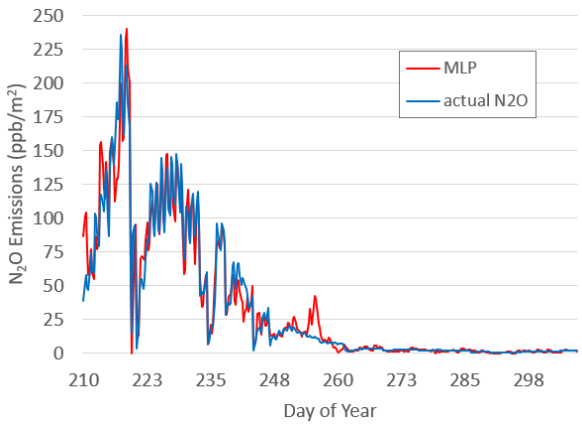
Takeaway. It was difficult to draw strong conclusions by analyzing average model performance over the entire season, since emission trends were non-stationary and tended to zero at the end of the season. What we observed is that RF and LSTM_L were among the models with the best global average performance, and performance was worse for 2023 experiments than for 2021 and 2022 experiments. Furthermore, the time series plots of model predictions vs. actual emissions revealed that most models did quite well and succeeded in predicting values to fill the gaps of missing data that were at least 24 h in size.



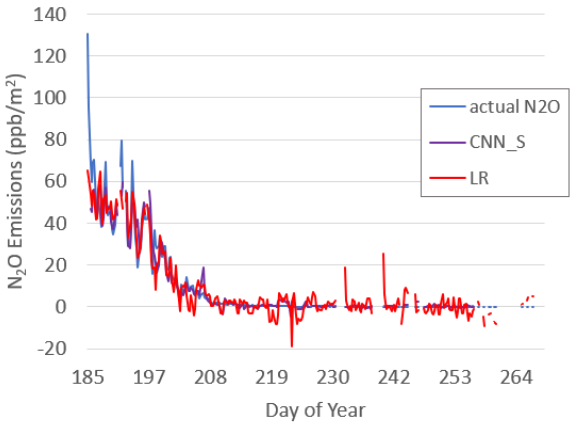
(a) 2021 All Season 6 h Resolution — Chamber C4



(b) 2021 Later Part of the Season 30 min Resolution — Chamber C4



(c) 2022 All Season 6 h Resolution — Chamber C1



(d) 2023 All Season 6 h Resolution — Chamber C4

Figure 5.4.1: Line plots of actual N_2O emissions vs. single-year interpolation model predictions (of various models using 30 min temporal resolution datasets) for 2021, 2022, and 2023.

Table 5.4.2: N_2O prediction experiments: single-year interpolation global model performance averaged over all chambers and all four dataset temporal resolutions, where the data shown are mean ± standard deviation, and bold font results are the best (in terms of average performance) for a given year.

Year	**Model**	**R^2**	**RMSE**	**MAPE (%)**
2021	CNN_L	0.75±0.25	6.49±2.92	83±77
2021	CNN_S	0.80±0.16	6.32±2.06	78±63
2021	MLP	0.78±0.17	6.67±2.27	63±63
2021	LR	0.70±0.06	8.06±0.80	165±66
2021	LSTM_L	0.88±0.11	**4.56**±1.58	51±33
2021	LSTM_S	0.88±0.12	4.94±1.77	54±56
2021	RF	**0.90**±0.05	4.60±1.58	**21**±6
2021	SVM	0.73±0.07	7.52±0.89	136±48
2021	ZeroR	0.00±0.00	15.23±1.05	332±115
2022	CNN_L	0.79±0.18	11.26±7.46	112±123
2022	CNN_S	0.83±0.14	10.49±7.32	101±107
2022	MLP	0.88±0.10	7.95±4.54	53±44
2022	LR	0.69±0.25	13.14±8.16	214±104
2022	LSTM_L	**0.89**±0.11	**7.57**±4.13	54±37
2022	LSTM_S	0.87±0.14	8.04±4.35	59±44
2022	RF	0.87±0.08	8.79±4.98	**38**±28
2022	SVM	0.76±0.15	11.46±6.16	238±107
2022	ZeroR	0.00±0.00	28.23±16.67	614±284
2023	CNN_L	0.56±0.35	4.54±4.40	109±154
2023	CNN_S	0.62±0.34	4.70±4.48	100±115
2023	MLP	0.69±0.31	4.73±5.08	88±113
2023	LR	0.56±0.33	8.34±13.84	233±178
2023	LSTM_L	**0.71**±0.31	**3.16**±3.11	69±99
2023	LSTM_S	0.68±0.32	4.16±4.19	67±87
2023	RF	0.70±0.27	4.81±4.56	**24**±37
2023	SVM	0.42±0.32	480.70±6,257.22	17E+3± 22E+4
2023	ZeroR	0.00±0.00	11.85±9.37	370±281

5.4.3.1.2 Seasonal Performance In this section we analyze model performance over different parts of the season.

By analyzing Tables 5.4.2 and 5.4.3, we can observe that better model performance occurred in early and middle season compared to the global average performance over the entire season, and we can also observe that in the later parts of the season some performance metrics were worse. We can see in Table 5.4.3 that the standard deviation of the performance was generally higher in 2022 and 2023 than in 2021, and this could be because a) 2022 and 2023 had several more chambers than 2021 and b) we found a significant difference in performance between the performance of broadcast without crop and broadcast with crop, and this could only have affected 2022 and 2023 performance results. We can also observe that average R^2 performance was lower for each model in 2022 early season compared to 2021 early season. This may be explained by the fact that 2022 emissions had larger variability in early season than 2021 early season. Furthermore, we can see RMSE and R^2 performance generally decreased as the season progressed. At first glance, the poor R^2 performance during the later parts of the seasons suggests models were poorly performing at the end of the season; however, the RMSE also showed decreasing trends (and hence, better model performance) as the season progressed, which conflicts with the R^2 seasonal trends. These conflicting evaluation metric trends may be explained as follows:

- **Hypothesis 1)** The higher R^2 and higher RMSE at the start of the season suggest that the models had difficulty predicting large emission spike events that occurred more frequently at the start of the season (hence the higher RMSE), but otherwise the models were able to learn the general macro-level emission patterns and could predict the direction of the emission events (hence the higher R^2), where dips and spikes were predicted correctly, but the scale of the spikes were not predicted very accurately.
- **Hypothesis 2)** The lower R^2 and lower RMSE at the end of the season suggests that the models were correctly predicting the low baseline emissions at the end of the season (hence the lower RMSE), but the models (other than RF) were generally failing to capture the micro fluctuations of the low baseline emissions in their predictions (hence the lower R^2).

To confirm these two hypotheses, we can observe Figure 5.4.1 for supporting evidence. In Figures 5.4.1a, 5.4.1c, and 5.4.1d we can see that hypothesis 1) is supported by the fact that a) the model predictions all followed the large emission trend patterns at the start of the season with only a bit of fluctuation and b) a few steep spikes were not predicted well. In terms of supporting hypothesis 2), we can see in Figure 5.4.1b that the model predictions captured the small scale of the low emission baseline, and the differences in micro fluctuations between RF and LSTM_L (which were the better performing models) were relatively small, other than in a few cases. We can conclude that even though the R^2 was poor for most of the models, the small RMSE and relatively small MAPE indicate that the models were doing well at the end of the season. We can also notice in Figures

5.4.1b and 5.4.1d that models with higher RMSE at the end of the season (SVM and LR) had more trouble predicting the scale of the low emission baseline (with high prediction variance around the mean zero emission baseline), but they were nevertheless able to follow the low baseline emission trend at the end of the season (predictions were centered around the zero emission baseline). If we only relied on R^2 to perform this analysis, a conclusion drawn about good R^2 performance later in the season may be somewhat incorrect, since, as we can see in Figure 5.4.1b, RF and LSTM_L were quite visually similar in their ability to predict the small scale nature of the low baseline emissions at the end of season, even though RF had much better R^2 in later parts of the season. Furthermore, since a) predicting emission spikes and capturing the non-stationary nature of emissions is important and b) high emissions occurred earlier in the season, placing more weight on analyzing model performance during earlier parts of the season should be favored, and conclusions about the best performing models at the end of the season should be attributed less weight.

Takeaway. The analysis done in this section has demonstrated that simply analyzing global model performance over an entire season may not be appropriate if the input dataset is a non-stationary time series dataset, since there may exist performance trends at different parts of the season that cannot be uncovered using only global average performance. A temporal analysis of the performance should be conducted, since we found that the better earlier-season R^2 model performance trends were masked out by the poor performance later in the season. Furthermore, it may not be sufficient to use only a single evaluation metric. From our findings, we suggest using both the RMSE and R^2 when evaluating model performance over a season, since it allowed us to better understand whether models could learn micro- and macro-level emission patterns.

5.4.3.1.3 Model Comparison In this section we perform an analysis to identify the better performing models, by first examining the global and seasonal model performance and then performing significance testing.

By examining Tables 5.4.2 and 5.4.3, we can see that LSTM_L and RF were among the models with the highest average R^2 and lowest average RMSE and MAPE. In some cases LSTM_S and MLP also had the best performance for some metrics. In terms of global performance, RF had the highest R^2, lowest RMSE, and lowest MAPE in 2021. RF also had the lowest MAPE in 2023. LSTM_L had the highest R^2 and lowest RMSE in 2022 and 2023. In terms of seasonal performance, LSTM_L and RF had the best performance generally earlier and later in the season, respectively. MLP, tied with LSTM_L, had the best average R^2 in 2022 early season, although MLP had higher average RMSE. LSTM_S had the lowest average RMSE in 2023 in late season. SVM and LR were among the models with the worst average performance. In particular, SVM had large RMSE in 2023 due to generally failing to capture the 2023 emission trends.

We now more formally compare models by using significance testing by using the results of each model as a result set. We performed a separate analysis for early season, mid-season, late season, end of season, and the entire season using RMSE, R^2, and MAPE. In all cases, the Friedman test rejected the null hypothesis that the models all had similar performance. The Friedman test rankings are shown in Table 5.4.4, which illustrates the

Table 5.4.3: N_2O prediction experiments: single-year interpolation seasonal model performance averaged over all chambers and all four dataset temporal resolutions, where the data shown are mean ± standard deviation, N/A = no results available due to zero samples in that part of the season, and bold font results are the best (in terms of average performance) for a given year.

		2021			2022			2023		
Season	**Model**	**R^2**	**RMSE**	**MAPE (%)**	**R^2**	**RMSE**	**MAPE (%)**	**R^2**	**RMSE**	**MAPE (%)**
Early	CNN_L	0.75±0.10	10.12±2.00	70±15	0.59±0.19	18.28±11.49	59±31	0.71±0.09	6.20±4.49	41±29
Early	CNN_S	0.80±0.03	9.25±0.73	68±24	0.64±0.11	16.97±9.30	49±14	0.72±0.10	7.02±5.31	38±27
Early	MLP	0.77±0.02	9.89±0.42	53±10	**0.77**±0.08	13.24±6.27	33±9	0.72±0.10	8.16±6.29	34±17
Early	LR	0.75±0.01	10.44±0.10	113±6	0.36±0.19	22.43±11.74	78±25	0.69±0.22	14.08±22.71	82±56
Early	LSTM_L	**0.91**±0.00	**6.07**±0.13	46±13	**0.77**±0.10	**12.90**±5.95	**28**±8	**0.84**±0.05	**4.36**±3.09	**25**±16
Early	LSTM_S	0.89±0.04	6.82±1.07	38±6	0.76±0.13	13.06±5.91	33±8	0.77±0.11	6.18±4.83	29±21
Early	RF	0.90±0.02	6.59±0.70	**27**±9	0.69±0.11	15.38±7.41	64±29	0.68±0.11	8.56±6.42	37±24
Early	SVM	0.77±0.03	9.84±0.64	97±10	0.68±0.06	16.15±8.78	62±23	0.30±0.30	505.30± $1.7E+3$	32E+2± 11E+3
Early	ZeroR	0.00±0.00	22.89±0.08	210±1	0.00±0.00	46.57±28.36	79±15	0.00±0.00	19.67±13.90	91±62
Mid	CNN_L	0.67±0.08	5.06±0.57	20±2	0.61±0.09	9.69±4.02	67±17	0.00±0.00	1.76±1.52	116±103
Mid	CNN_S	0.67±0.06	5.14±0.53	19±2	0.71±0.08	8.44±3.59	64±21	0.00±0.00	1.48±1.06	99±78
Mid	MLP	0.54±0.10	6.00±0.53	20±2	0.79±0.09	6.95±2.77	57±28	0.00±0.00	1.37±0.98	90±61
Mid	LR	0.28±0.05	7.54±0.29	30±2	0.42±0.21	11.91±6.17	105±47	0.00±0.00	5.38±5.03	317±265
Mid	LSTM_L	**0.79**±0.04	**4.08**±0.42	**15**±2	**0.85**±0.04	**5.88**±2.41	**35**±9	0.00±0.00	0.94±0.76	58±57
Mid	LSTM_S	0.75±0.04	4.46±0.41	16±2	0.79±0.13	6.73±2.85	45±15	0.00±0.00	0.96±0.67	57±46
Mid	RF	0.59±0.07	5.66±0.34	18±1	0.65±0.11	8.76±2.73	59±16	**0.06**±0.10	**0.58**±0.56	**11**±11
Mid	SVM	0.52±0.08	6.13±0.46	24±2	0.59±0.10	10.14±4.84	96±45	0.00±0.00	234.26± 782.38	92E+2± 3E+4
Mid	ZeroR	0.00±0.00	12.01±0.11	38±1	0.00±0.01	17.03±7.46	153±60	0.00±0.00	5.69±4.19	510±392
Late	CNN_L	0.47±0.04	2.28±0.12	34±4	0.09±0.18	2.73±1.79	92±54	0.00±0.00	1.58±1.36	105±88
Late	CNN_S	0.38±0.31	2.50±0.77	36±13	0.12±0.22	2.45±1.63	78±46	0.00±0.00	1.20±1.04	97±80
Late	MLP	0.34±0.12	2.53±0.32	33±6	0.32±0.39	1.08±0.41	30±14	0.00±0.00	1.33±1.07	95±68
Late	LR	0.00±0.00	5.45±0.15	92±3	0.00±0.00	6.85±4.13	240±126	0.00±0.00	4.14±3.07	315±207
Late	LSTM_L	**0.56**±0.28	**2.00**±0.78	25±7	0.29±0.39	1.06±0.39	36±15	0.00±0.00	0.86±0.73	66±58
Late	LSTM_S	0.52±0.11	2.14±0.24	26±6	0.19±0.33	1.44±1.07	39±15	0.04±0.11	**0.72**±0.61	58±49
Late	RF	0.45±0.18	2.27±0.32	**21**±4	**0.71**±0.15	**0.84**±0.74	**12**±7	**0.10**±0.16	0.79±0.80	**19**±25
Late	SVM	0.00±0.00	5.33±0.22	84±7	0.00±0.00	5.62±2.87	225±120	0.00±0.00	1.4E+3± 4.9E+3	35E+3± 12E+4
Late	ZeroR	0.00±0.00	8.12±0.07	198±7	0.00±0.00	18.87±10.81	784±378	0.00±0.00	5.94±4.36	528±403
End	CNN_L	0.00±0.00	2.60±0.49	154±44	0.00±0.00	2.38±1.68	142±83	N/A	N/A	N/A
End	CNN_S	0.00±0.00	2.58±1.02	160±68	0.00±0.00	2.26±1.74	141±93	N/A	N/A	N/A
End	MLP	0.00±0.00	1.81±0.91	108±42	0.01±0.04	0.83±0.38	46±19	0.00±0.00	1.77±1.80	118±132
End	LR	0.00±0.00	8.21±0.19	524±9	0.00±0.00	7.10±4.20	464±259	0.00±0.00	4.99±3.69	361±274
End	LSTM_L	0.00±0.00	1.45±0.36	96±11	0.02±0.05	0.89±0.49	67±34	N/A	N/A	N/A
End	LSTM_S	0.00±0.00	1.82±1.26	114±70	0.00±0.00	0.93±0.62	67±28	N/A	N/A	N/A
End	RF	**0.54**±0.19	**0.66**±0.13	**10**±3	**0.30**±0.31	**0.34**±0.24	**8**±6	0.00±0.00	**0.38**±0.82	**15**±52
End	SVM	0.00±0.00	4.96±0.37	324±14	0.00±0.00	5.82±3.06	376±195	0.00±0.00	5.2E+3± 1.8E+4	24E+4± 83E+4
End	ZeroR	0.00±0.00	13.34±0.00	1104±2	0.00±0.00	20.24±10.90	1544±830	0.00±0.00	6.09±4.39	537±408

average performance ranking of the models for each part of the season and identifies the best performing models. We can see that LSTM_L was among the best performing models in most of the cases, RF was the model next most frequently among the best models, and LSTM_S was in some cases among the best performing models. Model seasonal performance trends are discussed below:

- Early season: LSTM was the best performing model. LSTM_L was the superior model in terms of RMSE and R^2, and in terms of MAPE, LSTM_L and LSTM_S were the best models.
- Mid-season: LSTM_L was among the best models for all three metrics. In terms of MAPE, RF was also among the best performing models.
- Late season: RF was among the best models for all metrics. LSTM_L and LSTM_S were also among the best models in terms of RMSE.
- End of season: RF was the superior model.
- Entire season: RF and LSTM_L were among the best models. LSTM_L was the best model in terms of RMSE and R^2, whereas RF was the best in terms of MAPE.

In summary, LSTM was found to be the best model at the start of the season for all metrics, and as the season progressed, RF performed increasingly well and became the superior model for the end of season. We also found that larger input tensors for LSTM (LSTM_L) generally led to better performance than shorter tensors (LSTM_S). Furthermore, the fact that models performed differently in different parts of the season suggests EL (using models each for different parts of the season) could potentially lead to improved performance. In fact, in our study described in Section 5.5, we found that such ensemble models can do quite well at N_2O emission interpolation tasks.

Takeaway. LSTM and RF were among the best performing models. LSTM_L generally had better performance than LSTM_S. LSTM had significantly better performance earlier in the season, and in later parts of the season, RF had significantly better performance.

5.4.3.1.4 Effects of Dataset Temporal Resolution In this section we briefly discuss the effects of dataset temporal resolution on performance by using significance testing. We performed a separate analysis for early season, mid-season, late season, end of season, and the entire season using RMSE, MAPE, and R^2. We found that in many cases the Friedman test rejected the null hypothesis, meaning dataset temporal resolution occasionally had a significant impact on performance. Among the resolutions that led to the best performance are 60 min and 180 min, which were among the best dataset resolutions in all cases.

Takeaway. Although dataset temporal resolution did not always have a significant impact on performance, in cases when there was a significant impact, we found that 60 min and 180 min resolutions were always among the resolutions that led to the best performance.

Table 5.4.4: N_2O prediction experiments: list of model performance average rankings from the Friedman test for the single-year interpolation experiments, where a lower ranking number indicates better ranking performance and marked in bold are the best performing models[1].

			Average Performance Ranking for Each Model							
Metric	**Season**	**Friedman Test**	**CNN_L**	**CNN_S**	**MLP**	**LR**	**LSTM_L**	**LSTM_S**	**RF**	**SVM**
RMSE	Early	Reject	4.61	4.79	4.25	6.61	**1.82**	2.71	5.07	6.14
RMSE	Mid	Reject	5.04	4.75	3.89	7.21	**1.89**	2.82	3.71	6.68
RMSE	Late	Reject	4.96	4.32	4.04	7.50	**2.86**	**2.86**	**2.00**	7.46
RMSE	End	Reject	5.25	5.31	3.19	7.81	3.06	3.13	**1.06**	7.19
RMSE	All	Reject	4.47	4.57	3.97	6.44	**2.54**	3.30	4.02	6.69
R^2	Early	Reject	5.46	5.04	3.82	5.89	**2.00**	2.86	4.79	6.11
R^2	Mid	Reject	3.50	3.21	2.71	4.68	**1.32**	2.14	3.46	3.96
R^2	Late	Reject	2.64	2.61	2.21	3.21	1.96	2.11	**1.71**	3.21
R^2	End	Reject	1.88	1.88	1.81	1.88	1.75	1.88	**1.00**	1.88
R^2	All	Reject	4.73	4.53	3.70	5.98	**2.76**	3.30	3.74	6.36
MAPE	Early	Reject	5.00	4.71	3.14	7.21	**1.82**	**2.68**	4.68	6.75
MAPE	Mid	Reject	4.96	4.46	4.04	6.93	**2.00**	2.93	**3.04**	7.04
MAPE	Late	Reject	4.75	4.82	3.68	7.57	2.46	2.79	**1.29**	7.43
MAPE	End	Reject	5.19	5.25	2.69	7.88	3.44	3.44	**1.00**	7.13
MAPE	All	Reject	4.65	4.56	3.44	7.14	3.25	3.43	**2.10**	7.37

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

5.4.3.2 Multi-year Extrapolation

We will now explore and analyze the results of the multi-year extrapolation experiments described Section 5.4.2.2. Table 5.4.5 presents the global performance of the models, where the results were averaged over both SSNC and non-SSNC experiments. Tables 5.4.6 and 5.4.7 show the results of MY_{2021} and MY_{2022} experiments, respectively, over different parts of the season, where the results were averaged over SSNC and non-SSNC experiments.

5.4.3.2.1 Global Performance In this section we analyze the average model performance over the entire season. Specifically, we compare model performance for a) multi-year extrapolation and single-year interpolation experiments, b) MY_{2021} vs. MY_{2022} experiments, c) only MY_{2021} experiments, and d) only MY_{2022} experiments.

By observing Tables 5.4.2 and 5.4.5, we can see that multi-year extrapolation performance was worse than single-year interpolation performance, especially in terms of RMSE and MAPE. Although in some multi-year extrapolation cases (e.g., MY_{2021}), reasonable R^2 was still achieved, suggesting that the models were able to learn the emission trends from one year and generalize those trends to another year, although the models failed to properly capture the scale of the emissions.

We now compare MY_{2021} vs. MY_{2022} experiment model performance by observing Table 5.4.5. We can see that MY_{2022} experiments led to worse model performance than the MY_{2021} experiments, which is interesting, since the only difference between MY_{2021}

and MY_{2022} experiments was that the training and testing datasets were swapped. This performance difference suggests a) that the models were able to more easily predict larger unobserved emission trends than smaller unobserved emissions trends (given that 2022 had larger emissions than 2021) and b) more importantly, and in general, just because a model performed well for a set of multi-year N_2O emission extrapolation experiments involving data only gathered from a single study site, it does not mean the model will generalize well to new geographical regions.

We now focus on analyzing the MY_{2021} experiment results. We can see that each model (other than LR, which completely failed) did better than the ZeroR and DFSAMem baseline models in terms of R^2 and RMSE. We can see DFSAMem obtained zero R^2, and this indicates that the emissions in 2022 had variability that could not be explained by simply memorizing 2021 emissions and repeating the 2021 emissions as the 2022 predictions. This means that the other non-baseline models learned some of the processes that affect emission variability. However, it can be observed that all models did worse in terms of MAPE compared to DFSAMem, suggesting that the models were failing in some capacity; we proceed to elaborate one why this conclusion may not be entirely correct. A model with 50 % MAPE performance in two different parts of the season does not necessarily imply that the model has identical predictive capabilities in both parts of the season, since a) the scale of prediction errors is not captured by MAPE and b) the scale of the emissions changed over the season. This means earlier-season and/or later-season MAPE performance may have skewed the average MAPE model performance over the entire season. This point may also apply to other evaluation metrics, although we did not investigate on this point. Furthermore, it is more important to predict emissions tied to large emission spike events than low baseline emissions, and so, higher performance earlier in the season should be favored. Both 2021 and 2022 low baseline emissions in later parts of the season were quite similar (2021 and 2022 emissions had less than a 1 ppb/m^2 difference on average) even though 2021 and 2022 emissions were quite different in earlier parts of the season, so, this is probably why DFSAMem did well in terms of MAPE. Therefore, this evidence suggests that an analyst should be careful when drawing conclusions using MAPE as a performance metric for N_2O emission prediction model evaluation.

We now focus on analyzing the MY_{2022} experiment results. All the models appeared to have generally failed, since ZeroR (which always predicted 24.9 ppb/m^2 when making 2021 emission predictions) outperformed all other models in terms of RMSE. At first glance, this suggests the models did not learn anything, but we observed evidence that supports the idea that models had not completely failed and that they had at least learned something; that is, a) there were cases of positive R^2 from the non-baseline models (e.g., LSTM) and b) there were cases where the non-baseline models outperformed ZeroR in terms of MAPE (e.g., LSTM). Furthermore, although ZeroR did relatively well in terms of RMSE compared to the other models, this better performance was probably due to the fact that the average emissions from 2022 (24.9 ppb/m^2) was relatively small due to there being many smaller scale emission readings at the end of the season (1.9 ppb/m^2 in late season and 1.5 ppb/m^2 in the end of season); and since 2021 had smaller emission peaks than 2022, then by design, ZeroR had a decent chance of making reasonably accurate emission predictions. For example, ZeroR always predicted 24.9 ppb/m^2 in the MY_{2022} experiments, and this

was reasonably close to the average early season emissions (22.8 ppb/m^2) and mid-season emissions (21.5 ppb/m^2) in 2021. So, it is not surprising that ZeroR appeared to do well in terms of RMSE. On the other hand, this relatively good ZeroR performance and the fact models also failed to outperform the DFSAMem baseline model in terms of MAPE may indicate models were overfitting to the large scale emissions in 2022, meaning the models were actually failing, since they were overestimating the scale of the smaller 2021 emissions over the entire season. We cannot verify this by analyzing only global average performance. However, these MAPE performance trends may not be useful enough to draw any conclusion for reasons discussed earlier. Since this section analyzed results of model performance averaged over the entire season, we did not have enough information to fully understand model performance at different parts of the season and we could not verify all our hypotheses. It was difficult to understand when models were failing and when they were doing well. Therefore, we proceed with a seasonal analysis in the next section.

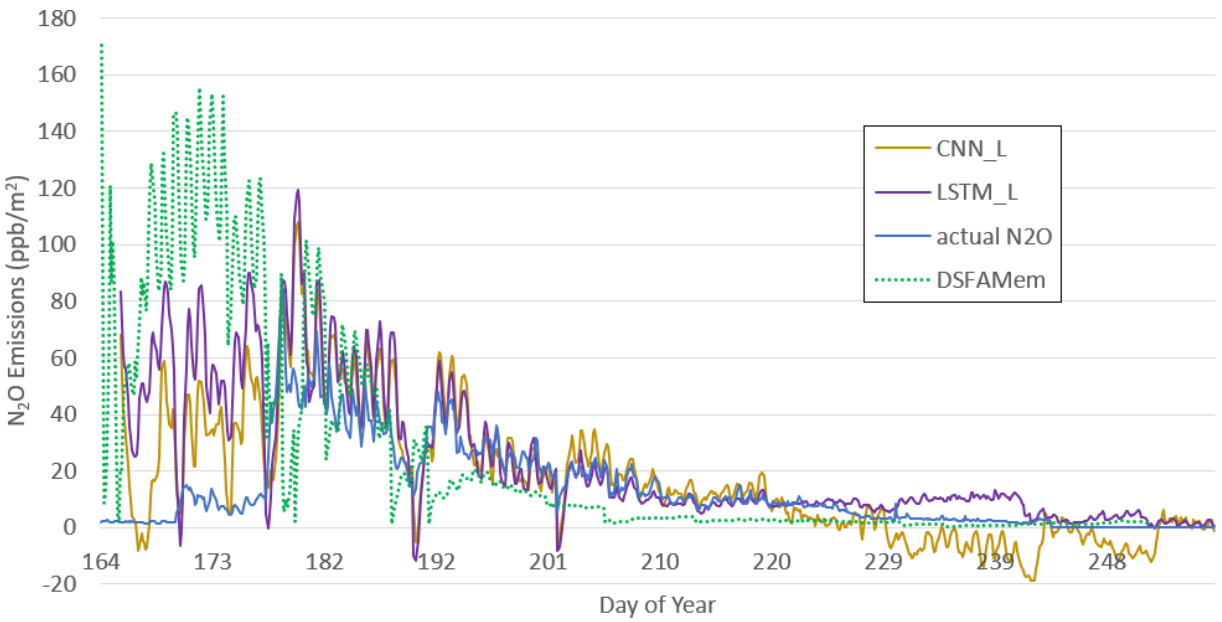
Takeaway. Similar to the single-year interpolation experiment analysis, it was difficult to draw strong conclusions by conducting only a global performance analysis. Nevertheless, what we can conclude is that a) single-year interpolation model performance was better than multi-year extrapolation model performance and b) just because a model did reasonably well, when trained using one year's data and tested on another year's data, does not mean reasonable performance will also be achieved when the training and testing datasets are swapped.

5.4.3.2.2 Seasonal Performance In this section we analyze model performance over different parts of the season. We first analyze only MY_{2021} experiment results, then MY_{2022} experiment results, and finally we analyze both MY_{2021} and MY_{2022} experiment results.

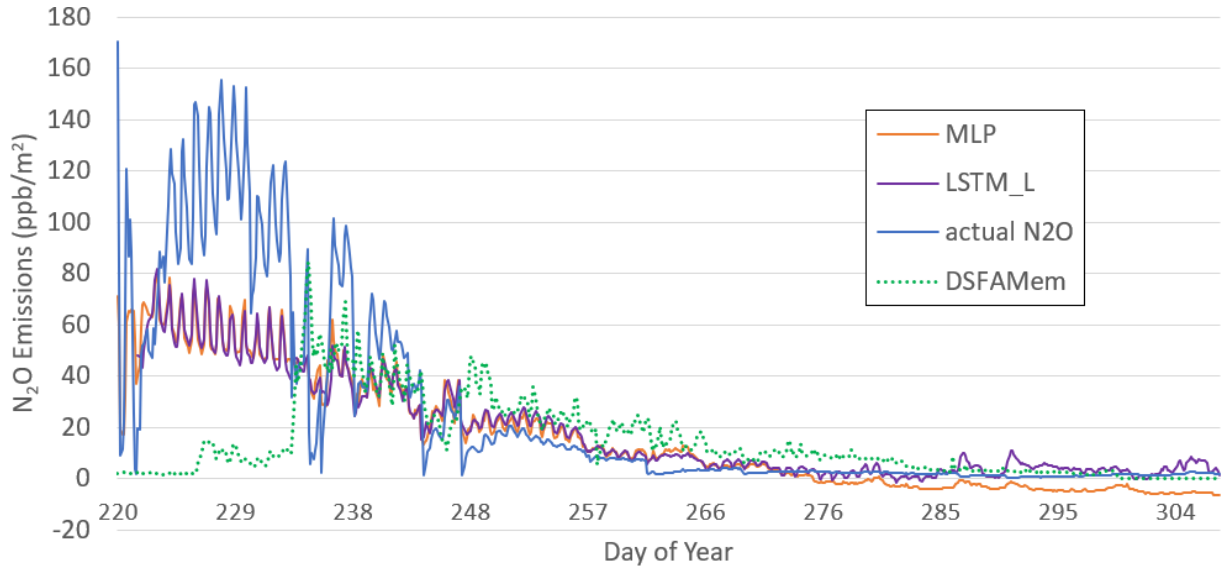
We begin by analyzing the MY_{2021} experiment results. By observing Table 5.4.6, we can see in terms of MAPE performance, compared to the global performance analysis conducted in Section 5.4.3.2.1, that models were no longer always being outperformed by the baseline model DSFAMem (except in the later parts of the season). We can also see that in the middle of the season, some models did worse than ZeroR in terms of RMSE (e.g., CNN_L), whereas in the other parts of the season (e.g., late season) there were no non-baseline models (other than LR, which completely failed) that did worse than ZeroR. This is likely the case, since the average emissions of 2021 (13.6 ppb/m^2) was similar to the average emissions in 2022 mid-season (10.3 ppb/m^2). In fact, every other part of the 2022 season other than mid-season had a larger difference in average emissions compared to the overall average emissions of 2021. Some models did worse than ZeroR (e.g., CNN_S in early season) in terms of MAPE. Furthermore, we can see that the R^2 performance of most models became nearly zero when splitting the performance analysis by season instead of doing a global analysis, which was a similar observation to the observations made in the seasonal analysis of the interpolation experiments discussed in Section 5.4.3.1.2. We can also see that the start of the season had the worst (highest) RMSE and the RMSE generally decreased over the season, suggesting model performance improved in later parts of the season. As previously stated, since the emissions also decreased over the season, the decrease in RMSE did not strictly indicate better performance, but the RMSE decrease did nevertheless suggest improved performance as the season progressed. We can see that

Table 5.4.5: N_2O prediction experiments: multi-year extrapolation global model performance averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, CV = cross-validation, the data shown are mean ± standard deviation, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance.

		MY_{2021}			**MY_{2022}**		
Inner CV Type	**Model**	**R^2**	**RMSE**	**MAPE (%)**	**R^2**	**RMSE**	**MAPE (%)**
Block CV	CNN_L	0.51±0.19	25.99±5.34	489±327	0.00±0.00	36.67±14.10	924±507
Block CV	CNN_S	0.58±0.12	24.22±3.05	382±263	0.01±0.03	32.71±15.58	812±613
Block CV	MLP	**0.62**±0.08	**23.42**±2.18	**236**±202	0.00±0.00	39.04±13.34	655±283
Block CV	LR	0.00±0.00	113.04±50.93	3169±1906	0.00±0.00	97.69±18.81	3956±902
Block CV	LSTM_L	0.59±0.12	23.91±3.26	347±242	**0.02**±0.07	**31.15**±11.07	565±317
Block CV	LSTM_S	0.60±0.15	23.47±3.72	350±315	0.00±0.00	32.08±10.08	**514**±158
Block CV	RF	0.33±0.08	31.21±1.81	280±47	0.00±0.00	45.31±4.27	877±318
Block CV	ZeroR	0.00±0.00	39.76±0.00	399±0	0.00±0.00	**18.86**±0.00	642±0
Block CV	DFSAMem	0.00±0.00	40.07±0.00	**158**±0.00	0.00±0.00	40.07±0.00	**444**±0
Standard CV	CNN_L	0.43±0.26	34.18±23.14	1016±1211	0.00±0.00	40.32±20.72	1165±758
Standard CV	CNN_S	0.56±0.14	25.92±9.33	547±550	0.01±0.05	41.39±22.53	1142±837
Standard CV	MLP	**0.62**±0.06	**23.29**±1.71	**276**±210	0.00±0.00	39.82±10.26	784±340
Standard CV	LR	0.00±0.00	113.04±50.93	3169±1906	0.00±0.00	97.69±18.81	3956±902
Standard CV	LSTM_L	0.55±0.12	24.96±3.10	358±276	**0.02**±0.08	32.31±15.55	**480**±236
Standard CV	LSTM_S	0.55±0.17	24.77±4.28	437±331	0.00±0.01	**30.61**±11.69	503±224
Standard CV	RF	0.26±0.05	32.75±1.10	331±54	0.00±0.00	43.41±3.36	676±189
Standard CV	ZeroR	0.00±0.00	39.76±0.00	399±0	0.00±0.00	**18.86**±0.00	642±0
Standard CV	DFSAMem	0.00±0.00	40.07±0.00	**158**±0	0.00±0.00	40.07±0.00	**444**±0



(a) 2021 emissions vs. model predictions



(b) 2022 emissions vs. model predictions

Figure 5.4.2: Line plots (at 180 min temporal resolution) of actual N_2O emissions vs. model predictions for the multi-year extrapolation non-SSNC experiments that used block CV for hyperparameter tuning, where SSNC = synthetic soil nitrogen content.

MAPE generally increased over the season, although we avoid making strong conclusions for reasons discussed earlier. We can see that the middle of the season was when models achieved the best R^2, which suggests mid-season achieved the best model performance. We can also see that some models did better during different parts of the season; for example, CNN_L had the best R^2 and RMSE in early season for standard CV, and MLP had the best performance in mid-season. We further elaborate on the performance differences between models later in Section 5.4.3.2.3.

We now analyze the MY_{2022} experiment results. By observing Table 5.4.7, we can see that ZeroR was no longer the best performing model: in all but early season, ZeroR did worse than all other non-baseline models (except LR, which failed completely). This supports our claim that ZeroR was only doing well in our global performance analysis (see Table 5.4.5) given that ZeroR predicted relatively low 2021 emissions and that 2021 had lower emissions than 2022. Our claim is supported, since early season 2021 emissions were near-zero and 2022 emissions were relatively large, meaning the non-baseline models probably failed to learn enough information to predict low emissions at the very start of the 2021 season when 2022 emissions in the training data were at their peak. We can also see, for similar reasons to those mentioned in our MY_{2021} experimental analysis above, that DSFAMem had the best MAPE performance only during the later parts of the season. We draw no conclusions from this MAPE-related observation for reasons discussed earlier.

By focusing on non-early season results in both Tables 5.4.6 and 5.4.7, we can see that the models did comparatively well to the single-year interpolation experiment results shown in Table 5.4.3. This is not surprising, since although 2021 and 2022 early season emissions were completely different, the emission trends of both years were relatively similar from mid-season onward when the emissions were aligned by DSFA. To get a better idea of how well the models were doing, we plotted the predictions (averaged over all CV folds and iterations) of some models over 2021 and 2022 in Figures 5.4.2a and 5.4.2b, respectively. We also included DSFAMem predictions to illustrate how different model predictions were from emissions at the same point in time (with respect to DSFA) from their training dataset. For the 2021 predictions, we can see that although models did poorly in early season, the model predictions were smaller than DSFAMem predictions, indicating that the models were taking advantage of patterns in predictor variables and understood that smaller emissions were present in the early season of 2021. For the 2022 predictions, a similar phenomenon can be observed, where training data emissions were relatively small in early season, but the models were nevertheless predicting larger values in 2022 when actual emissions were high. This showcases that the models learned some of the underlying patterns in predictor variables that indicate large emission events. In the MY_{2021} experiments, the models did not observe 2021 emissions as large as those in the 2022 testing dataset during training, meaning it is not surprising that the models could not accurately predict emissions that were approximately twice as large as the largest emissions in the training dataset. We can observe this training data limitation in Figure 5.4.2b. For 2022 model predictions, they did not exceed the maximum 2021 emissions of 84.3 ppb/m^2; MLP and LSTM_L made a maximum emission prediction of 78.7 ppb/m^2 and 81.6 ppb/m^2, respectively. In any case, although the models generally failed in terms of performance metrics in early season and were not visually accurate in Figure 5.4.2b, the models did successfully provide a rough

estimate of emissions and did learn the trends. Furthermore, predictions made beyond early season appeared quite reasonable. In particular, by observing the prediction line plots near DoY 192 in Figure 5.4.2a and DoY 248 in Figure 5.4.2b, the DSFAMem predictions deviated from the emission trend in the test year, yet the models still successfully followed the novel trend that was not present in the training data at that point in time with respect to DSFA. This means the models successfully learned emission trends and were not overfitting too much to the temporal structure of the training data. The model predictions clearly followed the emission trends of the testing data in later parts of the seasons, and since the present work is proposing an affordable PN that provides rough estimates of emissions, these results are a good starting point for satisfying these requirements. However, as mentioned earlier, we have no evidence to support the idea that the models can generalize reasonably well to new study sites. These preliminary results suggest that with more data from different years, model multi-year extrapolation performance could be improved.

Takeaway. We found that conducting a seasonal performance analysis was necessary for drawing meaningful conclusions about the model performance results. We also found baseline model results also helped our analysis. Applying a global average performance analysis over the entire season might hide performance trends that are different over various parts of the season (especially using MAPE), and without a baseline model to compare models to, it might be difficult to interpret results and make sense of the relative performance of a model. By a) conducting seasonal analysis, b) using baseline models, and c) observing model prediction line plots vs. actual emissions, we were able to show that the models in our multi-year extrapolation experiments were generally able to learn emission trends from one year and predict the trends in another year, although the models did fail to predict emissions in parts of the test season (early season) that had emission patterns and scales that were not well represented in the training data.

5.4.3.2.3 Model Comparison In this section we compare the performance of the non-baseline models to attempt to find the best performing models. We explore the performance of models for each metric over the entire season, for MY_{2021} experiments, and for MY_{2022} experiments. Then we perform significance testing to more formally identify superior models. Finally, we briefly compare single-year interpolation and multi-year extrapolation experiment model performance.

In terms of best average model performance over the entire season, by observing Table 5.4.5 we can see that for the block CV and standard CV MY_{2021} experiments, MLP had the highest average R^2 and lowest average RMSE, and MLP had the lowest average MAPE. For the block CV MY_{2022} experiments, LSTM_L had the highest average R^2 and lowest average RMSE, and LSTM_S had the lowest average MAPE. For the standard CV MY_{2022} experiments, LSTM_L had the highest average R^2 and lowest average MAPE, and LSTM_S had the lowest average RMSE.

For the MY_{2021} experiments, by observing Table 5.4.6, we can see that:

- for block CV experiments, in early season, LSTM_S had the lowest average RMSE and LSTM_L had the lowest MAPE; in mid-season, MLP had the best average R^2,

Table 5.4.6: N_2O prediction experiments: multi-year extrapolation seasonal model performance for the MY_{2021} experiments (experiments with models trained using 2021 data and tested using 2022 data) averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, the data shown are mean ± standard deviation, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance.

		Block Cross-validation			**Random Cross-validation**		
Season	**Model**	**R^2**	**RMSE**	**MAPE (%)**	**R^2**	**RMSE**	**MAPE (%)**
Early	CNN_S	0.00±0.00	40.34±1.85	99±8	0.00±0.00	39.42±1.09	99±1
Early	CNN_L	0.00±0.00	40.87±2.19	79±9	**0.01**±0.01	**38.53**±1.01	73±0
Early	MLP	0.00±0.00	41.34±0.32	77±1	0.00±0.00	40.47±0.32	77±2
Early	LR	0.00±0.00	94.16±52.42	389±189	0.00±0.00	94.16±52.42	389±189
Early	LSTM_S	0.00±0.00	**39.16**±0.69	77±10	0.00±0.00	40.61±1.28	82±1
Early	LSTM_L	0.00±0.00	41.21±1.08	**63**±2	0.00±0.00	43.44±0.67	**61**±0
Early	RF	0.00±0.00	56.53±0.37	89±0	0.00±0.00	59.26±0.37	85±0
Early	ZeroR	0.00±0.00	72.75±0.00	88±0	0.00±0.00	72.75±0.00	88±0
Early	DSFAMem	0.00±0.00	73.74±0.00	109±0	0.00±0.00	73.74±0.00	109±0
Mid	CNN_S	0.00±0.00	9.26±0.23	151±13	0.00±0.00	10.41±0.92	182±17
Mid	CNN_L	0.00±0.00	11.08±2.19	174±33	0.00±0.00	11.06±0.73	193±27
Mid	MLP	**0.23**±0.15	**6.10**±0.60	**100**±13	**0.38**±0.12	**5.48**±0.52	**100**±8
Mid	LR	0.00±0.00	149.73±81.43	2590±1456	0.00±0.00	149.73±81.43	2590±1456
Mid	LSTM_S	0.07±0.11	7.00±0.78	108±16	0.00±0.00	8.73±0.53	152±7
Mid	LSTM_L	0.12±0.17	7.10±1.46	107±24	0.00±0.00	7.31±0.25	124±0
Mid	RF	0.00±0.00	9.94±0.65	208±14	0.00±0.00	9.88±0.23	207±5
Mid	ZeroR	0.00±0.00	7.69±0.00	151±0	0.00±0.00	7.69±0.00	151±0
Mid	DSFAMem	0.00±0.00	13.79±0.00	216±0	0.00±0.00	13.79±0.00	216±0
Late	CNN_S	0.00±0.00	4.02±0.51	177±6	0.00±0.00	12.32±4.14	742±249
Late	CNN_L	0.00±0.00	5.84±2.47	304±183	0.00±0.00	11.58±2.32	671±177
Late	MLP	0.00±0.00	3.28±1.10	167±104	0.00±0.00	**3.81**±2.73	**209**±198
Late	LR	0.00±0.00	104.56±81.65	5768±4915	0.00±0.00	104.56±81.65	5768±4915
Late	LSTM_S	0.00±0.00	**2.17**±0.56	**104**±34	0.00±0.00	10.32±4.11	630±242
Late	LSTM_L	0.00±0.00	3.06±0.13	143±26	0.00±0.00	6.26±2.54	372±168
Late	RF	0.00±0.00	10.45±1.24	525±76	0.00±0.00	11.36±1.50	610±95
Late	ZeroR	0.00±0.00	11.73±0.00	711±0	0.00±0.00	11.73±0.00	711±0
Late	DSFAMem	0.00±0.00	4.82±0.00	202±0	0.00±0.00	4.82±0.00	**202**±0
End	CNN_S	0.00±0.00	3.33±1.03	177±23	0.00±0.00	12.46±5.75	879±418
End	CNN_L	0.00±0.00	7.27±2.09	427±93	0.00±0.00	9.72±7.12	679±515
End	MLP	0.00±0.00	3.93±3.64	261±268	0.00±0.00	**4.14**±3.66	**283**±281
End	LR	0.00±0.00	64.74±71.00	4630±5241	0.00±0.00	64.74±71.00	4630±5241
End	LSTM_S	0.00±0.00	2.88±1.62	180±101	0.00±0.00	9.79±5.49	706±401
End	LSTM_L	0.00±0.00	**2.72**±0.09	**161**±11	0.00±0.00	6.23±2.94	433±226
End	RF	0.00±0.00	4.30±1.35	314±98	0.00±0.00	6.89±1.78	502±128
End	ZeroR	0.00±0.00	12.09±0.00	872±0	0.00±0.00	12.09±0.00	872±0
End	DSFAMem	0.00±0.00	**1.57**±0.00	**58**±0	0.00±0.00	**1.57**±0.00	**58**±0

Table 5.4.7: N_2O prediction experiments: multi-year extrapolation seasonal model performance for the MY_{2022} (MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data) experiments averaged over both SSNC and non-SSNC experiments, where SSNC = synthetic soil nitrogen content, the data shown are mean ± standard deviation, and bold font results are for the best performing model (in terms of average performance) or second best performing model when a baseline model has the best performance.

		Block Cross-validation			**Random Cross-validation**		
Season	**Model**	**R^2**	**RMSE**	**MAPE (%)**	**R^2**	**RMSE**	**MAPE (%)**
Early	CNN_S	0.00±0.00	32.95±12.56	574±333	0.00±0.00	27.87±4.47	413±81
Early	CNN_L	0.00±0.00	**29.88**±5.46	**480**±163	0.00±0.00	**23.44**±0.85	**381**±7
Early	MLP	0.00±0.00	70.39±17.78	1598±453	0.00±0.00	67.65±1.61	1539±64
Early	LR	0.00±0.00	98.42±2.35	2315±4	0.00±0.00	98.42±2.35	2315±4
Early	LSTM_S	0.00±0.00	46.35±12.95	867±347	0.00±0.00	48.01±6.03	882±149
Early	LSTM_L	0.00±0.00	40.10±3.08	723±88	0.00±0.00	48.23±1.00	902±3
Early	RF	0.00±0.00	74.75±0.53	1670±6	0.00±0.00	79.97±6.68	1796±157
Early	ZeroR	0.00±0.00	**20.83**±0.00	**383**±0	0.00±0.00	**20.83**±0.00	383±0
Early	DSFAMem	0.00±0.00	73.74±0.00	1408±0	0.00±0.00	73.74±0.00	1408±0
Mid	CNN_S	0.00±0.00	12.35±0.60	54±0	0.00±0.00	20.87±1.66	91±10
Mid	CNN_L	0.11±0.15	10.92±4.34	45±23	0.00±0.00	21.56±3.36	88±18
Mid	MLP	**0.25**±0.22	**7.61**±1.11	**31**±2	0.00±0.00	10.01±0.95	46±8
Mid	LR	0.00±0.00	28.46±3.40	158±18	0.00±0.00	28.46±3.40	158±18
Mid	LSTM_S	0.08±0.05	8.47±0.24	**31**±4	0.03±0.04	9.62±1.45	38±1
Mid	LSTM_L	0.21±0.18	7.82±0.89	**31**±7	**0.19**±0.05	**7.94**±0.26	**33**±3
Mid	RF	0.00±0.00	19.38±4.34	90±19	0.00±0.00	13.31±3.61	53±16
Mid	ZeroR	0.00±0.00	9.49±0.00	51±0	0.00±0.00	9.49±0.00	51±0
Mid	DSFAMem	0.00±0.00	13.79±0.00	57±0	0.00±0.00	13.79±0.00	57±0
Late	CNN_S	0.00±0.00	22.21±4.85	524±116	0.00±0.00	34.77±2.27	820±53
Late	CNN_L	0.00±0.00	14.46±11.32	333±273	0.00±0.00	42.27±6.87	1000±166
Late	MLP	0.00±0.00	5.50±1.45	112±42	0.00±0.00	13.07±1.27	296±31
Late	LR	0.00±0.00	68.28±26.50	1582±668	0.00±0.00	68.28±26.50	1582±668
Late	LSTM_S	0.00±0.00	**4.54**±2.09	**98**±59	**0.13**±0.18	4.53±2.73	90±53
Late	LSTM_L	0.00±0.00	7.79±4.52	177±100	0.00±0.00	**3.67**±0.28	**60**±13
Late	RF	0.00±0.00	17.85±1.98	412±44	0.00±0.00	8.42±7.34	189±172
Late	ZeroR	0.00±0.00	19.15±0.00	443±0	0.00±0.00	19.15±0.00	443±0
Late	DSFAMem	0.00±0.00	4.82±0.00	**64**±0	0.00±0.00	4.82±0.00	64±0
End	CNN_S	0.00±0.00	24.81±5.41	1887±426	0.00±0.00	37.34±0.83	2685±57
End	CNN_L	0.00±0.00	16.64±12.22	1222±1035	0.00±0.00	43.43±5.71	3089±415
End	MLP	0.00±0.00	4.90±0.10	391±101	0.00±0.00	11.64±0.69	930±156
End	LR	0.00±0.00	164.76±60.53	14513±5376	0.00±0.00	164.76±60.53	14513±5376
End	LSTM_S	0.00±0.00	**4.21**±0.82	**208**±61	0.00±0.00	5.51±3.09	370±323
End	LSTM_L	0.00±0.00	7.13±4.02	480±398	0.00±0.00	**3.21**±1.07	**143**±62
End	RF	0.00±0.00	18.71±1.41	1438±96	0.00±0.00	9.42±6.87	637±495
End	ZeroR	0.00±0.00	24.39±0.00	2059±0	0.00±0.00	24.39±0.00	2059±0
End	DSFAMem	0.00±0.00	**1.57**±0.00	**62**±0	0.00±0.00	**1.57**±0.00	**62**±0

RMSE, and MAPE; in late season, LSTM_S had the lowest average RMSE and MAPE; and in end of season, LSTM_L had the lowest average RMSE and MAPE;

- and for the standard CV experiments, in early season, CNN_L had the highest average R^2 and lowest average RMSE, and LSTM_L had the lowest MAPE; in mid-season, MLP had the best average R^2, RMSE, and MAPE; in late season, MLP had the lowest average RMSE and MAPE; and in end of season, MLP had the lowest average RMSE and MAPE.

For the MY_{2022} experiments, by observing Table 5.4.7, we can see that:

- for block CV experiments, in early season, CNN_L had the lowest average RMSE and MAPE; in mid-season, MLP had the best average R^2, RMSE, and MAPE. Note that other models had similar average MAPE performance, but MLP had the smallest variability in its performance, making it arguably the best; in late season, LSTM_S had the lowest average RMSE and MAPE; and in the end of season, LSTM_S had the lowest average RMSE and MAPE;
- and for standard CV experiments, in early season, CNN_L had the lowest average RMSE and MAPE; in mid-season, LSTM_L had the best average R^2, RMSE, and MAPE; in late season, LSTM_S had the highest average R^2, and LSTM_L had the lowest average RMSE and MAPE; and in end of season, LSTM_L had the lowest average RMSE and MAPE.

To more thoroughly compare models, we performed significance testing by including the performance of each model as a result set in the analysis for early season, mid-season, late season, end of season, and the entire season when considering only MY_{2021} experiment results, only MY_{2022} experiment results, and both MY_{2021} and MY_{2022} experiment results. We included the baseline models for completeness in this analysis, since we found that in some cases the baseline models had better performance metrics than some of the non-baseline models; although we do not elaborate on the baseline model performance, since we have already discussed this topic. Table 5.4.8 illustrates the results of the significance testing analysis and provides the average performance ranking of each model. We can see that a) LSTM_S and LSTM_L were most frequently among the best performing models, b) MLP was next most frequently among the best models, and c) CNN_L and CNN_S were less frequently among of the best performing models, but in early season, they were consistently among the best models in terms of RMSE and MAPE. Interestingly, in one case RF was among the most important models in terms of RMSE in the end of season. This aligns with our findings from our single-year interpolation experiment analysis that RF was best at predicting the low baseline emissions at the end of the season.

We now briefly discuss the differences in model performance between single-year interpolation experiments and multi-year extrapolation experiments. As previously stated in Section 5.4.3.1, for the single-year interpolation experiments, we found that RF and LSTM_L were among the best performing models over the season such that LSTM_L and RF performed better in earlier parts and later parts of the season, respectively. For the multi-year extrapolation experiments, we found that RF was one of the worst models.

In other words, our results suggest RF is a model that better performs interpolation tasks than extrapolation tasks by taking advantage of the temporal autocorrelation that is more commonly available in single-dataset interpolation tasks. This finding is consistent with the findings reported in our study described in Section 4.4.

In summary, CNN, LSTM, and MLP (the deep learning models) had the best average performance over different parts of the season. It is not surprising that deep learning models outperformed the other two ML models (RF and LR), since deep learning has commonly been found to generalize better than more simple models. CNN was found to be best in early season, MLP had better average performance in mid-season (although it did well in later parts of the season, too), and LSTM did generally well across most parts of the season. This is evidence that an ensemble model could be used to increase model performance over each part of the season. Furthermore, what is interesting is that MLP, a tabular model, had comparable performance to that of the tensor-based models; tensor-based models are designed to model time series data, whereas tabular models struggle to learn complex temporal patterns from time series data [150]. MLP likely did reasonably well in this study, because we included lagged features and temporal features in the input datasets, providing MLP with the ability to learn temporal patterns that would be otherwise unavailable or limited without such features.

Takeaway. CNN was one of the best performing models for early season predictions. LSTM and MLP were the other best performing models that did generally well across multiple parts of the season, although MLP did particularly well in mid-season. By making use of temporal features and lagged feature, MLP, a tabular model, was able to compete with the tensor-based models, CNN and LSTM, which are more appropriate models for time series data analysis than tabular models. We found RF is a model that best performs in interpolation tasks and is worse at extrapolation tasks.

5.4.3.2.4 Block vs. Standard Cross-validation In this section we explore, over the entire season, the effects of the choice of CV (block CV vs. standard CV) used for tuning hyperparameters. We performed significance testing to identify any subtle performance differences between CV choices. We performed two different variations of significance testing, namely, multi-model result tests that included the results of all models (excluding LR, since no hyperparameter tuning was performed) and single-model result tests that included results from only individual models. Furthermore, we also performed these two types of tests using results from a) MY_{2021} and MY_{2022} experiments, b) only MY_{2021} experiments, and c) only MY_{2022} experiments. Table 5.4.9 summarizes the results of our significance testing and identifies models that were significantly affected by the CV choice, and identifies what choice of CV led to significantly better performance for each test case. Note that we included only the MY_{2021} experiments for R^2 tests in our analysis, since in the MY_{2022} experiment results, R^2 was mostly zero, meaning significant testing would not be very useful.

We can see in Table 5.4.9 that using standard CV for hyperparameter tuning in no case led to significantly better performance than using block CV, and in many cases using block CV led to significantly better performance than using standard CV. These findings make

Table 5.4.8: N_2O prediction experiments: list of model performance average rankings from the Friedman test for the multi-year extrapolation experiments, where a lower ranking number indicates better ranking performance, marked in bold are the best performing non-baseline models[1], DMem = DSFAMem, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and Both = $MY_{2021} \cup MY_{2022}$.

				Average Performance Ranking for Each Model								
Metric	**Results Analyzed**	**Season**	**Friedman Test**	**CNN_S**	**CNN_L**	**MLP**	**LR**	**LSTM_S**	**LSTM_L**	**RF**	**ZeroR**	**DMem**
RMSE	Both	Early	Reject	**2.75**	**2.25**	5.00	8.38	**3.50**	4.25	7.00	4.25	7.63
RMSE	Both	Mid	Reject	6.13	6.38	**1.63**	9.00	**3.13**	**2.00**	6.25	3.50	7.00
RMSE	Both	Late	Reject	6.25	6.25	**3.00**	9.00	**2.75**	**2.38**	5.50	6.63	3.25
RMSE	Both	End	Reject	6.38	6.75	**3.38**	9.0	**3.50**	**3.13**	**4.75**	6.88	1.25
RMSE	Both	All	Reject	**3.81**	4.27	4.10	8.94	**3.30**	**3.51**	6.20	4.17	6.70
RMSE	MY_{2021}	All	Reject	**3.1**3	3.92	**2.60**	8.94	**2.79**	**3.19**	5.73	6.85	7.85
RMSE	MY_{2022}	All	Reject	**4.50**	4.63	5.60	8.94	**3.81**	**3.83**	6.67	1.48	5.54
MAPE	Both	Early	Reject	**4.63**	**2.38**	**5.00**	9.00	3.88	**2.63**	6.63	3.75	7.13
MAPE	Both	Mid	Reject	5.75	6.00	**2.38**	9.00	**2.38**	**1.75**	6.63	4.25	6.88
MAPE	Both	Late	Reject	6.13	6.13	**3.50**	9.00	**2.75**	**2.63**	5.50	6.88	2.50
MAPE	Both	End	Reject	6.13	6.63	**3.75**	9.00	**3.63**	**3.13**	4.75	7.00	1.00
MAPE	Both	All	Reject	5.65	6.44	**4.33**	8.94	**3.96**	**3.60**	4.96	5.42	1.71
MAPE	MY_{2021}	All	Reject	5.27	6.04	**3.15**	8.92	4.40	**4.44**	4.54	6.13	2.13
MAPE	MY_{2022}	All	Reject	5.63	5.81	5.02	8.98	**3.40**	**3.08**	5.35	5.31	2.42

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

sense, since block CV was designed to favor hyperparameter choices that enable better extrapolation performance.

Takeaway. When used for hyperparameter tuning, block CV generally led to better (in some cases significantly better) performance than standard CV for multi-year N_2O emission extrapolation tasks.

5.4.3.2.5 Effects of Synthetic Soil Nitrogen Content In this section we explore, over the entire season, the effects on model performance of excluding and including the SSNC feature by using a similar approach to the significance testing applied in Section 5.4.3.2.4. For the multi-model result tests, we excluded LR results to avoid having the worst model bias the analysis. Table 5.4.10 summarizes the results of our significance testing and identifies models that were significantly affected by the inclusion or exclusion of the SSNC feature.

We did not find any significant performance differences from excluding or including the SSNC feature for the multi-model tests. This implies that the SSNC feature had no significant impact that consistently affected all the models. Although when examining the effects of the SSNC feature on individual models, we can see that MLP, LR, and RF were affected by the feature. LR performance significantly benefited from excluding the SSNC feature in all cases except for the MY_{2021} experiment R^2 performance, but this is because LR had zero R^2 in all folds and iterations. RF, on the other hand, had a significant benefit to its performance from including the SSNC feature in terms of MAPE. We can also see

Table 5.4.9: N_2O prediction experiments: list of models significantly[1] affected by the choice of cross-validation (CV) type for hyperparameter tuning for the multi-year extrapolation experiments, where MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, Both = $MY_{2021} \cup MY_{2022}$, and Multi-model = every model's performance results were included in the significance testing analysis.

		Models with significantly better performance ranking from using	
Metric	**Results Analyzed**	**Block CV**	**Standard CV**
RMSE	Both	None	None
RMSE	MY_{2021}	Multi-model, RF	None
RMSE	MY_{2022}	None	None
MAPE	Both	Multi-model, CNN_S	None
MAPE	MY_{2021}	Multi-model, RF	None
MAPE	MY_{2022}	None	None
R^2	MY_{2021}	Multi-model, RF	None

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

that MY_{2022}-MLP experiments RMSE performance was significantly better without the SSNC feature. This observation is important, since although RF benefited from including the SSNC feature, MLP was among the best performing models and RF was one of the worst performing models. In other words, to maximize performance, excluding the SSNC feature should be favored.

Takeaway. We showed that the best performing models (CNN, LSTM, and MLP) did not need the SSNC feature to obtain reasonable performance; and in fact, MLP's performance was significantly increased by the exclusion of the SSNC feature. Therefore, we can relax the strong assumption of an exponential decay in changes in nitrification rates over the season, which was a necessary assumption in one of our previous works [150] to obtain reasonable multi-year extrapolation emission prediction performance. Our empirical results suggest that no assumption needs to be made about the N_2O emission dataset's distribution and we can solely rely on features derived from proximal sensor data for performing multi-year N_2O emission extrapolation tasks.

5.4.4 Conclusion

N_2O is a powerful GHG that has approximately 273 times more GWP than CO_2 over a 100-year period, which is one of the reasons why government-enforced GHG reduction legislations have been put in place to pressure farmers into meeting emission reduction targets. By measuring N_2O emissions, a farmer can confirm whether an emission reduction strategy was successful. Unfortunately, high frequency N_2O emission measuring equipment (such as automated gas chambers) is quite expensive, so it is not feasible to expect farmers to purchase such equipment. Therefore, we propose replacing the expensive equipment with an

Table 5.4.10: N_2O prediction experiments: list of models significantly[1] affected by the exclusion or inclusion of the SSNC feature for the multi-year extrapolation experiments, where SSNC = soil synthetic nitrogen content, MY_{2021} = experiments with models trained using 2021 data and tested using 2022 data, MY_{2022} = experiments with models trained using 2022 data and tested using 2021 data, and Both = $MY_{2021} \cup MY_{2022}$.

		Models with significantly better performance ranking from	
Metric	**Results Analyzed**	**Excluding the SSNC feature**	**Including the SSNC feature**
RMSE	Both	LR	None
RMSE	MY_{2021}	LR	None
RMSE	MY_{2022}	MLP, LR	None
MAPE	Both	LR	RF
MAPE	MY_{2021}	LR	RF
MAPE	MY_{2022}	LR	RF
R^2	MY_{2021}	None	None

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

affordable in-field IoT sensing device equipped with data-driven N_2O emission prediction models. Once trained, the models will only require data from inexpensive off-the-shelf sensors to make reasonably accurate predictions. In this study we showed that models trained using only weather and soil data could achieve reasonable emission prediction accuracy. We found that RF and LSTM were the best models for gap-filling interpolation tasks, such that LSTM was significantly better at making earlier-season (when predicting emissions is arguably more important) predictions and RF was significantly better at making later-season (when predicting emissions is arguably less important) predictions. We also found that training a model using one year's data and predicting emissions from another year's data was not a trivial extrapolation task, but reasonable model performance could still be achieved. Deep learning models (CNN, MLP, and LSTM) were able to reasonably predict a) 2021 emissions trends, when trained using 2022 data, and b) 2022 emission trends, when trained using 2021 data. Our study did not have enough available data to conclude whether our models could perform well at predicting emissions from other regions, since we had weather, soil, and N_2O emission data from only a single smart farm. Nevertheless, the research conducted in the present work may provide a useful starting point for performing affordable N_2O emission regional mapping using high spatiotemporal N_2O emission estimations made by data-driven models that are enabled by inexpensive sensors.

Future work is as follows: a) adding soil moisture and temperature sensors that offer readings at various depths to our PN, b) including lagged features of sizes greater than 12 h (e.g., at least 24 h) to determine (via feature selection) if access to more historical data can improve model performance, c) using actual resource-constrained devices (e.g., an NVIDIA Jetson Nano board or a Raspberry Pi) to deploy and evaluate the N_2O emission prediction models, d) repeating the 2023 experiments, but remove emissions from 38 days after fertilizer application to address the class imbalance issue of 2023 datasets mostly containing low baseline emissions, e) deriving and including a feature similar to SSNC,

but instead of deriving SSNC from DSFA, use the DoY of the first large rain event since fertilizer was applied. This feature would presumably help the model identify the period of the largest emission peaks of the season and could improve performance; f) using ReLu as the output layer activation function (instead of the linear function) for the deep learning models to force the predictions to be positive, since N_2O emissions cannot be negative. This could improve model performance, and g) including N_2O emission datasets from different regions and soil types.

5.5 N_2O Emission Prediction Study: Using Temporal Cross-validation, Single-year Interpolation, and Distributed Machine Learning

Most of the content of this study was published in Killeen et al. [136].

The present work is a study that performed N_2O emission prediction using soil and weather sensor data. We made use of existing distributed ML techniques to avoid sharing private data. Specifically, we evaluated models using CL, LL, and three distributed ML methods, FL using FedAvg and two EL methods. We had data from only a single study site despite us wanting to explore the difficulties of extrapolating N_2O emission predictions to other regions (training a model using one farm's data and evaluating the model using a different farm's data), so we partitioned the data among the clients using data splits with varying degree of heterogeneity.

Our goal was to investigate whether accurate N_2O emission predictions could be produced, without needing to share farmers' private data, by training models using existing distributed ML methods (including FL and EL approaches). We used various temporal generalization evaluation methods and we compared models evaluated using IID datasets vs. non-IID client dataset splits.

In terms of how the present work relates to this thesis: the the temporal analysis methodologies applied in the present work tie into the spatiotemporal data analysis aspect of this thesis, and the distributed ML experiments applied are directly tied to the distributed ML aspects of this thesis.

The following is a list of limitations of the present work: a) the data gathered was from only a single study site, b) only FedAvg was considered, which is not an appropriate choice for non-IID client data, and we did not consider other FL aggregator algorithms, c) we did not explore the effects on model performance of varying the number of clients (the number of clients were fixed), d) the networking logic and nodes for the distributed ML experiments were simulated on a single machine, e) we did not perform cross-year extrapolation experiments for evaluating how well models would generalize for client datasets from different growing seasons (we only performed interpolation/gap-filling experiments), and f) we ran ML experiments on a desktop machine instead of evaluating and training models on a resource-constrained IoT device (the type of device we want to deploy the models to in a production setting).

5.5.1 Materials and Methods

5.5.1.1 Datasets

In the present work we included all of the available soil and weather data sources described in Section 5.3.2, although we limited the N_2O emission datasets to only those produced by chambers that monitored the emissions from plots that a) were treated by the conventional broadcast fertilizer application technique and b) did not have any crop growing in the same row as the chambers. That is, we limited our N_2O emission datasets to $C_{2021,4}$, $C_{2022,1}$, and $C_{2023,4}$ (one dataset for each year).

5.5.1.2 Data Pre-processing

We applied the same data pre-processing steps as those described in Section 5.4.1.2, except we only derived an additional 180 min temporal resolution dataset instead of also deriving 60 min and 360 min temporal resolution datasets. This resulted in a total of six datasets (a 30 min and 180 min temporal resolution dataset for each chamber). We then derived lagged features for each of the six datasets using the same process described in Section 5.4.1.2.1, except we did not derive any temporal features. This resulted in datasets having 314 to 419 different features.

5.5.1.2.1 Client Dataset Partitioning Since the present work involved distributed ML experiments, we had to partition our datasets into segments for each client. Given that a) our N_2O emission datasets were temporally heterogeneous, b) the FedAvg algorithm assumes IID client datasets [63], and c) we had data from only a single study site despite us wanting to explore the difficulties of extrapolating N_2O emission predictions to other regions (training a model using one farm's data and evaluating the model using a different farm's data), we partitioned the data among the clients using two types of splits of varying heterogeneity using methods inspired by McMahan et al. [170], as follows:

- **IID** datasets were created by taking advantage of the temporal autocorrelation commonly found in time series agri-data and assigning samples to clients by applying the modulo operation over the sample indices/timestamps. This resulted in client datasets that had a lower temporal resolution by a factor equal to the number of clients relative to the input dataset. For example, in a three-client system with a 12-sample dataset, samples would be assigned as follows:
  - Client 0 = t_0, t_3, t_6, t_9
  - Client 1 = t_1, t_4, t_7, t_{10}
  - Client 2 = t_2, t_5, t_8, t_{11}

  where t_i is the sample taken at time i.

- **non-IID** datasets were created by sorting datasets by timestamp, splitting the dataset into equally sized parts, and then assigning various parts to each client. This resulted in clients having data from different parts of the growing season. In this partitioning scheme, the temporal resolution of each client's dataset retained the original dataset's temporal resolution. For example, in a three-client system with a 12-sample dataset, samples would be assigned as follows:
  - Client 0 = t_0, t_1, t_2, t_3
  - Client 1 = t_4, t_5, t_6, t_7
  - Client 2 = t_8, t_9, t_{10}, t_{11}

  where t_i is the sample taken at time i.

Informally, the heterogeneity (or 'IID-ness') of the splits was quantified by the degree to which the temporal autocorrelation could lead to over-optimistic model performance as a result of having training and testing readings being sampled nearby in time. More formally, the heterogeneity of the splits was quantified by the average temporal distance of the samples between clients; that is, for random samples t_i and t_j taken from the datasets of clients c_0 and c_1, respectively, the time distance between i and j (the timestamps of the samples) would more likely be smaller for IID datasets than for non-IID datasets. It is important to note that the N_2O emissions were non-stationary time series, meaning the distribution of the emissions changed over the season, and we took this into account when defining the heterogeneity of the splits. Strictly speaking, the IID split was not truly IID, since the samples were temporally autocorrelated. A more appropriate definition would be referring to the IID and non-IID datasets as homogeneous and heterogeneous datasets, respectively. Nevertheless, we will loosely use the IID and non-IID terms for notation simplicity. Furthermore, in our FL experiments, we only used a dataset from a single year per experiment, since involving multi-year data would lose focus of the FL evaluation, given the difficulty of the N_2O emission extrapolation problem. We used the 30 min temporal resolution datasets (one for each year) in our FL experiments, which we used as the source dataset to partition into client datasets. It is important to note that this dataset partitioning scheme was not designed to represent realistic data distributions, since we were limited to datasets from a single study site. In practice, the data distributions could be completely different. Investigating under which conditions one should expect IID and non-IID datasets in a realistic N_2O emission prediction setting is beyond the scope of the present work. Nevertheless, we chose this partitioning scheme for its simplicity, since it enabled us to compare model performance using client datasets of varying heterogeneity.

5.5.1.3 Machine Learning Models

We used the same ML learning models as those listed in Section 5.4.1.3, except we did not include the DSFAMem model. We chose these models for the same reasons as the reasons described in Section 5.4.1.3.

5.5.1.4 Evaluation Methods

In addition to using RMSE, R^2, and MAPE as evaluation metrics, we also included model execution time, network communication cost of the simulated network, and FL positive transferability ratio (PTR) as evaluation metrics. FL PTR represents the ratio of clients that benefit from using the global model rather than their local model (e.g., 0.6 would mean 60 % of the clients get better performance from using FL rather than using LL). We also used PTR to compare LL vs. EL by replacing the global model performance with the ensemble model performance in the PTR computation. FL PTR is computed as follows (shown in Equation 5.1) [279]:

$$\text{PTR} = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\left(\theta_k^{\text{G}} - \theta_k^{\text{L}}\right) \tag{5.1}$$

where K is the number of clients, θ_k^G and θ_k^L is the performance of the global model (trained via FL) and client k's local model, respectively, evaluated against client k's testing dataset, and $\mathbb{I}(x) = 1$ if $x > 0$ and $\mathbb{I}(x) = 0$ if $x \leq 0$.

5.5.2 Experiments

In this section we explain our experiments. Our goal was to evaluate a) the accuracy of the predictions made by the models when trained using various learning methods and b) the resource utilization (execution time and network usage) of the models. The ML and deep learning models were implemented using the same execution environment as the one discussed in Section 4.6.2, except we did not use the *geopandas* package in the present work. We trained and evaluated models using the same machine as the one described in Section 4.5.2. The datasets and scripts used to run the experiments discussed in the present work are open source and can be found on GitHub[7].

5.5.2.1 Model Selection

In this section we explain how we performed model selection. This includes feature selection and choosing hyperparameters. Since our experiments involved FL and model sharing combined with distributed EL (we will simply refer to this as EL from now on) such that data was never shared, this means the clients must have previously agreed on the features to use. We also assumed clients agreed before-hand on which hyperparameters their models would use. As such, we initially performed model selection (i.e., feature selection and hyperparameter tuning) via CL to satisfy these assumptions before running the main experiments (described next in Section 5.5.2.2) involving CL, FL (using FedAvg), EL, and LL. We used block CV (described in Section 5.3.4.1) for model selection.

[7] https://github.com/patkilleen/fed_learning_n2o_prediction_2024, accessed on February 25 2025

5.5.2.1.1 Feature Selection Since there were many features in the datasets, we performed feature selection for each of the six datasets using the exact same steps as those described in Section 5.4.1.4. After applying feature selection, the number of selected features for each dataset ranged from 53 to 121.

5.5.2.1.2 Hyperparameter Tuning Each model had static hyperparameter choices for the CL, LL, FL, AE, and SE experiments described in Section 5.5.2.2. Hyperparameter tuning was conducted using random search and nested 4-3-fold block CV with a 4-day cluster size, 3 executions per trial, 100 trials, and a single iteration, using the methodology described in Section 4.3.5.2. The hyperparameter search spaces (and the neural network architectures for the deep learning models) for RF, MLP, 1D CNN, LSTM and SVM are described in Appendix C in Tables C.1, C.2, C.4, C.7, and C.8, respectively. We applied min-max scaling to scale dataset values between 0 and 1 before performing hyperparameter tuning. The datasets provided to the hyperparameter tuning step were the three 180 min temporal resolution datasets, using the features selected from the feature selection step. We chose the 180 min resolution datasets (instead of the 30 min resolution datasets) in order to speed up the tuning process. This should be fine, since a) our experiments involved six clients (meaning the IID experiments would have 180 min resolution datasets, given that 30 min resolution datasets were partitioned into to six client datasets) and b) in our study described in Section 5.4, we found that 180 min resolution datasets led to reasonable performance.

We used a tensor length (number of time steps of an input sample) of 14 for the tensor-based models (CNN and LSTM). We chose 14 as a tensor length, since we found it was a reasonable choice in our study described in Section 5.4.

5.5.2.2 Federated Learning and Model Sharing

We simulated a network of computing nodes on a single machine. We fixed the number of clients to six. The learning methods evaluated in the present work were the same as those explained in Section 4.6.2.2; that is, we evaluated AE, CL, FL, LL, and SE learning methods. For the FL experiments, we chose FedAvg as the FL aggregator algorithm, we set the number of selected clients per round to 4[8], and we set the number of training rounds to 64. Our FL experiments only included MLP, CNN, and LSTM models. Furthermore, we used the same methodology as the one described in Section 4.6.2.2 to train and evaluate the SE learning method.

We used the results of the model selection step (discussed in Section 5.5.2.1) to chose the features and hyperparameters applied by the experiments described in this section. Note that the features selected were those resulting from the feature selection step applied to the 30 min temporal resolution datasets. We applied min-max scaling to the original input dataset before splitting the dataset into client partitions, which required that each client have knowledge about minimum and maximum feature values of other clients (we

[8]each round, 2 clients were not selected, to simulate the cross-device FL setting where network connectivity in a field is unreliable

discuss this point later in this section). We then partitioned the datasets into IID and non-IID client dataset partitions, resulting in 180 min resolution IID datasets and 30 min resolution non-IID datasets. For each learning method we evaluated each model against the two different client dataset splits (IID and non-IID) using 2 iterations of 5-fold block CV with a 2-day cluster size. Note that for AE, FL, LL, and SE, for a given fold, we took the average model performance across all clients and treated this performance as the learning method's performance for that fold.

Note that we assumed for simplicity that the maximum and minimum of each feature in the original data are known by each client to simplify the min-max scaling process when rescaling the dataset values between 0 and 1. For the non-N_2O features, this should be a reasonable assumption, since the features are environmental variables, and so, determining their lower- and upper-bound should be straightforward in practice (e.g., 0 % to 100 % for a soil moisture sensor). For N_2O, estimating the lower- and upper-bound may be less straightforward; however, we argue this assumption is reasonable, since there exist works that perform maximum value consensus while also preserving privacy [62]. Therefore, we assumed some form of privacy preserving technology was used by clients to come to a consensus on what the maximum N_2O emission value of the original data was. For the minimum N_2O value, this can be trivially set to zero, so assuming clients have access to the minimum N_2O value in our experiments should be appropriate. We otherwise did not use (or assumed the use of) any differential privacy or other privacy preserving techniques to protect the model weight updates or model sharing steps.

5.5.3 Analysis and Results

In this section, we discuss the results of our N_2O emission prediction experiments. We explore and analyze the results of the experiments by a) comparing IID client dataset vs. non-IID client dataset results in Section 5.5.3.1, b) comparing learning method results in Section 5.5.3.2, c) comparing the performance of various models in Section 5.5.3.3, and d) analyzing resource utilization in Section 5.5.3.4.

Tables 5.5.1, 5.5.2, and 5.5.3 present the performance of the various training methods for each model for 2021, 2022, and 2023, respectively. Figure 5.5.1 illustrates bar plots of the R^2 performance (averaged over the 3 years) at different points in the season, for RF and LSTM, for the various learning methods. Table 5.5.4 shows the performance of FL and EL compared to LL using the RMSE PTR (PTR_{RMSE}) metric of each model, for each year, and for IID vs. non-IID client datasets. Figure 5.5.2 shows line plots of LSTM predictions vs. actual emissions for each year and for each learning method in both the IID and non-IID client dataset cases. Note that the plotted predictions at a given point in time are the average model predictions over each fold and iteration for an experiment. Table 5.5.6 lists the best performing non-baseline models, in terms of their ability to outperform other models in terms of at least one evaluation metric. Table 5.5.5 shows the average performance ranking of the various learning methods for both the IID and non-IID client dataset cases resulting from a significance test analysis. Table 5.5.7 shows the average performance ranking of the models for both IID and non-IID client dataset cases, for each

learning method, and for R^2 and RMSE, where we indicate whether the Friedman test rejected the null hypothesis. Table 5.5.8 shows the total execution time of the initial hyperparameter tuning experiments (the experiments used to make an appropriate choice of static hyperparameters for the distributed ML experiments). Table 5.5.9 shows the total (over all experiments) sequential and parallel execution times, and network communication costs, of the experiments.

5.5.3.1 IID vs. non-IID Client Datasets

In this section we compare the results of IID vs. non-IID client dataset experiments. By observing results for 2021, 2022, and 2023 in Tables 5.5.1, 5.5.2, and 5.5.3, respectively, we can see that IID results were better than non-IID results, in many cases. As previously mentioned, it is important to note that, by definition, the IID client dataset split results in training and testing data are in fact not truly IID. The samples are temporally autocorrelated, by definition of the lagged features, and a test sample may lie immediately adjacent in time to a training sample when combined for CL. Temporal autocorrelation may also affect the EL and FL experiments, since the local client testing data will be nearby in time to the data used to train each of the local models. The non-IID case may better represent model generalization performance by increasing the distance in time between training and testing samples; however, the non-IID case likely does not reflect the expected client dataset distribution of monitoring N_2O emissions across regions (high emissions in early season and low emissions at the end of season, for each clients' dataset). The non-IID client data split does not capture this, whereas the IID split does. So, the IID dataset experiments may have been slightly over-optimistic, but they used datasets that reflected a more realistic N_2O prediction task, whereas non-IID dataset experiments were not over-optimistic in terms of model performance, but evaluated less realistic scenarios.

Takeaway. IID client datasets led to better model performance for most learning methods than non-IID client datasets. The IID datasets likely better represented N_2O dataset distributions across various regions than the non-IID datasets; however, the IID datasets may have lead to over-optimistic model performance.

5.5.3.2 Learning Method Comparison

In this section we compare the performance of the AE, CL, FL, LL, and SE learning methods by a) comparing the learning methods' performance in the IID and/or non-IID case, b) using PTR_{RMSE} to compare AE, FL, and SE model performance to LL model performance, c) performing significance testing, and d) exploring learning method performance at various parts of the season.

By observing results for 2021, 2022, and 2023 in Tables 5.5.1, 5.5.2, and 5.5.3, respectively, we explore the learning method performance by comparing the results of a) IID vs. non-IID datasets, b) only IID datasets, and c) only non-IID datasets.

- We now compare the results of each learning method for IID vs. non-IID datasets. AE generally did well in the IID case and failed in the non-IID case. This is probably

Table 5.5.1: N_2O prediction experiments: 2021 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods.

		R^2		RMSE		MAPE (%)	
Model	**L.Method**	**IID**	**non-IID**	**IID**	**non-IID**	**IID**	**non-IID**
CNN	AE	0.59±0.28	0.01±0.07	8.40±3.88	11.90±6.99	159±109	348±482
CNN	CL	0.66±0.45	0.32±0.37	7.19±5.76	11.22±4.27	128±119	239±151
CNN	FL	0.09±0.28	0.00±0.00	13.76±4.87	12.71±8.44	334±162	329±412
CNN	LL	0.35±0.36	0.04±0.12	11.45±5.40	4.85±5.25	205±192	60±104
CNN	SE	**0.91**±0.07	**0.38**±0.27	**3.59**±1.47	**2.22**±2.44	**40**±23	**25**±34
MLP	AE	0.83±0.14	0.01±0.05	5.19±1.05	10.79±6.25	81±31	266±383
MLP	CL	**0.93**±0.05	**0.69**±0.24	**3.77**±0.90	7.10±2.20	**68**±20	93±38
MLP	FL	0.75±0.33	0.00±0.02	5.93±4.76	12.31±9.54	142±230	270±332
MLP	LL	0.65±0.22	0.08±0.17	7.48±1.58	4.72±4.58	127±58	60±84
MLP	SE	0.88±0.06	0.45±0.31	4.13±0.85	**1.90**±1.82	71±25	**20**±27
LR	AE	0.71±0.20	0.00±0.02	6.73±1.16	11.48±7.57	135±45	247±369
LR	CL	**0.81**±0.04	**0.67**±0.17	6.38±0.40	8.29±1.72	131±12	164±24
LR	LL	0.52±0.26	0.02±0.08	9.28±2.28	7.00±9.18	187±75	83±101
LR	SE	0.80±0.09	0.31±0.27	**5.58**±1.09	**2.25**±2.37	**116**±38	**21**±26
LSTM	AE	0.94±0.04	0.03±0.11	3.08±0.85	12.47±6.28	36±12	387±558
LSTM	CL	**0.98**±0.00	**0.79**±0.11	2.08±0.15	6.20±1.94	22±10	55±14
LSTM	FL	**0.98**±0.01	0.00±0.03	**2.03**±0.55	17.73±9.09	**21**±12	653±912
LSTM	LL	0.75±0.20	0.11±0.22	6.33±2.80	3.98±4.43	70±29	42±45
LSTM	SE	0.97±0.02	0.45±0.31	2.34±0.61	**1.88**±1.84	31±12	**23**±29
RF	AE	0.94±0.03	0.01±0.03	3.23±1.18	12.81±6.01	21±9	412±526
RF	CL	**0.99**±0.00	**0.81**±0.11	**1.80**±0.23	6.33±2.17	**7**±1	44±18
RF	LL	0.73±0.20	0.08±0.15	6.79±2.51	4.25±4.87	59±33	30±32
RF	SE	0.97±0.01	0.39±0.29	2.02±0.54	**1.99**±2.07	28±13	**19**±21
SVM	AE	0.69±0.18	0.01±0.03	7.07±1.20	12.80±5.54	153±50	376±459
SVM	CL	**0.79**±0.04	**0.71**±0.12	6.76±0.42	7.91±1.46	149±17	160±32
SVM	LL	0.58±0.22	0.03±0.09	8.38±1.55	6.84±4.90	179±66	104±116
SVM	SE	**0.79**±0.08	0.39±0.30	**5.89**±1.36	**1.94**±1.88	**125**±48	**18**±21
ZeroR	AE	0.00±0.00	0.00±0.00	**14.74**±3.83	12.03±7.77	**335**±115	333±420
ZeroR	CL	0.00±0.00	0.00±0.00	15.10±1.61	15.65±3.73	**335**±43	336±42
ZeroR	LL	0.00±0.00	0.00±0.00	14.99±3.78	**5.82**±6.21	340±129	**66**±137
ZeroR	SE	0.00±0.00	0.00±0.00	1.43E+4± 3.49E+4	745.91± 2.40E+3	403E+3± 126E+4	5462± 185E+2

Table 5.5.2: N_2O prediction experiments: 2022 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods.

		R^2		**RMSE**		**MAPE (%)**	
Model	**L.Method**	**IID**	**non-IID**	**IID**	**non-IID**	**IID**	**non-IID**
CNN	AE	0.95±0.04	0.03±0.10	9.06±3.09	30.94±26.49	46±21	461±517
CNN	CL	**0.98**±0.01	**0.90**±0.07	5.96±1.19	14.60±5.05	23±8	55±40
CNN	FL	**0.98**±0.03	0.03±0.10	**5.73**±2.63	28.74±26.73	**22**±16	416±681
CNN	LL	0.76±0.20	0.18±0.28	21.14±9.91	10.88±16.86	116±82	85±147
CNN	SE	0.96±0.07	0.74±0.18	6.15±2.87	**4.65**±7.08	44±19	**22**±38
MLP	AE	0.97±0.02	0.02±0.12	6.65±2.74	30.50±24.02	19±9	580±590
MLP	CL	**0.99**±0.00	**0.86**±0.07	5.04±1.03	15.27±4.50	19±9	34±17
MLP	FL	0.98±0.01	0.01±0.06	5.67±2.61	28.63±29.18	**15**±6	443±579
MLP	LL	0.86±0.15	0.26±0.31	14.88±5.43	8.93±12.48	41±21	45±93
MLP	SE	0.98±0.03	0.74±0.18	**5.02**±2.43	**3.98**±6.62	20±12	**16**±30
LR	AE	0.78±0.24	0.00±0.01	17.13±4.19	47.21±33.98	261±83	1338±2274
LR	CL	**0.89**±0.02	0.60±0.36	16.42±1.27	29.73±19.71	261±44	310±87
LR	LL	0.55±0.37	0.15±0.27	28.64±17.89	27.20±82.23	326±111	114±365
LR	SE	0.84±0.15	**0.73**±0.17	**13.93**±4.34	**4.81**±8.47	**242**±70	**23**±48
LSTM	AE	0.97±0.02	0.03±0.10	7.61±2.89	30.53±26.74	26±9	478±538
LSTM	CL	**0.99**±0.01	**0.90**±0.07	5.49±1.02	14.50±4.42	32±14	37±17
LSTM	FL	**0.99**±0.01	0.02±0.08	**4.71**±1.93	30.99±27.40	**14**±7	532±720
LSTM	LL	0.87±0.10	0.18±0.29	15.78±7.70	9.55±14.45	60±27	58±100
LSTM	SE	0.98±0.02	0.70±0.18	5.82±2.34	**4.82**±7.86	31±15	**22**±42
RF	AE	0.90±0.11	0.01±0.07	11.90±4.71	40.25±19.68	41±28	1076±1160
RF	CL	**0.97**±0.01	**0.65**±0.21	**7.67**±1.82	24.64±7.77	**14**±5	61±35
RF	LL	0.63±0.23	0.17±0.24	24.70±9.37	**11.12**±15.42	76±56	**53**±122
RF	SE	0.93±0.10	0.61±0.21	9.27±4.27	506E+5± 392E+6	46±30	519E+5± 402E+6
SVM	AE	0.73±0.23	0.01±0.08	19.85±3.46	34.30±22.84	366±128	807±923
SVM	CL	**0.88**±0.01	**0.73**±0.14	16.76±1.06	22.16±5.85	352±46	390±95
SVM	LL	0.65±0.26	0.09±0.21	22.90±4.95	10.74±14.10	380±131	68±81
SVM	SE	0.83±0.14	0.49±0.28	**14.83**±3.74	**5.59**±10.00	**291**±97	**25**±48
ZeroR	AE	0.00±0.00	0.00±0.00	**47.83**±12.12	39.20±23.60	**989**±354	955±1009
ZeroR	CL	0.00±0.00	0.00±0.00	49.44±4.66	46.68±14.11	1000±204	1004±236
ZeroR	LL	0.00±0.00	0.00±0.00	49.16±11.78	**15.44**±21.82	1018±446	**84**±190
ZeroR	SE	0.00±0.00	0.00±0.00	6.90E+4± 1.30E+5	697.47±3474.99	2.11E+6± 4.20E+6	2789±8799

Table 5.5.3: N_2O prediction experiments: 2023 model performance for IID and non-IID client datasets, where the data shown are mean ± standard deviation, L.Method = learning method, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and bold font results are the best (in terms of average performance) among the different types of learning methods.

		R^2		RMSE		MAPE (%)	
Model	**L.Method**	**IID**	**non-IID**	**IID**	**non-IID**	**IID**	**non-IID**
CNN	AE	0.82±0.30	0.02±0.08	2.73±1.38	9.65±11.30	64±26	276±278
CNN	CL	**0.99**±0.01	**0.84**±0.09	**1.61**±0.34	5.94±2.63	18±16	38±27
CNN	FL	0.88±0.25	0.00±0.00	1.71±0.89	11.16±14.07	**12**±16	280±183
CNN	LL	0.56±0.38	0.08±0.22	6.26±3.77	3.50±5.35	133±61	68±50
CNN	SE	0.88±0.20	0.43±0.34	1.81±0.97	**1.43**±2.44	55±21	**8**±11
MLP	AE	**0.88**±0.24	0.02±0.11	3.72±1.31	10.76±11.08	113±36	376±327
MLP	CL	0.87±0.29	**0.86**±0.05	4.84±4.99	6.61±2.84	193±238	183±92
MLP	FL	0.82±0.27	0.00±0.00	4.63±2.29	12.71±10.95	186±146	496±402
MLP	LL	0.80±0.26	0.10±0.23	5.71±2.04	3.46±4.57	168±65	82±51
MLP	SE	**0.88**±0.20	0.46±0.33	**3.17**±1.36	**1.46**±2.69	**103**±36	**8**±12
LR	AE	0.73±0.33	0.00±0.00	6.56±5.12	58.50±82.37	188±48	4127±6454
LR	CL	**0.92**±0.02	**0.80**±0.16	4.76±0.58	6.74±1.50	169±23	240±48
LR	LL	0.50±0.38	0.01±0.09	13.94±24.50	10.17±17.05	305±108	119±182
LR	SE	0.80±0.23	0.36±0.28	**4.61**±1.80	**1.88**±3.25	**165**±60	**10**±16
LSTM	AE	0.92±0.22	0.03±0.15	1.58±0.77	11.51±10.75	27±26	427±322
LSTM	CL	**0.99**±0.00	**0.93**±0.02	1.38±0.32	4.39±1.31	12±13	37±29
LSTM	FL	0.97±0.09	0.00±0.00	**1.13**±0.90	7.96±10.11	**5**±4	191±182
LSTM	LL	0.79±0.26	0.09±0.21	3.72±1.92	3.01±4.30	59±36	47±46
LSTM	SE	0.95±0.11	0.48±0.36	1.32±0.78	**1.28**±2.21	22±17	**7**±10
RF	AE	0.90±0.07	0.01±0.07	4.62±2.40	12.26±11.09	12±8	470±293
RF	CL	**0.97**±0.01	**0.80**±0.17	**2.90**±0.97	6.77±1.71	**6**±2	25±23
RF	LL	0.68±0.27	0.08±0.18	8.25±4.17	4.34±6.59	27±22	40±91
RF	SE	0.90±0.07	0.36±0.28	4.11±2.19	**2.05**±3.85	38±33	**10**±15
SVM	AE	0.71±0.30	0.01±0.08	6.94±0.88	11.95±11.43	374±82	428±339
SVM	CL	**0.84**±0.08	**0.66**±0.28	6.67±0.21	8.58±1.95	342±41	404±96
SVM	LL	0.59±0.30	0.06±0.15	9.55±1.92	4.25±5.55	476±116	56±106
SVM	SE	0.76±0.22	0.28±0.24	**5.39**±1.77	**2.22**±3.98	**232**±79	**12**±19
ZeroR	AE	0.00±0.00	0.00±0.00	**17.78**±6.01	14.13±11.43	610±139	611±368
ZeroR	CL	0.00±0.00	0.00±0.00	18.40±3.69	17.51±4.00	610±122	606±127
ZeroR	LL	0.00±0.00	0.00±0.00	18.17±5.89	**4.90**±7.56	**620**±191	**47**±112
ZeroR	SE	0.00±0.00	0.00±0.00	2.79E+4± 1.38E+5	5.81E+4± 4.49E+5	134E+4± 615E+4	232E+3± 169E+4

because averaging model predictions removes the ability of the AE model to predict global trends using local trends, since the local trends are removed when averaging each client's model predictions. This observation may also be due to the training data of the local models being quite similar to the local testing data in the IID case, due to temporal autocorrelation. CL was less affected by the heterogeneity of the client datasets. Non-IID dataset CL (non-IID-CL) results were only slightly worse than IID-CL results. This is probably because CL had the ability to learn global trends by analyzing the union of the entire season's dataset, so even in a more difficult setting where temporal structure will be less prevalent in the dataset, CL could still learn reasonably well. FL, like AE, failed in the non-IID case and generally did much better in the IID case. This may be the case, because both AE and FL performed some form of internal model averaging over all the clients; that is, AE averaged local model predictions and FL averaged local model weight updates. In both cases, this results in losing local trends and the resulting model does not provide utility for predicting global trends. This can be observed in the prediction line plots in Figure 5.5.2, where we can see that both FL and AE model predictions, in the non-IID case, were quite similar in the 2021, 2022, and 2023 seasons. For LL, it generally did much better in the IID case than in the non-IID case. It is important to note that non-IID-LL did not completely fail. This is probably because the emissions were more similar in different parts of the season (e.g., high emissions in early season and low emissions at the end of season), so the LL models were less likely to be hindered by large emission scale variability. That means the presence of large emission spikes that deviate from the average emissions, which would hinder model performance, were probably encountered less frequently by LL models. Similar to CL, SE did quite well in the IID case, and did worse, but reasonably well, in the non-IID case. There are a couple of reasons that could explain this observation:

- The results were over-optimistic and the SE meta-models were taking advantage of temporally autocorrelated residuals of the local models. The applied block CV resulted in samples (and hence predictions) in the folds keeping their order from the original dataset (a source of error that we described in Section 5.3.4.1). So, if model predictions were more similar when made closer in time, it may have made the SE meta-model results over-optimistic. Note that in our study previously described in Section 4.4, we found that LR was less prone to overfitting to the spatial structure in the data (which was caused by SA). Nevertheless, we have no way of ruling out the possibility that SE meta-models were only doing well due to autocorrelated residuals.
- Another possibility as for why the non-IID case enabled reasonable SE performance is that the non-IID case forced local models to be different, due to the different parts of the season having different emission trends. This means the predictions of each local model used to build the level-1 dataset, which were used to train the LR meta-model, would have been different, but would still have had some predictive power, allowing the LR meta-model to take advantage of the variability between model predictions. Ensemble models do best when models are slightly different.

- We now compare learning methods by analyzing only IID results. In general, we can see that LL was the worst performing method, although in some cases it outperformed other methods. Examples of cases that did worse (in terms of at least one evaluation metric) than LL follow: 2021-FL-CNN[9], 2021-FL-MLP, 2023-SE-RF, 2023-CL-MLP, and 2023-FL-MLP. Note that in 2023, there was generally an increased number of low baseline emission samples, and given a) that we set near-zero values to –1 and 1 before MAPE calculations (see Section 5.3.4) and b) reasons previously discussed in Section 5.4.3.2.1 about how MAPE may not be reliable for N_2O emission prediction evaluation, this might be why 2023 LL MAPE results showed superior performance over other methods. Furthermore, we can generally see that CL led to the best R^2 performance, although at times, FL, SE, and in one case AE (2023-AE-MLP), also led to the best performance. What is interesting is that 2021-SE-CNN had much better R^2 performance (0.91) than 2021-CL-CNN (0.66) and had less standard deviation. This might be due to a poor choice of static hyperparameters for CNN (resulting from the hyperparameter search space of random search being too wide), leading to some folds having large differences in performance. This difference in performance may be taken advantage of by SE to reduce variability and improve performance. In terms of RMSE and MAPE, CL, FL, and SE led to the best performance.
- We now compare learning methods by analyzing only non-IID results. We can see that a) CL was generally the best performing method, b) the next best performing method was SE, c) LL generally did much worse than CL and SE, and d) AE and FL generally failed. What is interesting is that CL was generally best in terms of R^2, whereas SE was best in terms of RMSE. In terms of the worst method, FL was the worst in terms of R^2 and was generally worst in terms of RMSE. In terms of MAPE, both AE and FL were the worst methods.

We now analyze the difference between LL and the three distributed ML methods (AE, FL, and SE) by using the PTR_{RMSE} metric (described in Section 5.5.1.4). Note that we only used RMSE for computing PTR, since a) R^2 had many zeros, making ties in model performance hinder the meaning of the PTR_{RMSE}, and b) we found that MAPE was somewhat unreliable in this setting due to the large frequency of low baseline emissions (described in Section 5.4.3.2.1). We did not include ZeroR in this analysis.

- We now examine the differences between FL and LL. By examining Table 5.5.4, we can see that in the IID case, most of the time, the majority of the clients achieved better performance from using the FL global model than their LL model, except for CNN, where on average, most of the clients (67 % = 100 % – 33 %) did better using their LL model. In the non-IID case, on average, the clients had better performance using their LL model than the FL global model.
- We now examine the differences between EL and LL. By examining Table 5.5.4, we can see that for the IID case, for every experiment (except for 2021-LL-CNN vs. 2021-AE-CNN), virtually all the clients had better performance using an ensemble model than their LL model. For the non-IID case, on average, the LL model led

[9] using a '<dataset year>-<learning method>-<model>' naming convention

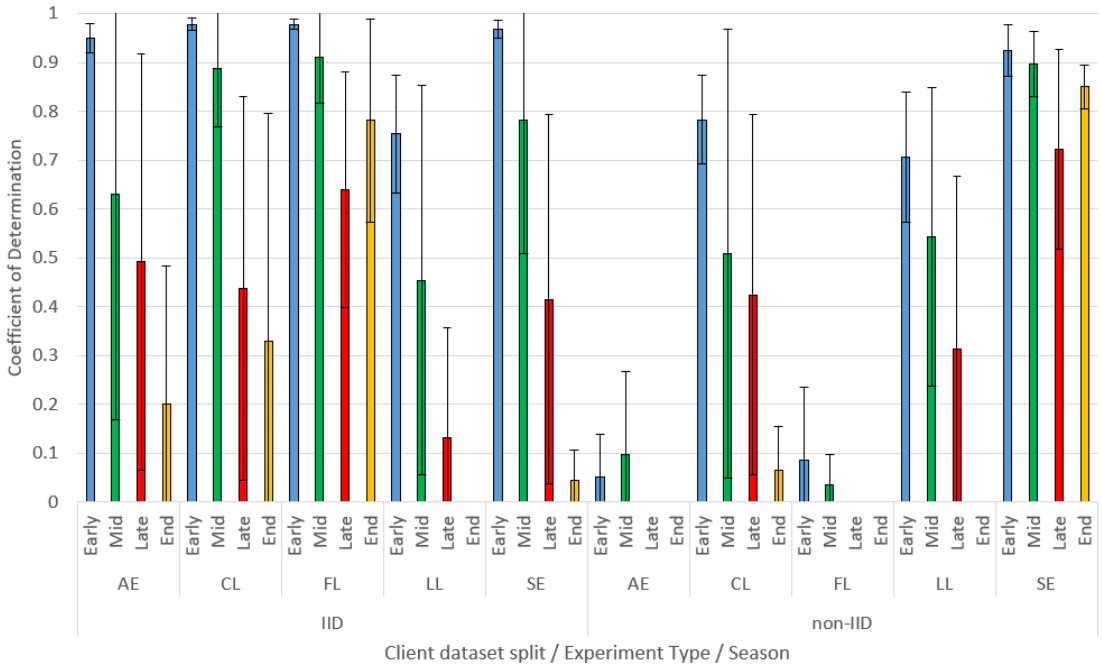
to better performance than the AE model for most clients. On the other hand, the clients in most cases still got better performance using their SE model than their LL model.

To more formally compare the various learning methods, we proceed to use significance testing. In this analysis, we averaged out the performance of each client for each fold and treated that average as the fold's model performance, since, by design, CL did not have a performance metric for each client due to CL gathering all client testing datasets and combining them. This is somewhat of a design limitation, since it would have been interesting to see the differences in performance of the CL model when evaluated against each clients' testing dataset. Note that in this analysis we did not include ZeroR. Furthermore, we performed two batches of significance testing: a) one where FL results were excluded, and both deep learning and ML results were included and b) a second batch where FL results were included, and only deep learning model results were included (since the ML methods did not have FL experiments). By observing Table 5.5.5, we can see that:

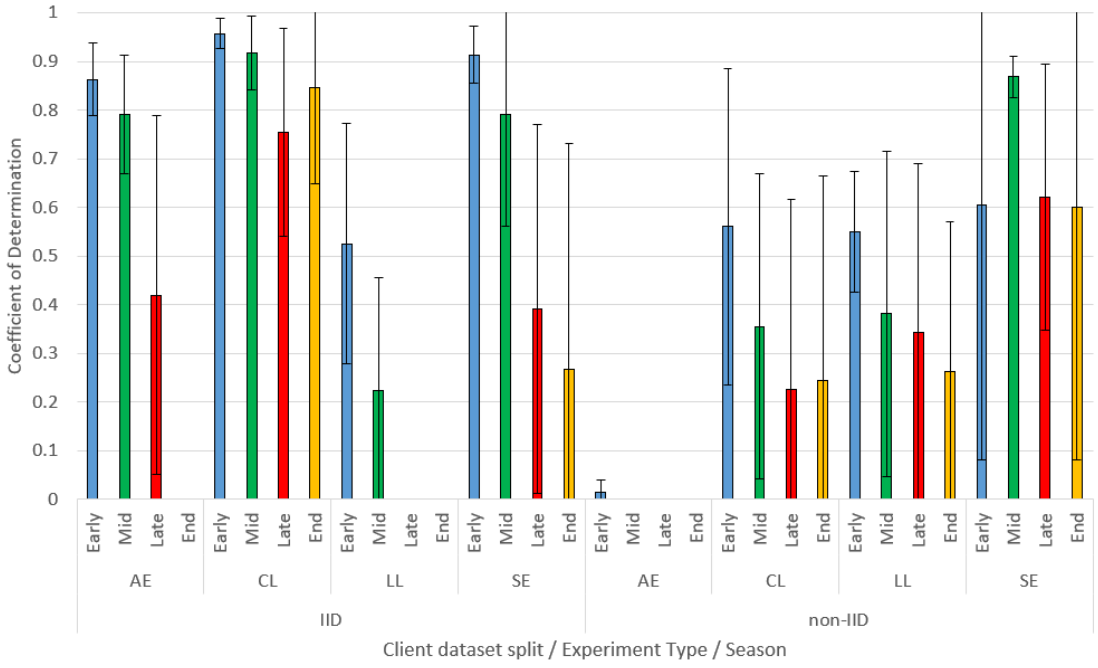
- For the significance tests that included FL results, CL, FL, and SE were among the best ranked learning methods. In three of the four cases (all but non-IID RMSE), CL was among the best ranked methods. In one of the four cases, FL and SE were each among the best ranked methods.
- For the significance test results that excluded FL results, in terms of RMSE, SE was exclusively the best ranked method in both the IID and non-IID case. In terms of R^2, CL was exclusively the best ranked method in both the IID and non-IID case.

To get a better understanding of the differences between learning methods, we analyzed the model performance by season (early season, mid-season, late season, and end of season). We chose to plot only LSTM and RF in Figure 5.5.1, since they were among the best performing models in our study described in Section 5.4. Furthermore, we chose to analyze R^2 performance, since Figure 5.5.1 shows the average performance over 2021, 2022, and 2023, making RMSE not appropriate, given the differences in emission scales between years.

- We can see that for IID-LSTM, FL generally did better than CL in the late season and end of season. This is interesting, although, since the emissions were generally lower at the end of the season regardless of environmental conditions, good end of season model performance is less important for evaluating a model's performance over the entire season.
- For both IID-RF and IID-LSTM, we noticed how SE generally did progressively worse as the season progressed, but for non-IID-RF and non-IID-LSTM, that the decreasing performance trend was less prevalent.
- What is also interesting is that, for both non-IID-RF and non-IID-LSTM, LL did relatively similarly to CL; where, early season LL performance was slightly worse than that of CL, but the mid-season LL performance was better than that of CL.



(a) LSTM



(b) RF

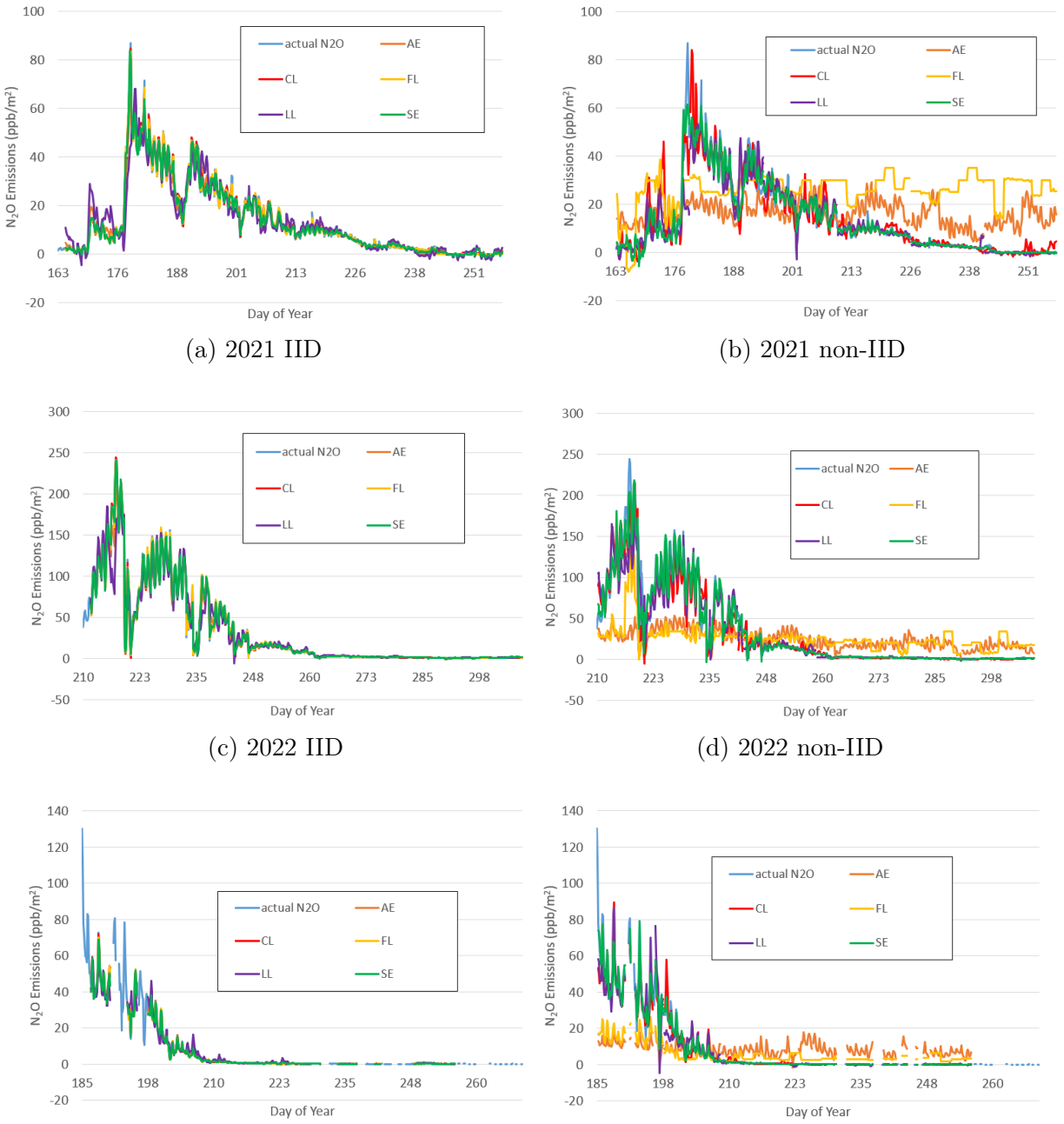
Figure 5.5.1: N_2O prediction experiments: bar plots of LSTM and RF R^2 seasonal performance averaged over 2021, 2022, and 2023 for each learning method, where the error bar represents 1 standard deviation, blue bars are early season, green bars are mid-season, red bars are late season, yellow bars are end of season performance, AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

Table 5.5.4: N_2O prediction experiments: PTR_{RMSE} performance for IID and non-IID client datasets over all three years, where the data shown are mean ± standard deviation, AE = average ensemble learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and N/A = not applicable.

		PTR_{RMSE}								
		2021			**2022**			**2023**		
Model	**Client Data Split**	**LL vs. FL**	**LL vs. AE**	**LL vs. SE**	**LL vs. FL**	**LL vs. AE**	**LL vs. SE**	**LL vs. FL**	**LL vs. AE**	**LL vs. SE**
CNN	IID	0.33±0.26	0.75±0.16	1.00±0.00	0.98±0.05	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
MLP	IID	0.75±0.41	1.00±0.00	1.00±0.00	0.97±0.11	1.00±0.00	1.00±0.00	0.63±0.41	1.00±0.00	1.00±0.00
LR	IID	N/A	1.00±0.00	1.00±0.00	N/A	1.00±0.00	1.00±0.00	N/A	0.98±0.05	1.00±0.00
LSTM	IID	1.00±0.00	1.00±0.00	1.00±0.00	0.95±0.08	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
RF	IID	N/A	1.00±0.00	1.00±0.00	N/A	1.00±0.00	1.00±0.00	N/A	0.98±0.05	0.97±0.11
SVM	IID	N/A	1.00±0.00	1.00±0.00	N/A	0.98±0.05	1.00±0.00	N/A	1.00±0.00	1.00±0.00
CNN	non-IID	0.05±0.08	0.10±0.14	1.00±0.00	0.12±0.11	0.08±0.12	1.00±0.00	0.10±0.09	0.05±0.08	1.00±0.00
MLP	non-IID	0.03±0.11	0.13±0.17	1.00±0.00	0.08±0.12	0.05±0.08	1.00±0.00	0.03±0.07	0.10±0.12	1.00±0.00
LR	non-IID	N/A	0.13±0.13	1.00±0.00	N/A	0.10±0.12	0.97±0.07	N/A	0.10±0.12	1.00±0.00
LSTM	non-IID	0.07±0.12	0.08±0.09	1.00±0.00	0.02±0.05	0.03±0.07	1.00±0.00	0.10±0.12	0.07±0.09	1.00±0.00
RF	non-IID	N/A	0.15±0.12	0.97±0.07	N/A	0.07±0.09	0.98±0.05	N/A	0.12±0.08	0.95±0.08
SVM	non-IID	N/A	0.22±0.14	1.00±0.00	N/A	0.07±0.09	0.97±0.07	N/A	0.12±0.08	1.00±0.00

In summary, we found CL to generally be the best performing learning method in both the IID and non-IID client dataset cases; however, CL does not respect farmer's privacy and does not scale well. In the IID case, FL and SE were also some of the best performing methods with comparable performance to CL. Furthermore, although on one hand the IID data split represents a more realistic emission data distribution between clients, it may have led to over-optimistic results, because of how similar (not independent) the client datasets were (due to temporal autocorrelation between samples). On the other hand, SE also did reasonably well in the non-IID dataset case, but its reasonable performance may have been in part due to autocorrelation between local model residuals. Nevertheless, the findings in this section showed that FL and SE have the potential to enable reasonably accurate region-wide N_2O predictions that somewhat respect the privacy of farmers and can be made using inexpensive off-the-shelf sensors. Furthermore, one of the benefits of SE is that any type of model could be used as a local model, whereas FL requires a model that has a FL aggregator algorithm (like FedAvg for deep learning models). Unfortunately, SE requires labeled data from clients that join the proposed SFS even after the system goes live; therefore, even though SE obtained decent performance, in practice the deployability of SE will be limited by this requirement. Furthermore, when using SE, an analyst must be, careful because the internal representation of data stored in the model could lead to potential privacy violations; for example, the split information of a decision tree's internal node may leak private information about a farmer's dataset [298].

Takeaway. FL and SE did similarly well and competed with CL in terms of performance in the IID client dataset case, whereas in the non-IID case, although FL failed, SE still did reasonably well. However, SE requires every client to have labeled data to train a local meta-learning model, even new clients that join the system after it goes live. Nevertheless, this is empirical evidence that supports the idea that FL and SE have the potential to enable region-wide N_2O mapping applications by providing reasonably accurate predictions while also somewhat respecting farmer's privacy.



(a) 2021 IID

(b) 2021 non-IID

(c) 2022 IID

(d) 2022 non-IID

(e) 2023 IID

(f) 2023 non-IID

Figure 5.5.2: Line plots (at 3 h resolution) of actual N_2O emissions vs. LSTM predictions using various learning methods for 2021, 2022, and 2023, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

Table 5.5.5: N_2O prediction experiments: learning method performance average ranking from the Friedman test, where a lower ranking number indicates better ranking performance, marked in bold are the best performing models[1], AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning.

				Learning Method Average Performance Ranking				
Includes FL	**Client Data Split**	**Metric**	**Friedman Test**	**AE**	**CL**	**FL**	**LL**	**SE**
Yes	IID	R^2	Reject	3.44	**1.43**	2.60	4.76	2.73
Yes	IID	RMSE	Reject	3.60	**2.03**	**2.33**	4.81	**2.22**
Yes	non-IID	R^2	Reject	3.82	**1.19**	4.20	3.06	1.87
Yes	non-IID	RMSE	Reject	4.31	3.10	4.52	2.07	**1.00**
No	IID	R^2	Reject	2.78	**1.23**	N/A	3.97	2.02
No	IID	RMSE	Reject	2.88	1.65	N/A	3.98	**1.49**
No	non-IID	R^2	Reject	3.69	**1.18**	N/A	3.06	1.87
No	non-IID	RMSE	Reject	3.88	2.97	N/A	2.13	**1.02**

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

5.5.3.3 Model Comparison

In this section we compare and analyze the performance differences between models. By observing results for 2021, 2022, and 2023 in Tables 5.5.1, 5.5.2, and 5.5.3, respectively, we compiled a list of the best performing models and present them in Table 5.5.6, such that models in this table are those that outperformed other models in terms of at least one evaluation metric.

We proceed to do significance testing to identify the best performing models. Table 5.5.7 shows the average performance ranking results. We can see that LSTM was among the highest ranked models in all cases. The models next most frequently among the best were MLP and CNN. In two cases RF was among the best ranked models; that is, in the non-IID-LL case. LR, SVM, and ZeroR were never among the best. In the two cases where the Friedman test passed (meaning there was no significant difference between the models), which was in the non-IID-FL case, the models generally failed, which is why they were all similar.

Takeaway. The deep learning models had generally better performance than the ML models. In particular, in all cases, LSTM was found to consistently produce significantly better performance than many other models. RF was the only ML model that could consistently compete in performance with some of the deep learning models.

5.5.3.4 Resource Utilization

In this section we explore the resource utilization of the experiments. It is important to note that because we ran a distributed learning environment on a single machine via

Table 5.5.6: N_2O prediction experiments: list of non-baseline models with the best performing evaluation metrics[1] for each client dataset split, learning method, and dataset, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, and None = all models generally fail.

		Best performing models		
Client Dataset Split	**Learning Method**	**2021**	**2022**	**2023**
IID	AE	LSTM, RF	LSTM, MLP	LSTM, RF
IID	CL	RF	LSTM, MLP, RF	CNN, LSTM, RF
IID	FL	LSTM	LSTM	LSTM
IID	LL	LSTM, RF	LSTM, MLP	LSTM, MLP, RF
IID	SE	LSTM, RF	LSTM, MLP	LSTM
non-IID	AE	None	None	None
non-IID	CL	LSTM, RF	CNN, LSTM, MLP	LSTM, RF
non-IID	FL	None	None	None
non-IID	LL	LSTM, RF	MLP	LSTM, MLP, RF
non-IID	SE	LSTM, MLP, SVM	CNN, MLP	LSTM

[1] from results reported in Tables 5.5.1, 5.5.2, and 5.5.3

simulations, the client model evaluation was done sequentially (models were not evaluated in parallel). We first cover the total execution time taken for the initial hyperparameter tuning and then cover the resource utilization of the main experiments performed in this study.

In terms of hyperparameter tuning, we can see in Table 5.5.8 that the deep learning models took the longest and ML models took much shorter amounts of time.

We now analyze the resource utilization results shown in Table 5.5.9. Note that the network communication cost includes the total amount of data uploaded and downloaded over a client's simulated network link. We proceed by discussing execution time and then network communication costs:

- We now analyze execution times. In terms of execution time, deep learning models took longer than ML models. Among the deep learning models, CNN and LSTM took longer than MLP, possibly because, as tensor-based models, their input samples were larger and so learning took more computing resources per sample. Among the ML models, RF had the longest execution time.
  - By analyzing the sequential execution time, we can see that a) CL had the shortest execution time, b) AE, LL, and SE had the next shortest execution times, and c) FL had the longest execution time. AE and SE always had at least as much execution time as LL, since LL was part of the AE and SE experiments.
  - We now analyze parallel processing execution times by dividing the total sequential execution time that could be parallelized by the total number of clients. We can see that CL no longer had the shortest execution time; that is, AE, LL, and

Table 5.5.7: N_2O prediction experiments: model performance average ranking from the Friedman test, where a lower ranking number indicates better ranking performance, marked in bold are the best performing models[1], AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, and SE = stacked ensemble learning, and N/A = not applicable.

				Average Performance Ranking for Each Model						
Client Data Split	**Learning Method**	**Metric**	**Friedman Test**	**CNN**	**MLP**	**LR**	**LSTM**	**RF**	**SVM**	**ZeroR**
IID	AE	R^2	Reject	3.69	2.61	4.59	**1.70**	2.98	5.33	6.78
IID	AE	RMSE	Reject	3.46	2.70	4.77	**1.62**	2.99	5.53	6.93
IID	CL	R^2	Reject	2.87	3.07	5.00	**1.63**	2.77	5.67	6.90
IID	CL	RMSE	Reject	2.97	3.03	5.03	**1.67**	2.67	5.70	6.93
IID	FL	R^2	Reject	2.39	2.25	N/A	**1.32**	N/A	N/A	N/A
IID	FL	RMSE	Reject	2.34	2.33	N/A	**1.32**	N/A	N/A	N/A
IID	LL	R^2	Reject	4.03	**2.46**	4.29	**2.14**	3.45	4.27	6.31
IID	LL	RMSE	Reject	3.88	2.64	4.66	**2.07**	3.54	4.51	6.69
IID	SE	R^2	Reject	2.88	2.94	4.88	**1.92**	2.95	5.42	6.99
IID	SE	RMSE	Reject	2.72	2.97	4.94	**1.89**	3.02	5.52	6.94
non-IID	AE	R^2	Reject	**1.22**	**1.20**	1.29	**1.13**	1.26	1.24	1.33
non-IID	AE	RMSE	Reject	**3.11**	**3.11**	4.98	**3.47**	4.51	3.92	4.92
non-IID	CL	R^2	Reject	3.53	2.97	4.43	**1.80**	3.57	4.70	6.80
non-IID	CL	RMSE	Reject	3.53	2.83	4.50	**1.93**	3.60	4.70	6.90
non-IID	FL	R^2	Pass	-	-	N/A	-	N/A	N/A	N/A
non-IID	FL	RMSE	Pass	-	-	N/A	-	N/A	N/A	N/A
non-IID	LL	R^2	Reject	**1.91**	**1.72**	2.12	**1.76**	**1.92**	2.10	2.43
non-IID	LL	RMSE	Reject	**4.18**	**3.76**	4.55	**3.47**	**3.06**	4.97	4.01
non-IID	SE	R^2	Reject	3.26	**2.81**	3.70	**2.89**	3.78	4.36	6.72
non-IID	SE	RMSE	Reject	**3.72**	**3.29**	3.76	**3.44**	3.92	4.08	5.78

[1] In terms of the Friedman test and Wilcoxon's signed-rank test methodology described in Section 4.3.6.

Table 5.5.8: N_2O prediction experiments: initial hyperparameter tuning process total execution time (min = minute) for each dataset.

	Total Execution Time (min)		
Model	**2021**	**2022**	**2023**
CNN	607	613	591
MLP	549	547	546
LSTM	715	709	712
RF	60	55	43
SVM	6	1	0

Table 5.5.9: N_2O prediction experiments: sequential and parallel total execution time and simulated network communication cost for each model and learning method, where AE = average ensemble learning, CL = centralized learning, FL = federated learning, LL = local learning, SE = stacked ensemble learning, N/A = not applicable, and NIY = not implemented yet (since the network cost of sending ML models over the network was not considered).

	Total Sequential Execution Time (min)					**Total Parallel Execution Time (min)**					**Total Network Communication (MB)**				
Model	**AE**	**CL**	**FL**	**LL**	**SE**	**AE**	**CL**	**FL**	**LL**	**SE**	**AE**	**CL**	**FL**	**LL**	**SE**
CNN	74	57	2973	73	74	13	57	496	12	13	1922	1190	27,662	0	1922
MLP	60	22	2387	60	60	11	22	398	10	11	270	92	3885	0	270
LR	0	0	N/A	0	0	0	0	N/A	0	0	NIY	92	N/A	0	NIY
LSTM	84	73	3289	83	84	15	73	548	14	15	182	1190	2614	0	182
RF	10	15	N/A	10	10	2	15	N/A	2	2	NIY	92	N/A	0	NIY
SVM	0	0	N/A	0	0	0	0	N/A	0	0	NIY	92	N/A	0	NIY
ZeroR	0	0	N/A	0	0	0	0	N/A	0	0	NIY	92	N/A	0	NIY

SE now had the shortest execution time, and FL still had the longest execution time.

- We now analyze network communication costs. By observing CL network costs, we can see that the tabular model network costs (92 MB) were smaller than the tensor-based model network costs (1190 MB). This observation is due to a limitation of our experiments' design. Instead of sending the raw datasets to the simulated cloud, data pre-processing was first applied. This meant that tensor-based models had to send approximately 14 times more data, given that their input samples had a temporal dimension and were 14 times larger than the tabular-based model samples. In practice the clients could first send their raw dataset file to the cloud, and then the server could do the pre-processing to save on bandwidth. Furthermore, we can also notice that CNN had more communication costs than MLP and LSTM, and this is probably because CNN had more parameters than the other two models.

  Furthermore, LL had no network communication cost, CL had the next least amount of network communication costs, AE and SE had approximately twice the network cost of CL, and FL had the largest network cost. It is interesting that CL had less network communication costs than FL, since one of the advantages of FL often claimed in the literature is that FL reduces communication costs. CL may have been less network intensive than FL, because the datasets used in this study were relatively small. If we had much larger datasets, then sending model weights over the network, over multiple rounds, would require less bandwidth than sending a single large dataset over the network. We can also notice that FL had approximately 20 times more communication costs than both AE and SE. This is interesting, since we used 64 communication rounds in FL, so clients would need to send model updates 64 times over the network. One would expect SE and AE to have had around 64 times less network communication than FL, given that model sharing was only done

once for EL methods. This can be explained as follows: in one round of FL using FedAvg, each of the 6 clients uploaded their local weight updates and downloaded the global weights, meaning $2 \times 6 = 12$ model updates were sent over the network for a single round of communication. Since we used 64 rounds of FedAvg, that means 768 (64×12) model updates were sent over the network during FL. For both EL approaches, there was only one model sharing phase when each client sent its model to each other client, meaning each client downloaded 5 ($6 - 1$) model updates and each client uploaded 5 model updates during EL. This resulted in a total of 10 model updates sent over the network per client, for a total of 60 model updates. EL had approximately 12.8 times fewer model updates than FL, which somewhat aligns with the relative difference in observed communication costs between FL and EL. Note that we did not include the ML network communication costs due to time constraints. We leave this as future work.

Takeaway. In a realistic environment, when model evaluation was done in parallel over multiple client devices, AE, LL, and SE had the shortest execution time and FL had the longer execution time. In terms of simulated network communication cost (for methods that communicate over the network), CL had the smallest communication cost and FL had the largest communication cost. FL was found to have poor resource utilization compared to the other learning methods, although this was partly due to the use of small datasets in the present work. In a scenario with larger datasets, FL would have had more a of chance of executing faster and requiring less network bandwidth than CL. Nevertheless, we found EL (via model sharing) to have some of the best resource utilization without needing to share private data.

5.5.4 Conclusion

We proposed an IoT-based smart farming architecture that supports inexpensive region-wide PA services enabled by data-driven model predictions. We showcased our architecture and learning methodology in the N_2O emission prediction use case. We showed that PA models combined with our architecture could be trained using distributed EL (via model sharing) and FL to achieve reasonable N_2O emission prediction accuracy, while also somewhat satisfying the privacy concerns of farmers by never sharing farmers' data. By somewhat respecting farmers' privacy, SFS adoption rates will increase, and this will lead to increased amounts of available training data. As a result, increased PA model performance can be expected from this additional training data. Our architecture has the potential to enable low-cost in-field gateway deployment for estimating agri-variables that would otherwise be too expensive to sense. This is achieved by replacing expensive sensing equipment with a gateway (a.k.a. a PN) that has a PA model deployed on-board, such that the model was trained using a ground truth dataset gathered using expensive sensing equipment. We gathered N_2O emission data from expensive automated gas chambers over three years (2021, 2022, and 2023), and we also gathered a) soil and weather information from nearby Pessl sensors and b) soil data from a Raspberry Pi-based prototype of our architecture's PN. We used FedAvg as our FL aggregator algorithm and included LR, RF,

SVM, LSTM, CNN, and MLP models in our experiments. We also compared the performance of models trained via FL, CL, LL, and two EL methods (AE and SE). Furthermore, we investigated the effects of homogeneous (IID) client datasets vs. heterogeneous (non-IID) client datasets on learning performance. We simulated a network of six clients on a single machine. We found that FL and SE do similarly well and compete with CL in terms of performance in the IID client dataset case, whereas in the non-IID case, SE still did reasonably well (although FL failed); however, SE requires new clients that join the system to have labeled training data, which may be expensive to gather and limits the practical applicability of SE in this setting. Nevertheless, this is empirical evidence that supports the idea that FL and SE have the potential to enable region-wide N_2O mapping by providing reasonable prediction performance while also somewhat respecting farmer's privacy. It also suggests that our architecture could be applied to other application domains for reducing sensing costs by replacing expensive sensing equipment with our PN. However, it is important to note that privacy issues are not all solved by simply applying FL for PA model training [195, 63]. Although raw data is never shared, sharing model weight updates can still infringe on farmer privacy, since data points could be reconstructed. This point is relevant to the EL as well. Additional cybersecurity mechanisms should be applied (e.g., homomorphic encryption and differential privacy) [195], which are often applied to model weight updates [6], and this should be clear to stakeholders and policy makers.

We proceed to list future work avenues: a) varying the number of clients to explore the effects on model performance; b) using other FL aggregator algorithms (e.g., FedProx [149] or Federated Deep Learning [98]) other than FedAvg. We only used FedAvg, which is a limitation of the present work, since FedAvg expects IID data to perform best [63] and N_2O emissions are quite heterogeneous by nature; c) performing similar experiments on datasets other than N_2O emissions, for example, yield prediction using imagery[10], d) evaluating N_2O prediction models on a resource-constrained devices such as a Raspberry Pi or an NVIDIA Jetson Nano board, and even deploying many such devices in the experiments to involve actual client devices instead of only simulating clients, e) performing extrapolation distributed ML experiments that involve training models using one year's data and testing the models on another year's data instead of only performing single-year interpolation experiments, and f) exploring the use of various privacy preserving technologies and analyzing the trade-offs between privacy, model accuracy, and computational complexity.

[10] We explored this in the study described in Section 4.6.

Chapter 6

Conclusion

In this thesis we analyzed spatiotemporal IoT sensor data and used privacy-aware distributed EL and FL, to avoid sharing raw data, in a fog computing environment for smart farming applications. We showcased our methodology for two use cases, crop yield prediction and N_2O emission prediction, where we gathered our own IoT sensor data from a single smart farm. This data enabled us to carry out our spatiotemporal data analysis and distributed ML experiments, and helped us confirm that there is potential for replacing expensive sensing equipment with affordable sensors combined with models trained via distributed ML. Our methodology is not limited to PA, it could be applied to other domains that have expensive and/or low spatiotemporal resolution sensing equipment that can be replaced by models combined with affordable sensors (e.g., air pollution prediction).

The remainder of this chapter concludes this thesis, summarizes the presented work, discusses limitations of the work, and presents potential future work avenues.

6.1 Summary

Farmers will be expected to produce more food due to an increasing world population despite challenges like climate change. PA and smart farming can be used to help farmers achieve this goal. By gathering site-specific agri-data using sensors and feeding the data to predictive analytics models, input cost reductions and agriculture resource optimizations can be achieved. Farmers are reluctant to share their data, which leads to poor SFS adoption rates. Lower adoption rates means less training data is available, and models need as much training data as possible to maximize performance. To address this, on-site local data analytics models could be applied to avoid sharing data, but applying LL to agri-data may lead to loss of useful geographical trends that could have been useful to a centralized model. SFSs will need to respect farmers' privacy concerns, and they will need to deal with the segmented islands of agri-data that are stored in different regions. Furthermore, an analyst should be careful when training and evaluating PA models, since agri-data tend to have a highly dynamic spatiotemporal nature. PA model performance

evaluation risks being over-optimistic if spatiotemporal structure is not considered in the model's evaluation.

This thesis involved the analysis of spatiotemporal IoT sensor data for smart farming applications by using privacy-aware distributed ML techniques. A privacy-aware IoT-based smart farming architecture that uses distributed ML was proposed, such that PA models are trained without needing to share farmers' private data. Expensive sensing equipment is first required to gather expensive-to-sense labeled agri-data for training the models, but once the PA models are trained, new farmers can join the system and benefit from the models without needing any expensive equipment. Only affordable sensors and an on-site device for hosting the local PA model logic is required. Furthermore, by somewhat respecting farmers' privacy and making the system affordable, our proposed system will have reasonable adoption rates, and this will lead to increased amounts of available training data. As a result, increased PA model performance can be expected. We had access to a Canadian smart farm called Area X.O and used it as the study site for our studies. We gathered data from 2020, 2021, 2022, and 2023 growing seasons. The data consisted of N_2O emissions, crop yield information, crop imagery, management information, soil, and weather data. We used this data to conduct PA experiments to showcase our proposed distributed ML methodology, where we focused on two use cases, namely crop yield prediction and N_2O emission prediction. We conducted yield prediction studies using UAV imagery and yield data. We conducted N_2O emission prediction studies using N_2O, weather, and soil data. We took into account the spatiotemporal heterogeneity of agri-data by making use of various block CV model evaluations techniques. These techniques included spatial and temporal CV, where the datasets were partitioned into spatially (or temporally) disjointed training and testing (and validation) datasets. We simulated a network of six client nodes on a single machine to perform distributed ML experiments. The distributed ML methods used were two EL approaches (SE and AE) and FL (using FedAvg), and we also included CL and LL as additional performance baselines. We also investigated the effects of homogeneous (IID) client datasets vs. heterogeneous (non-IID) client datasets on learning performance.

From the results of our PA experiments, we found that CL and SE were generally the best performing learning methods, although a) CL does not respect farmers' privacy concerns and does not scale well and b) SE requires new clients (clients that join after the distributed training is complete and the system goes live) to have labeled data. AE and FL somewhat respect farmers' privacy concerns, are more scalable than CL, and do not need new clients to have labeled data to benefit from a deployed PA model. We found that in cases when the datasets were IID in nature, AE and FL could do reasonably well and even compete with CL in terms of model performance, but it is not reasonable to assume agri-data are IID; they are non-IID by nature. For non-IID dataset experiments, AE and FL did more poorly and failed on multiple occasions. In particular, AE and FL completely failed when using non-IID datasets for N_2O emission prediction. Despite this, by using AE and FL, we were able to generate useful yield precision maps for most clients even when non-IID datasets was being used. Our results demonstrated that our smart farming architecture and methodology can potentially be used to replace expensive yield monitors and N_2O emission sensing equipment. This would be done using inexpensive sensors combined with fog-deployed predictive analytics models without needing to share

farmers' data. Furthermore, although we limit the scope of this thesis to only the yield prediction and N_2O emission prediction domains, our proposed architecture is general enough that it could be applied to other application domains (e.g., air pollution monitoring) for achieving high spatiotemporal resolution sensing in a a) privacy-aware, b) affordable, and c) scalable manner, at the cost of reduced sensing accuracy.

In terms of how relevant the results of this thesis will be in the future given the quick advancement pace of modern deep learning techniques, the methodology applied in this thesis should remain useful as long as there exist expensive and/or low spatiotemporal resolution sensing solutions that are replaceable (at the cost of lower sensing accuracy for increased spatiotemporal resolution sensing) by affordable sensors combined with models.

6.2 Limitations

It is important to note that this thesis has not solved everything related to building affordable PA models that a) respect farmers' privacy and b) generalize well to new regions (and periods) of interest. We proceed to discuss the limitations of this thesis.

- It is important to note that farmer privacy issues are not all solved by simply applying FL or distributed EL for PA model training. Our proposed architecture and distributed ML methodology do not provide any formal cybersecurity benefits other than slight improvements to baseline privacy by avoiding sharing raw data. This should be clear to stakeholders and policy makers, because, although raw data is never shared, sharing model weight updates (or entire models) can still infringe on farmer privacy, since data points can be reconstructed or model updates may leak private information. Additional cybersecurity mechanisms should be applied (e.g., homomorphic encryption and/or differential privacy) to harden distributed ML-based privacy-enabling solutions [63, 195]. This thesis did not use privacy-preserving technologies.

- Wireless network connectivity is assumed to be available in the proposed IoT architecture, but in practice wireless network coverage is a known challenge on farm land, especially for large farms [268].

- The proposed architecture was not formally or empirically evaluated; it was proposed simply as a means to an end for showcasing our distributed ML methodology for spatiotemporal IoT sensor data analysis.

- The distributed ML experiments done in this thesis were simulated on a single machine instead of deploying clients on a distributed network of machines.

- The agri-data used in this thesis were obtained from only a single study site. So strong conclusions about the spatial extrapolation performance of models were limited by the lack of data from other regions. Farms each have their own unique soil properties and local weather patterns, meaning without data from other farms, models trained using

data from only a single farm will be expected to fail to generalize to new locations when the models are given previously unseen soil and weather data patterns.

- Our FL experiments only used the FedAvg aggregator algorithm. FedAvg assumes IID data [63], but agri-data tend to be heterogenous (non-IID) by nature, so FedAvg was not appropriate for this setting. Nevertheless, it was chosen for its simplicity as a proof of concept of the proposed privacy-aware distributed ML methodology.
- We did not verify if the local model predictions used to train the SE meta-models were spatially or temporally autocorrelated, so we cannot rule out that the reasonable SE performance could have been over-optimistic due to autocorrelated residuals.

6.3 Future Work

We conclude this thesis by discussing future work avenues:

- In terms of the yield prediction work, evaluating model temporal generalizability by training a model using one year's data and then evaluating the model against another year's data.
- In terms of N_2O emission prediction work, performing extrapolation distributed ML experiments that involve training models using one year's data and testing the models on another year's data, instead of only performing single-year interpolation/gap-filling distributed ML experiments.
- Instead of considering data from only a single farm, performing N_2O emission prediction and yield prediction experiments involving multi-farm extrapolation performance evaluation by training a model using one farm's data and evaluating the model against another farm's data.
- Analyzing the importance of sensors by systematically leaving out dataset features derived from specific sensors, and examining the effects on N_2O emission prediction model performance. This could help us deploy our system by identifying how many redundant sensors or what types of sensors are required for reasonable model accuracy, which could help with deployment cost estimations.
- In terms of FL experiments:
  - Instead of only using FedAvg, including additional FL aggregator algorithms such as FedProx [149] or Federated Deep Learning [98] would be interesting.
  - Investigating the use of Personalized FL instead of FedAvg to tune the global model to the unique distribution of each client's local dataset [244, 279, 147], so that all clients can benefit from the global model.
- In terms of distributed EL experiments:

  - Evaluating a variation of AE that uses the maximum of local model predictions instead of the average. This could be interesting, since AE was able to reasonably identify low yield areas, so maybe the maximum operation could help identify high yield areas as well.
  - Evaluating a variation of AE, k-nearest-clients: when a new client i joins the system, building an ensemble of models would involve combining local models from only the k nearest clients in geographical space. The intuition is that the models of these k clients would more likely capture local weather patterns and other environmental trends that are unique to a region of interested, than the models from clients further away.
  - Evaluating a variation of AE that uses weighted averaging and defines the weights applied to each model's prediction based on the average yield (or N_2O emission) of the local model's private dataset. This could improve performance, although it does not meet privacy requirements, so privacy preserving technologies like homomorphic encryption would need to be applied to safeguard the shared average target variable values.
  - Investigating whether the local model predictions used to build the meta-model in SE were spatially (or temporally) autocorrelated in both the yield prediction and N_2O emission prediction experiments, to determine if the SE performance results in the present work were over-optimistic.

- In terms of the distributed ML experiments:
  - Instead of simulating distributed ML experiments on a single machine, using resource-constrained devices (e.g., an NVIDIA Jetson Nano board or a Raspberry Pi) to train and evaluate models to better explore the feasibility of deploying such models to resource-constrained execution environments.
  - Applying a leave-one-client-out CV evaluation methodology to evaluate how well models do when making predictions for a new client that joins the system.
  - Varying the number of clients, instead of fixing the number of clients to six.
  - Exploring the use of various privacy preserving technologies and analyzing the trade-offs between privacy, model accuracy, and computational complexity.

Appendices

This section contains the appendices of this thesis. Appendix A lists the equations and definitions of VIs used in this thesis. Appendix B presents the tables used to compare related work to the studies presented in this thesis. Appendix C presents the hyperparameter search spaces used and other model hyperparameter details related to the models used in this thesis. Appendix D provides additional supplementary material.

Appendix A

Vegetation Index Equations

In this section, the VIs used in our yield prediction studies (described in Chapter 4) are listed in the following tables: VIs derived using NIR and RGB bands are in Table A.1, VIs derived using only RGB bands are in Table A.2, and VIs derived using the red-edge band are in Table A.3.

Table A.1: Vegetation indices, derived using near-infrared (NIR) and red-green-blue (RGB) bands, involved in this thesis, where ρ_i = the reflectance of the spectral band i.

Vegetation Index	**Description**	**Equation**	**Reference**
$\text{DVI}_{\text{NIR,Red}}$	NIR-red difference vegetation index	$\rho_{\text{NIR}} - \rho_{\text{Red}}$	[263]
$\text{DVI}_{\text{NIR,Green}}$	NIR-green difference vegetation index	$\rho_{\text{NIR}} - \rho_{\text{Green}}$	[263]
NDVI	normalized difference vegetation index	$(\rho_{\text{NIR}} - \rho_{\text{Red}}) \,/\, (\rho_{\text{NIR}} + \rho_{\text{Red}})$	[296]
$\text{NDVI}_{\text{Blue}}$	blue normalized difference vegetation index	$(\rho_{\text{NIR}} - \rho_{\text{Blue}}) \,/\, (\rho_{\text{NIR}} + \rho_{\text{Blue}})$	[283]
$\text{NDVI}_{\text{Green}}$	green normalized difference vegetation index	$(\rho_{\text{NIR}} - \rho_{\text{Green}}) \,/\, (\rho_{\text{NIR}} + \rho_{\text{Green}})$	[283]
RDVI	renormalized difference vegetation index	$(\rho_{\text{NIR}} - \rho_{\text{Red}}) \,/\, \sqrt{(\rho_{\text{NIR}} + \rho_{\text{Red}})}$	[296]
WDRVI	wide range vegetation index	$(\alpha\rho_{\text{NIR}} - \rho_{\text{Red}}) \,/\, (\alpha\rho_{\text{NIR}} + \rho_{\text{Red}})$, where $\alpha = 0.2$	[167]
SAVI	soil-adjusted vegetation index	$(1 + L)\,(\rho_{\text{NIR}} - \rho_{\text{Red}}) \,/\, (\rho_{\text{NIR}} + \rho_{\text{Red}} + L)$, where $L = 0.5$	[107]

Table A.1: *Cont.*

Vegetation Index	**Description**	**Equation**	**Reference**
MSAVI	improved SAVI	$0.5\left(2\rho_{\text{NIR}}+1-\sqrt{\left(2\rho_{\text{NIR}}+1\right)^2-8\left(\rho_{\text{NIR}}-\rho_{\text{Red}}\right)}\right)$	[95]
OSAVI	optimized soil-adjusted vegetation index	$\left(1+L\right)\left(\rho_{\text{NIR}}-\rho_{\text{Red}}\right)/\left(\rho_{\text{NIR}}+\rho_{\text{Red}}+L\right)$, where $L=0.16$	[223]
$\text{SRI}_{\text{NIR,Red}}$	NIR-red simple ratio index	$\rho_{\text{NIR}}/\rho_{\text{Red}}$	[40]
$\text{SRI}_{\text{NIR,Green}}$	NIR-green simple ratio index	$\rho_{\text{NIR}}/\rho_{\text{Green}}$	[167]
MSR	modified simple ratio index	$\left(\left(\rho_{\text{NIR}}/\rho_{\text{Red}}\right)-1\right)/\left(\sqrt{\rho_{\text{NIR}}/\rho_{\text{Red}}+1}\right)$	[94]
GCVI	green chlorophyll vegetation index	$\left(\rho_{\text{NIR}}/\rho_{\text{Green}}\right)-1$	[112]
NG	normalized green index	$\rho_{\text{Green}}/\left(\rho_{\text{NIR}}+\rho_{\text{Red}}+\rho_{\text{Green}}\right)$	[166]
MCARI2	improved MCARI	$\frac{1.5\left[2.5\left(\rho_{\text{NIR}}-\rho_{\text{Red}}\right)-1.3\left(\rho_{\text{NIR}}-\rho_{\text{Green}}\right)\right]}{\sqrt{\left(2\rho_{\text{NIR}}+1\right)^2-\left(6\rho_{\text{NIR}}-5\sqrt{\rho_{\text{Red}}}\right)-0.5}}$	[94]
MTVI2	modified triangular vegetation index 2	$\frac{1.5\left[1.2\left(\rho_{\text{NIR}}-\rho_{\text{Green}}\right)-2.5\left(\rho_{\text{Red}}-\rho_{\text{Green}}\right)\right]}{\sqrt{\left(2\rho_{\text{NIR}}+1\right)^2-\left(6\rho_{\text{NIR}}-5\sqrt{\rho_{\text{Red}}}\right)-0.5}}$	[94]

Table A.2: Vegetation indices, derived using only red-green-blue (RGB) bands, involved in this thesis, where ρ_i = the reflectance of the spectral band i between 0 and 1.

Vegetation Index	**Description**	**Equation**	**Reference**
r	red chromatic coordinate	$\rho_{\text{Red}} / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[294]
g	green chromatic coordinate	$\rho_{\text{Green}} / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[294]
b	blue chromatic coordinate	$\rho_{\text{Blue}} / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[294]
$\text{MSR}_{\text{Green,Red}}$	MSR with NIR band replaced by green	$((\rho_{\text{Green}}/\rho_{\text{Red}}) - 1) / \left(\sqrt{\rho_{\text{Green}}/\rho_{\text{Red}} + 1}\right)$	This thesis
$\text{SRI}_{\text{Green,Red}}$	green-red simple ratio index	$\rho_{\text{Green}}/\rho_{\text{Red}}$	[294]
$\text{SRI}_{\text{Red,Green}}$	red-green simple ratio index	$\rho_{\text{Red}}/\rho_{\text{Green}}$	[227]
$\text{SRI}_{\text{Green,Blue}}$	green-blue simple ratio index	$\rho_{\text{Green}}/\rho_{\text{Blue}}$	[294]
$\text{SRI}_{\text{Red,Blue}}$	red-blue simple ratio index	$\rho_{\text{Red}}/\rho_{\text{Blue}}$	[294]
$\text{DVI}_{\text{Green,Red}}$	green-red difference vegetation index	$\rho_{\text{Green}} - \rho_{\text{Red}}$	[263]
$\text{DVI}_{r,g}$	normalized red-green difference vegetation index	$r - g$	[276]
$\text{DVI}_{g,b}$	normalized green-blue difference vegetation index	$g - b$	[276]
VIg	vegetation index green	$(\rho_{\text{Green}} - \rho_{\text{Red}}) / (\rho_{\text{Green}} + \rho_{\text{Red}})$	[294]
TVIg	Tucker index	$(\rho_{\text{Green}} + \rho_{\text{Red}}) / (\rho_{\text{Green}} - \rho_{\text{Red}})$	[263]
NGRDI	normalized green red difference index	$(g - r) / (g + r)$	[227]
IKAW1	Kawashima index 1	$(\rho_{\text{Red}} - \rho_{\text{Blue}}) / (\rho_{\text{Red}} + \rho_{\text{Blue}})$	[294, 123]
IKAW2	Kawashima index 2	$(\rho_{\text{Green}} - \rho_{\text{Blue}}) / (\rho_{\text{Green}} + \rho_{\text{Blue}})$	[123]
IKAW3	Kawashima index 3	$(\rho_{\text{Red}} - \rho_{\text{Green}}) / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[123]
IKAW4	Kawashima index 4	$(\rho_{\text{Red}} - \rho_{\text{Blue}}) / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[123]
IKAW5	Kawashima index 5	$(\rho_{\text{Green}} - \rho_{\text{Blue}}) / (\rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})$	[123]
CIVE[1]	color index of vegetation extraction	$0.441 \cdot R - 0.811 \cdot G + 0.385 \cdot B + 18.78745$	[294]
CIVEn	color index of vegetation extraction, where the normalized bands are used	$0.441 \cdot r - 0.811 \cdot g + 0.385 \cdot b + 18.78745$	[166]
ExR	excess red vegetation index	$1.4 \cdot r - g$	[294]
ExG	excess green vegetation index	$2 \cdot g - r - b$	[294]
ExB	excess blue vegetation index	$1.4 \cdot b - g$	[294]
ExGR	excess green minus excess red	ExG - ExR	[227]
I_{PCA}	principal component analysis index	$0.994 \cdot \lvert\rho_{\text{Red}} - \rho_{\text{Blue}}\rvert + 0.961 \cdot \lvert\rho_{\text{Green}} - \rho_{\text{Blue}}\rvert + 0.914 \cdot \lvert\rho_{\text{Green}} - \rho_{\text{Red}}\rvert$	[294]
GLI	green leaf index	$\frac{[(\rho_{\text{Green}} - \rho_{\text{Red}}) + (\rho_{\text{Green}} - \rho_{\text{Blue}})]}{(\rho_{\text{Green}} + \rho_{\text{Red}} + \rho_{\text{Green}} + \rho_{\text{Blue}})}$	[294]
VARI	visible atmospherically resistant index	$(\rho_{\text{Green}} - \rho_{\text{Red}}) / (\rho_{\text{Green}} + \rho_{\text{Red}} - \rho_{\text{Blue}})$	[294]

[1] R, G, and B are the digital pixel values of the red, green, and blue bands, respectively, ranging between 0 and 255.

Table A.3: Vegetation indices, derived using the red-edge band, involved in this thesis, where ρ_i = is the reflectance of the spectral band i.

Vegetation Index	**Description**	**Equation**	**Reference**
$SRI_{RedEdge,Red}$	red-edge-red simple ratio index	$\rho_{RedEdge}/\rho_{Red}$	This thesis
$DVI_{RedEdge,Red}$	red-edge-red difference vegetation index	$\rho_{RedEdge} - \rho_{Red}$	This thesis
$SRI_{NIR,RedEdge}$	NIR-red-edge simple ratio index	$\rho_{NIR}/\rho_{RedEdge}$	This thesis
$DVI_{NIR,RedEdge}$	NIR-red-edge difference vegetation index	$\rho_{NIR} - \rho_{RedEdge}$	This thesis
$NDVI_{RedEdge}$[1]	NDVI with red band replaced by red-edge	$(\rho_{NIR} - \rho_{RedEdge}) / (\rho_{NIR} + \rho_{RedEdge})$	[281]
$MSR_{RedEdge}$	MSR with red band replaced by red-edge	$((\rho_{NIR}/\rho_{RedEdge}) - 1) / \left(\sqrt{\rho_{NIR}/\rho_{RedEdge} + 1}\right)$	[281]
TCARI	transformed chlorophyll absorption in reflectance index	$3\left((\rho_{RedEdge} - \rho_{Red}) - 0.2\,(\rho_{RedEdge} - \rho_{Green})\left(\frac{\rho_{RedEdge}}{\rho_{Red}}\right)\right)$	[296]
MCARI	modified chlorophyll absorption ratio index	$((\rho_{RedEdge} - \rho_{Red}) - 0.2\,(\rho_{RedEdge} - \rho_{Green}))\left(\frac{\rho_{RedEdge}}{\rho_{Red}}\right)$	[95]
TCI	triangular chlorophyll index	$1.2\,(\rho_{RedEdge} - \rho_{Green}) - 1.5\,(\rho_{Red} - \rho_{Green})\sqrt{\rho_{RedEdge}/\rho_{Red}}$	[95]

[1] This VI is also referred to as normalized difference red-edge (NDRE) in the literature.

Appendix B

Comparison of Related Work

In this section, a detailed and qualitative comparison of most of the related work discussed in Section 2.9 can be found in the form of categorization criteria tables. These comparison tables also include studies described in this thesis. Tables B.1, B.2, and B.3, compare the related work discussed in Sections 2.9.2, 2.9.3, and 2.9.4, respectively.

Table B.1: Comparison of the related work, summarized in Section 2.9.2, that applied federated learning (FL) to agriculture settings, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, IMG = imagery, LL = local learning, CL = centralized learning, EL = distributed ensemble learning, and N/A = not applicable.

Reference	Number of Clients	FL Data Partitioning	FL Client Device Class	FL Architecture	IID Dataset	non-IID Dataset	LL	CL	EL	Feature Type
[4]	10	horizontal	cross-device	standard	Y	Y	N	Y	N	IMG
[5]	4	horizontal	cross-silo	standard	Y	Y	N	Y	N	IMG
[6]	2	horizontal	cross-device	standard	Y	Y	N	Y	N	IMG
[8, 9]	3	horizontal	cross-device	standard	N	Y	N	Y	N	IoT network status
[14]	U	horizontal	cross-silo	standard	U	U	N	N	N	IMG
[26]	N/A	horizontal	cross-device	hierarchical	N/A	N/A	N	Y	N	SOI, WEA
[27]	U	horizontal	cross-device	hierarchical	U	U	N	Y	N	SOI, WEA
[39]	18	horizontal	cross-device	hierarchical	U	U	N	N	N	animal physiology
[51]	N/A	U	cross-device	hierarchical	N	Y	N	Y	N	SOI, MNG, WEA
[56]	6	horizontal	cross-silo	standard	N	Y	Y	Y	N	IMG
[59]	N/A	horizontal	cross-device	hierarchical	N	Y	N	N	N	IMG, SOI, WEA
[63]	4, 9, 11	horizontal	cross-silo	standard	N	Y	Y	Y	Y	IMG, SOI, WEA, LOC, TIM, MNG
[64]	3	horizontal	cross-silo	standard	U	U	N	Y	N	IMG
[74]	5, 10, 15	U	cross-device	hierarchical	Y	Y	N	Y	N	IoT network traffic
[98]	2	horizontal	cross-silo	standard	N	Y	P	Y	N	IMG
[106]	1 to 5	horizontal	cross-device	standard	Y	Y	N	Y	N	IMG
[108]	U	horizontal	cross-device	hierarchical	U	U	N	Y	N	SOI, WEA
[110]	U	horizontal	cross-silo	standard	Y	N	N	N	N	IMG

Table B.1: *Cont.*

Reference	Number of Clients	FL Data Partitioning	FL Client Device Class	FL Architecture	IID Dataset	non-IID Dataset	LL	CL	EL	Feature Type
[125]	4	horizontal	cross-device	standard	U	U	N	U	N	IMG
[142]	2	U	cross-device	hierarchical	U	U	N	Y	N	IoT network traffic
[147]	9	horizontal	cross-silo	standard	N	Y	Y	Y	N	SOI, WEA, MNG
[153]	2	horizontal	cross-silo	P2P	N	Y	N	Y	N	IMG
[162]	3, 5, 7	horizontal	cross-silo	standard	U	U	N	N	N	IMG
[164]	9	horizontal	cross-silo	standard	N	Y	N	Y	N	SOI, WEA, MNG
[165]	5, 10, 15, 20, 25, 30	horizontal	cross-device	standard	N	Y	N	Y	N	accelerometer, gyroscope
[181]	U	horizontal	cross-silo	U	U	U	N	N	U	IMG
[195]	3	horizontal	cross-device	hierarchical	N	Y	Y	Y	N	HS IMG
[201]	3	horizontal	cross-device	standard	N	Y	Y	Y	N	HS IMG
[211]	U	horizontal	cross-device	standard	U	U	U	U	N	SOI
[207]	U	transfer	cross-device	hierarchical	U	U	U	U	U	IoT network traffic
[235]	N/A	horizontal	cross-silo	standard	N/A	N/A	N	N	N	IMG
[237]	9, 10, 15, 20	horizontal	cross-silo	standard	Y	N	N	Y	N	IMG
[243]	2,5,6	horizontal	cross-silo	standard	U	Y	Y	N	N	WEA, CO_2
[253]	5	horizontal	cross-device	P2P	U	U	N	N	N	IMG
[259]	3, 5, 7	horizontal	cross-silo	standard	U	U	N	N	N	IMG
[260]	6	horizontal	cross-silo	standard	Y	N	U	U	N	IMG
[262]	U	horizontal	cross-silo	standard	U	U	U	U	N	IMG
[271]	5, 7	horizontal	cross-silo	standard	Y	Y	N	Y	N	HS IMG
[279]	10, 11, 36, 72, 84	horizontal	cross-silo	standard	N	Y	Y	N	N	IMG, HS IMG
[287]	N/A	vertical, horizontal	cross-device	standard	N/A	N/A	N	N	N	IMG, SOI ,WEA
[290]	U	horizontal	cross-silo	standard	Y	Y	N	Y	N	IMG
[292]	10	vertical, horizontal	cross-device	standard	N	Y	N	Y	N	IMG
[298]	up to 248	vertical, horizontal	cross-silo	standard	N	Y	Y	Y	N	PHE, WEA
[199]	10, 20	horizontal	cross-device	hierarchical	U	U	N	N	Y	SOI, WEA
Sec 4.6	6	horizontal	cross-silo	hierarchical	Y	Y	Y	Y	Y	IMG
Sec 5.5	6	horizontal	cross-device	hierarchical	Y	Y	Y	Y	Y	SOI, WEA

Table B.2: Comparison of the yield prediction related work summarized in Section 2.9.3, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, IMG = imagery, GEN = genotype information, RGB = red-green-blue, MS = multispectral, HS = hyperspectral, VI = vegetation index, TI = texture index, and ∼ indicates an approximate value.

Reference	Imagery Type	Feature Types	Number of VIs	Number of TIs	Number of Raw Bands	Number of Raw Yield Samples	Study Period	Prediction Scale	Includes Temporal Features/ Structure	Includes Spatial Features/ Structure	Spatial Generalizability Evaluated	Temporal Generalizability Evaluated
[141]	RGB, MS	IMG	26	0	7	64	2021	field-level	N	N	N	N
[3]	RGB	IMG	15	0	3	U	2019	field-level	Y	N	N	N
[21]	MS	IMG	26	0	5	32	2017–2019	field-level	N	N	N	N
[232]	MS	IMG	10	0	4	48	2021	field-level	N	N	N	N
[61]	HS, LiDAR	IMG	U	U	269	369	2020	field-level	N	P	N	N
[191]	MS	IMG	3	0	10	U	2020–2021	field-level	N	U	N	N
[215]	MS	IMG	33	0	4	88	2017–2019	field-level	N	N	N	N
[282]	MS	IMG	12	8	5	57	2020	field-level	Y	P	N	N
[103]	MS	IMG	14	0	11	151	2015	field-level	N	N	N	N
[93]	RGB, MS	IMG	35	4	8	20	2019	field-level	N	P	N	N
[38]	RGB	IMG	12	0	3	U	2019	field-level	Y	P	N	N
[67]	HS	IMG	81	0	274	1429	2020	field-level	Y	N	N	N
[92]	RGB	IMG	8	0	3	20	2019	field-level	Y	N	N	N
[231]	RGB, MS	IMG, BIO, LAI	2	0	7	5	2021	field-level	N	N	N	N
[236]	MS	IMG, TOP	4	4	U	673	1998	pixel-level	N	P	N	N
[266]	HS	IMG	4	0	71	192	2000	pixel-level	N	N	N	N
[126]	MS, LiDAR	IMG, TOP, SOI	6	0	4	U	2013	pixel-level	N	N	N	N
[17]	MS	IMG, LOC, WEA, MNG	5	0	4	U	2020–2021	pixel-level	Y	Y	Y	Y
[127]	RGB, MS, LiDAR	IMG, TOP	6	0	7	U	2016–2018	pixel-level	Y	N	N	N
[132]	RGB, MS	IMG	33	0	8	18,106	2021	pixel-level	N	P	N	N
[286]	HS	IMG	0	0	240	172	2015	pixel-level	N	Y	N	N
[251]	RGB, MS	IMG	6	0	6	54–288	2019	pixel-level	Y	N	N	N
[69]	RGB	IMG	6	0	3	59	2018	pixel-level	N	N	N	N
[76]	RGB, MS	IMG	6	0	7	80	2018	pixel-level	N	P	N	N
[272]	MS	IMG, MNG	15	0	5	U	2020	pixel-level	Y	P	N	N
[297]	RGB	IMG	1	0	3	14,705	2016	pixel-level	N	N	N	N
[53]	MS	IMG, MNG, GEN	8	0	5	∼4500	2017–2019	pixel-level	N	Y	U	N
[18]	RGB, MS, thermal	IMG, MNG	U	0	7	72	2020–2021	pixel-level	N	N	N	N
[29]	RGB	IMG	0	0	3	U	2021	pixel-level	Y	Y	Y	N
Sec 4.4	RGB, MS	IMG	55	0	8	18,106	2021	pixel-level	N	P	Y	N
Sec 4.5	MS	IMG	2	0	5	13,264	2021	pixel-level	N	P	Y	N
Sec 4.6	MS	IMG	10	0	5	31,246	2020–2021	pixel-level	Y	Y	Y	P

Table B.3: Comparison of the N_2O emission prediction related work summarized in Section 2.9.4, where Sec = Section, Y = yes, N = no, P = partial, U = unclear, LOC = location, MNG = management, PHE = phenology, SOI = soil, TIM = time, TOP = topography, WEA = weather, and ∼ indicates an approximate value.

Reference	Included Process-based Models	N_2O Temporal Resolution	Features Types	Number of Features	Study Duration	Number of Study Sites	Number of N_2O Samples	Study Prediction Scale	Spatial Generalizability Evaluated	Temporal Generalizability Evaluated	Was Gap-filling Applied
[228]	Y	4/day and 1–2/week	SOI, MNG, WEA	9	2012–2014, 2016–2017	2	∼4100	field-level	N	N	N
[96]	U	a few samples per month	SOI, WEA	9	2012–2015, 2017	1	126	field-level	N	Y	N
[204]	N	U	SOI, MNG, WEA	19	U	multi	594	region-level	N	N	N
[83]	N	sub-weekly to seasonally	SOI, MNG, WEA, TOP	24	2013–2015	1	∼600	field-level	N	N	N
[289]	Y	multi/year	SOI, WEA	> 8	U	multi	1353	global-level	N	N	N
[255]	Y	U	SOI, WEA, MNG	>15	1980–2021	256	1767	global-level	Y	N	N
[171]	N	2/week	SOI, WEA	7	2017–2018	1	25	field-level	N	N	N
[22]	N	2–3/week	SOI, WEA, MNG	∼8	2013–2014	1	23	field-level	N	N	N
[139]	U	U	MNG	<17	1990–2010	multi	U	region-level	N	N	N
[198]	Y	U	SOI, WEA, MNG	12	<2018	multi	186	global-level	N	N	N
[270]	Y	U	SOI, WEA, MNG	11	U	multi	∼19,000	region-level	N	N	N
[302]	N	varies	SOI, WEA	∼5	U	64	209	global-level	N	N	N
[116]	N	6/day	SOI, WEA, MNG	5	2019–2020	1	U	field-level	N	N	N
[24]	N	U	MNG	U	2018	multi	U	global-level	N	N	N
[115]	N	∼daily	SOI, WEA, MNG	>11	2009–2011	multi	U	multi	Y	P	N
[155]	Y	simulated daily, collected hourly	SOI, WEA, MNG, PHE	92	2015–2018	1	$> 10^6$	region-level	N	P	N
[114]	N	U	SOI, MNG	3	U	multi	<500	global-level	U	U	N
[81]	Y	daily	SOI, WEA, MNG	>15	-	multi	U	region-level	P	N	N
[280]	N	U	SOI, WEA, MNG	21	U	multi	679	region-level	N	N	N
[160]	N	30 min	SOI, WEA, PHE	68	2019–2020	1	U	field-level	N	Y	Y
[70]	N	30 min	SOI, WEA, MNG, PHE	78	2019–2020	1	U	field-level	N	N	Y
[252]	N	hourly	SOI, WEA, TIM	9	2001–2006	1	1525	field-level	N	P	Y
[28]	N	4/day	SOI, WEA, MNG, PHE	25	2012–2016	1	U	field-level	N	N	Y
[86]	N	30 min	SOI, WEA, MNG	< 1081	2017–2019	1	U	field-level	N	Y	Y
[48]	N	30 min	SOI, WEA, MNG	5	2012,2016–2018	1	U	field-level	N	N	Y
[150]	Y	30 min	SOI, WEA, MNG	6	2021–2022, 2024	1	U	field-level	U	Y	N
Sec 5.4	N	30 min	SOI, WEA, TIM, MNG	< 422	2021–2023	1	≤ 4653	field-level	N	Y	Y
Sec 5.5	N	30 min	SOI, WEA	≤ 419	2021–2023	1	< 4653	field-level	N	Y	Y

Appendix C

Model Hyperparameters

In this section we explain the hyperparameter choices and hyperparameter search spaces (those used by random search) for most of the models involved in the studies described in this thesis. The ranges of the hyperparameter search spaces used by random search (and the neural network architecture for deep learning models) are shown in Tables C.1, C.2 and C.3, C.4, C.5, C.6, C.7, and C.8 for RF, two versions of MLP, 1D CNN, 2D rCNN, 2D sCNN, LSTM, and SVM, respectively; where brackets define the search space used for choosing values uniformly at random (with replacement), such that a) the curly brackets define the set of possible values that can be chosen and b) the square brackets define the lower-bounds and upper-bounds (with a given step) of the search space. For example, [16,64] with a step of 16 is the same as {16,32,48,64}.

In terms of the deep learning models, the Adam optimizer with MSE as the loss function was used. For all the deep learning models except the one described in Table C.3, the early stopping patience, number of epochs, batch size, and learning rate were tuned using the search space ranges found in Table C.9.

Table C.1: Hyperparameter search space ranges for the random forest (RF) model.

Hyperparameter	**Search Range**	**Step**
Number of trees	[50,500]	25
Maximum depth	[8,24]	1
Number of candidate attributes	[1,*number of features* -1]	1
Perform bootstrap sampling	{True,False}	-
Bootstrap sampling size	[0.65,0.95]	0.05
Minimum samples in leaf	[1,15]	1

Table C.2: Hyperparameter search space ranges and neural network structure for the multilayer perceptron (MLP) model used by all studies in this thesis (except the study described in Section 4.5), where i = the number of hidden layers, FC_i = fully connected dense layer i, and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Number of hidden layers i	{1,2,3,4,5,6}	-
N/A	Activation function a	{ReLu, tanh, Sigmoid}	-
FC_1	Layer size	[8,256]	1
	Activation function	$\{a\}$	-
FC_2	Layer size	[8,256]	1
	Activation function	$\{a\}$	-
...			
FC_i	Layer size	[8,256]	1
	Activation function	$\{a\}$	-
F_{i+1}	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.3: Hyperparameter search space ranges and neural network structure for the multilayer perceptron (MLP) model used by only the study described in Section 4.5, where i = the number of hidden layers, FC_i = fully connected dense layer i, and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Number of hidden layers i	{1,2,3,4,5,6}	-
N/A	Early stopping patience	{5}	-
N/A	Epochs	{500}	-
N/A	Batch size	{4096}	-
N/A	Learning rate	$\{10^{-3},10^{-4},10^{-5}\}$	-
FC_1	Layer size	[32,1024]	32
	Activation function	{ReLu}	-
FC_2	Layer size	[32,1024]	32
	Activation function	{ReLu}	-
...			
FC_i	Layer size	[32,1024]	32
	Activation function	{ReLu}	-
F_{i+1}	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.4: Hyperparameter search space ranges and neural network structure for the 1D convolutional neural network (CNN) model, where FC_i = fully connected dense layer i and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Activation function a	{ReLu, tanh, Sigmoid}	-
Conv1D	Number of filters/feature maps	[8,256]	1
	Filter/kernel size	[2,*input tensor length*]	1
	Stride	{1}	-
	Activation Function	{ReLu}	-
MaxPooling1D	Pool size	{2}	-
	Stride	{2}	-
Flatten	-	-	-
FC_1	Layer size	[8,256]	1
	Activation function	{a}	-
FC_2	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.5: Hyperparameter search space ranges and neural network structure for the 2D spectral reflectance convolutional neural network (rCNN) model, where FC_i = fully connected dense layer i and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Activation function a	{ReLu, tanh, Sigmoid}	-
Conv2D	Number of filters/feature maps	[8,256]	1
	Filter/kernel size	{1}	-
	Stride	{1}	-
	Activation Function	{ReLu}	-
AveragePooling2D	Pool size p	[2,*input tile size*]	1
	Stride	{p}	-
Flatten	-	-	-
FC_1	Layer size	[8,256]	1
	Activation function	{a}	-
FC_2	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.6: Hyperparameter search space ranges and neural network structure for the 2D spatial convolutional neural network (sCNN) model, where FC_i = fully connected dense layer i and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Activation function a	{ReLu, tanh, Sigmoid}	-
Conv2D	Number of filters/feature maps	[8,256]	1
	Filter/kernel size	[2,*input tile size*]	1
	Stride	{1}	-
	Activation Function	{ReLu}	-
MaxPooling2D	Pool size	{2}	-
	Stride	{2}	-
Flatten	-	-	-
FC_1	Layer size	[8,256]	1
	Activation function	{a}	-
FC_2	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.7: Hyperparameter search space ranges and neural network structure for the long short-term memory (LSTM) model, where FC_i = fully connected dense layer i, FC_1 only exists half the time (depending on the truth value of the hyperparameter f), and N/A = not applicable.

Layer	**Hyperparameter**	**Search Range**	**Step**
N/A	Activation function a	{ReLu, tanh, Sigmoid}	-
N/A	FC_1 conditional existence flag f	{True,False}	-
LSTM	Layer size	[3,50]	1
FC_1	Layer size	[8,256]	1
	Activation function	{a}	-
FC_2	Layer size	{1}	-
	Activation function	{Linear}	-

Table C.8: Hyperparameter search space ranges for the support vector machine (SVM) model, where loguniform = log uniform random sampling.

Hyperparameter	**Search Range**	**Step**
C	$[10^{-1},10^{3}]$	loguniform
gamma	$[10^{-3},10^{3}]$	loguniform
Kernel	{Linear, Polynomial, Radial Basis Function, Sigmoid}	-

Table C.9: Hyperparameter search space ranges common to the deep learning models (excluding the multilayer perceptron model detailed in Table C.3) , where loguniform = log uniform random sampling.

Hyperparameter	**Search Range**	**Step**
Early stopping patience	[3,20]	1
Epochs	[400,700]	50
Batch size (as a fraction of the training dataset size)	[0.15,0.9]	0.01
Learning rate	$[10^{-5},10^{-1}]$	loguniform

Appendix D

Supplementary Material

Supplementary material is found in this section. Figure D.1 shows a visual example of a NDVI and RGB orthomosaic. Table D.1 provides the mean size of the orthomosaic UAV imagery files used in our yield prediction experiments. Table D.2 shows the raw, cleaned, and interpolated yield datasets' statistical properties. Figure D.2 shows the selected features (and their importance) of our yield prediction study described in Section 4.6. Figure D.3 shows daily average air temperature for the 2021, 2022, and 2023 growing seasons. Figure D.4 shows daily total precipitation (rainfall) for the 2021, 2022, and 2023 growing seasons. The remaining tables and figures provide additional information about the N_2O studies described in Chapter 5. Figure D.5 illustrates a map of the field and the sensor deployment locations. Table D.3 presents summary statistics of the sensors' raw datasets. Table D.4 contains detailed information about sensors, including sampling rates, SN models, and soil and weather parameters monitored. Figure D.6 shows the complete diagram of all the components and wiring of the PN prototype.

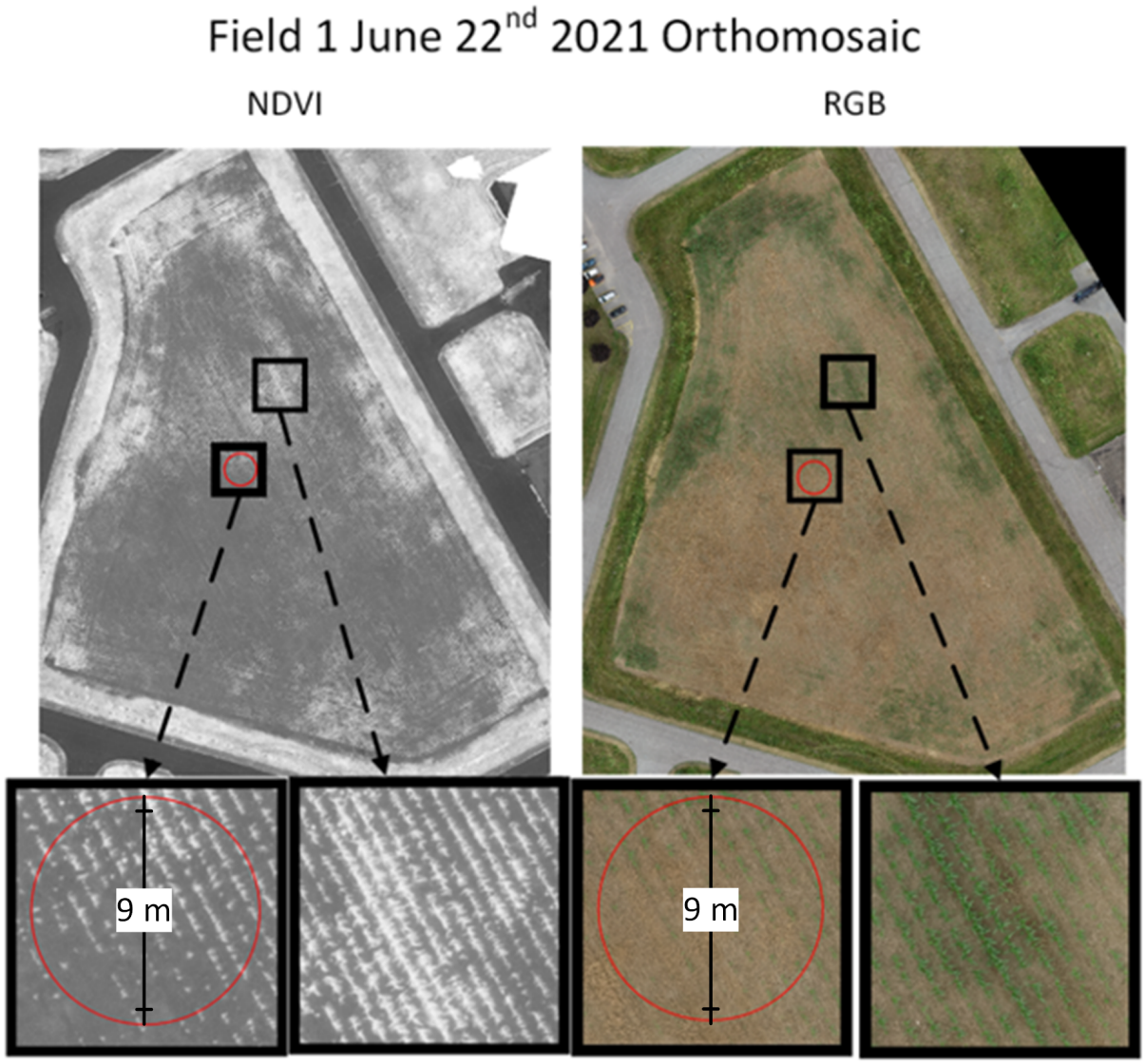


Figure D.1: Field 1 June 22, 2021, NDVI and RGB orthomosaics such that the size of a 9 m diameter (4.5 m radius) filter is visualized in the field, where NDVI = normalized difference vegetation index and RGB = red-green-blue.

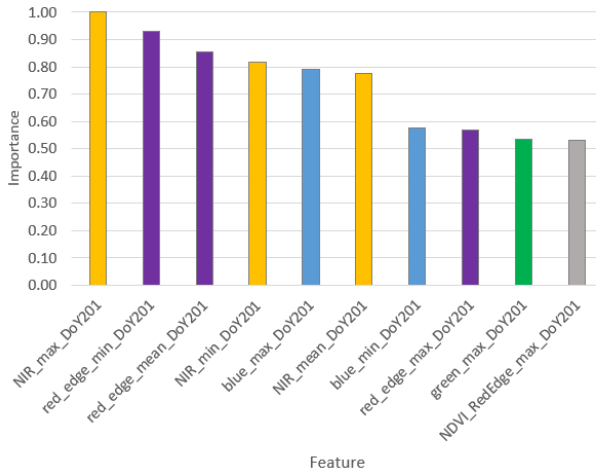
Table D.1: Mean (averaged over all the missions conducted in 2021) size of the orthomosaic UAV imagery files used in the yield prediction experiments, where RGB = red-green-blue and MS = multispectral.

Field	**Image Type**	**Orthomosaic Size (MB)**
1	RGB	806
2	RGB	793
3	RGB	677
4	RGB	473
9	RGB	410
11	RGB	1,582
11	MS	530
1, 2, 3, 4, and 9[1]	MS	691

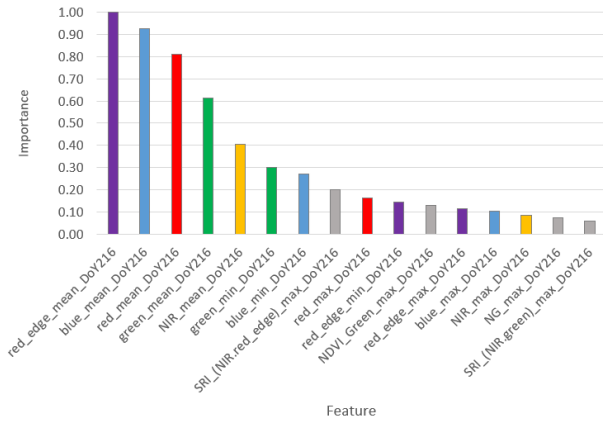
[1] Each MS image acquisition mission had a single orthomosaic containing the imagery of all the smaller fields.

Table D.2: Yield dataset yield (in bushels per acre) statistical properties, where Std Dev = standard deviation, Max = maximum, Min = minimum, Q1 = quartile 1, Q3 = quartile 3, IQR = interquartile range, and n = the number of samples.

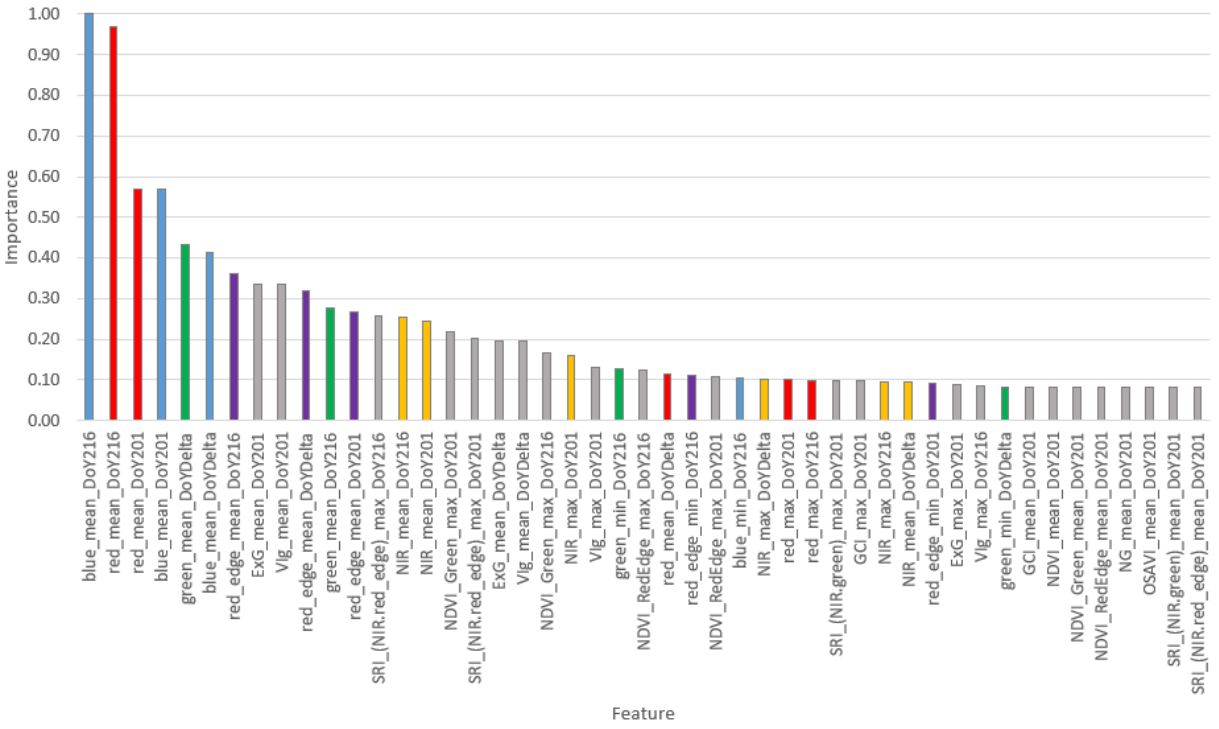
Yield Dataset	**Year**	**Field**	**Mean**	**Median**	**Std Dev**	**Max**	**Min**	**Q1**	**Q3**	**IQR**	**n**
Raw	2020	11	156	161	50	801	4	141	174	32	13,140
Raw	2021	1	113	106	66	812	5	84	128	43	1,038
Raw	2021	2	146	137	81	836	1	106	173	67	1,577
Raw	2021	3	113	107	62	677	4	73	147	75	840
Raw	2021	4	74	68	52	595	4	46	91	44	873
Raw	2021	9	119	111	69	592	20	88	130	42	514
Raw	2021	11	149	155	51	809	2	127	174	47	13,264
Cleaned	2020	11	158	163	28	308	24	148	175	27	9,545
Cleaned	2021	1	111	111	25	179	32	96	129	33	628
Cleaned	2021	2	145	141	37	252	56	116	173	56	1,175
Cleaned	2021	3	117	112	42	219	21	83	149	66	617
Cleaned	2021	4	77	74	25	148	26	58	94	36	584
Cleaned	2021	9	116	116	20	178	65	103	129	26	301
Cleaned	2021	11	153	159	35	274	17	138	177	39	9,194
Interpolated	2020	11	157	161	23	224	38	148	171	23	30,990
Interpolated	2021	1	108	108	17	150	63	97	120	23	2,016
Interpolated	2021	2	147	143	31	217	92	123	168	45	1,589
Interpolated	2021	3	122	119	40	200	24	88	157	68	1,086
Interpolated	2021	4	75	70	21	120	35	58	93	35	675
Interpolated	2021	9	117	116	15	172	87	107	125	18	610
Interpolated	2021	11	160	162	22	217	50	149	174	25	24,662



(a) mono-temporal-DoY201

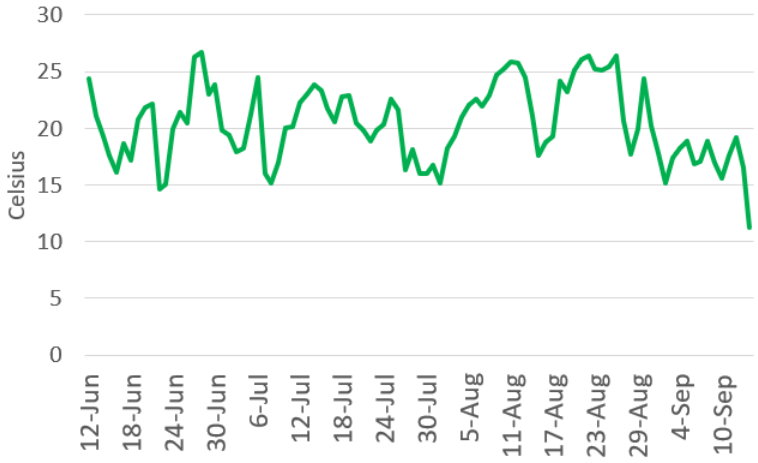


(b) mono-temporal-DoY216

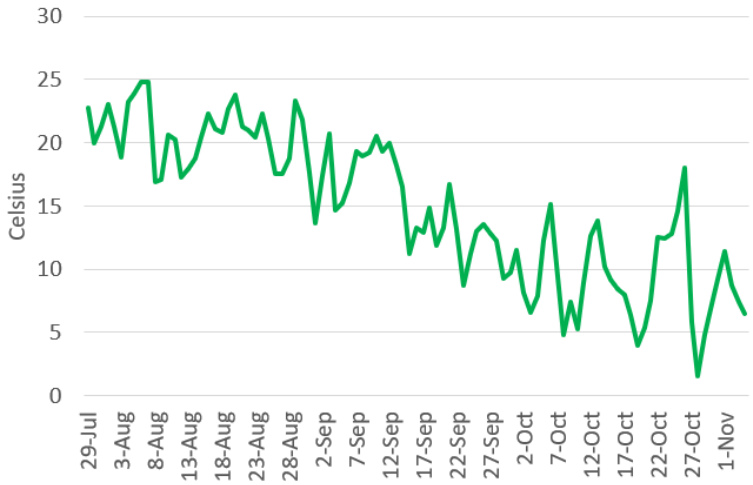


(c) multi-temporal

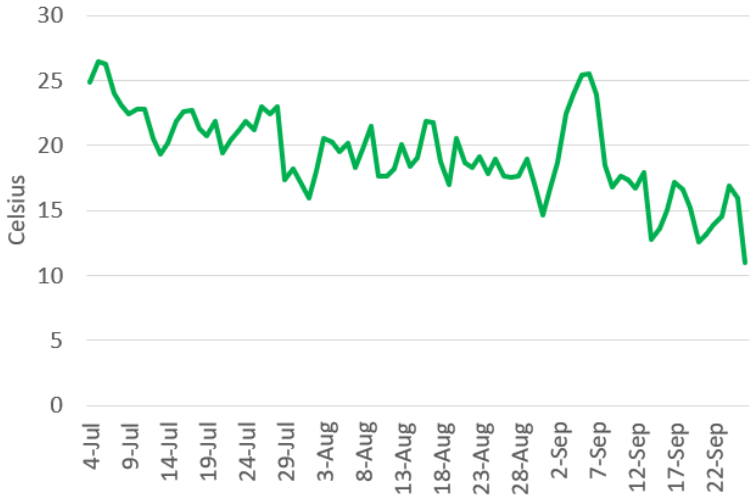
Figure D.2: 2021 UAV imagery-based yield prediction feature importance bar plots of the selected features in our study described in Section 4.6, where the importance is the normalized (via min-max scaling) absolute value of the PLSR coefficients of the features; the red, green, blue, orange, and purple bars are the red, green, blue, near-infrared (NIR), and red-edge spectral band features, respectively; the grey bars are for VI features; features have a '<imagery channel>_<aggregation filter>_<temporal feature type>' naming convention; and DoY = day of year.



(a) 2021

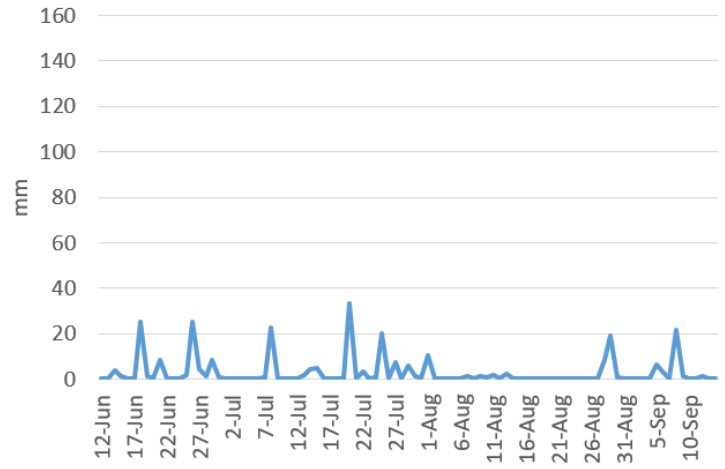


(b) 2022

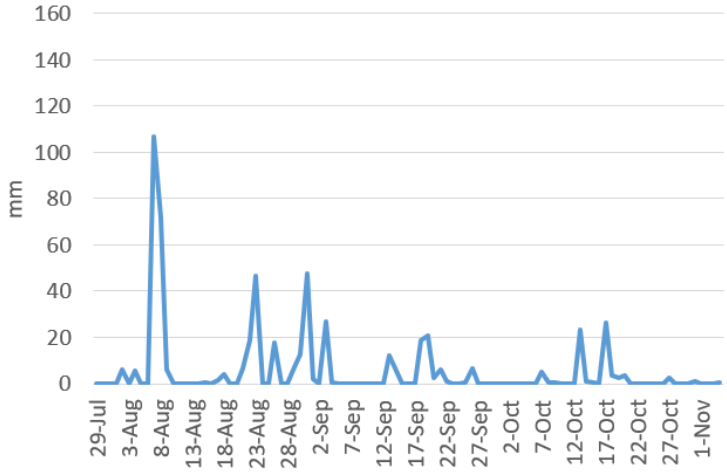


(c) 2023

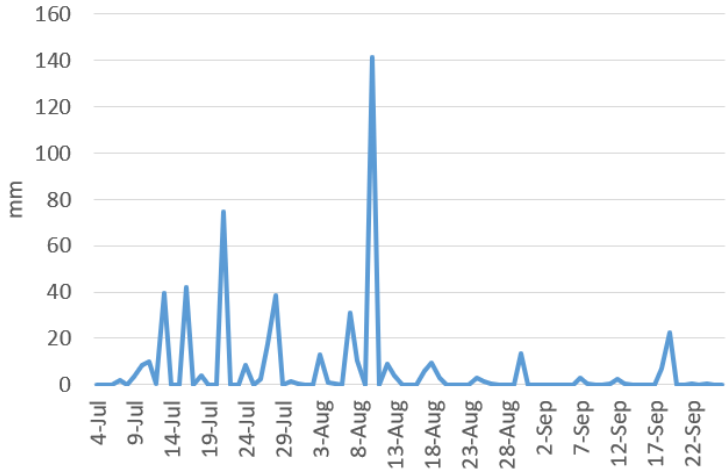
Figure D.3: Daily average air temperature time series plots for the 2021, 2022, and 2023 growing seasons.



(a) 2021



(b) 2022



(c) 2023

Figure D.4: Daily total precipitation time series plots for the 2021, 2022, and 2023 growing seasons.

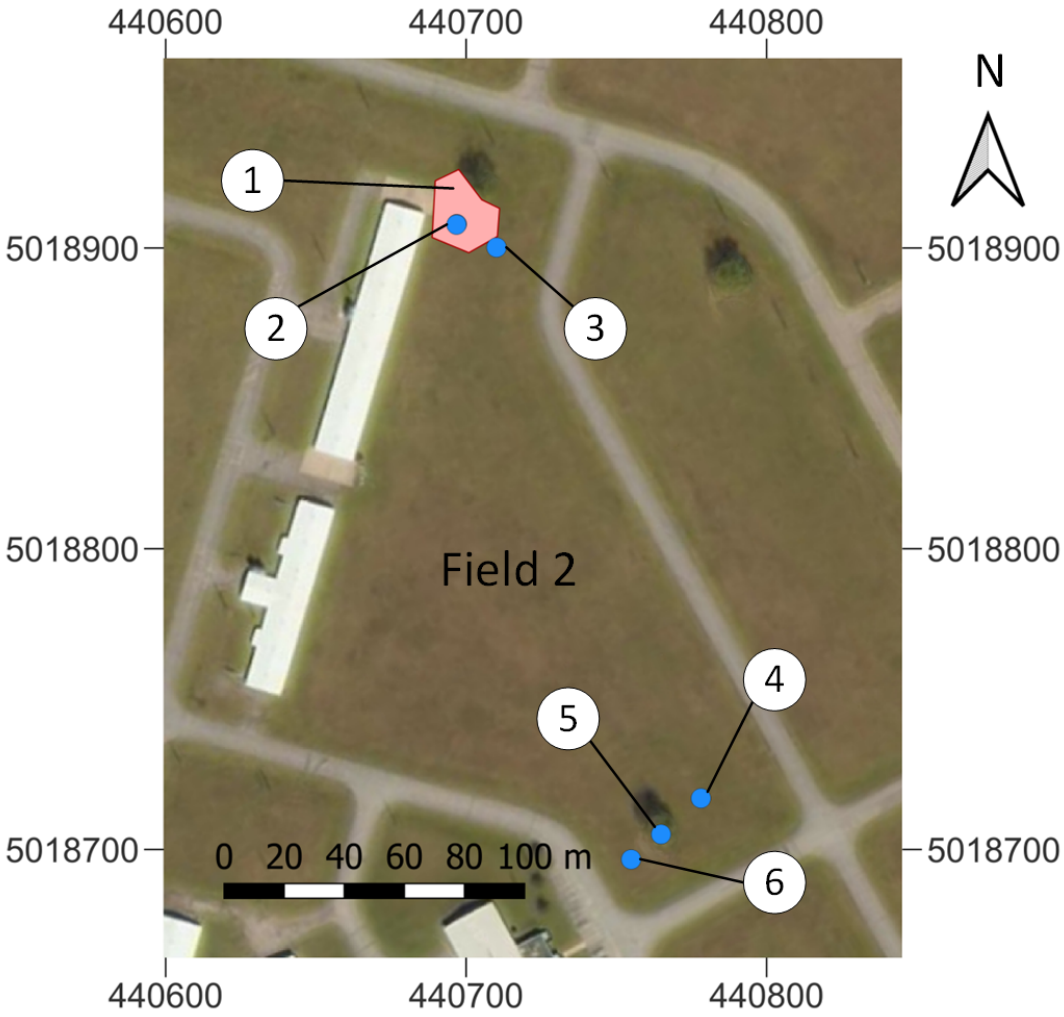


Figure D.5: N_2O prediction experiments: a sensor deployment map of the study area. The map illustrates the sensor node (SN) locations in Field 2 and includes a distance scale and coordinate grid that uses the coordinate reference system WGS 84 / UTM zone 18N. Sensors are identified on the map from 1 to 6, where 1 = the area where LI-COR gas chambers were deployed, 2 = the predictor node (PN), 3 = Pessl SN 01209E4F, 4 = Pessl SN 00209FC8, 5 = Pessl SN 01209E58, and 6 = Pessl SN 01209E52.

Table D.3: Summary statistics of the sensors' raw datasets (those used in the N_2O emission prediction experiments), where # nulls = number of missing readings, μ = mean, Med = median, Max = maximum, Min = minimum, σ = standard deviation, N/A = not applicable, and VPD = vapor pressure deficit; see Table D.4 for the units of the sensors readings.

									Autocorrelation at lag k (where k is in hours)[1]												
Sensor Node	**Sensor**	**Year**	**# nulls**	μ	**Med**	**Max**	**Min**	σ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
00209FC8	air_temperature	2021	0	20.47	20.40	33.53	5.67	4.99	0.97	0.90	0.80	0.69	0.56	0.43	0.31	0.19	0.09	0.00	-0.06	-0.08
00209FC8	dew_point	2021	0	15.51	15.50	24.30	-0.90	4.75	0.99	0.97	0.94	0.92	0.90	0.88	0.86	0.84	0.82	0.80	0.79	0.77
00209FC8	solar_radiation	2021	0	231.89	61.00	1284.00	0.00	322.90	0.90	0.73	0.54	0.32	0.10	-0.10	-0.26	-0.37	-0.43	-0.47	-0.49	-0.50
00209FC8	VPD	2021	0	0.66	0.48	2.80	0.00	0.64	0.95	0.85	0.71	0.55	0.38	0.21	0.04	-0.11	-0.24	-0.34	-0.41	-0.44
00209FC8	relative_humidity	2021	0	76.52	78.83	100.00	17.47	19.28	0.96	0.87	0.75	0.62	0.48	0.33	0.19	0.05	-0.06	-0.16	-0.23	-0.27
00209FC8	precipitation	2021	0	0.12	0.00	24.00	0.00	0.91	0.33	0.17	0.05	0.03	0.03	0.03	0.02	0.01	0.00	-0.01	0.00	0.00
00209FC8	leaf_wetness	2021	0	13.31	0.00	60.00	0.00	24.20	0.89	0.74	0.61	0.49	0.39	0.30	0.23	0.17	0.13	0.08	0.05	0.02
00209FC8	deltaT	2021	0	2.98	2.00	13.00	0.00	2.73	0.95	0.85	0.73	0.58	0.43	0.27	0.12	-0.03	-0.15	-0.25	-0.31	-0.35
01209E4F	precipitation	2021	0	0.11	0.00	22.80	0.00	0.84	0.32	0.17	0.05	0.04	0.03	0.03	0.02	0.01	0.00	-0.01	0.00	0.00
01209E4F	soil_moisture1	2021	2	24.00	23.80	39.10	12.60	4.99	0.99	0.98	0.97	0.96	0.95	0.94	0.94	0.93	0.92	0.91	0.91	0.90
01209E4F	soil_moisture2	2021	2	40.20	41.20	48.70	27.60	4.40	0.99	0.98	0.98	0.97	0.96	0.96	0.95	0.95	0.94	0.94	0.93	0.93
01209E4F	soil_moisture3	2021	2	48.25	48.50	52.40	40.40	1.99	0.98	0.97	0.96	0.95	0.95	0.94	0.94	0.93	0.92	0.91	0.91	0.90
01209E4F	soil_moisture4	2021	2	45.09	45.10	50.70	42.10	1.08	0.93	0.87	0.84	0.81	0.79	0.77	0.76	0.75	0.73	0.71	0.69	0.67
01209E4F	soil_moisture5	2021	2	40.49	40.70	46.70	39.30	0.71	0.94	0.89	0.85	0.81	0.77	0.75	0.71	0.68	0.63	0.59	0.55	0.53
01209E4F	soil_moisture6	2021	2	32.49	32.60	35.00	31.10	0.54	0.98	0.96	0.95	0.94	0.93	0.93	0.92	0.92	0.93	0.93	0.93	0.94
01209E4F	soil_temperature1	2021	2	20.04	19.90	24.40	15.00	1.82	0.99	0.97	0.95	0.91	0.87	0.83	0.79	0.76	0.73	0.70	0.68	0.67
01209E4F	soil_temperature2	2021	2	19.39	19.30	22.30	15.90	1.45	1.00	0.99	0.99	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.93	0.93
01209E4F	soil_temperature3	2021	2	19.25	19.30	21.80	16.30	1.34	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97
01209E4F	soil_temperature4	2021	2	18.82	18.90	21.10	16.00	1.31	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98
01209E4F	soil_temperature5	2021	2	18.51	18.70	20.60	15.60	1.32	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
01209E4F	soil_temperature6	2021	2	17.94	18.20	20.00	15.00	1.34	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
01209E52	air_temperature	2021	1117	20.90	20.98	33.54	9.07	4.92	0.97	0.90	0.81	0.71	0.60	0.49	0.38	0.28	0.19	0.12	0.07	0.06
01209E52	dew_point	2021	1117	15.15	14.80	24.00	-0.50	5.18	0.97	0.94	0.91	0.88	0.85	0.82	0.79	0.76	0.74	0.72	0.70	0.68
01209E52	VPD	2021	1117	0.75	0.56	2.90	0.00	0.71	0.94	0.84	0.70	0.54	0.38	0.21	0.06	-0.08	-0.19	-0.27	-0.32	-0.35
01209E52	relative_humidity	2021	1117	73.47	75.86	100.00	17.37	20.65	0.94	0.85	0.73	0.60	0.47	0.33	0.20	0.08	-0.03	-0.11	-0.17	-0.20
01209E52	precipitation	2021	1117	0.00	0.00	0.00	0.00	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
01209E52	deltaT	2021	1117	3.42	3.00	14.00	0.00	2.99	0.93	0.84	0.71	0.57	0.42	0.27	0.12	0.00	-0.11	-0.19	-0.24	-0.27
01209E58	precipitation	2021	0	0.12	0.00	22.60	0.00	0.84	0.29	0.17	0.05	0.04	0.03	0.03	0.02	0.01	0.00	0.00	0.01	0.01
01209E58	soil_moisture1	2021	2	13.73	13.60	16.80	13.40	0.42	0.90	0.76	0.63	0.53	0.46	0.40	0.34	0.28	0.23	0.20	0.16	0.14
01209E58	soil_moisture2	2021	2	2.12	2.00	4.50	1.70	0.35	0.88	0.73	0.58	0.48	0.41	0.33	0.26	0.20	0.16	0.12	0.09	0.07
01209E58	soil_moisture3	2021	2	0.00	0.00	0.30	0.00	0.01	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01209E58	soil_moisture4	2021	2	0.00	0.00	0.50	0.00	0.02	0.59	0.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01209E58	soil_moisture5	2021	2	0.00	0.00	0.30	0.00	0.01	0.34	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01209E58	soil_moisture6	2021	2	0.00	0.00	0.70	0.00	0.02	0.29	0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01209E58	soil_temperature1	2021	2	22.64	22.30	38.30	6.40	5.93	0.95	0.86	0.73	0.60	0.45	0.30	0.16	0.03	-0.09	-0.19	-0.26	-0.29
01209E58	soil_temperature2	2021	2	23.48	22.80	43.10	6.00	6.91	0.94	0.82	0.67	0.51	0.36	0.21	0.07	-0.06	-0.18	-0.28	-0.36	-0.38
01209E58	soil_temperature3	2021	2	23.64	22.90	44.00	5.60	7.19	0.94	0.82	0.67	0.51	0.34	0.19	0.04	-0.10	-0.22	-0.32	-0.39	-0.42
01209E58	soil_temperature4	2021	2	23.53	22.70	42.30	5.60	7.09	0.94	0.82	0.67	0.51	0.34	0.18	0.03	-0.11	-0.23	-0.33	-0.40	-0.43
01209E58	soil_temperature5	2021	2	23.20	22.30	42.30	5.30	6.98	0.95	0.84	0.69	0.53	0.36	0.19	0.04	-0.11	-0.23	-0.33	-0.39	-0.41
01209E58	soil_temperature6	2021	2	23.11	22.30	42.00	5.30	6.74	0.95	0.85	0.71	0.55	0.38	0.21	0.06	-0.09	-0.21	-0.30	-0.37	-0.39
LI-COR	N_2O from $C_{2021,4}$	2021	1	13.50	8.12	97.77	0.00	15.14	0.99	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.93	0.92	0.92	0.91
LI-COR	N_2O from $C_{2022,1}$	2022	1	34.23	7.30	255.44	0.10	49.55	0.99	0.98	0.97	0.96	0.95	0.94	0.92	0.91	0.90	0.89	0.88	0.88
LI-COR	N_2O from $C_{2022,2}$	2022	2	22.36	10.88	131.53	0.01	23.68	0.98	0.96	0.94	0.91	0.89	0.87	0.84	0.83	0.81	0.80	0.79	0.78
LI-COR	N_2O from $C_{2022,3}$	2022	1	7.39	1.12	98.18	0.00	12.61	0.98	0.97	0.95	0.93	0.91	0.89	0.87	0.86	0.85	0.84	0.83	0.83
LI-COR	N_2O from $C_{2023,1}$	2023	15	0.26	0.14	6.67	0.00	0.45	0.91	0.85	0.80	0.77	0.76	0.73	0.72	0.70	0.69	0.68	0.68	0.67
LI-COR	N_2O from $C_{2023,3}$	2023	11	5.17	0.21	171.16	0.01	14.75	0.96	0.92	0.89	0.87	0.85	0.83	0.81	0.79	0.78	0.76	0.75	0.74
LI-COR	N_2O from $C_{2023,4}$	2023	90	6.83	0.19	132.69	0.00	16.08	0.98	0.95	0.93	0.91	0.89	0.88	0.87	0.86	0.85	0.84	0.83	0.83

[1] The autocorrelation at lag k is the correlation of values within k time units of each other.

Table D.3: *Cont.*

									Autocorrelation at lag k (where k is in hours)[1]											
Sensor Node	**Sensor**	**Year**	**# nulls**	μ	**Med.**	**Max**	**Min**	σ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
00209FC8	air_temperature	2022	178	14.90	15.35	32.04	-5.69	7.11	0.98	0.94	0.89	0.82	0.75	0.68	0.61	0.55	0.49	0.45	0.42	0.41
00209FC8	dew_point	2022	178	11.13	11.90	26.00	-6.00	6.60	0.99	0.98	0.96	0.95	0.94	0.92	0.91	0.90	0.89	0.88	0.87	0.86
00209FC8	solar_radiation	2022	178	188.92	0.00	1485.00	0.00	298.98	0.89	0.72	0.52	0.30	0.09	-0.09	-0.22	-0.30	-0.36	-0.39	-0.41	-0.42
00209FC8	VPD	2022	178	0.40	0.21	2.49	0.00	0.49	0.95	0.84	0.70	0.53	0.36	0.19	0.04	-0.08	-0.18	-0.25	-0.29	-0.31
00209FC8	relative_humidity	2022	178	81.05	85.79	100.00	26.38	18.32	0.95	0.85	0.71	0.55	0.39	0.23	0.07	-0.06	-0.16	-0.24	-0.30	-0.32
00209FC8	precipitation	2022	178	0.02	0.00	26.00	0.00	0.37	0.25	0.20	0.17	0.10	0.04	0.07	0.06	0.01	0.01	-0.01	-0.01	0.00
00209FC8	leaf_wetness	2022	178	1.44	0.00	15.00	0.00	3.49	0.91	0.80	0.70	0.62	0.56	0.49	0.44	0.39	0.35	0.32	0.29	0.26
00209FC8	u-sonic wind speed	2022	2084	2.77	2.70	6.90	0.00	1.36	0.91	0.80	0.69	0.58	0.48	0.40	0.34	0.29	0.27	0.25	0.24	0.24
00209FC8	wind_gust	2022	2084	131.25	116.00	19.90	0.00	74.10	0.82	0.66	0.52	0.39	0.26	0.16	0.10	0.05	0.01	-0.02	-0.02	-0.02
00209FC8	deltaT	2022	178	2.06	1.00	11.00	-1.00	2.31	0.95	0.84	0.69	0.53	0.35	0.18	0.02	-0.11	-0.22	-0.30	-0.35	-0.37
01209E4F	precipitation	2022	4	0.02	0.00	17.60	0.00	0.29	0.29	0.21	0.20	0.12	0.05	0.10	0.07	0.01	0.02	-0.01	0.00	0.00
01209E4F	soil_moisture1	2022	9	33.09	33.70	41.10	23.40	3.53	1.00	0.99	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95
01209E4F	soil_moisture2	2022	9	47.31	47.30	59.20	40.80	2.12	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90	0.89	0.88
01209E4F	soil_moisture3	2022	9	47.50	47.40	55.60	44.60	1.13	0.99	0.98	0.97	0.95	0.93	0.92	0.90	0.89	0.88	0.86	0.85	0.84
01209E4F	soil_moisture4	2022	9	44.47	44.40	47.40	43.40	0.44	0.99	0.98	0.97	0.96	0.95	0.93	0.92	0.91	0.90	0.89	0.87	0.86
01209E4F	soil_moisture5	2022	9	38.33	38.30	39.80	37.80	0.33	1.00	0.99	0.99	0.99	0.98	0.98	0.97	0.97	0.97	0.97	0.96	0.96
01209E4F	soil_moisture6	2022	9	33.47	33.50	34.90	32.80	0.41	1.00	1.00	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.97
01209E4F	soil_temperature1	2022	9	16.46	17.30	24.80	6.40	4.74	1.00	0.99	0.99	0.98	0.97	0.96	0.96	0.95	0.94	0.94	0.93	0.93
01209E4F	soil_temperature2	2022	9	16.46	17.60	22.80	8.50	4.13	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98
01209E4F	soil_temperature3	2022	9	16.87	18.40	22.00	10.00	3.68	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99
01209E4F	soil_temperature4	2022	9	16.87	18.60	21.00	10.60	3.30	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99
01209E4F	soil_temperature5	2022	9	16.88	18.80	20.40	11.10	3.02	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99
01209E4F	soil_temperature6	2022	9	16.57	18.20	19.60	11.10	2.78	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99
01209E52	air_temperature	2022	0	14.96	15.35	32.56	-4.68	7.04	0.98	0.95	0.89	0.83	0.76	0.69	0.62	0.56	0.51	0.47	0.45	0.43
01209E52	dew_point	2022	0	11.27	12.00	25.80	-5.00	6.55	0.99	0.97	0.96	0.94	0.93	0.91	0.90	0.89	0.89	0.88	0.87	0.86
01209E52	VPD	2022	0	0.39	0.17	2.62	0.00	0.52	0.95	0.84	0.69	0.52	0.35	0.19	0.04	-0.08	-0.17	-0.24	-0.28	-0.29
01209E52	relative_humidity	2022	0	81.95	88.71	100.00	24.02	19.45	0.95	0.84	0.70	0.54	0.38	0.21	0.06	-0.06	-0.17	-0.24	-0.29	-0.32
01209E52	precipitation	2022	0	0.13	0.00	25.80	0.00	1.14	0.19	0.21	0.15	0.11	0.03	0.09	0.05	0.02	0.01	-0.01	-0.01	0.00
01209E52	deltaT	2022	0	1.99	1.00	11.00	-1.00	2.47	0.94	0.83	0.69	0.52	0.35	0.18	0.03	-0.10	-0.20	-0.28	-0.32	-0.35
01209E58	precipitation	2022	0	0.14	0.00	25.00	0.00	1.16	0.18	0.21	0.15	0.11	0.03	0.09	0.04	0.02	0.01	-0.01	-0.01	0.00
01209E58	soil_moisture1	2022	2	0.01	0.00	0.70	0.00	0.05	0.82	0.59	0.39	0.24	0.14	0.08	0.03	0.00	-0.01	-0.01	-0.01	-0.01
01209E58	soil_moisture2	2022	2	32.29	31.90	46.90	25.30	4.16	0.98	0.95	0.93	0.90	0.88	0.86	0.84	0.81	0.79	0.77	0.76	0.74
01209E58	soil_moisture3	2022	2	62.40	62.60	73.40	58.00	1.71	0.99	0.97	0.95	0.93	0.92	0.91	0.90	0.89	0.88	0.87	0.86	0.85
01209E58	soil_moisture4	2022	2	63.84	64.00	78.80	59.10	1.84	0.99	0.97	0.94	0.92	0.89	0.86	0.83	0.80	0.78	0.76	0.74	0.73
01209E58	soil_moisture5	2022	2	59.32	59.40	77.60	51.40	3.29	0.99	0.98	0.97	0.96	0.94	0.93	0.92	0.90	0.89	0.87	0.86	0.85
01209E58	soil_moisture6	2022	2	53.95	53.50	78.00	42.40	5.20	0.99	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90	0.89
01209E58	soil_temperature1	2022	2	17.37	17.10	45.00	-5.10	9.72	0.95	0.86	0.75	0.63	0.50	0.37	0.26	0.17	0.11	0.06	0.03	0.02
01209E58	soil_temperature2	2022	2	17.19	16.90	35.50	3.30	6.54	0.98	0.95	0.90	0.84	0.78	0.73	0.68	0.63	0.60	0.58	0.56	0.56
01209E58	soil_temperature3	2022	2	17.39	18.10	26.60	7.30	5.04	1.00	0.99	0.98	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.92	0.92
01209E58	soil_temperature4	2022	2	17.60	18.65	25.00	9.10	4.55	1.00	1.00	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97
01209E58	soil_temperature5	2022	2	17.56	19.00	23.70	10.10	4.14	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
01209E58	soil_temperature6	2022	2	17.77	19.50	23.00	11.00	3.81	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
00209FC8	air_temperature	2023	178	19.34	19.33	33.48	4.46	5.00	0.97	0.90	0.80	0.68	0.55	0.42	0.29	0.17	0.06	-0.02	-0.07	-0.09
00209FC8	dew_point	2023	178	15.51	15.70	24.90	4.40	3.87	0.98	0.95	0.92	0.89	0.87	0.85	0.83	0.81	0.80	0.78	0.77	0.75

Table D.3: *Cont.*

									Autocorrelation[1] at lag k (where k is in hours)											
Sensor Node	**Sensor**	**Year**	**# nulls**	μ	**Med.**	**Max**	**Min**	σ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
00209FC8	solar_radiation	2023	178	253.89	53.00	1476.00	0.00	352.23	0.90	0.73	0.54	0.33	0.12	-0.08	-0.23	-0.35	-0.44	-0.49	-0.52	-0.53
00209FC8	VPD	2023	178	0.52	0.31	3.15	0.00	0.56	0.95	0.83	0.67	0.49	0.30	0.11	-0.07	-0.23	-0.36	-0.45	-0.51	-0.53
00209FC8	relative_humidity	2023	178	81.15	85.13	100.00	32.14	17.32	0.95	0.83	0.68	0.51	0.32	0.13	-0.05	-0.22	-0.36	-0.47	-0.54	-0.56
00209FC8	precipitation	2023	178	0.03	0.00	16.20	0.00	0.35	0.29	0.16	0.10	0.04	0.01	0.00	-0.01	-0.01	0.00	0.00	-0.01	-0.01
00209FC8	leaf_wetness	2023	178	0.96	0.00	10.00	0.00	2.92	0.86	0.70	0.57	0.47	0.38	0.30	0.23	0.15	0.09	0.05	0.02	-0.02
00209FC8	deltaT	2023	178	2.35	2.00	11.00	0.00	2.39	0.95	0.83	0.67	0.49	0.30	0.11	-0.07	-0.24	-0.38	-0.48	-0.54	-0.57
01209E4F	precipitation	2023	4618	0.04	0.00	18.20	0.00	0.47	0.25	0.14	0.09	0.04	0.01	0.00	-0.01	-0.01	-0.01	0.00	-0.01	-0.01
01209E4F	soil_moisture1	2023	4638	40.90	40.40	68.60	35.00	3.55	0.94	0.88	0.82	0.77	0.72	0.68	0.65	0.62	0.60	0.57	0.55	0.53
01209E4F	soil_moisture2	2023	4638	50.93	50.20	86.80	47.80	2.81	0.95	0.88	0.83	0.77	0.72	0.68	0.64	0.60	0.57	0.54	0.52	0.50
01209E4F	soil_moisture3	2023	4638	49.38	48.90	88.40	45.70	2.48	0.93	0.84	0.78	0.71	0.64	0.59	0.54	0.49	0.46	0.43	0.40	0.38
01209E4F	soil_moisture4	2023	4638	46.78	46.20	88.40	45.70	1.95	0.82	0.73	0.67	0.60	0.53	0.48	0.43	0.40	0.37	0.34	0.32	0.31
01209E4F	soil_moisture5	2023	4638	40.28	39.40	89.70	38.80	2.22	0.72	0.62	0.55	0.47	0.40	0.35	0.32	0.30	0.29	0.28	0.26	0.25
01209E4F	soil_moisture6	2023	4638	34.41	33.80	87.00	33.30	2.04	0.60	0.54	0.49	0.43	0.36	0.32	0.29	0.28	0.26	0.25	0.24	0.22
01209E4F	soil_temperature1	2023	4638	21.65	21.70	26.60	15.10	2.05	0.92	0.83	0.74	0.65	0.58	0.52	0.47	0.43	0.40	0.37	0.35	0.34
01209E4F	soil_temperature2	2023	4638	21.17	21.30	23.70	16.80	1.33	0.92	0.84	0.77	0.71	0.67	0.64	0.61	0.59	0.57	0.56	0.54	0.53
01209E4F	soil_temperature3	2023	4638	21.06	21.10	22.50	17.40	0.95	0.90	0.80	0.72	0.66	0.62	0.60	0.58	0.57	0.56	0.55	0.53	0.52
01209E4F	soil_temperature4	2023	4638	20.58	20.80	21.80	17.40	0.69	0.88	0.76	0.66	0.60	0.56	0.53	0.52	0.51	0.50	0.50	0.49	0.48
01209E4F	soil_temperature5	2023	4638	20.21	20.30	21.20	17.50	0.56	0.87	0.75	0.65	0.59	0.55	0.53	0.52	0.51	0.51	0.50	0.49	0.48
01209E4F	soil_temperature6	2023	4638	19.57	19.60	20.50	17.20	0.51	0.90	0.79	0.72	0.67	0.64	0.62	0.62	0.61	0.60	0.59	0.59	0.58
01209E52	air_temperature	2023	0	19.39	19.31	33.77	4.99	4.96	0.97	0.90	0.80	0.68	0.55	0.42	0.29	0.17	0.07	-0.01	-0.06	-0.09
01209E52	dew_point	2023	0	16.27	16.50	25.70	4.10	3.98	0.97	0.93	0.88	0.84	0.81	0.78	0.76	0.75	0.75	0.74	0.73	0.72
01209E52	VPD	2023	0	0.43	0.11	3.19	0.00	0.58	0.94	0.81	0.64	0.46	0.26	0.08	-0.09	-0.22	-0.32	-0.39	-0.43	-0.45
01209E52	relative_humidity	2023	0	85.16	94.44	100.00	30.28	18.01	0.94	0.82	0.66	0.48	0.29	0.10	-0.07	-0.22	-0.34	-0.43	-0.48	-0.50
01209E52	precipitation	2023	0	0.16	0.00	58.80	0.00	1.61	0.23	0.15	0.10	0.06	-0.01	0.00	-0.01	-0.01	0.00	0.00	-0.01	-0.01
01209E52	deltaT	2023	0	1.90	1.00	12.00	0.00	2.47	0.94	0.81	0.64	0.46	0.27	0.09	-0.08	-0.22	-0.33	-0.41	-0.46	-0.47
01209E58	precipitation	2023	0	0.17	0.00	58.80	0.00	1.63	0.23	0.15	0.09	0.06	-0.01	0.00	-0.01	-0.01	0.00	0.00	-0.01	-0.01
01209E58	soil_moisture1	2023	0	0.01	0.00	1.90	0.00	0.07	0.52	0.37	0.21	0.08	0.00	0.00	-0.01	-0.02	-0.01	-0.01	-0.02	-0.02
01209E58	soil_moisture2	2023	0	37.74	37.30	71.60	24.00	8.68	0.99	0.97	0.96	0.94	0.92	0.91	0.89	0.88	0.86	0.85	0.84	0.83
01209E58	soil_moisture3	2023	0	59.87	61.20	79.00	49.40	5.40	1.00	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.94
01209E58	soil_moisture4	2023	0	66.49	67.00	77.20	59.90	3.13	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.95	0.94
01209E58	soil_moisture5	2023	0	63.37	63.10	73.00	58.50	3.22	1.00	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95
01209E58	soil_moisture6	2023	0	59.82	59.10	74.90	52.80	5.41	1.00	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.94
01209E58	soil_temperature1	2023	0	22.08	20.70	43.90	5.60	7.84	0.94	0.83	0.69	0.54	0.38	0.21	0.05	-0.08	-0.20	-0.29	-0.36	-0.38
01209E58	soil_temperature2	2023	0	21.48	21.00	36.60	11.00	4.70	0.97	0.89	0.80	0.68	0.55	0.42	0.30	0.20	0.11	0.04	0.00	-0.01
01209E58	soil_temperature3	2023	0	21.20	21.20	28.10	14.50	2.67	0.99	0.97	0.94	0.91	0.86	0.82	0.78	0.74	0.71	0.68	0.66	0.65
01209E58	soil_temperature4	2023	0	21.04	21.00	25.50	16.00	2.07	1.00	0.99	0.98	0.97	0.96	0.94	0.93	0.92	0.90	0.89	0.89	0.88
01209E58	soil_temperature5	2023	0	20.64	20.60	23.60	16.60	1.67	1.00	1.00	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95
01209E58	soil_temperature6	2023	0	20.54	20.60	22.60	17.20	1.37	1.00	1.00	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97
PN	soil_nitrogen	2023	905	0.15	0.00	18.00	0.00	0.96	0.96	0.90	0.82	0.75	0.70	0.65	0.60	0.56	0.52	0.48	0.44	0.40
PN	soil_phosphorous	2023	910	39.37	38.00	88.00	36.00	4.38	0.98	0.95	0.92	0.90	0.88	0.85	0.83	0.81	0.79	0.78	0.76	0.75
PN	soil_potassium	2023	911	31.52	30.00	81.00	28.00	4.43	0.98	0.95	0.92	0.90	0.87	0.85	0.83	0.81	0.79	0.77	0.76	0.75
PN	soil_temperature	2023	920	21.93	21.75	36.90	11.80	3.94	0.97	0.91	0.84	0.75	0.65	0.55	0.46	0.38	0.32	0.27	0.24	0.22
PN	soil_moisture	2023	907	11.40	10.80	100.00	0.00	4.75	0.91	0.82	0.77	0.72	0.67	0.63	0.61	0.58	0.56	0.55	0.53	0.51
PN	soil_pH	2023	926	4.89	4.73	5.88	4.43	0.30	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.91	0.90	0.89	0.89

Table D.4: A description of the data sources used in our N_2O emission prediction studies (described in Chapter 5) along with host device identifiers, sensor nodes, the available sensors, sensor units, sensor sampling rates, the placement of the sensors relative to the LI-COR gas chambers, and approximate elevation, where min = minute and ID = identification number.

Host Sensor Node	**Sensor Node**	**Sensors (units)**	**Sampling Rate**	**Approximate Distance from $C_{2023,1}$**	**Approximate Elevation**
PN	NPK soil sensor	• nitrogen (mg/L) • phosphorous (mg/L) • potassium (mg/L)	15 min	< 1 m	≈ 93.3 m
PN	Seeed Studio pH sensor (S-pH-01)	• pH (logarithmic) • soil temperature (°C)	15 min	< 1 m	≈ 93.3 m
PN	Soil humidity sensor	• moisture (%)	15 min	< 1	≈ 93.3 m
LI-COR	LI-COR automated gas chamber	• N_2O emission flux (ppb/m^2)	30 min	0 m	≈ 93.3 m
01209E52	Pessl iMetos ECO D3®	• air temperature (°C) • dew point (°C) • vapor pressure deficit (kPa) • air relative humidity (%) • precipitation (mm) • DeltaT (°C)	60 min	≈ 220 m	≈ 93.7 m
00209FC8	Pessl iMetos ECO D3®	• air temperature (°C) • dew point (°C) • solar radiation (W/m^2) • vapor pressure deficit (kPa) • air relative humidity (%) • precipitation (mm) • leaf wetness (min) • DeltaT (°C)	10 min (60 min in 2021)	≈ 210 m	≈ 93.7 m
01209E58	Pessl iMetos ECO D3®	• $\times 6$ soil moisture (%) • $\times 6$ soil temperature (°C) • precipitation (mm)	60 min	≈ 205 m	≈ 93.7 m
01209E4F	Pessl iMetos ECO D3®	• $\times 6$ soil moisture (%) • $\times 6$ soil temperature (°C) • precipitation (mm)	10 min (60 min in 2021)	≈ 15 m	≈ 93.2 m

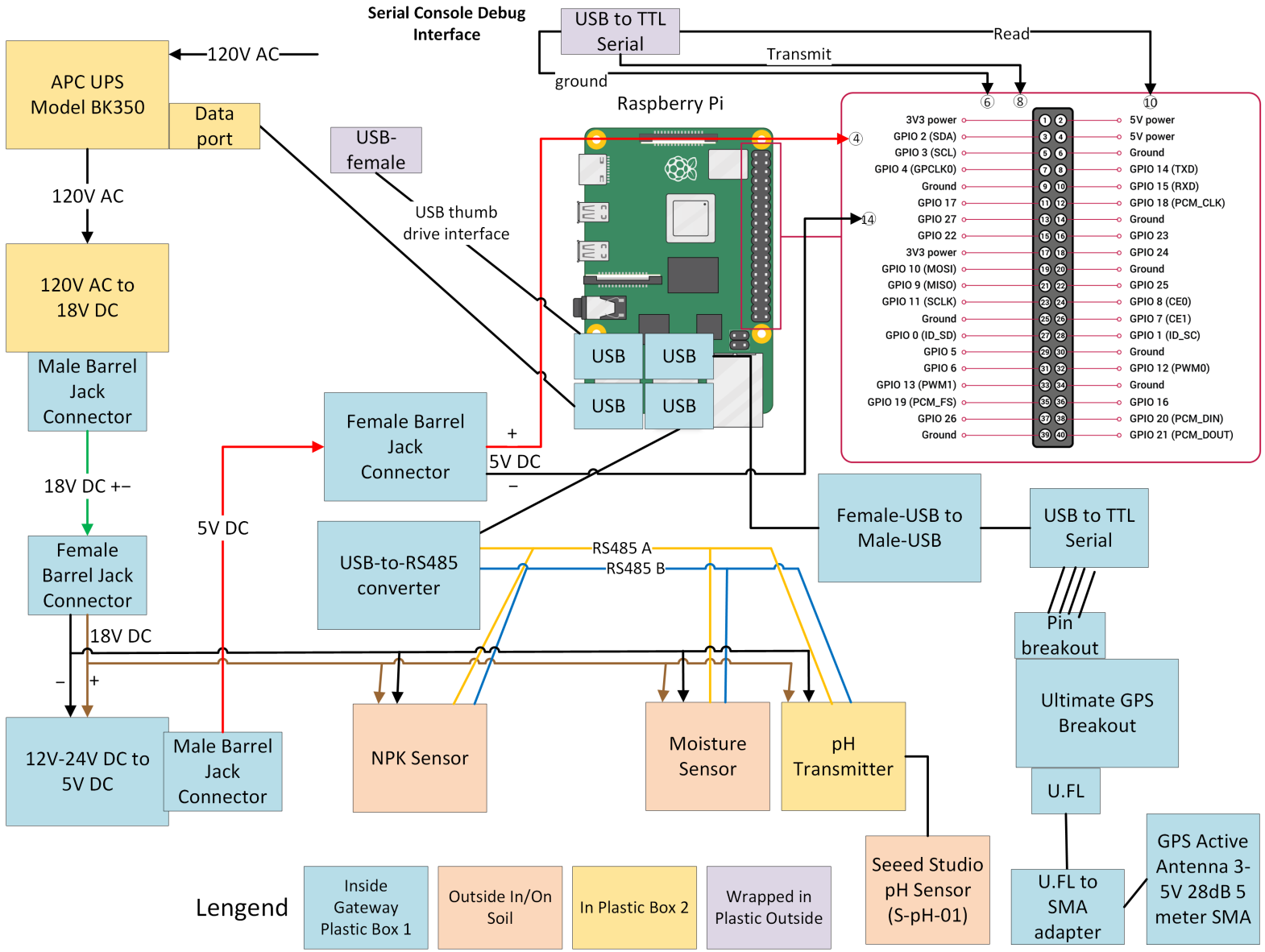


Figure D.6: N_2O prediction experiments: component and wiring diagram of the Summer 2023 PN prototype described in Section 5.3.3, where APC UPS = American Power Conversion uninterruptible power supply, NPK = Nitrogen-Phosphorus-Potassium, TTL = Transistor-Transistor Logic, and SMA = SubMiniature version A; the Raspberry Pi and pin diagram was taken from [247].

References

[1] L. J. Abendroth, R. W. Elmore, M. J. Boyer, and S. K. Marlay, "Understanding corn development: A key for successful crop management," *Proceedings of the Integrated Crop Management Conference*, no. 4, 2010.

[2] A. Abu-Khadrah, A. M. Ali, and M. Jarrah, "An amendable multi-function control method using federated learning for smart sensors in agricultural production improvements," *ACM Transactions on Sensor Networks*, 2023, just accepted.

[3] A. Adak, S. C. Murray, S. Božinović, R. Lindsey, S. Nakasagga, S. Chatterjee, S. L. Anderson, and S. Wilde, "Temporal vegetation indices and plant height from remotely sensed imagery can predict grain yield and flowering time breeding value in maize via machine learning regression," *Remote Sensing*, vol. 13, no. 11, p. 2141, 2021.

[4] M. Aggarwal, V. Khullar, N. Goyal, A. Alammari, M. A. Albahar, and A. Singh, "Lightweight federated learning for rice leaf disease classification using non independent and identically distributed images," *Sustainability*, vol. 15, no. 16, p. 12149, 2023.

[5] M. Aggarwal, V. Khullar, N. Goyal, R. Gautam, F. Alblehai, M. Elghatwary, and A. Singh, "Federated transfer learning for rice-leaf disease classification across multiclient cross-silo datasets," *Agronomy*, vol. 13, no. 10, p. 2483, 2023.

[6] M. Aggarwal, V. Khullar, N. Goyal, and T. A. Prola, "Resource-efficient federated learning over IoAT for rice leaf disease classification," *Computers and Electronics in Agriculture*, vol. 221, p. 109001, 2024.

[7] N. Aileen, *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media, 2019.

[8] M. Akbari, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Constrained federated learning for AoI-limited SFC in UAV-aided MEC for smart agriculture," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 277–295, 2023.

[9] ——, "AoI-aware energy-efficient SFC in UAV-aided smart agriculture using asynchronous federated learning," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1222–1242, 2024.

[10] M. N. Akhtar, A. J. Shaikh, A. Khan, H. Awais, E. A. Bakar, and A. R. Othman, "Smart sensing with edge computing in precision agriculture for soil assessment and heavy metal monitoring: A review," *Agriculture*, vol. 11, no. 6, p. 475, 2021.

[11] N. Akthar, M. V. Ahamad, and S. Ahmad, "MapReduce model of improved k-means clustering algorithm using Hadoop MapReduce," in *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE, 2016, pp. 192–198.

[12] M. R. Al Asif, K. F. Hasan, M. Z. Islam, and R. Khondoker, "STRIDE-based cyber security threat modeling for IoT-enabled precision agriculture systems," in *2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)*. IEEE, 2021, pp. 1–6.

[13] H. A. Alharbi and M. Aldossary, "Energy-efficient edge-fog-cloud architecture for IoT-based smart agriculture environment," *IEEE Access*, vol. 9, pp. 110 480–110 492, 2021.

[14] T. M. Antico, L. F. R. Moreira, and R. Moreira, "Evaluating the potential of federated learning for maize leaf disease prediction," in *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 2022, pp. 282–293.

[15] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (IoT)-based smart agriculture: Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129 551–129 583, 2019.

[16] M. Bacco, P. Barsocchi, E. Ferro, A. Gotta, and M. Ruggeri, "The digitisation of agriculture: a survey of research activities on smart farming," *Array*, vol. 3-4, p. 100009, 2019.

[17] L. Baghdasaryan, R. Melikbekyan, A. Dolmajain, and J. Hobbs, "Deep density estimation based on multi-spectral remote sensing data for in-field crop yield forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2014–2023.

[18] F. H. R. Baio, D. C. Santana, L. P. R. Teodoro, I. C. d. Oliveira, R. Gava, J. L. G. de Oliveira, C. A. d. Silva Junior, P. E. Teodoro, and L. S. Shiratsuchi, "Maize yield prediction with machine learning, spectral variables and irrigation management," *Remote Sensing*, vol. 15, no. 1, p. 79, 2022.

[19] A. Barbosa, T. Marinho, N. Martin, and N. Hovakimyan, "Multi-stream cnn for spatial resource allocation: A crop management application," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[20] A. Barbosa, R. Trevisan, N. Hovakimyan, and N. F. Martin, "Modeling yield response to crop management using convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 170, p. 105197, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919308543

[21] R. Barzin, R. Pathak, H. Lotfi, J. Varco, and G. C. Bora, "Use of UAS multispectral imagery at different physiological stages for yield prediction and input resource optimization in corn," *Remote Sensing*, vol. 12, no. 15, p. 2392, 2020.

[22] L. M. Bastos, C. W. Rice, P. J. Tomlinson, and D. Mengel, "Untangling soil-weather drivers of daily N2O emissions and fertilizer management mitigation strategies in no-till corn," *Soil Science Society of America Journal*, vol. 85, no. 5, pp. 1437–1447, 2021.

[23] R. Beigaitė, M. Mechenich, and I. Žliobaitė, "Spatial cross-validation for globally distributed data," in *International Conference on Discovery Science*. Springer, 2022, pp. 127–140.

[24] K. Benghzial, H. Raki, S. Bamansour, M. Elhamdi, Y. Aalaila, and D. H. Peluffo-Ordóñez, "GHG global emission prediction of synthetic N fertilizers using expectile regression techniques," *Atmosphere*, vol. 14, no. 2, p. 283, 2023.

[25] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis, "Machine learning in agriculture: A comprehensive updated review," *Sensors*, vol. 21, no. 11, p. 3758, 2021.

[26] S. Bera, T. Dey, A. Mukherjee, and R. Buyya, "E-CropReco: a dew-edge-based multiparametric crop recommendation framework for internet of agricultural things," *The Journal of Supercomputing*, vol. 79, no. 11, pp. 11 965–11 999, 2023.

[27] S. Bera, T. Dey, A. Mukherjee, and D. De, "FLAG: Federated learning for sustainable irrigation in agriculture 5.0," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2303–2310, 2024.

[28] L. Bigaignon, R. Fieuzal, C. Delon, and T. Tallec, "Combination of two methodologies, artificial neural network and linear interpolation, to gap-fill daily nitrous oxide flux measurements," *Agricultural and Forest Meteorology*, vol. 291, p. 108037, 2020.

[29] B. S. Bisht, I. Kiringa, and T. Yeap, "Corn yield prediction using spatial-temporal data and deep learning," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 1792–1798.

[30] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[31] P. Bose, N. K. Kasabov, L. Bruzzone, and R. N. Hartono, "Spiking neural networks for crop yield estimation based on spatiotemporal analysis of image time series," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 11, pp. 6563–6573, 2016.

[32] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[33] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53 841–53 849, 2020.

[34] G. Brown, "Ensemble learning." *Encyclopedia of machine learning*, vol. 312, pp. 15–19, 2010.

[35] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018, accessed June 11 2024. [Online]. Available: https://machinelearningmastery.com/better-deep-learning/

[36] J. Budakoti, "An IoT gateway middleware for interoperability in SDN managed internet of things," Ph.D. dissertation, Carleton University, 2018.

[37] T. F. Canata, M. C. F. Wei, L. F. Maldaner, and J. P. Molin, "Sugarcane yield mapping using high-resolution imagery data and machine learning technique," *Remote Sensing*, vol. 13, no. 2, p. 232, 2021.

[38] S. Chatterjee, A. Adak, S. Wilde, S. Nakasagga, and S. C. Murray, "Cumulative temporal vegetation indices from unoccupied aerial systems allow maize (zea mays l.) hybrid yield to be estimated across environments with fewer flights," *PLoS ONE*, vol. 18, no. 1, p. e0277804, 2023.

[39] D. Chen, P. Yang, I.-R. Chen, D. S. Ha, and J.-H. Cho, "SusFL: Energy-aware federated learning-based monitoring for sustainable smart farms," *arXiv preprint arXiv:2402.10280*, 2024.

[40] J. M. Chen, "Evaluation of vegetation indices and a modified simple ratio for boreal applications," *Canadian Journal of Remote Sensing*, vol. 22, no. 3, pp. 229–242, 1996.

[41] T. Cheng, Y. Zhu, D. Li, X. Yao, and K. Zhou, "Hyperspectral Remote Sensing of Leaf Nitrogen Concentration in Cereal Crops," in *Hyperspectral Indices and Image Classifications for Agriculture and Vegetation: Hyperspectral Remote Sensing of Vegetation*, 2nd ed., P. S. Thenkabail, J. G. Lyon, and A. Huete, Eds. CRC Press, 2018, vol. 2, ch. 6.

[42] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

[43] S. Choudhury, R. Singh, A. Gehlot, P. Kuchhal, S. V. Akram, N. Priyadarshi, B. Khan *et al.*, "Agriculture field automation and digitization using internet of things and machine learning," *Journal of Sensors*, vol. 2022, no. 1, p. 17, 2022.

[44] S. B. Choudhury, P. Jain, S. Kallamkuth, S. Ramanath, P. V. Bhatt, S. Sarangi, and P. Srinivasu, "Precision crop monitoring with affordable IoT: Experiences with okra," in *2019 Global IoT Summit (GIoTS)*. IEEE, 2019, pp. 1–6.

[45] S. Christensen, P. Ambus, J. Arah, H. Clayton, B. Galle, D. Griffith, K. Hargreaves, L. Klenzedtsson, A.-M. Lind, M. Maag *et al.*, "Nitrous oxide emission from an agricultural field: Comparison between measurements by flux chamber and micrometerological techniques," *Atmospheric Environment*, vol. 30, no. 24, pp. 4183–4190, 1996.

[46] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari, "LoRaFarM: A LoRaWAN-based smart farming modular IoT architecture," *Sensors*, vol. 20, no. 7, p. 2028, 2020.

[47] L. O. Colombo-Mendoza, M. A. Paredes-Valverde, M. d. P. Salas-Zárate, and R. Valencia-García, "Internet of things-driven data mining for smart crop production prediction in the peasant farming domain," *Applied Sciences*, vol. 12, no. 4, p. 1940, 2022.

[48] N. Cowan, P. Levy, J. Maire, M. Coyle, S. Leeson, D. Famulari, M. Carozzi, E. Nemitz, and U. Skiba, "An evaluation of four years of nitrous oxide fluxes after application of ammonium nitrate and urea fertilisers measured using the eddy covariance method," *Agricultural and Forest Meteorology*, vol. 280, p. 107812, 2020.

[49] M. Cruz, S. Mafra, and E. Teixeira, "An IoT crop recommendation system with k-NN and LoRa for precision farming," *XL SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS-SBrT 2022*, pp. 25–28, 2022.

[50] M. Cruz, S. Mafra, E. Teixeira, and F. Figueiredo, "Smart strawberry farming using edge computing and IoT," *Sensors*, vol. 22, no. 15, p. 5866, 2022.

[51] F. Cui, Z. A. Al-Sudani, G. S. Hassan, H. A. Afan, S. J. Ahammed, and Z. M. Yaseen, "Boosted artificial intelligence model using improved alpha-guided grey wolf optimizer for groundwater level prediction: Comparative study and insight for federated learning technology," *Journal of Hydrology*, vol. 606, p. 127384, 2022.

[52] A. Dahane, R. Benameur, and B. Kechar, "An IoT low-cost smart farming for enhancing irrigation efficiency of smallholders farmers," *Wireless Personal Communications*, vol. 127, no. 4, pp. 3173–3210, 2022.

[53] M. F. Danilevicz, P. E. Bayer, F. Boussaid, M. Bennamoun, and D. Edwards, "Maize yield prediction at an early developmental stage using multispectral images and genotype data for preliminary hybrid selection," *Remote Sensing*, vol. 13, no. 19, p. 3976, 2021.

[54] I. C. Davis and G. G. Wilkinson, "Crop yield prediction using multipolarization radar and multitemporal visible/infrared imagery," in *Remote Sensing for Agriculture, Ecosystems, and Hydrology Viii*, vol. 6359. SPIE, 2006, pp. 134–145.

[55] DEKALBASGROWDELTAPINE. (2020, Oct) Determining Corn Growth Stages. Accessed January 27 2025. [Online]. Available: https://www.dekalbasgrowdeltapine.com/en-us/agronomy/corn-growth-stages-and-gdu-requirements.html

[56] F. Deng, W. Mao, Z. Zeng, H. Zeng, and B. Wei, "Multiple diseases and pests detection based on federated learning and improved faster R-CNN," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[57] L. Deng, Z. Mao, X. Li, Z. Hu, F. Duan, and Y. Yan, "UAV-based multispectral remote sensing for precision agriculture: A comparison between different cameras," *ISPRS journal of photogrammetry and remote sensing*, vol. 146, pp. 124–136, 2018.

[58] S. X. Deng, "Assessment of greenhouse gas emissions from agriculture in canada: A comparative analysis," 2022, accessed February 28 2025. [Online]. Available: https://lfs-mlws-2020.sites.olt.ubc.ca/files/2023/03/dengsylvia_113064_22011205_Sylvia-Deng-Major-Project.pdf

[59] H. Devaraj, S. Sohail, B. Li, N. Hudson, M. Baughman, K. Chard, R. Chard, E. Casella, I. Foster, and O. Rana, "RuralAI in tomato farming: Integrated sensor system, distributed computing and hierarchical federated learning for crop health monitoring," *IEEE Sensors Letters*, vol. 8, no. 5, pp. 1–4, 2024.

[60] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, 2016.

[61] K. Dilmurat, V. Sagan, and S. Moose, "AI-DRIVEN MAIZE YIELD FORECASTING USING UNMANNED AERIAL VEHICLE-BASED HYPERSPECTRAL AND LIDAR DATA FUSION," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-3-2022, pp. 193–199, 2022.

[62] X. Duan, J. He, P. Cheng, Y. Mo, and J. Chen, "Privacy preserving maximum consensus," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 4517–4522.

[63] A. Durrant, M. Markovic, D. Matthews, D. May, J. Enright, and G. Leontidis, "The role of cross-silo federated learning in facilitating data sharing in the agri-food sector," *Computers and Electronics in Agriculture*, vol. 193, p. 106648, 2022.

[64] J. Ebenezer, P. G. Krishna, M. Poojitha, and A. V. Krishna, "Plant leaf disease detection and classification with CNN and federated learning approach," in *Conference of Innovative Product Design and Intelligent Manufacturing System*. Springer, 2022, pp. 513–523.

[65] D. Elavarasan, D. R. Vincent, V. Sharma, A. Y. Zomaya, and K. Srinivasan, "Forecasting yield by integrating agrarian factors and machine learning models: A survey," *Computers and Electronics in Agriculture*, vol. 155, pp. 257–282, 2018.

[66] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758–3773, 2018.

[67] J. Fan, J. Zhou, B. Wang, N. de Leon, S. M. Kaeppler, D. C. Lima, and Z. Zhang, "Estimation of maize yield and flowering time using multi-temporal UAV-based hyperspectral data," *Remote Sensing*, vol. 14, no. 13, p. 3052, 2022.

[68] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of iot in agriculture for the implementation of smart farming," *Ieee Access*, vol. 7, pp. 156 237–156 271, 2019.

[69] H. Fathipoor, H. Arefi, R. Shah-Hosseini, and H. Moghadam, "Corn forage yield prediction using unmanned aerial vehicle images at mid-season growth stage," *Journal of Applied Remote Sensing*, vol. 13, no. 3, p. 034503, 2019.

[70] I. Feigenwinter, L. Hörtnagl, and N. Buchmann, "N2O and CH4 fluxes from intensively managed grassland: The importance of biological and environmental drivers vs. management," *Science of The Total Environment*, vol. 903, p. 166389, 2023.

[71] P. Filippi, E. J. Jones, N. S. Wimalathunge, P. D. Somarathna, L. E. Pozza, S. U. Ugbaje, T. G. Jephcott, S. E. Paterson, B. M. Whelan, and T. F. Bishop, "An approach to forecast grain crop yield using multi-layered, multi-farm data sets and machine learning," *Precision Agriculture*, vol. 20, no. 5, pp. 1015–1029, 2019.

[72] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[73] P. Forster, T. Storelvmo, K. Armour, W. Collins, J.-L. Dufresne, D. Frame, D. Lunt, T. Mauritsen, M. Palmer, M. Watanabe *et al.*, "The earth's energy budget, climate feedbacks, and climate sensitivity," in *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Changed [Masson-Delmotte, V., P. Zhai, A. Pirani, S.L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M.I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T.K. Maycock, T. Waterfield, O. Yelekçi, R. Yu, and B. Zhou (eds.)]*. Open Access Te Herenga Waka-Victoria University of Wellington, 2021, pp. 923–1054.

[74] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, and M. Nafaa, "FELIDS: Federated learning-based intrusion detection system for agricultural internet of things," *Journal of Parallel and Distributed Computing*, vol. 165, pp. 17–31, 2022.

[75] J. N. Galloway, A. R. Townsend, J. W. Erisman, M. Bekunda, Z. Cai, J. R. Freney, L. A. Martinelli, S. P. Seitzinger, and M. A. Sutton, "Transformation of the nitrogen cycle: recent trends, questions, and potential solutions," *Science*, vol. 320, no. 5878, pp. 889–892, 2008.

[76] H. García-Martínez, H. Flores-Magdaleno, R. Ascencio-Hernández, A. Khalil-Gardezi, L. Tijerina-Chávez, O. R. Mancilla-Villa, and M. A. Vázquez-Peña, "Corn grain yield estimation from vegetation indices, canopy cover, plant density, and a neural network using multispectral and RGB images acquired with unmanned aerial vehicles," *Agriculture*, vol. 10, no. 7, p. 277, 2020.

[77] P. H. Garthwaite, "An interpretation of partial least squares," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 122–127, 1994.

[78] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica chimica acta*, vol. 185, pp. 1–17, 1986.

[79] E. Ghaderpour, S. D. Pagiatakis, and Q. K. Hassan, "A survey on change detection and time series analysis with applications," *Applied Sciences*, vol. 11, no. 13, p. 6141, 2021.

[80] E. Gilmore Jr and J. Rogers, "Heat units as a method of measuring maturity in corn," *Agronomy Journal*, vol. 50, no. 10, pp. 611–615, 1958, Contribution from Department of Agronomy, Texas Agr. Exp. Sta., College Station. Part of a thesis submitted by Mr. Gilmore in partial fulfillment of the requirements for an M.S. degree, Texas A. M. College.

[81] D. L. Giltrap and A.-G. E. Ausseil, "Upscaling NZ-DNDC using a regression based meta-model to estimate direct N2O emissions from New Zealand grazed pastures," *Science of the Total Environment*, vol. 539, pp. 221–230, 2016.

[82] D. Glaroudis, A. Iossifides, and P. Chatzimisios, "Survey, comparison and research challenges of IoT application protocols for smart farming," *Computer Networks*, vol. 168, p. 107037, 2020.

[83] A. J. Glenn, A. P. Moulin, A. K. Roy, and H. F. Wilson, "Soil nitrous oxide emissions from no-till canola production under variable rate nitrogen fertilizer management," *Geoderma*, vol. 385, p. 114857, 2021.

[84] A. Goap, D. Sharma, A. K. Shukla, and C. R. Krishna, "An IoT based smart irrigation management system using machine learning and open source technologies," *Computers and electronics in agriculture*, vol. 155, pp. 41–49, 2018.

[85] H. Gomaa, *Real-time software design for embedded systems*. Cambridge University Press, 2016.

[86] J. Goodrich, A. Wall, D. Campbell, D. Fletcher, A. Wecking, and L. Schipper, "Improved gap filling approach and uncertainty estimation for eddy covariance N2O fluxes," *Agricultural and Forest Meteorology*, vol. 297, p. 108280, 2021.

[87] S. Goryczka, L. Xiong, and V. Sunderam, "Secure multiparty aggregation with differential privacy: A comparative study," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013, pp. 155–163.

[88] T. W. Griffin, "The spatial analysis of yield data," in *Geostatistical Applications for Precision Agriculture*, O. A. Margareth, Ed., 2010, ch. 4.

[89] Y. Gueye and M. Mbaye, "eFarm-Lab: Edge AI-IoT framework for agronomic labs experiments," in *Research in Computer Science and Its Applications: 11th International Conference, CNRIA 2021, Virtual Event, June 17-19, 2021, Proceedings*. Springer, 2021, pp. 101–112.

[90] M. A. Guillén, A. Llanes, B. Imbernón, R. Martínez-España, A. Bueno-Crespo, J.-C. Cano, and J. M. Cecilia, "Performance evaluation of edge-computing platforms for the prediction of low temperatures in agriculture using deep learning," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 818–840, 2021.

[91] L. Gunisetti, S. B. Koduri, and V. Jagannathan, "Optimized deep learning system for smart maize leaf disease detection in IoT platform via routing algorithm," *Multimedia Tools and Applications*, vol. 82, no. 9, pp. 13 533–13 555, 2023.

[92] Y. Guo, H. Wang, Z. Wu, S. Wang, H. Sun, J. Senthilnath, J. Wang, C. Robin Bryant, and Y. Fu, "Modified red blue vegetation index for chlorophyll estimation and yield prediction of maize from visible images captured by UAV," *Sensors*, vol. 20, no. 18, p. 5055, 2020.

[93] Y. Guo, X. Zhang, S. Chen, H. Wang, S. Jayavelu, D. Cammarano, and Y. Fu, "Integrated UAV-based multi-source data for predicting maize grain yield using machine learning approaches," *Remote Sensing*, vol. 14, no. 24, p. 6290, 2022.

[94] D. Haboudane, J. R. Miller, E. Pattey, P. J. Zarco-Tejada, and I. B. Strachan, "Hyperspectral vegetation indices and novel algorithms for predicting green LAI of crop canopies: Modeling and validation in the context of precision agriculture," *Remote Sensing of Environment*, vol. 90, no. 3, pp. 337–352, 2004.

[95] D. Haboudane, N. Tremblay, J. R. Miller, and P. Vigneault, "Remote estimation of crop chlorophyll content using spectral indices derived from hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 2, pp. 423–437, 2008.

[96] A. Hamrani, A. Akbarzadeh, and C. A. Madramootoo, "Machine learning for predicting greenhouse gas emissions from agricultural soils," *Science of The Total Environment*, vol. 741, p. 140338, 2020.

[97] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 6, pp. 610–621, 1973.

[98] P. Hari, M. P. Singh, and A. K. Singh, "An improved federated deep learning for plant leaf disease detection," *Multimedia Tools and Applications*, vol. 83, p. 83471–83491, 2024.

[99] M. Hashemi and H. A. Karimi, "Weighted machine learning for spatial-temporal data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3066–3082, 2020.

[100] S. Hawinkel, S. De Meyer, and S. Maere, "Spatial regression models for field trials: a comparative study and new ideas," *Frontiers in Plant Science*, p. 592, 2022.

[101] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Hamilton, Ontario, Canada: Pearson Prentice Hall, 2009, vol. 1.

[102] A. Hensen, U. Skiba, and D. Famulari, "Low cost and state of the art methods to measure nitrous oxide emissions," *Environmental Research Letters*, vol. 8, no. 2, p. 025022, 2013.

[103] I. Herrmann, E. Bdolach, Y. Montekyo, S. Rachmilevitch, P. A. Townsend, and A. Karnieli, "Assessment of maize yield and phenology by drone-mounted superspectral camera," *Precision Agriculture*, vol. 21, no. 1, pp. 51–76, 2020.

[104] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[105] M. Homaira and R. Hassan, "Prediction of agricultural emissions in malaysia using the arima, LSTM, and regression models," *International Journal on Perceptive and Cognitive Computing*, vol. 7, no. 1, pp. 33–40, 2021.

[106] Y. Huang, X. Yang, J. Guo, J. Cheng, H. Qu, J. Ma, and L. Li, "A high-precision method for 100-day-old classification of chickens in edge computing scenarios based on federated computing," *Animals*, vol. 12, no. 24, p. 3450, 2022.

[107] A. Huete, "A soil-adjusted vegetation index (SAVI)," *Remote Sensing of Environment*, vol. 25, no. 3, pp. 295–309, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/003442578890106X

[108] G. Idoje, T. Dagiuklas, and M. Iqbal, "Federated learning: Crop classification in a smart farm decentralised network," *Smart Agricultural Technology*, vol. 5, p. 100277, 2023.

[109] A. Imtiaz, H. Gousia, K. Parmod *et al.*, "Machine learning-based peer-to-peer platform for precision agriculture in crop growth and disease monitoring," *Journal of Computer Technology & Applications*, vol. 13, no. 03, pp. 26–39, 2023.

[110] K. A. Jani, N. K. Chaubey, E. Panchal, P. Tripathi, and S. Yagnik, "A novel secure approach for enhancing accuracy of pest detection with private federated learning using dpsgd," in *World Conference on Information Systems and Technologies*. Springer, 2023, pp. 305–317.

[111] N. Japkowicz and Z. Boukouvalas, *Machine Learning Evaluation Towrads Reliable and Responsible AI*. Cambridge University Press, 2025.

[112] G. R. Jeffries, T. S. Griffin, D. H. Fleisher, E. N. Naumova, M. Koch, and B. D. Wardlow, "Mapping sub-field maize yields in Nebraska, USA by combining remote sensing imagery, crop simulation models, and machine learning," *Precision Agriculture*, pp. 1–17, 2019.

[113] J. H. Jeong, J. P. Resop, N. D. Mueller, D. H. Fleisher, K. Yun, E. E. Butler *et al.*, "Random forests for global and regional crop yield predictions," *PLoS ONE*, vol. 11, no. 6, p. e0156571, 2016.

[114] Z. Jiang, S. Yang, X. Chen, Q. Pang, Y. Xu, S. Qi, W. Yu, and H. Dai, "Controlled release urea improves rice production and reduces environmental pollution: A research based on meta-analysis and machine learning," *Environmental Science and Pollution Research*, vol. 29, pp. 3587–3599, 2022.

[115] Z. Jiang, S. Yang, P. Smith, and Q. Pang, "Ensemble machine learning for modeling greenhouse gas emissions at different time scales from irrigated paddy fields," *Field Crops Research*, vol. 292, p. 108821, 2023.

[116] D. R. Joshi, D. E. Clay, S. A. Clay, J. Moriles-Miller, A. L. Daigh, G. Reicks, and S. Westhoff, "Quantification and machine learning based N2O-N and CO2-C emissions predictions from a decomposing rye cover crop," *Agronomy Journal*, vol. 116, no. 3, pp. 795–809, 2022.

[117] P. Kairouz and H. B. McMahan *et al.*, "Advances and Open Problems in Federated Learning," *Foundations and Trends® in Machine learning*, vol. 14, pp. 1–210, 2021, no. 1–2.

[118] C. Kamienski, J.-P. Soininen, M. Taumberger, R. Dantas, A. Toscano, T. Salmon Cinotti, R. Filev Maia, and A. Torre Neto, "Smart water management platform: IoT-based precision irrigation for agriculture," *Sensors*, vol. 19, no. 2, p. 276, 2019.

[119] A. Kamilaris, A. Kartakoullis, and F. X. Prenafeta-Boldú, "A review on the practice of big data analysis in agriculture," *Computers and Electronics in Agriculture*, vol. 143, pp. 23–37, 2017.

[120] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[121] A. Kar, M. Ahmed, A. A. May, and H. T. Le, "High spatio-temporal resolution predictions of PM2.5 using low-cost sensor data," *Atmospheric Environment*, vol. 326, p. 120486, 2024.

[122] P. K. Kashyap, S. Kumar, A. Jaiswal, M. Prasad, and A. H. Gandomi, "Towards precision agriculture: IoT-enabled intelligent irrigation systems using deep learning neural network," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 479–17 491, 2021.

[123] S. Kawashima and M. Nakatani, "An algorithm for estimating chlorophyll content in leaves using a video camera," *Annals of Botany*, vol. 81, no. 1, pp. 49–54, 1998.

[124] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Analytics*. Cambridge, MA: The MIT Press, 2015.

[125] F. S. Khan, S. Khan, M. N. H. Mohd, A. Waseem, M. N. A. Khan, S. Ali, and R. Ahmed, "Federated learning-based UAVs for the diagnosis of plant diseases," in *2022 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, 2022, pp. 1–6.

[126] S. Khanal, J. Fulton, A. Klopfenstein, N. Douridas, and S. Shearer, "Integration of high resolution remotely sensed data and machine learning techniques for spatial prediction of soil properties and corn yield," *Computers and Electronics in Agriculture*, vol. 153, pp. 213–225, 2018.

[127] S. Khanal, A. Klopfenstein, K. Kushal, V. Ramarao, J. Fulton, N. Douridas, and S. A. Shearer, "Assessing the impact of agricultural field traffic on corn grain yield using remote sensing and machine learning," *Soil and Tillage Research*, vol. 208, p. 104880, 2021.

[128] T. P. Kharel, A. J. Ashworth, P. R. Owens, and M. Buser, "Spatially and temporally disparate data in systems agriculture: Issues and prospective solutions," *Agronomy Journal*, vol. 112, no. 5, pp. 4498–4510, 2020.

[129] A. Khattab, A. Abdelgawad, and K. Yelmarthi, "Design and implementation of a cloud-based IoT scheme for precision agriculture," in *2016 28th International Conference on Microelectronics (ICM)*. IEEE, 2016, pp. 201–204.

[130] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, and M. Nordlund, "Open-source federated learning frameworks for IoT: A comparative review and analysis," *Sensors*, vol. 21, no. 1, p. 167, 2020.

[131] P. Killeen, "Knowledge-based predictive maintenance for fleet management," Master's thesis, Université d'Ottawa/University of Ottawa, 2020.

[132] P. Killeen, I. Kiringa, and T. Yeap, "Corn grain yield prediction using UAV-based high spatiotemporal resolution multispectral imagery," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2022, pp. 1054–1062.

[133] ——, "Unsupervised dynamic sensor selection for IoT-based predictive maintenance of a fleet of public transport buses," *ACM Transactions on Internet of Things*, vol. 3, no. 3, pp. 1–36, 2022.

[134] P. Killeen, I. Kiringa, T. Yeap, and P. Branco, "Using UAV-based multispectral imagery, data-driven models, and spatial cross-validation for corn grain yield prediction," in *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2023, pp. 220–228.

[135] ——, "Corn grain yield prediction using UAV-based high spatiotemporal resolution imagery, machine learning, and spatial cross-validation," *Remote sensing*, vol. 16, no. 4, p. 683, 2024.

[136] P. Killeen, C. Lin, F. Li, I. Kiringa, and T. Yeap, "IoT-based smart farming architecture using federated learning: a nitrous oxide emission prediction use case," *ACM Journal on Computing and Sustainable Societies*, vol. 3, no. 2, pp. 1–38, 2025.

[137] ——, "Nitrous oxide emission prediction using IoT soil and weather sensor data," *ACM Journal on Computing and Sustainable Societies*, 2025, just accepted.

[138] P. Killeen, T. Yeap, and I. Kiringa, "UAV imagery-based yield prediction using distributed machine learning via model sharing for smart farming applications," unpublished.

[139] A. Kolasa-Więcek, "Use of artificial neural networks in predicting direct nitrous oxide emissions from agricultural soils," *Ecological Chemistry and Engineering S*, vol. 20, no. 2, pp. 419–428, 2013. [Online]. Available: https://doi.org/10.2478/eces-2013-0030

[140] M. Kulbacki, J. Segen, W. Knieć, R. Klempous, K. Kluwak, J. Nikodem, J. Kulbacka, and A. Serester, "Survey of drones for agriculture automation from planting to harvest," in *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2018, pp. 000 353–000 358.

[141] C. Kumar, P. Mubvumba, Y. Huang, J. Dhillon, and K. Reddy, "Multi-stage corn yield prediction using high-resolution uav multispectral data and machine learning models," *Agronomy*, vol. 13, no. 5, p. 1277, 2023.

[142] P. Kumar, G. P. Gupta, and R. Tripathi, "PEFL: Deep privacy-encoding-based federated learning framework for smart agriculture," *IEEE Micro*, vol. 42, no. 1, pp. 33–40, 2021.

[143] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, 2017.

[144] L. Lambrinos, "Internet of things in agriculture: A decision support system for precision farming," in *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2019, pp. 889–892.

[145] G. Lavanya, C. Rani, and P. GaneshKumar, "An automated low cost IoT based fertilizer intimation system for smart agriculture," *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100300, 2020.

[146] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[147] A. Li, M. Markovic, P. Edwards, and G. Leontidis, "Model pruning enables localized and efficient federated learning for yield forecasting and data sharing," *Expert Systems with Applications*, vol. 242, p. 122847, 2024.

[148] J. Li, H. Zhang, C.-Y. Chao, C.-H. Chien, C.-Y. Wu, C. H. Luo, L.-J. Chen, and P. Biswas, "Integrating low-cost air quality sensor networks with fixed and satellite monitoring systems to study ground-level PM2.5," *Atmospheric Environment*, vol. 223, p. 117293, 2020.

[149] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[150] C. Lin, F. Li, P. Killeen, T. Yeap, and I. Kiringa, "Agriculture-informed neural networks for predicting nitrous oxide emissions," *ACM Transactions on Internet of Things*, vol. 5, no. 4, pp. 1–23, 2024.

[151] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[152] J. Lindblom and M. Ljung, "Agriculture through sustainable IT failures and success factors in AgriDSS," in *The 11th European IFSA Symposium*, vol. 2, 2014, pp. 49–57.

[153] K. Liu, "Tomato leaf disease detection based on peer-to-peer federated learning in wireless networks," in *Fourth International Conference on Artificial Intelligence and Electromechanical Automation (AIEA 2023)*, vol. 12709. SPIE, 2023, pp. 1207–1212.

[154] L. Liu, M. Prodanović, and M. J. Pyrcz, "Impact of geostatistical nonstationarity on convolutional neural network predictions," *Computational Geosciences*, pp. 1–10, 2022.

[155] L. Liu, S. Xu, J. Tang, K. Guan, T. J. Griffis, M. D. Erickson, A. L. Frie, X. Jia, T. Kim, L. T. Miller *et al.*, "KGML-ag: a modeling framework of knowledge-guided machine learning to simulate agroecosystems: a case study of estimating N2O emission using data from mesocosm experiments," *Geoscientific Model Development*, vol. 15, no. 7, pp. 2839–2858, 2022.

[156] Q. Liu, W.-j. Xie, and J.-b. Xia, "Using semivariogram and Moran's I techniques to evaluate spatial distribution of soil micronutrients," *Communications in Soil Science and Plant Analysis*, vol. 44, no. 7, pp. 1182–1192, 2013.

[157] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz, "From industry 4.0 to agriculture 4.0: Current status, enabling technologies, and research challenges," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4322–4334, 2020.

[158] Z. Liu, Z. Zhu, J. Gao, and C. Xu, "Forecast methods for time series data: A survey," *IEEE Access*, vol. 9, pp. 91 896–91 912, 2021.

[159] G. Lyle, B. A. Bryan, and B. Ostendorf, "Post-processing methods to eliminate erroneous grain yield measurements: review and directions for future development," *Precision Agriculture*, vol. 15, no. 4, pp. 377–402, 2014.

[160] R. Maier, L. Hörtnagl, and N. Buchmann, "Greenhouse gas fluxes (CO2, N2O and CH4) of pea and maize during two cropping seasons: Drivers, budgets, and emission factors for nitrous oxide," *Science of the Total Environment*, vol. 849, p. 157541, 2022.

[161] M. Maimaitijiang, V. Sagan, P. Sidike, S. Hartling, F. Esposito, and F. B. Fritschi, "Soybean yield prediction from uav using multimodal data fusion and deep learning," *Remote Sensing of Environment*, vol. 237, p. 111599, 2020.

[162] D. Mamba Kabala, A. Hafiane, L. Bobelin, and R. Canals, "Image-based crop disease detection with federated learning," *Scientific Reports*, vol. 13, no. 1, p. 19220, 2023.

[163] R. Manikandan, G. Ranganathan, and V. Bindhu, "Deep learning based IoT module for smart farming in different environmental conditions," *Wireless Personal Communications*, vol. 128, no. 3, pp. 1715–1732, 2023.

[164] T. Manoj, K. Makkithaya, and V. Narendra, "A federated learning-based crop yield prediction for agricultural production risk management," in *2022 IEEE Delhi Section Conference (DELCON)*. IEEE, 2022, pp. 1–7.

[165] A. Mao, E. Huang, H. Gan, and K. Liu, "FedAAR: A novel federated learning framework for animal activity recognition with wearable sensors," *Animals*, vol. 12, no. 16, p. 2142, 2022.

[166] M. d. J. Marcial-Pablo, A. Gonzalez-Sanchez, S. I. Jimenez-Jimenez, R. E. Ontiveros-Capurata, and W. Ojeda-Bustamante, "Estimation of vegetation fraction using RGB and multispectral images from UAV," *International Journal of Remote Sensing*, vol. 40, no. 2, pp. 420–438, 2019.

[167] Á. Maresma, M. Ariza, E. Martínez, J. Lloveras, and J. A. Martínez-Casasnovas, "Analysis of vegetation indices to determine nitrogen application and yield prediction in maize (Zea mays L.) from a standard UAV service," *Remote Sensing*, vol. 8, no. 12, p. 973, 2016.

[168] G. McGraw, *Software Security: Building Security In*. Pearson Education, 2006.

[169] J. McKenzie, "'Mass delusion and wishful thinking': Why everything you think you know about methane is probably wrong," Dec. 2023, accessed December 3 2024. [Online]. Available: https://thebulletin.org/2023/12/mass-delusion-and-wishful-thinking-why-everything-you-think-you-know-about-methane-is-probably-wrong/

[170] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[171] A. Mehmandoost Kotlar, J. Singh, and S. Kumar, "Prediction of greenhouse gas emissions from agricultural fields with and without cover crops," *Soil Science Society of America Journal*, vol. 86, no. 5, pp. 1227–1240, 2022.

[172] L. Merbold, W. Eugster, J. Stieger, M. Zahniser, D. Nelson, and N. Buchmann, "Greenhouse gas budget (CO2, CH4 and N2O) of intensively managed grassland following restoration," *Global Change Biology*, vol. 20, no. 6, pp. 1913–1928, 2014.

[173] H. Meyer and E. Pebesma, "Predicting into unknown space? estimating the area of applicability of spatial prediction models," *Methods in Ecology and Evolution*, vol. 12, no. 9, pp. 1620–1633, 2021.

[174] MicaSense. (2017, Nov.) MicaSense RedEdge-M Multispectral Camera user manual. Accessed March 10 2022. [Online]. Available: https://www.geotechenv.com/Manuals/Leptron_Manuals/RedEdge-M_User_Manual.pdf

[175] ——. (2022, Jan.) Overview of Agriculture Indices. Accessed March 22 2022. [Online]. Available: https://support.micasense.com/hc/en-us/articles/227837307-Overview-of-Agricultural-Indices

[176] C. Mila, J. Mateu, E. Pebesma, and H. Meyer, "Nearest neighbour distance matching leave-one-out cross-validation for map validation," *Methods in Ecology and Evolution*, vol. 13, no. 6, pp. 1304–1316, 2022.

[177] Monsanto Company, "Corn growth stages and GDU requirements," *Agronomic Spotlight*, 2015, accessed February 28 2025. [Online]. Available: https://www.corn-states.com/app/uploads/2018/07/corn-growth-stages-and-gdu-requirements.pdf

[178] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*, second edition ed. John Wiley & Sons, 2015.

[179] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, 5th ed. John Wiley & Sons, 2010.

[180] R. Moreira, L. F. R. Moreira, P. L. A. Munhoz, E. A. Lopes, and R. A. A. Ruas, "AgroLens: A low-cost and green-friendly smart farm architecture to support real-time leaf disease diagnostics," *Internet of Things*, vol. 19, p. 100570, 2022.

[181] B. L. Muhammad and M. B. Kusharki, "Federated learning for collaborative crop disease monitoring in wheat production," in *2023 2nd International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS)*, vol. 1. IEEE, 2023, pp. 1–5.

[182] S. Mythili, K. Nithya, M. Krishnamoorthi, and M. Kalamani, "Standard and protocols for agro-IoT," in *Cloud IoT systems for smart agricultural engineering*, 1st ed., S. Krishnan, J. B. R. Rose, N. R. Rajalakshmi, and N. Prasanth, Eds. CRC Press, 2022, ch. 12.

[183] N. Naik, "Choice of effective messaging protocols for iot systems: MQTT, CoAP, AMQP and HTTP," in *2017 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 2017, pp. 1–7.

[184] P. Nevavuori, N. Narra, P. Linna, and T. Lipping, "Crop yield prediction using multitemporal UAV data and spatio-temporal deep learning models," *Remote Sensing*, vol. 12, no. 23, p. 4000, 2020.

[185] P. Nevavuori, N. Narra, and T. Lipping, "Crop yield prediction with deep convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 163, p. 104859, 2019.

[186] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[187] Nielsen, R. L. (Bob). (2019, May) Predict Leaf Stage Development in Corn Using Thermal Time. Accessed January 27 2025. [Online]. Available: https://www.agry.purdue.edu/ext/corn/news/timeless/VStagePrediction.html

[188] ——. (2020, April) Heat Unit Concepts Related to Corn Development. Accessed January 27 2025. [Online]. Available: https://www.agry.purdue.edu/ext/corn/news/timeless/heatunits.html

[189] B. Nikparvar and J.-C. Thill, "Machine learning of spatial data," *ISPRS International Journal of Geo-Information*, vol. 10, no. 9, p. 600, 2021.

[190] T. Nleya, C. Chungu, and J. Kleinjan. Corn growth and development. Accessed February 28 2025. [Online]. Available: https://www.researchgate.net/profile/Thandiwe-Nleya/publication/331683778_Corn_Growth_and_Development/links/5c87c7fb92851c831974f6c8/Corn-Growth-and-Development.pdf

[191] C. Oglesby, A. A. Fox, G. Singh, and J. Dhillon, "Predicting in-season corn grain yield using optical sensors," *Agronomy*, vol. 12, no. 10, p. 2402, 2022.

[192] A. Oliveira-Jr, C. Resende, A. Pereira, P. Madureira, J. Gonçalves, R. Moutinho, F. Soares, and W. Moreira, "IoT sensing platform as a driver for digital farming in rural Africa," *Sensors*, vol. 20, no. 12, p. 3511, 2020.

[193] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4118–4149, 2018.

[194] G. Onoufriou, P. Mayfield, and G. Leontidis, "Fully homomorphically encrypted deep learning as a service," *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 819–834, 2021.

[195] M. P.-L. Ooi, S. Sohail, V. G. Huang, N. Hudson, M. Baughman, O. Rana, A. Hinze, K. Chard, R. Chard, I. Foster, T. Spyridopoulos, and H. Nagra, "Measurement and applications: Exploring the challenges and opportunities of hierarchical federated learning in sensor applications," *IEEE Instrumentation & Measurement Magazine*, vol. 26, no. 9, pp. 21–31, 2023.

[196] F. Ortenberg, "Hyperspectral sensor characteristics: Airborne, spaceborne, handheld, and truck-mounted; integration of hyperspectral data with LiDAR," in *Fundamentals, Sensor Systems, Spectral Libraries, and Data Mining for Vegetation: Hyperspectral Remote Sensing of Vegetation*, 2nd ed., P. S. Thenkabail, J. G. Lyon, and A. Huete, Eds. CRC Press, 2018, vol. 1, ch. 2.

[197] C. S. O'Connell, T. L. Anthony, M. A. Mayes, T. Pérez, D. Sihi, and W. L. Silver, "Utilizing novel field and data exploration methods to explore hot moments in high-frequency soil nitrous oxide emissions data: Opportunities and challenges," *Frontiers in Forests and Global Change*, vol. 5, p. 674348, 2022.

[198] B. Pan, S. K. Lam, E. Wang, A. Mosier, and D. Chen, "New approach for predicting nitrification and its fraction of N2O emissions in global terrestrial ecosystems," *Environmental Research Letters*, vol. 16, no. 3, p. 034053, 2021.

[199] R. Patole, N. Singh, M. Adhikari, and A. K. Singh, "Multi-view ensemble federated learning for efficient prediction of consumer electronics applications in fog networks," *IEEE Transactions on Consumer Electronics*, 2023.

[200] G. Patrizi, A. Bartolini, L. Ciani, V. Gallo, P. Sommella, and M. Carratù, "A virtual soil moisture sensor for smart farming using deep learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[201] P. Patros, M. Ooi, V. Huang, M. Mayo, C. Anderson, S. Burroughs, M. Baughman, O. Almurshed, O. Rana, R. Chard *et al.*, "Rural AI: Serverless-powered federated learning for remote applications," *IEEE Internet Computing*, vol. 27, no. 2, pp. 28–34, 2022.

[202] R. Peeriga, D. R. Rinku, J. U. Bhaskar, R. Nagalingam, F. M. Aldosari, H. M. Albarakati, A. A. Alharbi, and A. Y. Jaffar, "Real-time rain prediction in agriculture using ai and iot: A bi-directional LSTM approach," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15 805–15 812, 2024.

[203] D. Pelliccia. (2018, Jul.) A variable selection method for PLS in Python. Accessed November 7 2023. [Online]. Available: https://nirpyresearch.com/variable-selection-method-pls-python/

[204] A. Philibert, C. Loyce, and D. Makowski, "Prediction of N2O emission from local information with random forest," *Environmental pollution*, vol. 177, pp. 156–163, 2013.

[205] PIX4D. (2025) Computer requirements - PIX4Dmapper. Accessed October 20 2025. [Online]. Available: https://support.pix4d.com/hc/en-us/articles/115002439383

[206] P. Ploton, F. Mortier, M. Réjou-Méchain, N. Barbier, N. Picard, V. Rossi, C. Dormann, G. Cornu, G. Viennois, N. Bayol, A. Lyapustin, S. Gourlet-Fleury, and R. Pélissier, "Spatial validation reveals poor predictive performance of large-scale ecological mapping models," *Nature Communications*, vol. 11, no. 1, p. 4540, Sep 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-18321-y

[207] L. Praharaj, M. Gupta, and D. Gupta, "Hierarchical federated transfer learning and digital twin enhanced secure cooperative smart farming," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 3304–3313.

[208] M. J. Prather, C. D. Holmes, and J. Hsu, "Reactive greenhouse gas scenarios: Systematic exploration of uncertainties and the role of atmospheric chemistry," *Geophysical Research Letters*, vol. 39, no. 9, 2012.

[209] S. Premkumar and A. Sigappi, "IoT-enabled edge computing model for smart irrigation system," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 632–650, 2022.

[210] J. W. M. Pullens, D. Abalos, S. O. Petersen, and A. R. Pedersen, "Identifying criteria for greenhouse gas flux estimation with automatic and manual chambers: A case study for N2O," *European Journal of Soil Science*, vol. 74, no. 1, p. e13340, 2023.

[211] V. Puri, V. K. Solanki, and G. J. R. Aponte, "Blockchain and federated learning based integrated approach for agricultural internet of things," in *The International Conference on Intelligent Systems & Networks*. Springer, 2023, pp. 240–246.

[212] M. Qiao, X. He, X. Cheng, P. Li, H. Luo, L. Zhang, and Z. Tian, "Crop yield prediction from multi-spectral, multi-temporal remotely sensed imagery using recurrent 3d convolutional neural networks," *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, p. 102436, 2021.

[213] X. Quan and R. Doluschitz, "Unmanned aerial vehicle (UAV) technical applications, standard workflow, and future developments in maize production–water stress detection, weed mapping, nutritional status monitoring and yield prediction," *Agricultural Engineering. EU*, vol. 76, no. 1, 2021.

[214] R. Raghu, V. Jayaramaan, J. Jayaraman, S. S. V. Nukala, and C. E. Montenegro-Marin, "A user-centered security approach to create an IoT based multi-layered fog-cloud architecture for data optimization in raised bed farming." *International Journal of Safety & Security Engineering*, vol. 12, no. 6, pp. 767–776, 2022.

[215] A. P. M. Ramos, L. P. Osco, D. E. G. Furuya, W. N. Gonçalves, D. C. Santana, L. P. R. Teodoro, C. A. da Silva Junior, G. F. Capristo-Silva, J. Li, F. H. R. Baio *et al.*, "A random forest ranking approach to predict yield in maize with uav-based vegetation spectral indices," *Computers and Electronics in Agriculture*, vol. 178, p. 105791, 2020.

[216] T. D. Rapson and H. Dacres, "Analytical techniques for measuring nitrous oxide," *TrAC Trends in Analytical Chemistry*, vol. 54, pp. 65–74, 2014.

[217] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.

[218] N. Rathore and A. Rajavat, "Smart farming based on IoT-edge computing: Applying machine learning models for disease and irrigation water requirement prediction in

potato crop using containerized microservices," in *Precision Agriculture for Sustainability*, N. Khatri, A. K. Vyas, C. Iwendi, and P. Chatterjee, Eds. Apple Academic Press, 2024, pp. 399–424.

[219] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[220] N. G. Rezk, E. E.-D. Hemdan, A.-F. Attia, A. El-Sayed, and M. A. El-Rashidy, "An efficient IoT based smart farming system using machine learning algorithms," *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 773–797, 2021.

[221] D. R. Roberts, V. Bahn, S. Ciuti, M. S. Boyce, J. Elith, G. Guillera-Arroita, S. Hauenstein, J. J. Lahoz-Monfort, B. Schröder, W. Thuiller *et al.*, "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, no. 8, pp. 913–929, 2017.

[222] J. P. Rodríguez, A. I. Montoya-Munoz, C. Rodriguez-Pabon, J. Hoyos, and J. C. Corrales, "IoT-Agro: A smart farming system to colombian coffee farms," *Computers and Electronics in Agriculture*, vol. 190, p. 106442, 2021.

[223] G. Rondeaux, M. Steven, and F. Baret, "Optimization of soil-adjusted vegetation indices," *Remote sensing of environment*, vol. 55, no. 2, pp. 95–107, 1996.

[224] A. Roukh, F. N. Fote, S. A. Mahmoudi, and S. Mahmoudi, "Big data processing architecture for smart farming," *Procedia Computer Science*, vol. 177, pp. 78–85, 2020.

[225] G. Ruß and A. Brenning, "Data mining in precision agriculture: management of spatial information," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2010, pp. 350–359.

[226] M. M. Saad, M. T. R. Khan, M. A. Tariq, and D. Kim, "LSTM enabled artificial intelligent smart gardening system," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2020, pp. 136–141.

[227] M. Saberioon, M. Amin, A. Anuar, A. Gholizadeh, A. Wayayok, and S. Khairunniza-Bejo, "Assessment of rice leaf chlorophyll content using visible bands at different growth stages at both the leaf and canopy scale," *International Journal of Applied Earth Observation and Geoinformation*, vol. 32, pp. 35–45, 2014.

[228] D. Saha, B. Basso, and G. P. Robertson, "Machine learning improves predictions of agricultural nitrous oxide (N2O) emissions from intensively managed cropping systems," *Environmental Research Letters*, vol. 16, no. 2, p. 024004, 2021.

[229] V. Saiz-Rubio and F. Rovira-Más, "From smart farming towards agriculture 5.0: A review on crop data management," *Agronomy*, vol. 10, no. 2, p. 207, 2020.

[230] J. J. Salazar, L. Garland, J. Ochoa, and M. J. Pyrcz, "Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy," *Journal of Petroleum Science and Engineering*, vol. 209, p. 109885, 2022.

[231] S. Sapkota and D. R. Paudyal, "Growth monitoring and yield estimation of maize plant using unmanned aerial vehicle (UAV) in a hilly region," *Sensors*, vol. 23, no. 12, p. 5432, 2023.

[232] D. Saravia, W. Salazar, L. Valqui-Valqui, J. Quille-Mamani, R. Porras-Jorge, F.-A. Corredor, E. Barboza, H. V. Vásquez, A. V. Casas Diaz, and C. I. Arbizu, "Yield predictions of four hybrids of maize (zea mays) using multispectral images obtained from UAV in the coast of peru," *Agronomy*, vol. 12, no. 11, p. 2630, 2022.

[233] R. M. Schmidt, "Recurrent neural networks (RNNs): A gentle introduction and overview," *arXiv preprint arXiv:1912.05911*, 2019.

[234] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data," *Ecological Modelling*, vol. 406, pp. 109–120, 2019.

[235] S. Seilov, B. Pandey, A. Nurzhaubayev, D. Abildinov, A. Konyrkhanova, and B. Zhursinbek, "Federated learning based smart horticulture and smart storage of fruits using E-Nose, and blockchain: A proposed model," in *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*. IEEE, 2024, pp. 1–5.

[236] C. Z. Serele, Q. H. J. Gwyn, J. B. Boisvert, E. Pattey, N. McLaughlin, and G. Daoust, "Corn yield prediction with artificial neural network trained using airborne remote sensing and topographic data," in *IGARSS 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No. 00CH37120)*, vol. 1. IEEE, 2000, pp. 384–386.

[237] A. Sharma, D. Hazara, S. K. Gupta, R. Kushwaha, and D. Kumari, "Potato leaf disease classification using federated learning," in *International Conference on Recent Trends in Image Processing and Pattern Recognition*. Springer, 2023, pp. 191–201.

[238] S. Shekhar, Z. Jiang, R. Y. Ali, E. Eftelioglu, X. Tang, V. M. Gunturi, and X. Zhou, "Spatiotemporal data mining: A computational perspective," *ISPRS International Journal of Geo-Information*, vol. 4, no. 4, pp. 2306–2338, 2015.

[239] C. Shi and Y. Wang, "Non-parametric machine learning methods for interpolation of spatially varying non-stationary and non-gaussian geotechnical properties," *Geoscience Frontiers*, vol. 12, no. 1, pp. 339–350, 2021.

[240] A. M. Sibley, P. Grassini, N. E. Thomas, K. G. Cassman, and D. B. Lobell, "Testing remote sensing approaches for assessing yield variability among maize fields," *Agronomy Journal*, vol. 106, no. 1, pp. 24–32, 2014.

[241] A. P. Singh and P. Tomar, "Chapter 4 - AI and IoT capabilities: Standards, procedures, applications, and protocols," in *Artificial Intelligence to Solve Pervasive Internet of Things Issues*, G. Kaur, P. Tomar, and M. Tanque, Eds. Academic Press, 2021, pp. 67–83. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128185766000046

[242] R. K. Singh, R. Berkvens, and M. Weyn, "AgriFusion: An architecture for IoT and emerging technologies based on a precision agriculture survey," *IEEE Access*, vol. 9, pp. 136 253–136 283, 2021.

[243] I. Siniosoglou, K. Xouveroudis, V. Argyriou, T. Lagkas, D. Margounakis, A.-A. A. Boulogeorgos, and P. Sarigiannidis, "Applying federated learning on decentralized smart farming: A case study," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2023, pp. 1295–1300.

[244] D. Sirohi, N. Kumar, P. S. Rana, S. Tanwar, R. Iqbal, and M. Hijjii, "Federated learning for 6G-enabled secure communication systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 56, no. 10, pp. 11 297–11 389, 2023.

[245] S. Solomon, D. Qin, M. Manning, R. B. Alley, T. Berntsen, N. L. Bindoff, Z. Chen, A. Chidthaisong, J. M. Gregory, G. C. Hegerl *et al.*, "Technical summary," *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, KB Averyt, M. Tignor and HL Miller (eds.)]*, 2007.

[246] A. I. Sourav, A. W. R. Emanuel, and D. B. Setyohadi, "Smart system architecture design in the field of precision agriculture based on IoT in Bangladesh," *ICIC Express Lett*, vol. 16, no. 10, pp. 1111–1118, 2022.

[247] Sparkfun. Introduction to the Raspberry Pi GPIO and Physical Computing. Accessed February 7 2025. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/1/5/9/5/GPIO.png

[248] J. Spisic, J. Balen, D. Zagar, and V. Galić, "IoT based network model and sensor node prototype for precision agriculture application," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*. IEEE, 2022, pp. 1–8.

[249] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Pearson Education, 2006.

[250] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 3rd ed. Pearson Education, 2008.

[251] S. Sunoj, J. Cho, J. Guinness, J. van Aardt, K. J. Czymmek, and Q. M. Ketterings, "Corn grain yield prediction and mapping from unmanned aerial system (UAS) multispectral imagery," *Remote Sensing*, vol. 13, no. 19, p. 3948, 2021.

[252] R. Taki, C. Wagner-Riddle, G. Parkin, R. Gordon, and A. VanderZaag, "Comparison of two gap-filling techniques for nitrous oxide fluxes from agricultural soil," *Canadian Journal of Soil Science*, vol. 99, no. 1, pp. 12–24, 2018.

[253] K. Thonglek and P. Rattanatamrong, "Decentralized federated learning for agricultural plant diseases identification on edge devices," in *2023 18th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. IEEE, 2023, pp. 1–6.

[254] P. J. Thorburn, J. S. Biggs, K. Collins, and M. Probert, "Using the APSIM model to estimate nitrous oxide emissions from diverse australian sugarcane production systems," *Agriculture, Ecosystems & Environment*, vol. 136, no. 3-4, pp. 343–350, 2010.

[255] X. Tian, J. Cong, H. Wang, H. Zheng, Z. Wang, Y. Chu, Y. Wang, Y. Xue, Y. Yin, and Z. Cui, "Cropland nitrous oxide emissions exceed the emissions of RCP 2.6: A global spatial analysis," *Science of The Total Environment*, vol. 858, p. 159738, 2023.

[256] K. H. Tiessen, D. N. Flaten, P. R. Bullock, D. L. Burton, C. A. Grant, and R. E. Karamanos, "Transformation of fall-banded urea: Application date, landscape position, and fertilizer additive effects," *Agronomy Journal*, vol. 98, no. 6, pp. 1460–1470, 2006.

[257] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[258] L. Ting, M. Khan, A. Sharma, and M. D. Ansari, "A secure framework for IoT-based smart climate agriculture system: Toward blockchain and edge computing," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 221–236, 2022.

[259] R. G. Tiwari, H. Maheshwari, A. K. Agarwal, N. K. Trivedi, and V. Gautam, "Federated learning for mango ripeness classification: Evaluating pre-trained deep learning frameworks," in *2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART)*. IEEE, 2023, pp. 23–28.

[260] J. Tong, Y. Zou, Z. Zeng, and F. Deng, "Detection technology of several pests and diseases in orchard based on federated learning and improved faster R-CNN," in *CAIBDA 2022; 2nd International Conference on Artificial Intelligence, Big Data and Algorithms*. VDE, 2022, pp. 1–4.

[261] B. Trawiński, M. Smętek, Z. Telec, and T. Lasota, "Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms," *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 4, pp. 867–881, 2012.

[262] R. Tripathy, J. Mandala, S. R. Pappu, and G. K. D. Gopisetty, "Optimization based rice leaf disease classification in federated learning," *Multimedia Tools and Applications*, vol. 83, no. 29, pp. 72 491–72 517, 2024.

[263] C. J. Tucker, "Red and photographic infrared linear combinations for monitoring vegetation," *Remote sensing of Environment*, vol. 8, no. 2, pp. 127–150, 1979.

[264] M. A. Uddin, M. Ayaz, A. Mansour, E.-H. M. Aggoune, Z. Sharif, and I. Razzak, "Cloud-connected flying edge computing for smart agriculture," *Peer-to-Peer Networking and Applications*, vol. 14, no. 6, pp. 3405–3415, 2021.

[265] V. Udutalapally, S. P. Mohanty, V. Pallagani, and V. Khandelwal, "sCrop: A novel device for sustainable automatic disease prediction, crop selection, and irrigation in internet-of-agro-things for smart agriculture," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 525–17 538, 2020.

[266] Y. Uno, S. Prasher, R. Lacroix, P. Goel, Y. Karimi, A. Viau, and R. Patel, "Artificial neural networks to predict corn yield from compact airborne spectrographic imager data," *Computers and electronics in agriculture*, vol. 47, no. 2, pp. 149–161, 2005.

[267] P. C. van Oorschot, *Computer Security and the Internet: Tools and Jewels*, 1st ed. Springer, 2020.

[268] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha *et al.*, "FarmBeats: An IoT platform for data-driven agriculture," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 515–529.

[269] C. Verdouw, H. Sundmaeker, B. Tekinerdogan, D. Conzon, and T. Montanaro, "Architecture framework of IoT-based food and farm systems: A multiple case study," *Computers and Electronics in Agriculture*, vol. 165, p. 104939, 2019.

[270] N. Villa-Vialaneix, M. Follador, M. Ratto, and A. Leip, "A comparison of eight metamodeling techniques for the simulation of N2O fluxes and N leaching from corn crops," *Environmental Modelling & Software*, vol. 34, pp. 51–66, 2012.

[271] D. Vimalajeewa, C. Kulatunga, D. P. Berry, and S. Balasubramaniam, "A service-based joint model used for distributed learning: Application for smart agriculture," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 838–854, 2021.

[272] C. N. Vong, L. S. Conway, J. Zhou, N. R. Kitchen, and K. A. Sudduth, "Corn emergence uniformity at different planting depths and yield estimation using uav imagery," in *2022 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers, 2022, p. 1.

[273] A. M.-C. Wadoux, G. B. Heuvelink, S. De Bruin, and D. J. Brus, "Spatial cross-validation is not the right way to evaluate map accuracy," *Ecological Modelling*, vol. 457, p. 109692, 2021.

[274] B. Whelan, A. McBratney, B. Minasny *et al.*, "Vesper 1.5–spatial prediction software for precision agriculture," in *Precision Agriculture, Proc. 6th Int. Conf. on Precision Agriculture, ASA/CSSA/SSSA, Madison, WI, USA*, vol. 179. Citeseer, 2002.

[275] B. Whelan, A. McBratney, and R. Viscarra Rossel, "Spatial prediction for precision agriculture," in *Proceedings of the Third International Conference on Precision Agriculture*. Wiley Online Library, 1996, pp. 331–342.

[276] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen, "Color indices for weed identification under various soil, residue, and lighting conditions," *Transactions of the ASAE*, vol. 38, no. 1, pp. 259–269, 1995.

[277] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big data in smart farming–a review," *Agricultural systems*, vol. 153, pp. 69–80, 2017.

[278] G. Wu, N. D. Miller, N. De Leon, S. M. Kaeppler, and E. P. Spalding, "Predicting zea mays flowering time, yield, and kernel dimensions by analyzing aerial images," *Frontiers in Plant Science*, vol. 10, p. 1251, 2019.

[279] J. Wu, W. Bao, E. Ainsworth, and J. He, "Personalized federated learning with parameter propagation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2594–2605.

[280] Q. Wu, J. Wang, Y. He, Y. Liu, and Q. Jiang, "Quantitative assessment and mitigation strategies of greenhouse gas emissions from rice fields in China: A data-driven approach based on machine learning and statistical modeling," *Computers and Electronics in Agriculture*, vol. 210, p. 107929, 2023.

[281] Q. Xie, J. Dash, W. Huang, D. Peng, Q. Qin, H. Mortimer, R. Casa, S. Pignatti, G. Laneve, S. Pascucci, Y. Dong, and H. Ye, "Vegetation indices combining the red and red-edge spectral information for leaf area index retrieval," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, 02 2018.

[282] B. Yang, W. Zhu, E. E. Rezaei, J. Li, Z. Sun, and J. Zhang, "The optimal phenological phase of maize for yield prediction with high-frequency UAV remote sensing," *Remote Sensing*, vol. 14, no. 7, p. 1559, 2022.

[283] C. Yang, J. H. Everitt, J. M. Bradford, and D. Murden, "Airborne hyperspectral imagery and yield monitor data for mapping cotton yield variability," *Precision Agriculture*, vol. 5, no. 5, pp. 445–461, 2004.

[284] Q. Yang, L. Shi, J. Han, Y. Zha, and P. Zhu, "Deep convolutional neural networks for rice grain yield estimation at the ripening stage using UAV-based remotely sensed images," *Field Crops Research*, vol. 235, pp. 142–153, 2019.

[285] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[286] W. Yang, T. Nigon, Z. Hao, G. D. Paiao, F. G. Fernández, D. Mulla, and C. Yang, "Estimation of corn yield based on hyperspectral imagery and convolutional neural network," *Computers and Electronics in Agriculture*, vol. 184, p. 106092, 2021.

[287] Z. M. Yaseen, "The next generation of soil and water bodies heavy metals prediction and detection: New expert system based edge cloud server and federated learning technology," *Environmental Pollution*, vol. 313, p. 120081, 2022.

[288] P.-Y. Yin, C.-C. Tsai, R.-F. Day, C.-Y. Tung, and B. Bhanu, "Ensemble learning of model hyperparameters and spatiotemporal data for calibration of low-cost PM2.5 sensors," *Mathematical Biosciences and Engineering*, vol. 16, no. 6, pp. 6858–6873, 2019.

[289] Y. Yin, Z. Wang, X. Tian, Y. Wang, J. Cong, and Z. Cui, "Evaluation of variation in background nitrous oxide emissions: A new global synthesis integrating the impacts of climate, soil, and management conditions," *Global Change Biology*, vol. 28, no. 2, pp. 480–492, 2022.

[290] J. Yoon, A. Kumar, J. Jang, J. Kim, and B. J. Choi, "Privacy-preserving pest detection using personalized federated learning," in *International Conference on Agriculture-Centric Computation*. Springer, 2023, pp. 58–70.

[291] J. You, X. Li, M. Low, D. Lobell, and S. Ermon, "Deep gaussian process for crop yield prediction based on remote sensing data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[292] C. Yu, S. Shen, K. Zhang, H. Zhao, and Y. Shi, "Energy-aware device scheduling for joint federated learning in edge-assisted internet of agriculture things," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1140–1145.

[293] K. R. Žalik and M. Žalik, "A review of federated learning in agriculture," *Sensors*, vol. 23, no. 23, p. 9566, 2023.

[294] L. Zeng, G. Peng, R. Meng, J. Man, W. Li, B. Xu, Z. Lv, and R. Sun, "Wheat yield prediction based on unmanned aerial vehicles-collected red–green–blue imagery," *Remote Sensing*, vol. 13, no. 15, p. 2937, 2021.

[295] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[296] L. Zhang, H. Zhang, Y. Niu, and W. Han, "Mapping maize water stress based on UAV multispectral remote sensing," *Remote Sensing*, vol. 11, no. 6, p. 605, 2019.

[297] M. Zhang, J. Zhou, K. A. Sudduth, and N. R. Kitchen, "Estimation of maize yield and effects of variable-rate nitrogen application using UAV-based RGB imagery," *biosystems engineering*, vol. 189, pp. 24–35, 2020.

[298] Q. Zhang, X. Zhao, Y. Han, F. Yang, S. Pan, Z. Liu, K. Wang, and C. Zhao, "Maize yield prediction using federated random forest," *Computers and Electronics in Agriculture*, vol. 210, p. 107930, 2023.

[299] R. Zhang and X. Li, "Edge computing driven data sensing strategy in the entire crop lifecycle for smart agriculture," *Sensors*, vol. 21, no. 22, p. 7502, 2021.

[300] M. Zhao, P. Olsen, and R. Chandra, "Seeing through clouds in satellite images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[301] X. Zhou, Y. Kono, A. Win, T. Matsui, and T. S. Tanaka, "Predicting within-field variability in grain yield and protein content of winter wheat using uav-based multispectral imagery and machine learning approaches," *Plant Production Science*, vol. 24, no. 2, pp. 137–151, 2021.

[302] Q. Zhuang, Y. Lu, and M. Chen, "An inventory of global N2O emissions from the soils of natural terrestrial ecosystems," *Atmospheric Environment*, vol. 47, pp. 66–75, 2012.