UNIVERSITY OF OTTAWA

PH.D THESIS

---

# Analysis of Complex Dynamical Systems by Combining Recurrent Neural Networks and Mechanistic Models

---


*Author:*
Ci LIN

*Supervisors:*
Dr. Tet YEAP
Dr. Iluju KIRINGA


*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
School of Electrical Engineering and Computer Science
Faculty of Engineering

# Declaration of Authorship

I, Ci Lin, declare that this thesis titled, “Analysis of Complex Dynamical Systems by Combining Recurrent Neural Networks and Mechanistic Models” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Ci Lin

---

Date: November 20, 2024

---

# *Abstract*

The aim of this study is to analyze the time series data in an innovative manner, which combined the deep learning model with the mechanistic model to form a hybrid model. In order to demonstrate the feasibility and theoretical background of the hybrid model, the Lorenz System is used as an example to explore the underlying mathematical mechanism. Through several simple exams, it is observed that the Bidirectional Long Short Term Memory (BDLSTM) in the hybrid model could be trained to learn the dynamical behavior of the complex dynamical system properly and the mechanistic model in the hybrid model could adapt to different situations flexibly.

In practical exploration, a real hybrid model, namely the Argiculture-informed Neural Network (AINN) model, that consists of deep neural networks (DNN), including but not limited to: Long Short Term Memory (LSTM) , Convolutional Neural Network (CNN), and Transformer, and Dynamic Land Ecosystem Model (DLEM), are proposed to predict nitrous oxide emissions in agricultural fields. The model is trained using data collected from smart farm, named Area X.O, in Ottawa, Ontario, Canada, during the 2021 growing season and tested using data from the 2022 growing season. During the data preprocessing stage, we utilized the Robust Scaler (or MinMax Scaler) to scale the data into a narrow range. Our statistical analysis revealed that temperature and humidity are closely related and share similar time series patterns, suggesting that they contain the same information for predicting nitrous oxide ($N_2O$) emissions.

To evaluate the model's performance, several metrics were used, including mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and coefficient of determination ($R^2$). Our results indicate that the AINN outperforms the pure DNN and DLEM in both the training and testing datasets, because in the hybrid model, the Recurrent Neural Network (RNN) part could catch the dynamical behavior of the emission of $N_2O$, and the DLEM part could regulate the training path and lead the entire model to converge in a limited number of basins of attraction.

Additionally, since it is costly to collected the data from the field, it is better for us to do time series data augmentation using Generative Adversarial Network (GAN), with the aim of closely matching the original data distribution while also preserving the dynamic behavior of the original data. However, even state-of-the-art GAN models like TimeGAN fall short in preserving the temporal dynamics present in the original time series due to the absence of first-order difference information. To address this limitation, this study proposes a novel process for generating multivariate

time series data. The proposed process comprises four essential modules: a) the GAN module for generating multivariate time series data, b) the sampling module for preserving the first-order difference distribution, c) the smoothing module for refining the generated data, and d) an evaluation module using the Kolmogorov-Smirnov Test (KS-test) and Hilbert-Schmidt Independence Criterion (HSIC), along with other metrics to test the synthetic time series data. This comprehensive approach ensures that the synthetic time series data maintains both the distribution and the dynamic behavior of the original data.

With the advent of quantum computing, we transitioned from the conventional LSTM to quantum LSTM and formulated the Quantum Long Short Term Memory-Dynamical Land Ecosystem Model (QLSTM-DLEM) model, showcasing enhanced generalization capability and stability. Experimental results indicated that QLSTM-DLEM achieved comparable performance to Long Short Term Memory-Dynamical Land Ecosystem Model (LSTM-DLEM) using several quantum bits.

# *Acknowledgements*

I would like to express my heartfelt gratitude to all those who have supported me throughout my Ph.D. journey. First and foremost, I would like to thank my supervisors, Dr. Tet Yeap and Dr. Iluju Kiringa, for their invaluable guidance, expertise, and unwavering support. Your encouragement and constructive feedback have inspired me to pursue excellence in my research and have been instrumental in shaping my academic development.

I am also deeply grateful to my committee members, Dr. Raymond Ng, Dr. Qi-Jun Zhang, Dr. Amiya Nayak, and Dr. Paula Bronco, for their insightful suggestions and for challenging me to think critically about my work. Your diverse perspectives have significantly enriched my research.

A special thanks to my colleagues Patrick Killeen, Futong Li, and Yuri for fostering a collaborative and stimulating environment in the Smart IoT Lab. The discussions, camaraderie, and support we shared made this journey not only productive but also enjoyable.

I would like to acknowledge the Ontario Research Fund for their financial support throughout my studies, as well as the admission scholarship provided by the University of Ottawa, which enabled me to pursue my Ph.D. I am also grateful for the experience gained as a research assistant and teaching assistant over the past six years, which has greatly contributed to my growth as a scholar and educator.

Lastly, I would like to extend my deepest appreciation to my family for their unwavering support and encouragement. Your belief in my abilities and your sacrifices have been the foundation of my academic success.

This thesis is dedicated to all those who have been a part of my academic journey. Thank you for your contributions, support, and encouragement along the way.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **HNN** | **H**opfield **N**eural **N**etwork |
| **MHN** | **M**odern **H**opfield **N**etwork |
| **RNN** | **R**ecurrent **N**eural **N**etwork |
| **ANN** | **A**rtificial **N**eural **N**etwork |
| **CNN** | **C**onvolutional **N**eural **N**etwork |
| **WGAN** | **W**asserstein **G**enerative **A**dversarial **N**etwork |
| **LSTM** | **L**ong **S**hort-**T**erm **M**emory |
| **BDLSTM** | **Bid**irectional **L**ong **S**hort-**T**erm **M**emory |
| **QLSTM** | **Q**uantum **L**ong **S**hort-**T**erm **M**emory |
| **GRU** | **G**ated **R**ecurrent **U**nit |
| **TDNN** | **T**ime **D**elay **N**eural **N**etwork |
| **AR** | **A**uto-**R**egressive |
| **ARMA** | **A**uto-**R**egressive **M**oving **A**verage |
| **ARIMA** | **A**uto-**R**egressive **I**ntegrated **M**oving **A**verage |
| **SARIMA** | **S**easonal **A**uto-**R**egressive **I**ntegrated **M**oving **A**verage |
| **MA** | **M**oving **A**verage |
| **HSIC** | **H**ilbert–**S**chmidt **I**ndependence **C**riterion |
| **SPSA** | **S**imultaneously **P**roximity **S**tochastic **A**pproximation |
| **DLEM** | **D**ynamic **L**and **E**cosystem **M**odel |
| **NMIP** | **N**itrous Oxide **M**odel **I**ntercomparison **P**roject |
| **MLP** | **M**ulti-**L**ayer **P**erceptrons |
| **TAR** | **T**hreshold **A**uto-**R**egressive |
| **MSM** | **M**arkov **S**witching **M**odels |
| **MCMC** | **M**arkov **C**hain **M**onte **C**arlo |
| **LASSO** | **L**east **A**bsolute **S**hrinkage and **S**election **O**perator |
| **DBN** | **D**eep **B**elief **N**etwork |
| **PCA** | **P**rincipal **C**omponent **A**nalysis |
| **ACF** | **A**uto-**C**orreletion **F**unction |
| **PACF** | **P**artial **A**uto-**C**orreletion **F**unction |

# Chapter 1

# Introduction

The recent unprecedented success of recurrent neural network models such as LSTM, GRU, and Transformer in video processing, natural language processing, and pattern recognition has made them popular as data-driven models. Their applications have expanded into non-traditional areas like earth and environmental sciences, where they are currently challenging the dominant role of knowledge-driven, process-based models in these fields.

## 1.1 Motivation and Importance

It is a challenging task to conduct time series analysis. In our efforts to develop time series analysis methodologies, we face several hurdles in terms of prediction accuracy and computation efficiency that need to be addressed within our architecture, as highlighted below:

1. Assessment of Uncertainty. For time series analysis, it is necessary to take into account the uncertainty originating not only from the noise in the observed measurements, but also from the accuracy of our model.

2. Vulnerability to Initial Conditions. In time series analysis, even a minor alteration in the initial conditions can result in significant differences over the long term. Therefore, it is crucial for the model to be capable of adjusting its analysis over time.

3. Variability in Dataset Size. The size of time series data can vary greatly, with some measurements yielding only a few data points, while others may have thousands. The time series model must be able to handle both scenarios efficiently and in a generalizable manner.

4. Difficulty in Detecting Long-Term, Non-Linear Relationships. Time series models must be able to uncover long-term temporal dependencies in the data.

Additionally, traditional methodologies often struggle with analyzing the non-linear relationships between inputs and observed time series measurements.

5. Handling Multi-Dimensional Time Series Data. In even a simple system, a single measurement can be influenced by multiple random variables. These variables may be interrelated or have causal relationships, but for the sake of simplicity, researchers often assume that they are independent and identically distributed. This assumption leads to a loss of cross-correlation information among the random variables and can result in an incorrect analysis.

As demonstrated in Table 1.1, RNN models are known to be capable of effectively capturing long-term and non-linear relationships when trained properly. Additionally, they are well-suited for handling high-dimensional problems. On the other hand, mechanistic modeling is indifferent to the size of the time series dataset and the assessment of uncertainty. Given these complementary strengths, it makes sense to explore the combination of these two methodologies into a single model that can address all the challenges outlined above.

TABLE 1.1: The pros and cons of RNN Modeling and Mechanistic Modeling

| RNN Modeling | Mechanistic Modeling |
|---|---|
| seeks to identify statistical relationships and correlations between inputs and outputs, but may have difficulties in accurately determining the uncertainty of its analysis. | aims to establish a causal relationship between inputs and outputs and is not concerned with assessing uncertainty. |
| effective in handling high-dimensional problems and can process large amounts of data. | struggle in handling high-dimensional problems and can be susceptible to model misspecification. |
| requires large datasets and can be computationally demanding, requiring significant processing power and memory resources. | capable of handling small datasets and is simple to implement, with less computational requirements. |
| limited to making predictions based on patterns found within the given data. | once validated, can be used as a predictive tool in situations where conducting experiments is difficult or costly. |

## 1.2 Research Problem

### 1.2.1 Time Series Analysis and its Application in Agriculture

Time series data can be found in a wide range of scientific and economic fields, including daily stock market quotations, monthly unemployment figures, population birthrates, school enrollments, blood pressure measurements over time, and weather time series, among others.

The correlations among sequential data points in time series data present a significant challenge for conventional statistical methods, which rely on the assumption of independent and identically distributed observations. To address this challenge, time series analysis provides a systematic way to examine the correlations between adjacent data points.The ultimate objective of time series analysis is to create mathematical or statistical models that can detect underlying patterns, dynamics, or mechanisms from sample data, and make accurate predictions or estimates for unseen data.

Applying time series analysis to predict nitrous oxide ($N_2O$) emissions in agriculture is a crucial step in understanding and mitigating greenhouse gas impacts. Time series models are particularly effective in capturing the temporal dependencies and dynamic interactions among the factors influencing emissions, such as soil moisture, soil temperature, nitrogen content, and agricultural practices. By leveraging historical data and real-time monitoring, deep learning models like LSTM and GRU can provide accurate and timely predictions. These models not only account for seasonal and cyclical patterns in emissions but also adapt to anomalies caused by extreme weather events or changes in farming practices.

### 1.2.2 Limitations of Current Solutions

Traditionally, researchers have used theoretical autocorrelation and cross-correlation functions to describe the properties of time series data in sampled form. With these methods, researchers can only estimate the mean, autocovariance, and autocorrelation functions from the sampled points $x_1, x_2, \ldots, x_n$. However, since it is impossible to have sufficient independent and identically distributed (i.i.d.) copies (usually only one realization is available) of $x_t$ for estimating the covariance and correlation functions, it raises questions in classical statistics. To overcome this challenge, it is common to assume that the time series data is stationary if only one observation is available for estimating the population means and covariance functions.

From a statistical perspective, the AR and ARMA models are the conventional models for stationary time series analysis, while for non-stationary time series, the

ARIMA model is widely accepted for modeling purposes. ARMA is a linear model for time series analysis, but most observations are non-linear in nature. Linear models may provide good approximations for some applications, but in most cases, a non-linear model can provide more insight into the time series data than a linear model.

### 1.2.3 Key Idea

When dealing with multivariate time series data, conventional approaches from the statistical field, such as AR, ARMA, ARIMA, and others, are typically employed. However, these traditional methods necessitate time series that are free of seasonal or temporary trends, requiring the model to be trained on a stationary time series. This limitation hinders practical applications of traditional approaches, prompting the exploration of alternative solutions for time series analysis.

The most intuitive approach is to identify a mechanistic model that analytically and comprehensively describes the system's behavior when the underlying mechanism behind the time series data is fully understood. However, this is not always the case, and empirical models are often established based on partial knowledge and available data. In such scenarios, the system typically becomes a combination of theoretical and empirical models.

Recent advancements in deep learning have popularized the use of RNN models, such as Transformer, LSTM, or GRU, for time series analysis. While these RNN models effectively model time series data, they often lack interpretability and are considered black box models. In this research, RNN models and mechanistic models are combined as a dynamical system to analyze time series data generated by a time-varying system. The goal is to find a parsimonious solution that accurately represents the underlying patterns and dynamics of the data while providing a simple and interpretable explanation of the results.

Moreover, with the emergence of quantum computing, the traditional RNN is replaced by its quantum counterpart, aiming to achieve better generalization and stability performance. In the field of quantum neural networks, utilizing several quantum bits, state-of-the-art regression performance can be attained.

To further understand the learning mechanism of RNN, additional neural networks are developed and compared, including MLP, TDNN, CNN, and transformer. The objective is to select the most appropriate model for our specific application.

### 1.2.4 Thesis Statement

We accurately predict N2O emissions in agriculture by integrating deep learning and mechanistic modeling to develop a hybrid model known as the Agricultural-informed Neural Network (AINN), including but not limited to LSTM-DLEM, CNN-DLEM, and Transformer-DLEM.

## 1.3 Objectives

The main objectives of this thesis are as follows:

1. To analyze time series data, especially for prediction of $N_2O$ emission from farming, using a combination of the recurrent neural network and the mechanistic model.

2. To generate time series data using WGAN and MCMC method, which could preserve the temporal dynamic in the original dataset.

3. To develop a quantum model to analyze the time series data.

The focus of this research is to explore the potential of combining the RNN with the Mechanistic Model as a hybrid dynamical system. The models being studied include: the Lorenz System (a simple case to demonstrate the feasibility of hybrid model, the hybrid combination model (the LSTM, CNN, Transformer, QLSTM and the DLEM) to predict nitrous oxide ($N_2O$) emissions, and the WGAN and the MCMC models to generate time series data.

Another key objective of this research is to apply RNNs or machine learning approaches to tackle time series data. The AINNs (CNN-DLEM, LSTM-DLEM, Transformer-DLEM) are developed to predict $N_2O$ emissions and to uncover the underlying mechanism of $N_2O$ production in the natural environment.

## 1.4 Contributions

The main contributions of this research are as follows:

- Feasibility of Hybrid Model Integration: This study explores the potential of combining a neural network model (a black-box model) with a mechanistic model. The hybrid model effectively learns the dynamical aspects of the system and adapts flexibly to varying conditions.

- Improved Prediction of $N_2O$ Emissions: Development of the Agriculture-informed Neural Network (AINN) combines neural networks (such as LSTM, CNN, and Transformer) with a mechanistic model, the Dynamic Land Ecosystem Model (DLEM). This integrated model enhances prediction accuracy compared to traditional single-model approaches by capturing both data-driven patterns and underlying physical dynamics. The study also develops a quantum LSTM variant alongside the DLEM, showing comparable performance to the classical LSTM-DLEM model but with greater generation capacity.

- Enhanced Data Augmentation for Agricultural Time Series: Given the challenges in collecting large-scale agricultural data, this work introduces an innovative data augmentation technique that combines WGAN (Wasserstein GAN) and MCMC (Markov Chain Monte Carlo). This approach generates realistic time series data, maintaining the original temporal dynamics, and offers more robust datasets for training models without extensive field data collection.

### 1.4.1 List of Publications

During my PhD studies, I wrote and published several papers, as shown below:

**Published Paper**

1. Ci Lin, Tet Yeap, and Iluju Kiringa. "Stacked Bidirectional LSTM for Predicting Emission of Nitrous Oxide". In: 35th Canadian Conference on Artificial Intelligence, Toronto, Ontario, Canada, May 30 - June 3, 2022.

2. Ci Lin, Futong Li, Patrick Killeen, Tet Hin Yeap, Iluju Kiringa. "Agriculture-informed Neural Networks for Predicting Nitrous Oxide Emissions", ACM transaction on Internet of Things.

3. Ci Lin, Tet Yeap, Iluju Kiringa (2023). "On the Basin of Attraction and Capacity of Restricted Hopfield Network as an Auto-Associative Memory", 2023 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC).

4. Ci Lin, Tet Yeap, Iluju Kiringa (2024). "Subspace Rotation Algorithm for Training Restricted Hopfield Network", 36th International Conference on Tools with Artificial Intelligence (ICTAI 2024).

**Submitted Papers**

1. Ci Lin, Patrick Killeen, Futong Li, Tet Yeap, Iluju Kiringa. "Preserving Temporal Dynamics in Synthetic Multivariate Time Series Using Generative Neural Networks and Monte Carlo Markov Chain", Submitted to IEEE transaction on Artificial Intelligence.

2. Ci Lin, Tet Yeap, Iluju Kiringa. "Implementing Restricted Hopfield Network as Robust Auto-associative Memory Using Subspace Rotation Algorithm", Submitted to IEEE transaction on Knowledge and Data Engineering.

**Preprinted Papers**

1. Ci Lin, Patrick Killeen, Futong Li, Tet Yeap, and Iluju Kiringa. "A Unified QLSTM-DLEM Model for Predicting Agricultural $N_2O$ Emissions", Authorea Preprints (2024).

2. Ci Lin, Tet Yeap, and Iluju Kiringa, "Improving Adversarial Robustness of Conjugate Neural Networks with Guided Diversity", Authorea Preprints (2024).

3. Ci Lin, Tet Yeap, and Iluju Kiringa. "Beyond Gradient: Subspace Rotation Algorithm", Authorea Preprints (2024).

## 1.5 Methodology

This research investigated the potential of combining Recurrent Neural Networks (RNNs) with Mechanistic Models to form a hybrid model and achieve improved performance compared to either model individually. For Theoretical exploration, the Lorenz system was used to generate an sample data and a manually designed mechanistic model, which consisted of linear part and nonlinear part, was used to evaluate the validity and feasibility of hybrid model. While, in practice, a hybrid model, which consisted of BDLSTM and DLEM, was proposed to predict the emission of $N_2O$ in the agricultural field.

In this exploration, the original dataset, which was a multivariate time series and included the data of temperature, moisture, precipitation, and flux of $N_2O$, was collected using LI-COR soil-gas measurement equipment. In order to study whether different agricultural practices have varied impact on the emission of $N_2O$, the LI-COR sensors were placed in a variety of places with different approaches to spread the fertilizer. A variety of variables, including temperature, moisture precipitation,

and flux of $N_2O$, were measured every 30 minutes during the 2021 and 2022 growing seasons in Area X.O, Ottawa, Ontario, Canada.

In order to analyze the multivariate time series data collected by the LI-COR sensor from the agricultural field, firstly, the traditional approaches, such as ACF and PACF, were utilized to analyze the data of interest. After obtaining the general information of the series data and before feeding the data into the hybrid model, the time series data needed to be pre-processed using a scaler or other data transformation technique. Finally, the hybrid models were used to learn the dynamical pattern in the time series data and predict the emission of $N_2O$ on the unseen dataset.

Since the time series dataset was difficult to collect once a year, we could only collect a time series of one growing season. Therefore, it is necessary for us to augment the data and reduce the generalized error of the hybrid model being trained. In this study, the WGAN-GP and MCMC methods were combined to augment the multivariate time series data. The WGAN-GP was used to generate the time series vector source and the MCMC approach was used to sample the time series source and generate time series dataset. Finally, the generated time series was needed to pass into a smoothing filter. Hilbert–Schmidt Independence Criterion (HSIC) was used to measure the quality of the time series data in comparison to its original data.

The simulation and experiment results showed that the hybrid model could perform better in a variety of cases than an individual deep learning model or a pure mechanistic model. Additionally, for the data augmentation approach mentioned in this section, the data generated by this technique were fed into the hybrid model and achieved similar performance in comparison with the real data.

## 1.6 Thesis Outline

The rest of this thesis is structured as follows. In Chapter 2, some important concepts of time series analysis, Bayesian statistics, and dynamical systems are introduced. Concurrently, several recurrent neural networks, which are usually used to address time series data, and one mechanistic model (Dynamic Land Ecosystem Model (DLEM)), are reviewed.

In Chapter 3, the time series analysis is formulated, and the conceptual design of the models is proposed along with two applications. In Chapter 4, experiments are conducted to test the performance of the AINN. In Chapter 5, a new process is used to generate the multivariate time series dataset for predicting the emission of $N_2O$. In Chapter 6, the quantum neural network is introduced and combined with DLEM to form a new hybrid model for predicting the agricultural emission of $N_2O$. In Chapter

7, it summarizes the contributions of this thesis proposal and presents some research directions for future work.

# Chapter 2

# Literature Reviews

RNN are a type of ANN specifically designed to handle sequential data. They are ideal for tasks that require modeling the temporal dependencies between time steps in a time series such as classification, sequence generation, and forecasting [1, 2].

Mechanistic Models are mathematical models that describe the underlying principles and mechanisms of a system based on scientific knowledge and first principles [3]. They are widely used in fields such as physics, engineering, and biology to model physical systems such as electrical circuits, biochemical reactions, and control systems [4].

In this research, we aim to combine both RNNs and Mechanistic Models as a Dynamical System for Time Series Analysis. A Dynamical System is a mathematical model that describes how a system changes over time in response to inputs and rules [5]. Time Series Analysis is a method of analyzing and modeling time-based data to uncover patterns, trends, and relationships [6]. With the emergence of data-driven Dynamical Systems, these two approaches are closely related to each other [7, 8].

The relationship between Time Series Analysis and Dynamical Systems is also bidirectional, as the insights and knowledge gained from Time Series Analysis can inform the development and refinement of Dynamical System models and vice versa [9]. This close relationship between the two fields allows for a more complete and accurate understanding of the behavior of systems over time.

In this chapter, we review and analyze significant contributions in the fields of RNNs, Mechanistic Models, Dynamical Systems, and Time Series Analysis. We start by defining the specific characteristics of our research problem and highlighting the limitations of previous studies on discovering accurate behavior of complex time-evolving systems.

## 2.1 Time Series Analysis

Time series data is a collection of data points that are measured at regular intervals over a period of time. The objective of time series analysis is to understand the underlying patterns and structures within the data, such as trends, seasonal variations, and autocorrelation. To produce accurate and reliable results, a substantial amount of data must be gathered.

Time series data is widely utilized in a range of fields, including language processing, electrical signal analysis, traffic analysis, weather forecasting, and emission analysis, among others [10, 11, 12, 13]. Time series analysis is used to develop mathematical, statistical or even mechanistic models that can accurately describe the behavior of the data and make predictions about future trends.

Historically, statistical models such as AR [14], ARMA [15], and ARIMA [16] have been used for time series analysis. With the advent of deep learning, neural network models have emerged as a powerful tool for time series analysis, including LSTM [17, 18], RHN [19, 20, 21], GNU [22].

The application of these models includes:

- Classification: This involves identifying and categorizing the data [23].
- Curve fitting: This involves plotting the data along a curve to understand the relationships between variables within the data [24].
- Descriptive analysis: This identifies patterns in time series data such as trends, cycles, or seasonal variations [25].
- Explanative analysis: This involves attempting to understand the data and the relationships within it as well as cause and effect [26].
- Exploratory analysis: This highlights the main characteristics of the time series data, typically in a visual format [27].
- Forecasting: This predicts future data based on historical trends, using historical data as a model for future predictions [28].
- Intervention analysis: This studies how events can impact the data [29].
- Segmentation: This divides the data into segments to reveal the underlying properties of the information source [30].

### 2.1.1 Challenges in Current Time Series Analysis

Time series analysis is a widely used method for analyzing data collected over time. Despite its widespread use, there are several challenges associated with time series analysis, including:

1. Non-stationarity: Many time series exhibit non-stationary behavior, meaning that their statistical properties change over time. This can make it difficult to accurately model the time series and make predictions with traditional statistical models.

2. Seasonality: Many time series exhibit regular patterns of behavior that repeat over time such as daily, weekly, or yearly patterns. This can make it challenging to separate the seasonal behavior from the underlying trends in the data.

3. Noisy data: Time series data can be noisy, meaning that it may contain outliers or other unexpected observations. This can make it difficult to identify patterns and trends in the data.

4. Model selection: There are many different models that can be used for time series analysis, such as ARIMA, SARIMA, exponential smoothing, and neural networks. Choosing the right model for a particular time series can be challenging and require a lot of experimentation.

5. High-dimensional data: Time series data can often have a large number of dimensions such as multiple time-based variables or multiple observations at each time step. This can make it difficult to identify the most relevant patterns and trends in the data.

6. Data availability: In some cases, time series data may not be available or may be limited in terms of frequency or quality. This can make it challenging to develop accurate models and make predictions.

### 2.1.2 Linear Time Series Model

The following section provides an overview of the three most widely used linear models for time series analysis: linear AR, MA, and ARMA. Although these models have different designs for the conditional mean ($f$), they all assume a constant conditional variance ($g^2$).

**Autoregressive (AR) Model - AR(p)**

The linear autoregressive (AR) model was first introduced in a seminar paper in 1927 [31]. It is a regression model in which the predictors are the past $p$ values of the response variable $Z_t$, with $p$ being referred to as the AR process order. The AR process can be described as follows:

$$Z_t = \phi_0 + \phi_1 * Z_{t-1} + \phi_2 * Z_{t-2} + ... + \phi_p * Z_{t-p} + \sigma * e_t \tag{2.1}$$

where $\phi_0$ to $\phi_p$ are the coefficients of the AR process, and $e_t$ is independent identity distributed white noise processes.

The AR model represents the mean $f$ of $Z_t$, given its past values, as a linear combination of its $p$ previous values, while the variance remains constant and equal to $\sigma^2$. It is widely known that an AR process is considered to be stationary only when the roots of its characteristic equation $1 - \phi_1 y - \phi_2 y^2 - \cdots - \phi_p y^p = 0$ are within the unit circle, which means $|y_i| < 1, i = 1, \ldots, p$.

**Moving Average (MA) Model - MA(q)**

Moving Average models simulate the behavior of a time-varying signal by considering the past average of $q$ error terms, also known as the residuals. This MA process is depicted as:

$$Z_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i} + \epsilon_t \tag{2.2}$$

where $\epsilon_t$ are error terms that are independent identity distributed $(0, \sigma^2)$, q is the MA process order and $(\alpha_0, \alpha_1, \ldots, \alpha_q)$ are the coefficients of MA process.

In the MA model, the conditional mean $f$ of current observation $Z_t$, given its past values, is a linear function of the past $q$ prediction errors, whereas its conditional variance $g^2$ is constant and equal to $\sigma^2$.

**Autoregressive Moving Average Model - ARMA(p; q)**

The $ARMA(p;q)$ model is the combination of the autoregressive $AR(p)$ and moving average $MA(q)$ models [32]. In an ARMA model, the current value of the time series is expressed as a linear combination of both the past values of the signal and the past error terms. The order of the ARMA model is defined by two parameters: the order of the AR, represented by the parameter "p", and the order of the MA, represented by the parameter "q". The ARMA process could be depicted as:

$$Z_t = \phi_0 + \sum_{i=1}^{p} \phi_i Z_{t-i} + \sum_{j=1}^{q} \alpha_j \epsilon_{t-j} + \epsilon_t \tag{2.3}$$

where $Z_t$ is the value of the time series at time t, $\phi_0, \phi_1, ..., \phi_p$ are the autoregression coefficients, $\epsilon_t$ is the error term at time t, $\alpha_1, \alpha_2, ..., \alpha_q$ are the moving average coefficients, and $p$ and $q$ are the orders of the autoregression and moving average components, respectively.

We emphasize that the ARMA(p; q) process is stationary if and only if its AR component is stationary. If $\phi_0 = 0$, the mean of the ARMA process will be zero.

### 2.1.3 Nonlinear Time Series Model

Time series analysis often relies on linear models and processes, which are generally suitable for making statistical inferences in many practical applications. However, it is still essential to consider nonlinearity in time series. In real-world, the empirical time series are often nonlinear, and nonlinear models can make significant contributions in various applications. In this section, we will focus on several nonlinear time series analysis techniques, including Threshold Autoregressive Models (TAR), Markov Switching Models, and Time-varying Coefficient Models, which can help to better understand and analyze complex nonlinear patterns in time series data.

#### Threshold Autoregressive Models (TAR)

The self-exciting TAR model is one of the most commonly used nonlinear time series models for scalar data. It is an extension of the segmented linear regression model, where the changes in the regression structure occur in the threshold space. While this model has many variants, we will focus on the simpler two-regime TAR model in this discussion.

A time series $x_t$ follows a two-regime TAR model of order p with threshold variable $x_{t-d}$ if it satisfies

$$x_t = \begin{cases} \phi_0 + \sum_{i=1}^{p} \phi_i x_{t-i} + \sigma_1 \epsilon_t, & if\ x_{t-d} \leq r \\ \theta_0 + \sum_{i=1}^{p} \theta_i x_{t-i} + \sigma_2 \epsilon_t, & if\ x_{t-d} > r \end{cases} \tag{2.4}$$

where $\epsilon_t$ represents a sequence of independent and identically distributed random variables with zero mean and unit variance. The model also includes real-valued parameters $\theta_i$ and $\phi_j$, with the additional assumption that $\theta_i$ and $\phi_j$ are not equal for some value of $i$. The parameters $d$ and $r$ are positive integers representing the delay and threshold, respectively. For the sake of simplicity, we use the same order

$p$ for both regimes. However, it is worth noting that different orders can be used if necessary.

### Markov Switching Models

Markov Switching Models (MSMs) are a class of time series models that allow for changes in the structure of the data generating process over time. In MSMs, the observed time series $y_t$ is modeled as a function of an unobserved Markov chain $S_t$ that determines the current state of the system. The Markov chain is assumed to have a finite number of states, denoted as $1, 2, \ldots, K$, with $K \geq 2$. The probability of being in state $i$ at time $t$, denoted as $P(S_t = i)$, depends on the previous state and is given by the transition probabilities $P(S_t = i|S_{t-1} = j) = \pi_{j,i}$, where $\pi_{j,i}$ is the probability of switching from state $j$ to state $i$.

To simplify our discussion, let us consider a time series $x_t$ and define $S_t$ as the state of the process at time $t$. In a two-state Markov switching model (MSM), $S_t$ can take on two possible values, namely $S_t = 1$ or $S_t = 2$. We say that the time series $x_t$ follows a two-state autoregressive MSM if it meets the following conditions:

$$x_t = \begin{cases} \phi_{0,1} + \phi_{1,1}x_{t-1} + \cdots + \phi_{p,1}x_{t-p} + \sigma_1\epsilon_t, \ if \ S_t = 1; \\ \phi_{0,2} + \phi_{1,2}x_{t-1} + \cdots + \phi_{p,2}x_{t-p} + \sigma_2\epsilon_t, \ if \ S_t = 2; \end{cases} \tag{2.5}$$

where $\phi_{i,j}$ are real numbers, $\sigma_i > 0$, and $\epsilon_t$ is a sequence of i.i.d. random variables with mean zero and variance 1.0.

The autoregressive polynomial of the $j$ state is defined as $j(B) = 1 - \phi_{1,j}B - \cdots - \phi_{p,j}B^j$. In many cases, it is necessary for the roots of $j(B) = 0$ to lie outside the unit circle for $j = 1$ and 2. The state transition of the model is governed by the transition probabilities, as depicted in the following:

$$P(S_t = 2|S_{t-1} = 1) = \eta_1, P(S_t = 1|S_{t-1} = 2) = \eta_2 \tag{2.6}$$

where $0 < \eta_j < 1$.

### Time-Varying Coefficient Models

Time-varying coefficient (TVC) models are a popular choice when weak stationary conditions or linearity assumptions are no longer appropriate. Unlike linear models, TVC models are nonlinear and offer greater flexibility. The way in which the coefficients change over time is a crucial factor in determining the properties of TVC models. In this section, we present one simple TVC model. A time-varying coefficient AR (tvAR) model can be written as:

$$x_t = \phi_{0,t} + \sum_{i=1}^{p} \phi_{i,t} x_{t-i} + \epsilon_t; \tag{2.7}$$

where $\phi_{j,t}$, for $j = 0, \ldots, p$, are random variables and $\epsilon_t$ is a sequence of iid random variables with mean zero and variance $\sigma_2$. At the same time $\phi_{j,t}$ are independent of $\epsilon_t$.

## 2.2 Bayesian Inference

Bayesian inference is a process of using data analysis to estimate the properties of an underlying probability distribution based on observed data [33]. This process utilizes Bayes' theorem to update the probability of a hypothesis as more evidence is obtained. In the context of time series analysis, Bayesian updating is crucial, and the Bayesian Structural Time Series (BSTS) model has been developed for various applications such as feature selection, time series forecasting, decision-making, and causal impact analysis [34, 35].

In the field of machine learning, the term "inference" is often used interchangeably with "prediction." By evaluating a well-trained model, one can make predictions about the next time measurement. In this context, learning or estimating the properties of the model is referred to as "training," while using the model for prediction is referred to as "inference."

### 2.2.1 Bayes' Theorem

When new data is obtained, the level of confidence in a hypothesis is updated using Bayes' theorem. Given two events $A, B \subseteq \Omega$, where $P(\Omega) > 0$, the conditional probability of event A given that B has occurred can be expressed as follows [36]:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.8}$$

where $P(B) \neq 0$. The prior probability $P(A)$ represents one's initial beliefs about event $A$ before event $B$ occurs. The likelihood function $P(B \mid A)$ can be understood as the probability of event $B$ occurring given that event $A$ has occurred. The posterior probability $P(A \mid B)$ represents the updated belief in event $A$ after taking into consideration the occurrence of event $B$.

### 2.2.2 Monte Carlo Methods

Monte Carlo methods, also referred to as Monte Carlo experiments, employ repetitive random sampling to estimate expectations. They are commonly used to address three types of problems: optimization, numerical integration, and sampling from a probability distribution [37]. The principle behind Monte Carlo methods is that by obtaining $M$ random samples, $\theta(1), ..., \theta(M)$, from the density $\pi$, the expectation can be calculated through linearity of expectation, as demonstrated below:

$$E_\pi\{\frac{1}{M}\sum_{i=1}^{M} g(\theta^i)\} = E_\pi\{g(\theta)\} \tag{2.9}$$

### 2.2.3 Markov Chain Monte Carlo

In some instances, it may not be possible or practical to sample from the target distribution $\prod(\theta)$ for Monte Carlo integration directly. Or in some cases, it is difficult for us to identify a suitable sampling density. This is where Markov Chain Monte Carlo (MCMC) methods come in handy. Researchers developed MCMC methods to provide an approximate solution for sampling from any target density [38]. The basic concept behind MCMC is straightforward: if a Markov chain with the desired equilibrium (stationary) distribution is created, then by tracking state transitions on the chain, a sequence of samples with the desired distribution can be obtained. Generally, the more samples that are collected, the better the obtained sample distribution will approximate the actual desired distribution. The most widely used algorithm for creating Markov chains is the Metropolis-Hastings algorithm.

**Algorithm 1** Metropolis–Hastings algorithm

**Output**: M approximate samples from $\pi(\theta)$
**Initialization**: Draw $\theta(0) \in \theta$ from an initial distribution;
**for** i $\leftarrow$ 1 to M **do**
  Draw $\theta^{'} \sim$ q($\theta^{'} \mid \theta^{(i-1)}$);
  Draw u $\sim$ Uniform(0, 1);
  **if** $u < \alpha\ (\theta, \theta^{'})$ **then**
    $\theta^i = \theta^{'}$
  **else**
    $\theta^i = \theta^{i-1}$
  **end if**
**end for**

## 2.3 Dynamical System

Dynamical systems provide a mathematical framework for describing the long-term evolution of systems in which parameters change over time according to fixed rules or equations. This framework can be used to model complex interactions between objects or components in a system that co-evolve over time. Traditionally, researchers used differential equations or iterative mappings to model dynamical systems and to analyze, predict, and understand their behavior over time. The versatility of the mathematical framework used to describe dynamical systems has made it relevant to a wide range of areas of study, including physics, biology, meteorology, astronomy, economics, and more.

In recent years, advances in machine learning and the availability of more data have led to a shift in the study of dynamical systems towards data-driven approaches. Instead of relying solely on analytical models and first principles, experts are now able to extract essential features and gain deeper insights into the behavior of dynamical systems by analyzing data directly.

From a differential equation perspective, dynamical systems can be expressed in the following form:

$$\frac{d}{dt}x(t) = f(x(t), t; \beta) \tag{2.10}$$

where $x$ represents the state of the system and $f$ is a vector field that may depend on the state $x$, time $t$, and a set of parameters $\beta$.

For data-driven modeling of dynamical systems, it is important for researchers to train the model with data collected from the target system so that it can learn its dynamics and accurately predict its behavior. In the machine learning field, the models could be neural networks, support vector regression, Gaussian processes, etc. This problem could be formulated in an abstract manner. Assuming a training data set collected from the target system $D_{target}^{train} = \{(z_1, z_2, ..., z_L)\}_{i=1}^{K} \in R^{K \times L \times N}$ with K trajectories of L step states, the predictive model could be expressed as:

$$\hat{z}_{k+1} = f_\theta(z_k) \tag{2.11}$$

which is parameterized by $\theta$. In order to find the optimal solution for the dynamical function $f$ Eq.2.11, the $\theta$ needs to be found, as shown in the following:

$$\theta^* = argmin_\theta \mathcal{L}(D_{target}^{train}; \theta) \tag{2.12}$$

where $\mathcal{L}$ is an object function to calculate the difference between future values $\hat{z}_{k+1}$ and true values $z_{k+1}$. The obtained model $f_\theta$ is tested on a testing data set $D^{test}_{target}$.

## 2.4 Neural Network Models

### 2.4.1 Convolutional Neural Network

CNN is a type of neural network designed specifically for processing data with a grid-like structure. First proposed by LeCun et al. [39], CNN performs the convolution operation on input data, which involves applying a filter to each subsection of the data to extract relevant features. This operation is represented mathematically as shown in Equation 2.13.

$$s[n] = (f * g)[n] = \sum_{m=0}^{n} f[m]g[n-m] \tag{2.13}$$

where $f$ is the input signal of length $N$, $g$ is the kernel or filter of length $M$, $n$ is the index of the output signal, and $m$ is the index of the input signal that the kernel overlaps with at each step.

After the convolutional layer, a pooling layer is often added to reduce the spatial dimensions of the feature maps while retaining the most informative features. A pooling layer operates by partitioning the input volume into non-overlapping regions such as segments or rectangles and then computing a summary statistic for each region such as the maximum or average value. This process is commonly referred to as max-pooling or average-pooling, respectively. By summarizing the information in each region, a pooling layer helps to reduce the computation required in subsequent layers while preserving important spatial relationships in the data.

### 2.4.2 Long Short-Term Memory

LSTM is a type of recurrent neural network that was introduced in [40]. It is designed to overcome the problems of gradients vanishing or explosions in standard RNNs, which can make it difficult to remember long-term dependencies.

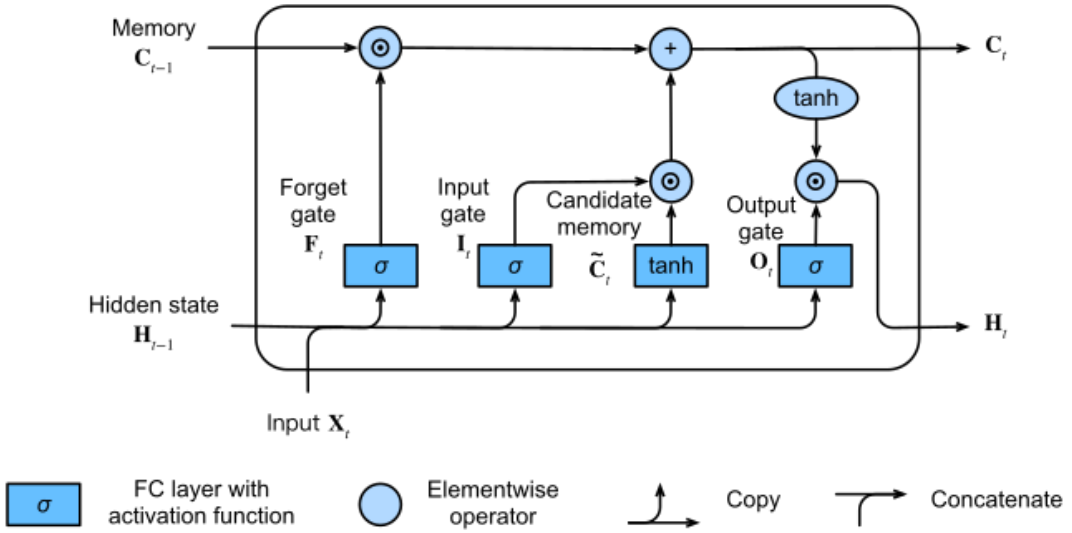



FIGURE 2.1: Structure of Long Short Term Memory

As illustrated in Figure 2.1, LSTM has three gates and one candidate memory cell, which accepts the input $X_t$ at the current time step and the hidden state $H_{t-1}$ at the previous time step. Assuming a LSTM has $d$ input nodes and $h$ hidden nodes and the batch size is $n$, then the three gates and one candidate memory cell could be calculated as follows:

$$\begin{aligned} I_t &= \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \\ F_t &= \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \\ O_t &= \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \\ \tilde{C}_t &= tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \end{aligned} \tag{2.14}$$

where the input nodes are $X_t \in R^{n \times d}$ and the hidden nodes are $H_{t-1} \in R^{n \times d}$. The output of the three gates (input gate, forget gate, and output gate) and the memory cell are denoted by $I_t$, $F_t$, $O_t$, and $\tilde{C}_t$, respectively. Each of these matrices has dimensions $R^{n \times h}$. The weight matrices between the input nodes and the three gates plus the memory cell are $W_{xi}, W_{xf}, W_{xo}$, and $W_{xc} \in R^{d \times h}$, respectively. Similarly, the weight matrices between the hidden nodes and the three gates plus the memory cell are $W_{hi}, W_{hf}, W_{ho}$, and $W_{hc} \in R^{h \times h}$, respectively. The biases for the gates and the memory cell are denoted by $b_i, b_f, b_o$, and $b_c \in R^{1 \times h}$, respectively. Finally, the sigmoid activation function, denoted by $\sigma$, is used to activate the gates.

LSTM utilizes two specialized gates, namely, the input gate and the forget gate, to manage input and memory retention, respectively. The input gate, denoted as $I_t$, controls the amount of new data that is incorporated through the memory cell $C_t$. On the other hand, the forget gate, represented as $F_t$, decides how much of the previous

memory content $C_{t-1}$ is maintained. By employing the point-wise multiplication technique, we can derive the following updated equation:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{2.15}$$

The proposed design has been effective in mitigating the vanishing gradient problem and improving the ability to capture long-term dependencies in sequential data [40].

Same as the $C_t$ computation, the output gate and the $tanh$ function of the memory cell are used to calculate the hidden state, as follows:

$$H_t = O_t \odot tanh(C_t) \tag{2.16}$$

### Bidirectional Long Short-Term Memory

Unlike unidirectional LSTM, BDLSTM processes time series data in both forward and backward directions using two separate hidden layers. As shown in Figure 2.2, BDLSTM leverages both posterior and prior probabilities for prediction and is generally more effective than unidirectional LSTM in several applications, including phoneme classification [41] and speech recognition [42].

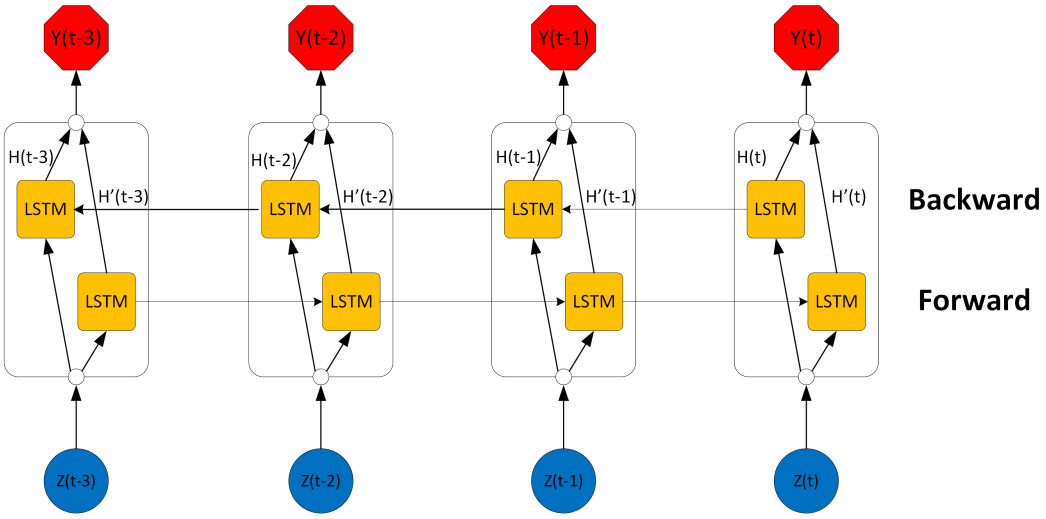



FIGURE 2.2: Architecture of Bidirectional Long Short-Term Memory

The output of the BDLSTM layers is a vector $Y_T$, in which each element is computed using the following equation:

$$Y_T = \gamma(H, H^{'}) \tag{2.17}$$

where $H$ and $H^{'}$ denote the outputs in the forward direction and backward direction, respectively. The two output sequences are combined using a concatenation function denoted by $\gamma$.

## 2.5 Dynamic Land Ecosystem Model

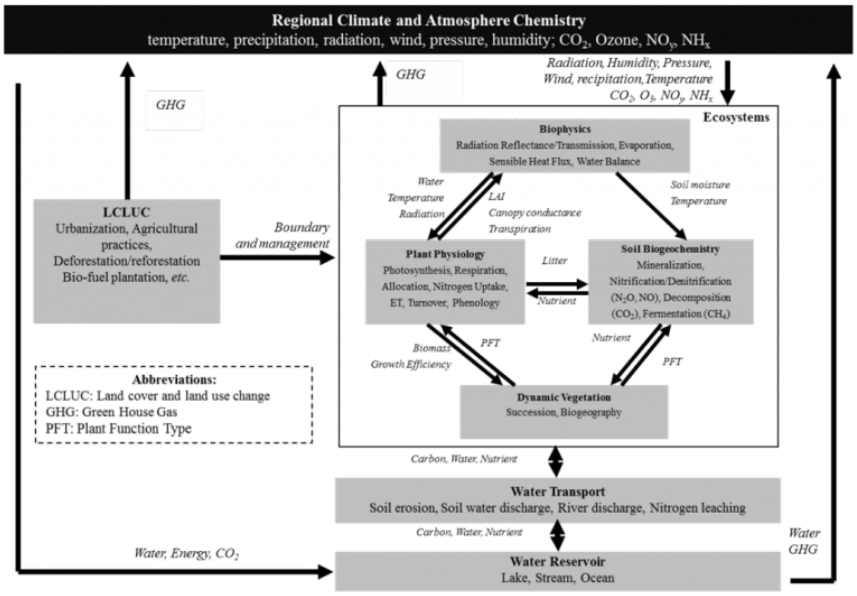



FIGURE 2.3: Interaction among five components in DLEM

The primary objective behind the development of the DLEM was to address crucial requirements for comprehending and forecasting the wide-ranging patterns and processes of terrestrial ecosystems and continental margins, as well as the intricate interactions involving climate, ecosystems, and human activities within the context of multifactor global change. The DLEM integrates significant biophysical, biogeochemical, vegetation dynamical, and land use processes, functioning across various temporal scales spanning from daily to yearly, and spatial resolutions ranging from meters to kilometers, encompassing regional and global extents. The DLEM exhibits the following characteristics: 1) It considers multiple driving factors; 2) it incorporates fully-coupled cycles of carbon, nitrogen, and water; 3) it concurrently simulates major greenhouse gases ($CO_2$, $CH_4$, $N_2O$, and $H_2O$); 4) and it dynamically tracks alterations in land cover/use and vegetation distribution. The model has undergone validation against site-specific measurements worldwide and has been employed at different scales [43]. The DLEM comprises five core elements: biophysics, plant physiology, soil biogeochemistry, dynamic vegetation, and land use and management. The interactions among these components are shown in Figure 2.3.

- Within the biophysical component, the model captures the instantaneous interactions involving energy, water, and momentum with the atmosphere. It encompasses micrometeorology, canopy physiology, soil physics, radiative transfer, hydrology, and the influence of surface fluxes of energy, moisture, and momentum on the simulated surface climate.

- The plant physiology module replicates crucial physiological processes, such as photosynthesis, autotrophic respiration, carbon allocation among different plant parts (root, stem, and leaf), turnover of living biomass, nitrogen uptake and fixation, transpiration, phenology, and more.

- The soil biogeochemistry component simulates N mineralization, nitrification, denitrification, $NH_3$ volatilization, leaching of soil mineral N, decomposition, and fermentation.

- In the dynamic vegetation component, two types of processes are simulated. Firstly, it accounts for biogeographical redistribution in response to climate changes. Secondly, it captures plant competition and succession during vegetation recovery following disturbances. Similar to other DGVMs (Dynamic Global Vegetation Models), DLEM utilizes the concept of PFT (Plant Functional Type) to describe vegetation distributions.

- The land use and management component replicates the biogeochemistry of managed ecosystems, encompassing agricultural ecosystems, plantation forests, and pastures.

### 2.5.1 The Global Nitrous Oxide Model Intercomparison Project (NMIP)

According to [45], the radiative forcing of $N_2O$ is approximately 265 times greater than that of $CO_2$ over a 100-year time horizon. As a result, it is estimated that $N_2O$ contributes approximately 6% to the overall global warming effect. Additionally, it is considered to be the single most important ozone-depleting emission in the twenty-first century [46]. Human activities, including industrial $N_2$ fixation through fossil fuel combustion or the Haber–Bosch process and manure nitrogen (N) application, play an increasingly significant role in the perturbation of the global N cycle [47, 48, 49]. As a result, atmospheric $N_2O$ concentration has increased by approximately 21%, from 271 ppb at preindustrial levels to 329 ppb in 2015 [50, 51, 52]. This highlights the urgent need to reduce $N_2O$ emissions, especially in the agricultural

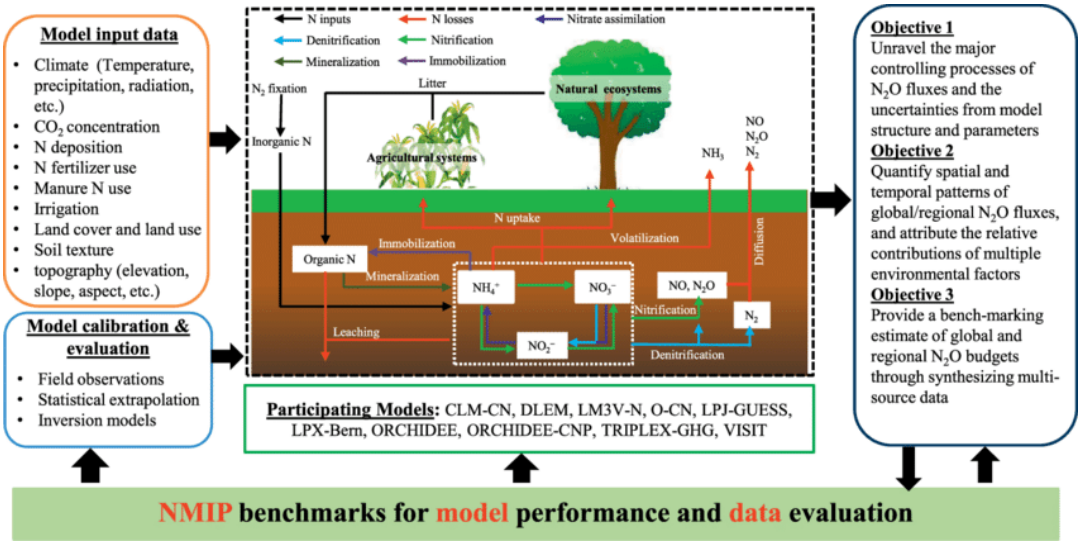



FIGURE 2.4: The Framework of Global $N_2O$ Model Intercomparison Project (NMIP) [44]

sector, which is one of the major contributors to increased $N_2O$ emissions. Therefore, reducing $N_2O$ emissions must be a top priority as part of our efforts to mitigate the effects of climate change.

However, compared to $CO_2$-related studies, global investigations into $N_2O$ fluxes have been significantly limited. Consequently, significant uncertainties persist regarding the primary drivers and their impacts on nitrification, denitrification, and above-below ground nitrogen exchange processes. Table 2.1 displays the commonly used process-based mechanistic models in evaluating and predicting changes to the N cycle and $N_2O$ emissions in response to various global factors. While the resulting estimates of global terrestrial $N_2O$ budgets and spatiotemporal patterns have varied significantly among these models, these discrepancies are primarily due to differences in the input data used, the structure of the models, and the methods used to determine model parameters. Our understanding of how model structures and parameters influence $N_2O$ estimation in response to multifactor global changes in both natural and managed ecosystems remains inadequate. To address these issues, the NMIP has been established, utilizing consistent input data and simulation protocols to estimate global terrestrial $N_2O$ fluxes while also considering uncertainties associated with model structure and parameters, as shown in Figure 2.4.

### 2.5.2 DLEM for Nitrous Oxide Emission

In this investigation, the emission of $N_2O$, which is influenced by all these five factors, is studied. According to the description in the [67], the hierarchy structure of

TABLE 2.1: Mechanistic Models for the Prediction of $N_2O$ Emission

| Model | Contact | Affiliation |
|---|---|---|
| CLM-CN [53] | E. Saikawa | Emory University |
| DLEM [54, 55] | H. Tian | Auburn University |
| LM3V-N [56] | S. Gerber | University of Florida |
| LPJ-GUESS [57, 58] | S. Olin/A.Arneth | Lund University, Sweden/Karlsruhe Institute of Technology, Germany |
| LPX-Bern [59, 57] | S. Lienert/F. Joos | Institute for Climate and Environmental Physics, University of Bern, Switzerland |
| O-CN [60] | S. Zaehle | Max Planck Institute for Biogeochemistry |
| ORCHIDEE [61] | N.Vuichard | L'Institut Pierre-Simon Laplace Laboratoire des Sciences du Climat et de l'Environnement (IPSL–LSCE), France |
| ORCHIDEE-CNP [62] | J. Chang/D. Goll | IPSL–LSCE, France |
| TRIPLEX-GHG [63, 64] | C. Peng | University of Quebec at Montreal, Canada |
| VISIT [65, 66] | A. Ito | National Institute for Environmental Studies, Japan |

DLEM for the emission of $N_2O$ could be represented in Figure 2.5.

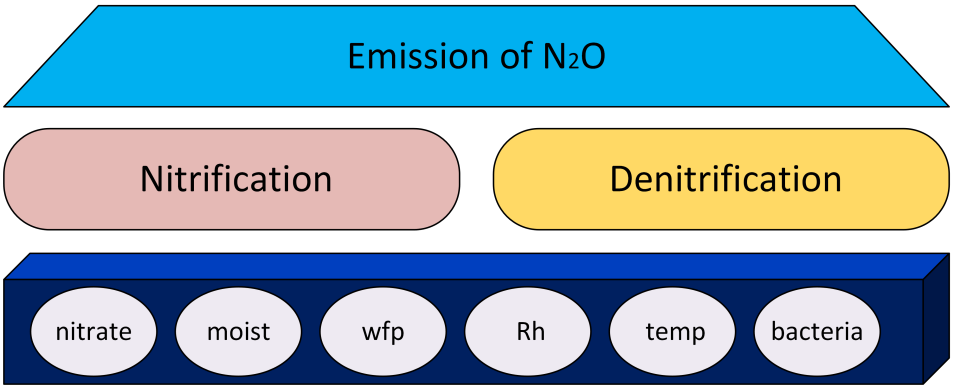



FIGURE 2.5: Hierarchy Structure in DLEM

### General Emission of Nitrous Oxide

$N_2O$ production in soils is majorly dominated by biological nitrification and denitrification processes.

$$\begin{aligned} G_{N_2O} &= (N_{nit} + N_{denit}) \times f(T_{soil}) \times (1 - f(wfp)) \\ f(T_{soil}) &= \frac{1}{1 + e^{-0.64+0.08T_{soil}}} \\ f(wfp) &= 0.0116 + \frac{1.36}{1 + e^{-\frac{wfp-0.815}{0.0896}}} \end{aligned} \tag{2.18}$$

where $G_{N_2O}$ is the $N_2O$ produced from soil, $N_{nit}$ is the daily nitrification rate ($gNm^{-2}day^{-1}$), $N_{denit}$ is the daily denitrification rate ($gNm^{-2}day^{-1}$), $f(T_{soil})$ is the scaled factor of soil temperature $(T_{soil}, ^\circ C)$ on the $N_2O$ emission process (unitless), and $f(wfp)$ is the effects of water-filled porosity.

### Nitrification

Nitrification is a biological process that involves the conversion of ammonia ($NH_3$) and ammonium ($NH_4^+$) into nitrite ($NO_2^-$) and nitrate ($NO_3^-$) by soil microorganisms, which are nitrifying bacteria. This process is an essential part of the nitrogen cycle and plays a crucial role in maintaining soil fertility and ecosystem productivity. The reactions involved in nitrification are the following:

$$\begin{aligned} 2\,NH_3{}^+ + 3\,O_2 &\longrightarrow 2\,NO_2{}^- + 2\,H^+ + 2\,H_2O \\ 2\,NO_2{}^- + O_2 &\longrightarrow 2\,NO_3{}^- \end{aligned} \tag{2.19}$$

And the nitrification rate is calculated as follows:

$$
\begin{aligned}
N_{nit} &= k_{nit} \times f(T_{Nsoil}) \times g(wfp) \times D_{NH_4} \\
f(T_{Nsoil}) &= 7.24 \times e^{-3.432+0.618\times T_{soil}\times(1-0.5\times T_{soil}/36.9)} \\
g(wfp) &= -12.904 \times wfp^4 + 17.651 \times wfp^3 + \\
&\quad 5.5368 \times wfp^2 + 0.9975 \times wfp - 0.0243
\end{aligned}
\tag{2.20}
$$

where $k_{nit}$ is the daily maximum fraction of $NH_4^+$ that is converted into $NO_3^-$ and nitrogen gases, $f(T_{Nsoil})$ is the soil temperature's effect on nitrification (unitless), and $g(wfp)$ is the soil moisture effect.

### Denitrification

Denitrification is the biological process that converts $NO_3^-$ into nitric oxide (NO), $N_2O$, and dinitrogen ($N_2$).The reactions involved in denitrification are the following:

$$
\begin{aligned}
{NO_3}^+ + 2\,H^+ + 2\,e^- &\longrightarrow {NO_2}^- + H_2O \\
{NO_2}^- + 2\,H^+ + e^- &\longrightarrow NO + H_2O \\
2\,NO + 2\,H^+ + 2\,e^- &\longrightarrow N_2O + H_2O \\
N_2O + 2\,H^+ + 2\,e^- &\longrightarrow N_2 + H_2O
\end{aligned}
\tag{2.21}
$$

And the denitrification rate is calculated as follows:

$$
\begin{aligned}
N_{denit} &= N_{pot_{denit}} \times f(T_{soil}) \times f(wfp) \times f(D_{NO_3}) \\
N_{pot_{denit}} &= (0.151 + 0.015 \times P_{clay}) \times Rh \times k_{den} \\
f(D_{NO_3}) &= 1.17 \times \frac{D_{NO_3}}{32.7 + D_{NO_3}} \\
D_{NO_3} &= av_{NO_3}/BD_{soil}
\end{aligned}
\tag{2.22}
$$

where $N_{pot_{denit}}$ is the potential rate of denitrification ($gNm^{-2}day^{-1}$), $P_{clay}$ is the percentage of clay content in soil, $Rh$ is the soil respiration rate ($gCm^{-2}day^{-1}$), $k_{den}$ is a parameter depending on the plant functional type to tune the potential denitrification rate ($gNm^{-2}day^{-1}$), $f(D_{NO_3})$ represents the effect of $NO_3^-$ concentration ($gNg^{-1}soil$), $av_{NO_3}$ is the $NO_3^-$ content in the soil per unit area ($gNm^{-2}$), and $BD_{soil}$ is the soil bulk density ($gsoilm^{-3}$).

## 2.6 Machine Learning Models for Prediction of Nitrous Oxide Emission

Despite being calibrated to specific locations, process-based biogeochemical models typically cannot accurately forecast daily or monthly emissions with more than a 20% margin of error. To address this limitation, researchers have started using machine learning techniques to study environmental phenomena with high variability across time and space. For example, machine learning models like classical regression, shallow learning, and deep learning have been introduced to predict soil greenhouse gas (GHG) emissions from agricultural fields. Table 2.2 displays the specific machine learning models that have been employed for this purpose [68].

## 2.7 Information Bottleneck Theory

In information theory, which underlies most of the analysis presented in this thesis, this trade-off is treated through the sub-field of rate distortion theory. In particular, the complexity of the model is then characterized through its coding length, which in turn is proportional to the amount of (mutual) information between data points and their new representatives (precise definitions of all these concepts will be given shortly). If we term this information the "compression-information," simpler models correspond to models with low values of compression-information that enable more efficient communication. However, these models typically suffer from a relatively high (expected) distortion. Hence, this fast communication comes with the cost of lower precision of the sent messages. Thus, the familiar precision-complexity trade-off, which we already encountered for supervised and unsupervised learning, arises again in the context of communication through an information-theoretic analysis [76].

### 2.7.1 Mutual Information

Mutual information is a measure of the amount of information that is shared between two random variables. It quantifies the degree of dependence between the variables and can be used to identify relationships and patterns in data.

The mutual information between two discrete random variables X and Y is defined as:

TABLE 2.2: Machine Learning Models for Prediction of $N_2O$ Emission

| Models | Descriptions |
|---|---|
| Support Vector Machine (SVM) [69] | SVM creates linear and nonlinear decision boundaries, or hyperplanes, in variable space by optimizing a margin-based formulation using a subset of the training data called support vectors. |
| Random Forest (RF) [70] | Decision trees grow concurrently by randomly splitting and searching for the best combination of variables. |
| Least Absolute Shrinkage and Selection Operator (LASSO) [71] | The LASSO model and ridge regression are quite similar, with the key distinction being that the LASSO model uses the L1 norm while ridge regression uses the L2 norm. |
| Feed-Forward Neural Network (FNN) [72] | FNN automatically adapts its synaptic weights and biases using the back-propagation learning algorithm, resulting in the creation of a distinct model that corresponds to the input/output relationship in the network. |
| Radial Basis Function Neural Network (RBFNN) [73] | RBFNN is a type of FNN that uses radial basis functions as activation functions. Their ability to provide global approximation, a compact scheme, and effective fitting of continuous and noisy datasets distinguish them from other types of supervised neural networks. |
| Long Short-Term Memory (LSTM) [40] | LSTM is a type of recurrent neural network that is especially well-suited for sequence-to-sequence learning tasks. |
| Deep Belief Network (DBN) [74] | DBN is primarily composed of Restricted Boltzmann Machines (RBMs), with a linear regression layer typically included to facilitate prediction tasks. |
| Convolutional Neural Network (CNN) [75] | CNN relies on the principle of local connectivity, utilizing convolutions in place of the traditional weighted sums used by FNNs. |

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.23}$$

where p(x,y) is the joint probability mass function of X and Y, and p(x) and p(y) are the marginal probability mass functions of X and Y, respectively. The logarithm is typically taken with respect to base 2, so the resulting units of mutual information are bits.

For continuous random variables, the mutual information can be defined using integrals instead of sums:

$$I(X;Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)}, dx, dy \tag{2.24}$$

where p(x,y) is the joint probability density function of $X$ and $Y$, and p(x) and p(y) are the marginal probability density functions of $X$ and $Y$, respectively.

Mutual information can also be expressed in terms of entropy:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2.25}$$

where H(X) and H(Y) are the entropy of $X$ and $Y$, respectively, and H(X|Y) and H(Y|X) are the conditional entropy of $X$ given $Y$ and the conditional entropy of $Y$ given $X$, respectively.

In summary, mutual information is a powerful tool for analyzing the relationships between random variables, and its calculation involves computing probabilities and entropy functions of the variables.

### Information Bottleneck Theory

The information bottleneck (IB) method is a technique in information theory introduced by Naftali Tishby et.al [76]. The central idea behind IB is to discover the best trade-off between accuracy and compression: an optimal hidden representation $H$ of the input $X$ should compress the information as much as possible, while still allowing for accurate prediction of the output $Y$.

IB algorithm minimizes the following functional with respect to conditional distribution $p(h|x)$:

$$\inf_{p(h|x)} \left( I(X;H) - \beta I(H;Y) \right) \tag{2.26}$$

where $I(X;H)$ and $I(H;Y)$ are the mutual information of $X$ and $H$, and of $H$ and $Y$, respectively, and $\beta$ is a Lagrange multiplier.

### 2.7.2 Hilbert-Schmidt Independence Criterion

HSIC, which stands for Hilbert-Schmidt Independence Criterion, is a statistical measure used to test the independence between two random variables. HSIC is a nonparametric method that can be used to detect any kind of dependence between variables, including nonlinear and high-dimensional relationships.

The basic idea behind HSIC is to compare the cross-covariance of two variables in their respective feature spaces with the product of their individual covariances. If the cross-covariance is close to zero, then the two variables are considered independent.

The HSIC between two random variables X and Y with observations $x_1, ..., x_n$ and $y_1, ..., y_n$, respectively, is defined as:

$$\text{HSIC}(X, Y) = \frac{1}{(n-1)^2}\text{tr}(K_X H K_Y H) \tag{2.27}$$

where $tr$ denotes the trace of a matrix, $K_X$ and $K_Y$ are the kernel matrices for X and Y, respectively, with entries $K_X(i, j) = k(x_i, x_j)$ and $K_Y(i, j) = k(y_i, y_j)$, where k is a kernel function, and $H = I - 1/n 1_n 1_n^T$, where I is the $n \times n$ identity matrix.

The HSIC statistic can be centered and normalized to obtain a test statistic that follows a standard normal distribution under the null hypothesis of independence:

$$\text{HSIC}_n(X, Y) = \frac{\text{HSIC}(X, Y) - \mathbb{E}[\text{HSIC}(X', Y')]}{\text{Var}[\text{HSIC}(X', Y')]} \tag{2.28}$$

where $X^{'}$ and $Y^{'}$ are independent copies of $X$ and $Y$, respectively.

In summary, HSIC is a nonparametric method for testing the independence between two variables, and its calculation involves computing the kernel matrices and their cross-covariance.

#### HSIC-Bottleneck Theory in Neural Network

Assuming there is an MLP composed of L hidden layers $T() : R^{d_{i-1}} \rightarrow R^{d_i}$, resulting in hidden representations $Z_i \in R^{m \times d_i}$, where $i \in 1, \ldots, L$, and $m$ denotes batch size. According to the Information Bottleneck principle, the original mutual information terms are replaced by HSIC, as shown in Equation 2.28, as the learning target:

$$Z_i^* = \arg\min_{Z_i} HSIC(Z_i, X) - \beta HSIC(Z_i, Y) \tag{2.29}$$

where $X \in R^{m \times d_x}$ is the input, $Y \in R^{m \times d_y}$ is the label, $i \in \{0, 1, 2, \ldots, L\}$ and L is the number of hidden layers, $d_x$ and $d_y$ are the dimensionalities of the input and output variables, respectively.

## 2.8 Training Algorithm for Neural Network

In the field of neural networks, data representation has two major applications: pattern recognition and data compression [77]. In pattern recognition, the challenge lies in discovering the regularities within the data, while in data compression, the process involves encoding, restructuring, or modifying data to reduce its dimensions, often using fewer bits than the original representation.

In these applications, it is essential to extract features that are invariant or robust to variations within each category. This process is referred to as feature extraction in the field of neural networks [96]. From the perspective of information theory, the goal of feature extraction is to minimize information loss during the transformation of data from a high-dimensional space to a low-dimensional space [97]. Building upon this idea, neural network-based principal component analysis has been developed. Additionally, machine learning algorithms can benefit from from feature extraction and selection when the model learns a mapping from input data to target data [98, 99].

### 2.8.1 Neural Network-Based Principal Component Analysis

Principal Component Analysis (PCA) is a well-known feature extraction method that aims to identify the optimal directions in which the data exhibit the largest variances and capture the most information [100]. Traditional PCA is limited to linear regression and matrix analysis, while the neural network-based PCA method estimates the PC from the input data "online," making it particularly suitable in addressing high-dimensional data. This approach circumvents the need for computing a large covariance matrix and is also effective in tracking nonstationary data, where the covariance matrix exhibits slow variations over time [101].

In recent decades, several neural network-based PCA learning algorithms have been proposed, with the Hebbian and Oja's learning rules serving as the foundation. These algorithms can be categorized into different classes, including Hebbian rule-based PCA algorithms, least mean squared error-based PCA algorithms, other optimization-based PCA algorithms, anti-Hebbian rule-based PCA algorithms, and nonlinear PCA algorithms, as illustrated in Table 2.3.

TABLE 2.3: Neural-Based PCA Algorithms

| Title | Categories | Update Rule |
|---|---|---|
| 1 [78, 79] | Subspace Learning Algorithms [80] | $w_{t+1} = w_t + \eta y_t x_t - y_t^2 w_t$ |
| | Generalized Hebbian Algorithm [81] | $w_{i,t+1} = w_{i,t} + \eta_{i,t} y_{i,t}[x_t - \hat{x}_{i,t}]$ |
| | LEAP Algorithm [82] | $w_{i,t+1} = w_{i,t} + \eta\{B_{i,t} y_{i,t}[x_t - w_{i,t} y_{i,t}] - A_{i,t} w_{i,t}\}$ |
| | DPD algorithm [83] | $w_{i,t+1} = w_{i,t} + \eta_t[x_t y_{i,t} - (\sum_{j=1}^{J_2} w_{j,t} w_{j,t}^T)\frac{w_{i,t}}{\|w_{i,t}\|}]$ |
| 2 [84] | Rubner-Tavan PCA Algorithm [85] | $w_{i,t+1} = w_{i,t} + \eta_t y_{i,t}[x_t - \hat{x}_t]$ |
| | APEX Algorithm [86] | $w_{i,t+1} = w_{i,t} + \eta_{i,t}[y_{i,t} x_t - y_{i,t}^2 w_{i,t}]$ |
| 3 | LMSER Algorithm [87] | $w_{i,t+1} = w_{i,t} + \eta_t\{2A_t - C_{i,t} - A_t C_{i,t} - \gamma[B_{i,t} A_t + A_t B_{i,t}]\} w_{i,t}$ |
| | PAST Algorithm [88] | $w_{i,t} = w_{i,t-1} + [x_{i,t} - \hat{x}_{i,t}]\frac{y_{i,t}^*}{\delta_{i,t}}$ |
| | Robust RLS Algorithm [89] | $w_{i,t} = w_{i,t-1} + [x_{i,t} - \hat{x}_{i,t}] y_{i,t}$ |
| 4 | NIC algorithm [90] | $W_{t+1} = (1 - \eta) W_t + \eta \hat{C}_{t+1} W_t [W_t^T \hat{C}_{t+1} W_t]^{-1}$ |
| | coupled PCA/MCA algorithms [91] | $w_{t+1} = w_t + \eta_t(\frac{x_t y_t}{\lambda_t} - w_t)$ |
| 5 | Kernel PCA [92, 93] | $\lambda v = C_1 v = \frac{1}{N}\sum_{j=1}^{N}(\phi(x_i)^T v)\phi(x_i)$ |
| | Robust/Nonlinear PCA [94] | $w_{i,t+1} = w_{i,t} + \eta_t \phi(y_{i,t}) e_{i,t}$ |
| | Autoassociative Network-Based Nonlinear PCA [95] | normal backpropagation approach |

[1] Hebbian Rule-Based PCA.
[2] Anti-Hebbian Rule-Based PCA.
[3] Least Mean Squared Error-Based PCA.
[4] Optimization-Based PCA.
[5] Nonlinear PCA

### 2.8.2 Gradient-Based Optimization Approaches

In the domain of machine learning, gradient descent (GD) is currently considered the state-of-the-art optimization strategy [102]. Unlike PCA algorithms, which aim to discover the principal direction for representing the data with minimal information loss, gradient-based optimization approaches focus on iteratively updating the parameters to minimize the target function, also known as the loss function in neural network terminology. Based on the processing of input data and the updating method, gradient descent algorithms can be categorized as follows [103]:

All the methods mentioned above fall under the category of gradient-based optimizations. Consequently, they suffer from several drawbacks, including local optima [115], gradient vanishing and exploding [116], hyper-parameter sensitivity [117], and difficulties with plateaus and flat regions, among others. In this paper, the proposed SRA approach overcomes the issues associated with gradient-based methods and achieves state-of-the-art performance on training MLP as a classifier.

### 2.8.3 Machine Learning Methods for Time Series Data Augmentation

Table 2.5 provides an overview of common data augmentation methods for time series data. Similar to data augmentation techniques employed in computer vision, most time series data augmentation approaches rely on random transformations, including operations like cropping [141], scaling [121], and rotation [142]. The difficulty in implementing random transformation-based data augmentation arises from the inherent diversity among time series datasets. Not all transformations are suitable for every time series dataset. For instance, the application of jittering (adding noise) assumes that it is typical for the time series patterns in a specific dataset to include noise. While this assumption is valid for sensor data, audio recordings, or Electroencephalogram (EEG) data, it may not be applicable to time series data obtained from object contours.

In addition to random transformations, researchers have also explored the utilization of intrinsic dataset information for synthesizing time series data. This can be accomplished through methods such as pattern mixing, decomposition, and generative models. Pattern mixing encompasses the practice of blending two or more existing time series to create fresh, unique patterns. The fundamental idea behind this technique is that by amalgamating distinct existing patterns, new samples can be

TABLE 2.4: Gradient-Based Optimization Approaches

| Title | Update Rule |
|---|---|
| SGD [104] | $\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ |
| BGD [105] | $\theta = \theta - \eta \nabla_\theta J(\theta)$ |
| Mini-BGD [106] | $\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$ |
| Momentum [107] | $\nu_t = \gamma \nu_{t-1} + \eta \nabla_\theta J(\theta)$<br>$\theta = \theta - \nu_t$ |
| NAG [108] | $\nu_t = \gamma \nu_{t-1} + \eta \nabla_\theta J(\theta - \gamma \nu_{t-1})$<br>$\theta = \theta - \nu_t$ |
| Adagrad [109] | $\theta_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$ |
| Adadelta [110] | $\Delta\theta_{t-1} = -\frac{RMS(\Delta\theta_{t-1})}{RMS(g_{t-1})} g_{t-1}$<br>$\theta_t = \theta_{t-1} + \Delta\theta_{t-1}$ |
| RMSprop [111] | $E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$ |
| Adam [112] | $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$<br>$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$<br>$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$<br>$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$ |
| AdaMax [113] | $u_t = \beta_2^\infty v_{t-1} + (1-\beta_2^\infty) \lvert g_t \rvert^\infty$<br>$= max(\beta_2 v_{t-1}, \lvert g_t \rvert)$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{u_t} \hat{m}_t$ |
| Nadam [114] | $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$ |

TABLE 2.5: Machine Learning Approaches for Time Series Data Augmentation

| Family | Methods |
|---|---|
| Random Transformation | Jittering [118, 119, 120], Rotation [118], Scaling [121], Magnitude Warping, Flipping [121], Permutation, Slicing, Time Warping, Time Masking, Frequency Masking, Frequency Warping, Fourier Transform |
| Pattern Mixing | Deviation From Mean (DFM) [122], Interpolation [123], Time Aligned Averaging [124], Guided Warping [125], Equalized Mixture Data Augmentation (EMDA) [126], Stochastic Feature Mapping (SFM) [127] |
| Decomposition | Seasonal-Trend Decomposition using Loess (STL) [128], Independent Component Analysis (ICA) [129], Empirical Mode Decomposition(EMD) [130] |
| Generative Models | Gaussian Trees [131], Posterior Sampling [132], Linear Model [133], Markov Chain [134], Local and Global Trend (LGT) [135], Generating Time Series (GRATIS) [136], Long Short-Term Memory (LSTM) [135], Generative Adversarial Network (GAN) [137, 138], WaveNet [139], Autoencoders [140] |

crafted, combining distinctive features from each contributing pattern. Some common approaches to pattern mixing include Deviation From Mean (DFM) [122], interpolation [123], and Equalized Mixture Data Augmentation (EMDA) [126]. Decomposition methods extract features from the dataset, such as trend components [128] and independent components [143], to generate new patterns based on these extracted features. Generative models take a somewhat indirect approach by utilizing the feature distributions within the dataset to produce new patterns. More recently, generative models employing neural networks such as GANs [144] have gained prominence.

### 2.8.4 Generative Adversarial Network for Time Series Data Augmentation

As shown in Table 2.6, GANs represent one of the most significant advancements in generative technology. Since their initial introduction in a seminal research paper [144], these frameworks have been applied extensively to generate synthetic sequences in diverse domains, including renewable scenarios [145], sensor data [146], finance [147], biosignals [148], smart grid data [149], and text [150]. Recently, researchers have shifted their focus to synthesizing time series data, leading to the development of several GANs specifically designed for this purpose.

TABLE 2.6: Generative Adversarial Network for Time Series Data Augmentation

| GAN Models | Characteristic |
|---|---|
| WGAN [151, 152] | Enhance the stability of learning, mitigate issues such as mode collapse, and offer meaningful learning curves that can aid in debugging and optimizing hyperparameters. |
| C-RNN-GAN [153] | Work with continuous sequential data and apply it through training on a collection of classical music. |
| RCGAN [154] | Utilize recurrent neural networks in both the generator and the discriminator, with both models being conditioned on auxiliary information. |
| WaveGAN [155] | Capable of generating one-second audio waveform segments with global coherence; suitable for sound effect creation. |
| TimeGAN [138] | Generate authentic time series data that blends the flexibility of the unsupervised approach with the control offered by supervised training. |
| GT-GAN [156] | Capable of generating both regular and irregular time series data. |

The earliest model for generating time series data, C-RNN-GAN [153], employs a standard GAN framework adapted for sequential data by utilizing LSTM in both its generator and discriminator. The Recurrent Conditional GAN (RCGAN) [154] takes a similar approach but introduces minor architectural differences such as removing the dependence on previous output while conditioning on additional input [157]. WaveGAN [155] generates time series data by estimating the conditional probability of the preceding data using dilated causal convolution. TimeGAN [138] presents a framework that combines adversarial training of GANs with supervised training to predict $x_{i+1}$ from $x_i$, where $x_i$ and $x_{i+1}$ denote two multivariate time series values at time $t_i$ and $t_{i+1}$, respectively. Building upon TimeGAN, GT-GANs [156] are capable of synthesizing both regular and irregular time series data. However, when dealing with time series data collected from dynamical systems, capturing the underlying dynamic behavior using these methods presents a significant challenge. This challenge arises because the dynamic behavior is implicitly encoded in the first-order difference or the first-order derivative over time.

## 2.9 Quantum Neural Network Models

In the early 1980s, Richard Feynman proposed that quantum computers could effectively address challenges in physics and chemistry by leveraging their ability to

TABLE 2.7: A Variety of Quantum Neural Network Models

| Models | Application |
|---|---|
| Quantum Restricted Boltzmann Machine Network [158] (QRBM) | classifier [159], anomaly detection [160] |
| Quantum Convolutional Neural Network (QCVNN) [161] | classifier [162], data analysis [163] |
| Quantum Generative Neural Network (QGAN) [164, 165] | image generation [166], discrete data generation [167] |
| Quantum Graph Neural Network (QGNN) [168] | particle track reconstruction [169], materials search [170], financial fraud detection [171] |
| Quantum Recurrent Neural Network (QRNN) [172] | prediction of solar irradiance [173], language translation [174], prediction of rotating machinery [175] |
| Quantum Tensor Neural Network (QTNN) [176] | tiny object classification [177], classifier [178] |
| Quantum Perceptron (QP) [179] | universal approximator [180], pattern classification [181] |

exponentially outperform classical computers in simulating large quantum systems [182]. In 2019, IBM introduced a quantum processor featuring 53 qubits, open for programming by external researchers. Simultaneously, Google achieved "Quantum Supremacy" with its 53-qubit Sycamore chip, surpassing the computational speed of the world's fastest supercomputer, IBM Summit. Google completed calculations in 200 seconds that would take 10,000 years on traditional computers [183].

The rapid advancements in quantum computer hardware have paved the way for the realization of Quantum Neural Networks (QNN). Capitalizing on quantum properties, QNN exhibits superior storage capacity and computational efficiency compared to classical counterparts [184]. QNN finds extensive application in diverse domains, including image processing [185, 186, 187], speech recognition [188, 189], disease prediction [190, 191], and various other fields. Table 2.7 illustrates various numerical types of quantum neural networks proposed for a variety of applications.

# Chapter 3

# Problem Formulation and Methodology

In this chapter, the research problem is succinctly stated based on the comprehensive analysis of relevant literature. Additionally, the approach to address the identified research challenges is presented in detail.

## 3.1 Problem Definition

In this study, we are analyzing a finite multivariate time series consisting of $T$ real-valued measurements taken at regular intervals, represented by $X = [X_1, X_2, ..., X_T]$. In the context of regression analysis, each measurement, represented by $X_t$ with a real vector, is considered as the outcome of $M$ input variables, represented by $Z_t \in R^M$ subjected by a random error term $\epsilon$, as shown in the following:

$$X_t = f(Z_t; \theta) + \epsilon_t \tag{3.1}$$

where $f$ is the model and $\theta$ are the model parameters.

The determination of input variables, referred to as regressors $Z_t$, is heavily influenced by the particular problem being analyzed. It can comprise of observations from the recent past, such as $X_{t-1}, ..., X_{t-P}$ over a look-back horizon $P$, as well as any relevant external observations. If the time series data shows signs of (quasi-) seasonality, the regressors may also include observations from further back in time, like $X_{t_0}$, where $t_0$ is significantly less than $t$.

### 3.1.1 Recurrent Neural Network as Universal Approximators

Recurrent Neural Networks (RNNs) are universal approximators for open dynamical systems. This means they can learn complex, time-dependent patterns and approximate the state transitions and outputs of systems with stochastic inputs to arbitrary

accuracy [192].

An RNN can be represented in a state-space form, where at each time step $k$, it computes a new hidden state $\tilde{s}_k$ based on the previous state $\tilde{s}_{k-1}$ and an external input $\alpha_k$:

$$\tilde{s}_{k+1} = \eta(\tilde{s}_k, \alpha_{k+1})$$

The output at each time step $k$ is then calculated from the hidden state:

$$\tilde{\beta}_k = \xi(\tilde{s}_k)$$

where $\eta$ represents the state transition function, $\xi$ represents the output function and $\alpha_k$ is a stochastic input, meaning it has a probabilistic distribution rather than fixed values.

For a target system defined by:

$$s_{k+1} = \eta(s_k, \alpha_{k+1})$$
$$\beta_k = \xi(s_k)$$

we want the RNN-based model's hidden states $\tilde{s}_k$ and outputs $\tilde{\beta}_k$ to closely match $s_k$ and $\beta_k$ from the target system, respectively.

The error between the target and the RNN's state can be expressed as:

$$e_k = \|s_k - \tilde{s}_k\|_1$$

Using a Lipschitz condition on $\eta$, it is shown that:

$$e_{k+1} \leq C_{\eta 1} e_k + \epsilon$$

where $C_{\eta 1}$ is a constant that must satisfy $|C_{\eta 1}| < 1$ to ensure convergence over time, and $\epsilon$ is a small error term that can be minimized.

By iterating this inequality, the paper shows that the error at any future time step $k$ can be bounded as:

$$\lim_{k \to \infty} e_k \leq \frac{\epsilon}{1 - C_{\eta 1}}$$

This formula means that, with a sufficiently small $C_{\eta 1}$, the error between the RNN state $\tilde{s}_k$ and the target state $s_k$ can be made arbitrarily small.

To ensure the output $\tilde{\beta}_k$ also approximates $\beta_k$ accurately, the paper requires the output function $\zeta$ to be continuous. Then, as the state error $e_k$ becomes small, the output error also diminishes, satisfying:

$$\lim_{k\to\infty} \|\beta_k - \tilde{\beta}_k\|_1 < \epsilon$$

where $\epsilon$ represents the overall approximation accuracy of the RNN for the entire dynamical system, including both state and output.

In order to extend universal approximation theorem, which states that feedforward networks can approximate any continuous function over compact domains, to RNNs for dynamical systems, we want to prove the Equation 3.2. Given a compact domain $S \subset \mathbb{R}^n$, and a continuous function $f : S \to \mathbb{R}$, there exists an RNN function $\zeta$ that satisfies:

$$\sup_{x\in S} |f(x) - \zeta(x)| < \delta \tag{3.2}$$

for any $\delta > 0$, where $\zeta$ is defined by an RNN's state transition and output functions.

The RNN's ability to approximate complex dynamical systems stems from its recursive nature, enabling it to capture both deterministic and stochastic dependencies. The approximation holds because the RNN can mimic the transitions and outputs of the system to any desired precision, provided certain conditions (like Lipschitz continuity and boundedness) are met.

The main results are captured in these inequalities and the recursive relationship, which prove RNNs' capacity as universal approximators of dynamical systems with stochastic inputs.

## 3.2 Conceptional Design

Our goal is to propose a hypothesis $f$ that effectively captures the complex, nonlinear relationships between the input regressors $Z_t$ and the temporal evolution of $X_t$, while also being computationally efficient to infer from observed measurements. Additionally, as both the inputs and outputs are observed sequentially, it is crucial that the model $f$ be capable of characterizing the temporal dependencies between observations.

Multiple hypotheses that can explain measurements based on a task-dependent criterion can be inferred from a single set of observations. It should be noted, however, that the objective of inferring the model $f$ is not only to explain the observed measurements, but also to generalize the out-of-sample predictive horizons. To this

end, the principle of parsimony, known as Occam's razor, is often followed by researchers. This principle suggests that when there are competing hypotheses that explain the data equally well, the simplest one should be chosen. Models that fit the data too well may be overfitting or failing to capture the underlying relationship between regressors and time series. How to avoid overfitting is, therefore, a crucial requirement for time series prediction tasks.

For time series analysis, there are three main types of models being used: statistical models, mathematical models, and deep learning models, as detailed in the following:

1. Statistical models: A statistical model is a mathematical model that embodies a set of statistical assumptions about how sample data and similar data from a larger population is generated. These models represent the data-generating process in an idealized manner. Typically, a statistical model is defined by a mathematical relationship between one or more random variables and other non-random variables. Therefore, a statistical model is regarded as a formal representation of a theory [193]. Statistical models play a critical role in statistical inference, as all statistical hypothesis tests and estimators are derived using them.

2. Mechanistic Model: Mechanistic models are a kind of mathematical model that aims to explain the behavior of a system or process by specifically representing the underlying mechanisms that influence its behavior. These models require a deep understanding of the scientific principles governing the system, such as physics, chemistry, or biology, in order to accurately account for the key factors that contribute to its behavior. By using mechanistic models, it is possible to make predictions about how the system will behave in different circumstances, optimize its performance, and gain a deeper understanding of its underlying mechanisms. These models are widely used across different areas such as engineering, physics, chemistry, biology, ecology, and economics.

3. Recurrent Neural Network Models: A recurrent neural network (RNN) is a kind of artificial neural network that has connections between its nodes that can form a cycle. This feature allows information to move in a loop, allowing the output of some nodes to affect the input of the same nodes in the future. Consequently, RNNs are able to display temporal dynamic behavior, which is useful for handling input sequences that have varying lengths. Moreover, because they are theoretically Turing complete, recurrent neural networks possess the computational power to process sequences of input in any order and

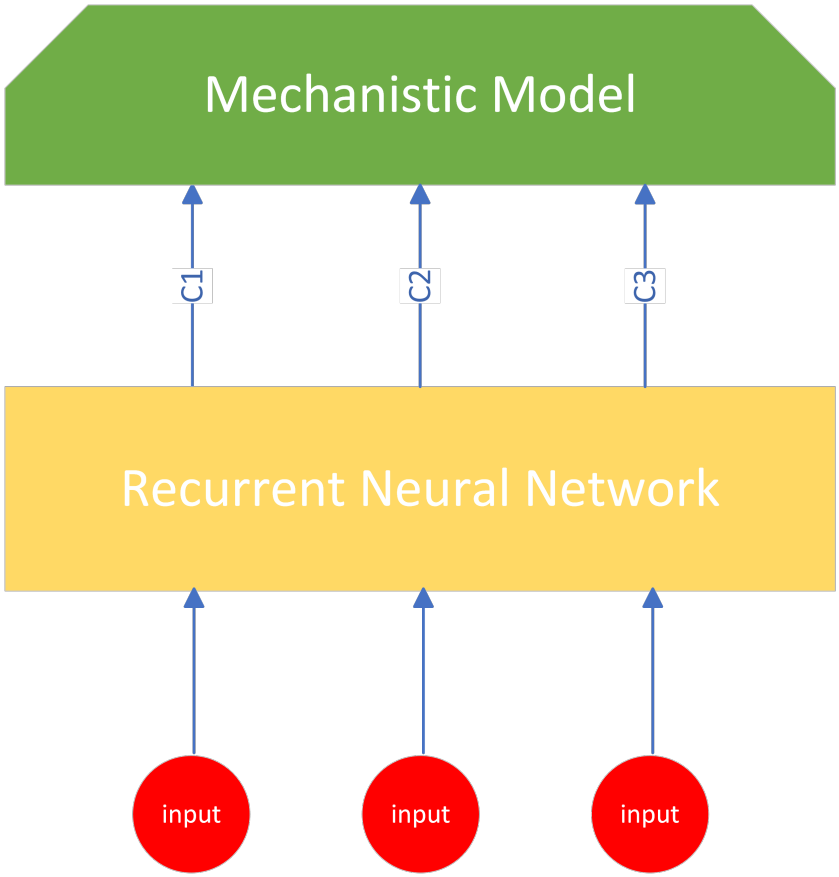



FIGURE 3.1: Conceptual Design of Hybrid Model

run any program. Thus, they are a versatile tool that has many applications in various fields, including natural language processing, image processing, and robotics.

In this research, we aim to combine an RNN with a mechanistic model to perform time series analysis, as shown in Figure 3.1.

From the first principle, the fertilizer system could be considered as a reaction-diffusion system, as shown in Equation 3.3.

$$\begin{aligned} \frac{\partial C}{\partial t} &= D_0 \nabla^2 C + F(C,B) \\ \frac{\partial B}{\partial t} &= D_1 \nabla^2 B + R(C,B) \end{aligned} \tag{3.3}$$

where $C$ represents the unknown vector function, $D$ is a diagonal matrix of diffusion coefficients, and $R$ accounts for all local reactions.

In practice, it is a challenge for researchers to find the differential equations to describe the dynamical behavior of the system. Basically, we can only study the differential equations theoretically. In most cases, researchers will find an algebraic

equation to approximate the behavior of the dynamical system. However, most of the time, the algebraic equation cannot approximate the dynamical system perfectly. As introduced in this thesis, the DLEM model, which is a terrestrial ecosystem model that couples major biogeochemical cycles, the hydrological cycle, and vegetation dynamics to estimate daily carbon, nitrogen, and water fluxes, is less powerful in representing the dynamical system. The differences between these two types of equations (differential equations and algebraic equations) are discussed in the following.

1. Many dynamical systems, particularly those involving changes over time or space, are described by differential equations. Differential equations involve derivatives, which represent rates of change. Algebraic equations, on the other hand, typically do not involve derivatives and describe relationships at specific points or states without capturing the dynamic behavior over time.

2. Dynamical systems often involve continuous changes in variables, while algebraic equations are typically used to represent relationships between discrete values or states. This discreteness can lead to a loss of precision when approximating continuous and smooth phenomena.

3. Many real-world dynamical systems exhibit complex, nonlinear, or chaotic behavior that cannot be accurately captured by simple algebraic equations. Differential equations provide a more flexible framework for modeling these complex dynamics.

4. Many dynamical systems require initial conditions or boundary conditions to specify their behavior.

## 3.3 Methodology

As mentioned in Section 3.2, the approach we take for time series analysis is to combine RNN with the mechanistic model and take advantage of the benefits of both models to achieve better performance. This investigation will apply this method to predict the emissions of $N_2O$, which will be detailed in Chapter 4. In this chapter, a simple example is presented to illustrate the underlying mechanism of the methodology and discuss the validity and effectiveness of the proposed hybrid model.

Since the RNN simulates the behavior of a dynamical system, it can be represented as a system of differential equations. Knowledge of the differential equation allows us to define a solution space (usually a functional space), denoted as the M space in Figure 3.2. For effective RNN learning from the dynamical system using

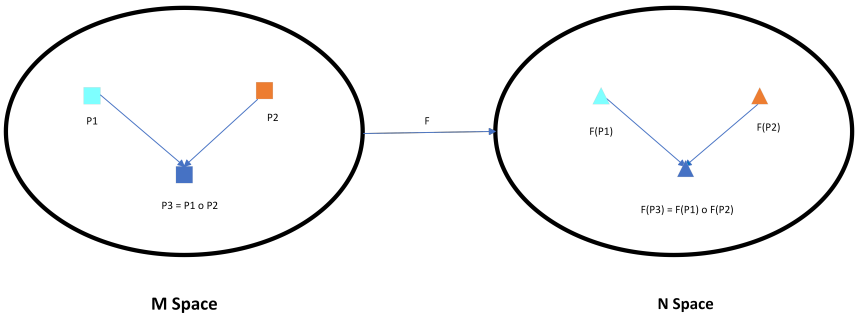



FIGURE 3.2: Neural Network Learning from Dynamical System

real-life data, it is essential to scale the dataset to the range of [0, 1] or [-1, 1]. This scaling is achieved through a mapping function F, illustrated in Figure 3.2. The mapping function, practically a min-max scaling or other types of transformations, allows us to convert data with varying units and ranges into the range of [0, 1], forming the N space (Figure 3.2). As a result, the mapping function F serves as a structure-preserving mapping and is considered an isomorphism.

Ideally, with a sufficient amount of data collected from the dynamical system, an adequately complex RNN model can learn the entire structure of the N space, which serves as an isomorphism of the M space, and effectively preserve the relations (mappings) of the elements within the M space. However, in practice, gathering ample data poses challenges, leading to the collection of only local information in the state-space. Consequently, our RNN model is often trained on insufficient and imbalanced data. In such situations, to enhance the capability of our model in simulating the real dynamical system, we can provide additional information, such as the general algebraic relationships between variables. Incorporating this supplemental knowledge empowers the model to navigate complexities more efficiently and improve its accuracy.

### 3.3.1 Mathematical Formulation

In this chapter, our objective is to establish a comprehensive theory that validates the mathematical soundness of the hybrid model. We aim to demonstrate that the hybrid model can be rigorously formulated through a strict mathematical process. To illustrate the application of this theory, we will utilize the Lorenz System as a specific example. The entire process can be outlined as follows:

$$G_t = C(Lorenz(x_t, x_{t-1}, x_{t-2} \ldots, x_{t-n})) \tag{3.4}$$

where $C$ represents the mechanistic model, *Lorenz* is the dynamical system we want to simulate, and $G_t$ is the output of the system. In order to illustrate the validity, it is necessary for us to use the LSTM to replace the Lorenz System part, as shown in the following:

$$G_t = C(LSTM(x_t, x_{t-1}, x_{t-2} \ldots, x_{t-n})) \tag{3.5}$$

In the following experiments, five experiments are designed for serving different purposes. In the first four experiments, there is a data transformation in the interface of $C$ and $LSTM$. Then the entire system could actually represented as:

$$G_t = C(T(LSTM(x_t, x_{t-1}, x_{t-2} \ldots, x_{t-n}))) \tag{3.6}$$

where $T$ is the data transformation operation. This interface separates the LSTM from the mechanistic model; thus, the analysis is very straightforward. You can find the detail and concrete analysis in Section 3.5.6.

In this section, we focused on the fifth experiment, which scales both the input and output in the range of [-1, 1] or [0, 1] without interface $T$. In essence, the mechanistic model is fed with scaling data but not the real data, while the output is also in the range of [0, 1], not the real output. It is difficult to understand at first sight. Let us discuss it step by step. Without loss of generality, the N is used as the operator to scale the data to the range of [-1, 1] or [0, 1]; therefore, we could have the following formulas:

$$N(G_t) = N(C(LSTM(x_t, x_{t-1}, x_{t-2}))) \tag{3.7}$$

In order to make the discussion more reasonable, the C, LSTM, and N are considered to be operators. Then, we need to introduce the concept of commutator, as shown in the following:

$$[A, B] = AB - BA \tag{3.8}$$

where A and B are both operators. If their commutator is considered to be 0, then A and B commute. However, in our experiment, it became apparent that C, LSTM, and N do not commute. For convenience, we can treat the combination of C and LSTM as a single operator, denoted as D. Subsequently, we define $\Delta$ as the difference between ND and DN, as shown in the following:

$$[N, D] = ND - DN = \Delta \tag{3.9}$$

Equation 3.7 can be transformed into the following form:

$$N(G_t) = ND = DN + \Delta = C(LSTM(N(x_t), N(x_{t-1}), N(x_{t-2}))) + \Delta \quad (3.10)$$

In what situation is Equation 3.10 met? We could consider the entire map as $\mathcal{T}$, which consistes of $C$ and $LSTM$; thus. we have

$$\mathcal{T} : [0,1]^n \longrightarrow [0,1] \quad (3.11)$$

It is straightforward to observe that in our experiment, the hybrid model is continuous in the space of $[0,1]^n$ or $[-1,1]^n$, and thus in practicality, the model will output values. Then in this situation, if we could not obtain the accurate range of the parameters, we could simply scale the both input and output in the range of [0, 1] or [-1, 1]. However, we could still pay attention to the additional $\Delta$ in Equation 3.10; it actually adds more of a work load in training the LSTM operator. This analysis is consistent with the observation we have from the experiments 1 to 5.

## 3.4 A Lorenz System Example

### 3.4.1 Description of the Lorenz System

In the year 1963, Edward Norton Lorenz, an esteemed meteorologist and mathematician, introduced a groundbreaking three-dimensional system known as the Lorenz System. This system emerged from a fluid convection model and offered a simplified representation of atmospheric convection [194]. The normalized form of the Lorenz System is expressed as follows:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \quad (3.12)$$

where $(x, y, z) \in \mathbb{R}^3$, and $\sigma$, $\rho$, and $\beta$ are all positive values. Here, $\sigma$ represents the Prandtl coefficient, $\rho$ corresponds to the Rayleigh coefficient, and $\beta$ signifies the aspect ratio coefficient.

FIGURE 3.3: Example solutions of the Lorenz System for different coefficients of $\sigma$, $\beta$, and $\rho$

As is widely recognized, the Lorenz System displays chaotic behavior. It showcases distinct trajectories when subjected to varying parameters and initial conditions, as depicted in Figure 3.3.

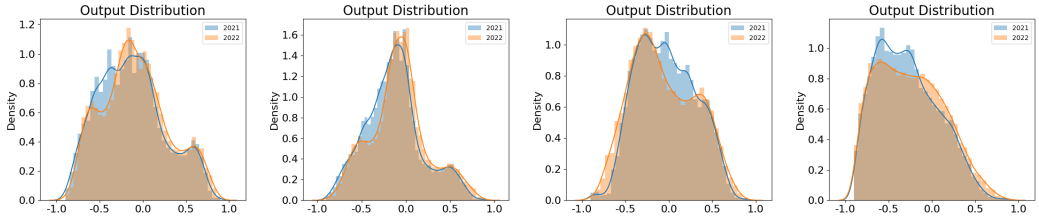



FIGURE 3.4: Example solutions of the Lorenz System for different Coefficients of $\sigma$, $\beta$, and $\rho$

The distribution of x, y, and z are shown in Figure 3.4.

### 3.4.2 Data Collection and Description

The training data and testing data used in this experiment are generated by the Lorenz System with different parameters and initial conditions. The detailed information is shown in the following:

- For training data, 10000 data points were generated from solving the Lorenz System with the parameters $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ and with initial conditions $x = 0.1$, $y = 1$, and $z = 2.05$. The time span for these 10000 samples range from 0 to 50. The trace of each dimension is shown in Figure 3.5.

- For testing data, 10000 data points were generated from solving the Lorenz System with the parameters $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ and with initial conditions $x = -1$, $y = 2$, and $z = 1.05$. The time span for these 10000 samples range from 0 to 50. The trace of each dimension is shown in Figure 3.6.

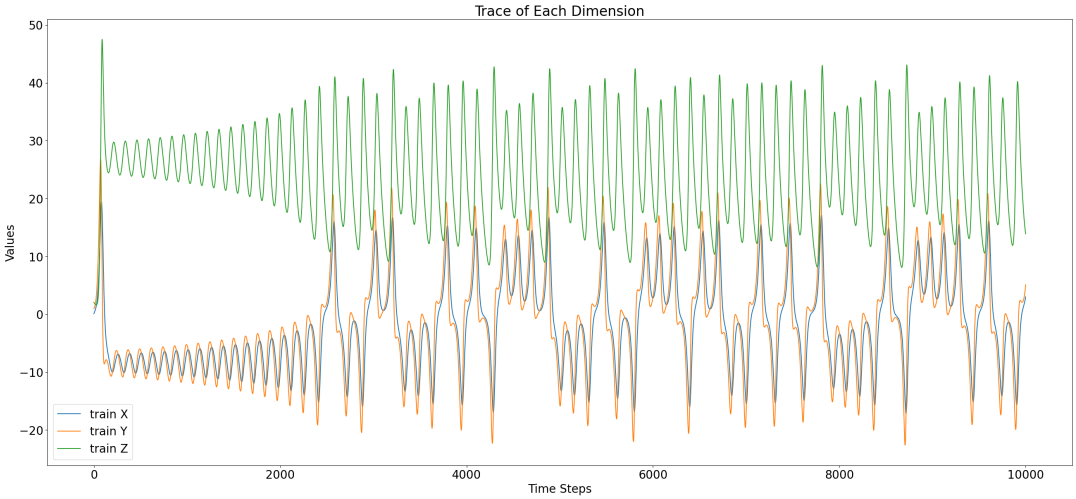



FIGURE 3.5: Training Dataset

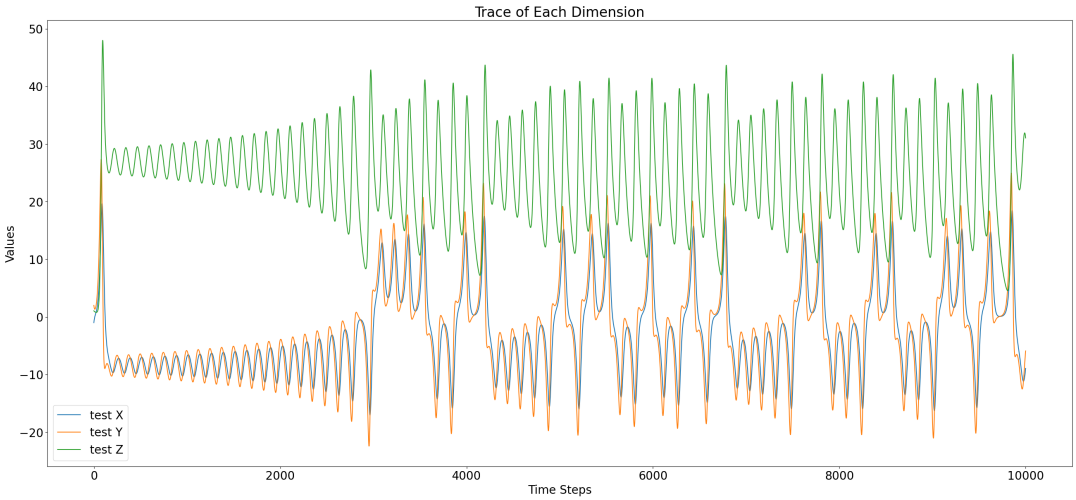



FIGURE 3.6: Testing Dataset

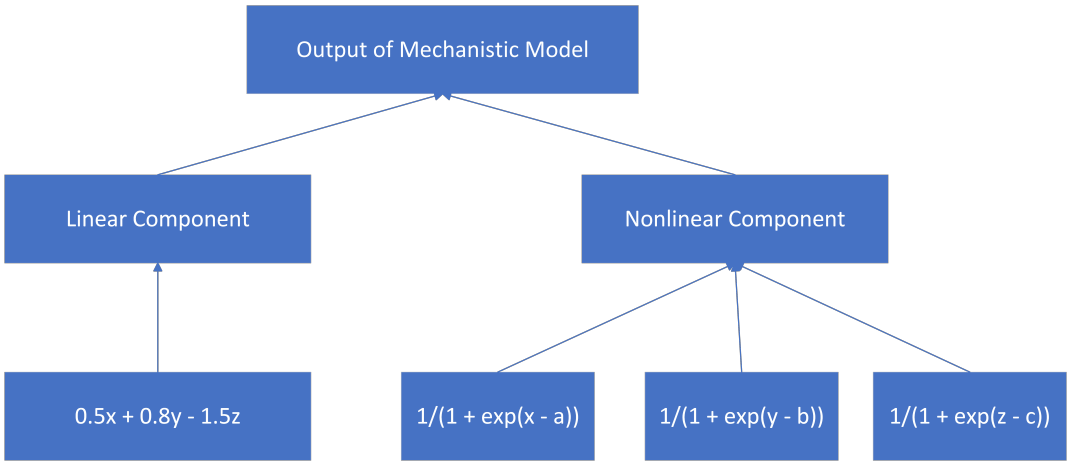



FIGURE 3.7: Illustration of the Mechanistic Model

The mechanistic model is designed to incorporate both the linear and nonlinear components, as demonstrated below:

- linear component: $0.5x + 0.8y - 1.5z$ .
- nonlinear component: $\frac{1}{1+exp(x-\alpha)} + \frac{1}{1+exp(y-\zeta)} + \frac{1}{1+exp(z-\gamma)}$ .
- summary = linear component + nonlinear component .

Then, the training dataset and testing dataset are fed into the mechanistic model to calculate their respective values. Please note that $\alpha$, $\zeta$, and $\gamma$ represent the expected values of $x$, $y$, and $z$, respectively, in this case. Since the training dataset differs from the testing dataset, the corresponding $\alpha$, $\zeta$, and $\gamma$ used to evaluate the training data and testing data also differ. The purpose of this design is to demonstrate whether the hybrid model can adapt to various situations. If the hybrid model can successfully fit into different environments, it would be convenient for us to design a universal model capable of training on one dataset and testing on another, which may be collected under diverse conditions. Here, I present a simple case to explore the feasibility of this idea.

### 3.4.3 Data Pre-Processing

In this particular case, the generated data falls outside the range of $[-1, 1]$, posing a challenge for the neural network in handling such data. Therefore, it is necessary for us to scale the data within the range of $[-1, 1]$ to ensure optimal performance of the recurrent neural network model. In this study, the min-max normalization approach is employed to scale the data within the range of $[-1, 1]$, as illustrated below:

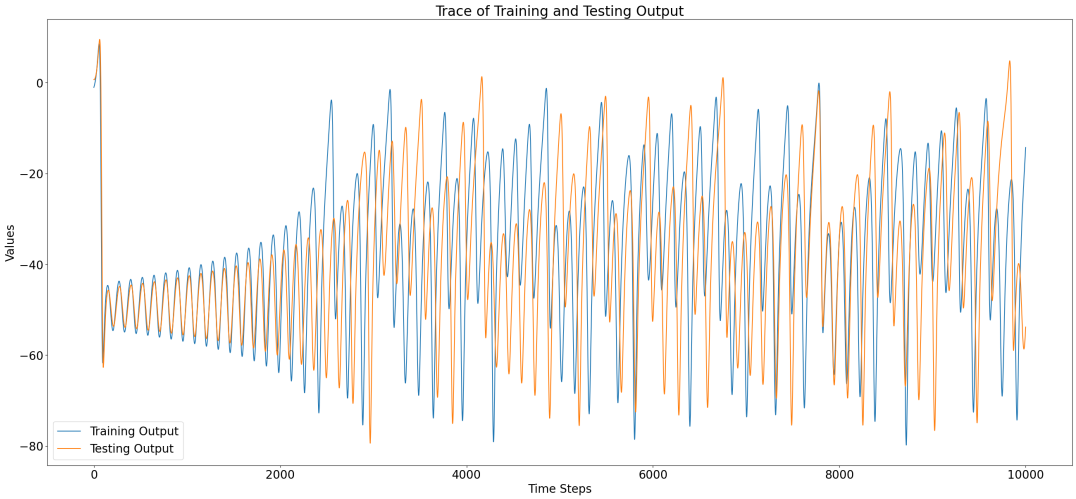



FIGURE 3.8: Trace of Training and Testing Output

$$x^{'} = \frac{(x - min(x))(b - a)}{max(x) - min(x)} \tag{3.13}$$

Here, $x'$ represents the scaled value, while $x$ represents the original value. The variables $a$ and $b$ correspond to the maximum and minimum values, respectively. The scaled data can be represented as a vector at each time-step, as depicted below:

$$[X_t, Y_t, Z_t]$$

where $X_t$, $Y_t$, and $Z_t$ represent the values of the random variables $x$, $y$, and $z$ at time step $t$. To construct a block of time series, data points from the previous n time steps are required. Therefore, the input can be represented as a data matrix, as illustrated below:

$$\begin{bmatrix} M_{t-n} \\ M_{t-n+1} \\ \vdots \\ M_{t-1} \end{bmatrix} = \begin{bmatrix} X_{t-n} & Y_{t-n} & Z_{t-n} \\ X_{t-n+1} & Y_{t-n+1} & Z_{t-n+1} \\ \cdots & & \\ X_{t-1} & Y_{t-1} & Z_{t-1} \end{bmatrix}$$

Please note that in this section, there is no need to pre-process the output of both the training set and the testing set. Instead, data transformation will be performed at the interface between the recurrent neural network model and the mechanistic model. This transformation switches the coefficient output from the recurrent neural network to the actual value in the mechanistic model, as demonstrated in Equation 3.14. The detailed process will be discussed in the following section.

## 3.5 Simulation and Experiment

In this study, the BDLSTM is employed as the recurrent neural network to capture the dynamic features of sequential data. Subsequently, the output of the BDLSTM is fed into the mechanistic model. However, prior to that, it is necessary to convert the coefficients received from the BDLSTM, which are within the range of $[-1,1]$, into practical data. The formulas for this conversion are presented below:

$$M = (C+1) \times \frac{(X_{max} - X_{min})}{2} + X_{min} \tag{3.14}$$

where $C$ is the coefficients received from the BDLSTM. Once the real value of $M$ is obtained, we feed it into the mechanistic model and calculate the output. In the final step, we compare the expected output with the actual output and take the mean square error as the loss value to tune the weight of the network with error backpropagation approach. For this case, the LSTM neural network models are implemented using PyTorch. The Adam optimizer [112] combined with an early stopping strategy is used to train the hybrid model.

### 3.5.1 The First Case: A General Case

In this case, the normal training dataset and testing dataset, generated from the aforementioned process, are utilized without any modifications. Both the training dataset and testing dataset are then fed into the mechanistic model, allowing us to calculate the outputs for each dataset. The results are presented in Figure 3.9.

Following the aforementioned process, once the hybrid model is appropriately trained, we proceed to feed the training data and testing data into the hybrid model, with varied $[\alpha, \zeta, \gamma]$ values depending on the model's status. The final regression performance is depicted in Figure 3.10. For the sake of detailed regression accuracy, only the first 400 points are illustrated.

It is important to note that during the training and testing phases, the $[\alpha, \zeta, \gamma]$ values vary. Specifically, in this particular case, the $[\alpha, \zeta, \gamma]$ values for the training dataset are $[-1.8645, -1.8585, 23.9183]$, while for the testing dataset, they are $[-2.6575, -2.6880, 23.7362]$.

When the hybrid model is evaluated using the testing dataset, if the $[\alpha, \zeta, \gamma]$ values do not change from $[-1.8645, -1.8585, 23.9183]$ to $[-2.6575, -2.6880, 23.7362]$, the output exhibits slight differences but remains relatively consistent, as depicted in

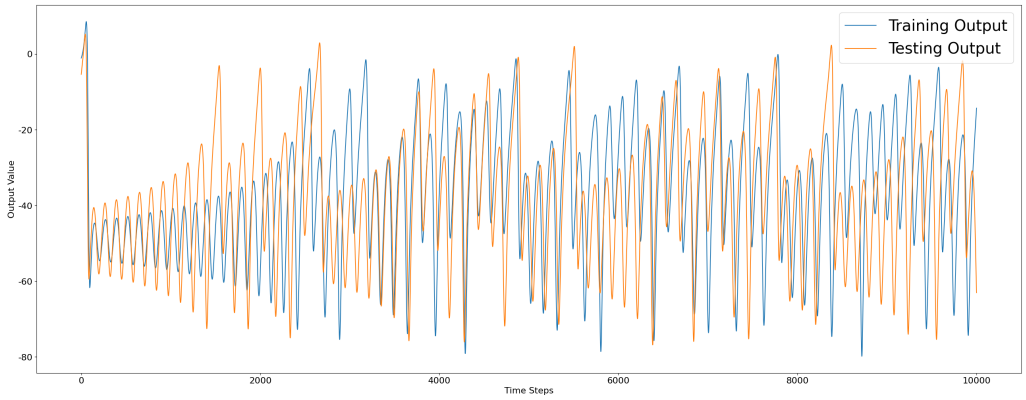



FIGURE 3.9: The Outputs of the Training Dataset and the Testing Dataset

Figure 3.11. However, another experiment is intentionally designed to create significant deviations. This experiment aims to demonstrate the capability and advantages of the hybrid model with varying parameters.

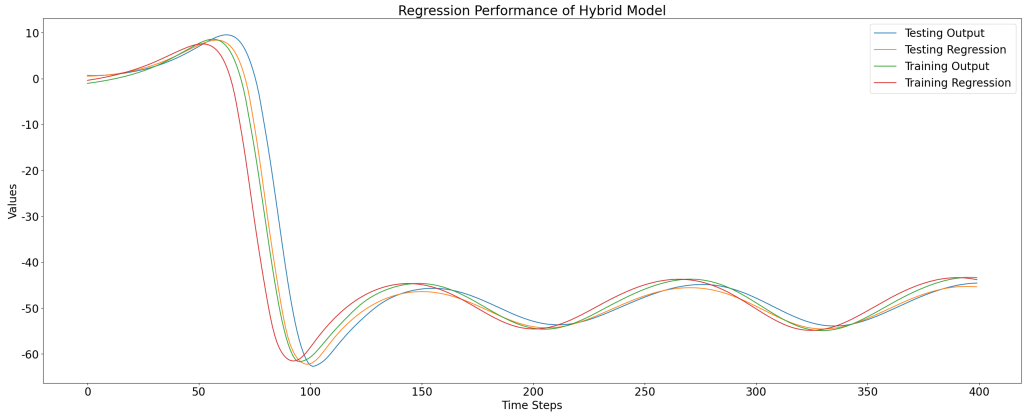



FIGURE 3.10: Regression Performance of the Hybrid Model

### 3.5.2 The Second Case: Further Study of the Adaptation Capability of the Hybrid Model

In the second case, to evaluate the adaptation capability of the hybrid model, it becomes necessary to modify the testing dataset. In this instance, we can apply a linear transformation to the dataset by subtracting 5 from all numbers in the x-dimension, adding 3 to all numbers in the y-dimension, and subtracting 10 from all numbers in the z-dimension. These transformations preserve the periodicity and distribution of the original data. However, the $[\alpha, \zeta, \gamma]$ values for the testing dataset change from $[-2.6575, -2.6880, 23.7362]$ to $[-7.6575, 0.3120, 13.7362]$. The training dataset

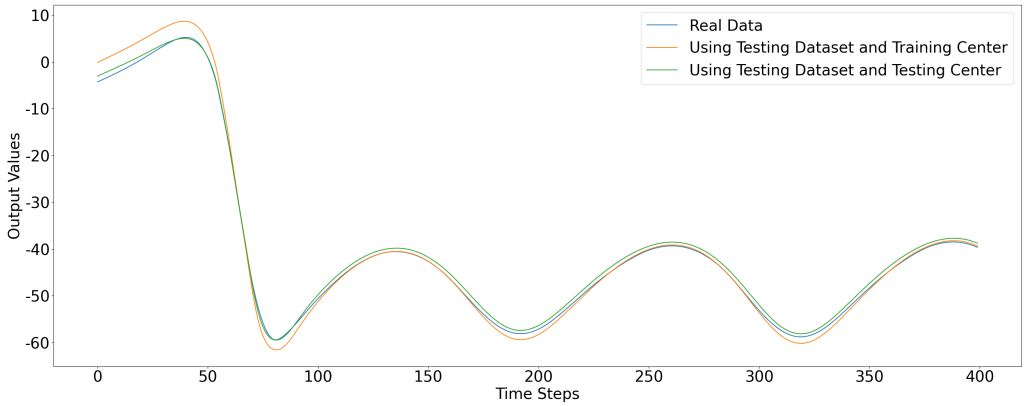



FIGURE 3.11: Regression Performance of the Hybrid Model in different Coefficients $[\alpha, \beta, \gamma]$

remains unchanged. We can now repeat the experiment, and the outputs for both the training dataset and testing dataset are presented in Figure 3.12.

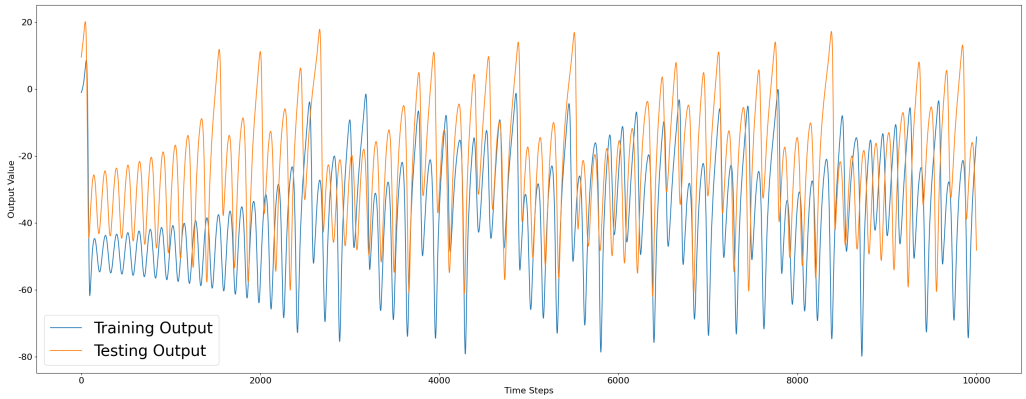



FIGURE 3.12: The Outputs of the Training Dataset and the Testing Dataset

However, in the second experiment, the outputs of the training dataset and testing dataset exhibit noticeable differences. If the $[\alpha, \zeta, \gamma]$ values do not change from $[-1.8645, -1.8585, 23.9183]$ to $[-7.6575, 0.3120, 13.7362]$ during the evaluation of the hybrid model using the testing dataset, the final results will demonstrate a larger deviation than anticipated, as depicted in Figure 3.13. The final outputs for both the training dataset and testing dataset are presented in Figure 3.14.

### 3.5.3 The Third Case: Scaling the Entire Testing Dataset

In the third case, to assess the adaptation capability of the hybrid model, it is necessary to modify the testing dataset. In this scenario, all the values in the x-dimension, y-dimension, and z-dimension are multiplied by a factor of 0.1. As a result, the periodicity and distribution of the original data remain unchanged. However, the

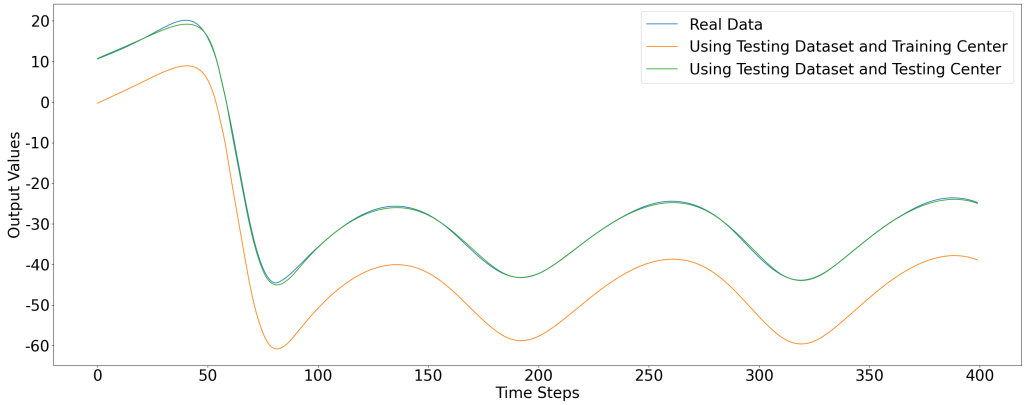



FIGURE 3.13: The Adaptive Outputs of the Testing Dataset

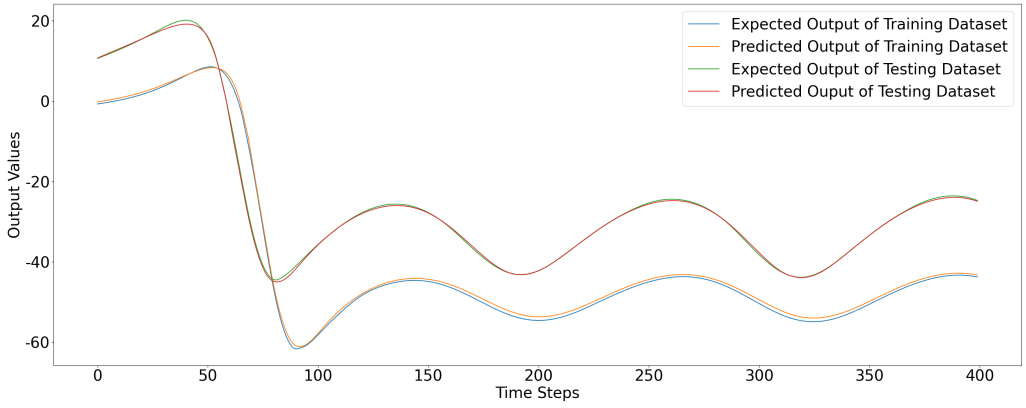



FIGURE 3.14: The Regression Performance of Training Dataset and the Testing Dataset

$[\alpha, \zeta, \gamma]$ values for the testing dataset change from $[-2.6575, -2.6880, 23.7362]$ to $[-0.2658, -0.2688, 2.3736]$. The training dataset remains the same. We proceed to repeat the experiment, and the outputs of the training dataset and the testing dataset are displayed in Figure 3.15.

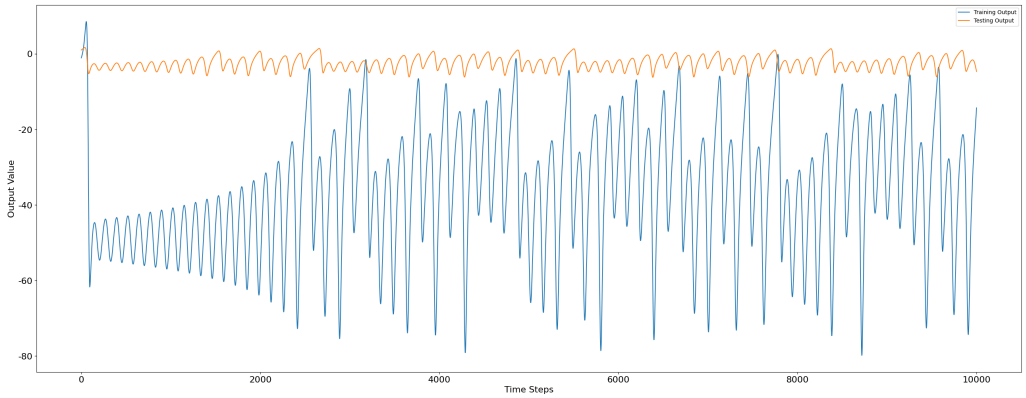



FIGURE 3.15: The Outputs of the Training Dataset and the Testing Dataset

However, in the third experiment, the outputs of the training dataset and testing dataset exhibit notable differences. If the $[\alpha, \zeta, \gamma]$ values do not change from $[-1.8645, -1.8585, 23.9183]$ to $[-0.2658, -0.2688, 2.3736]$ during the evaluation of the hybrid model using the testing dataset, the final results will demonstrate a larger deviation than anticipated, as depicted in Figure 3.16. The final outputs for both the training dataset and the testing dataset are presented in Figure 3.17.

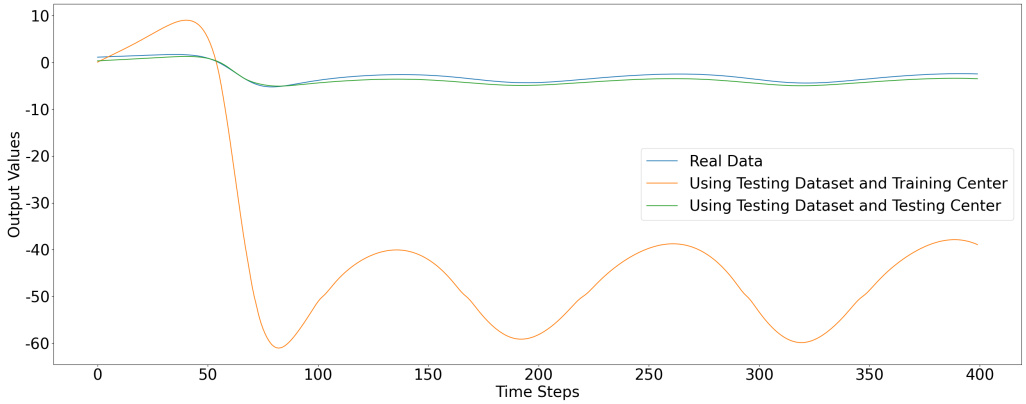



FIGURE 3.16: The Adaptive Outputs of the Testing Dataset

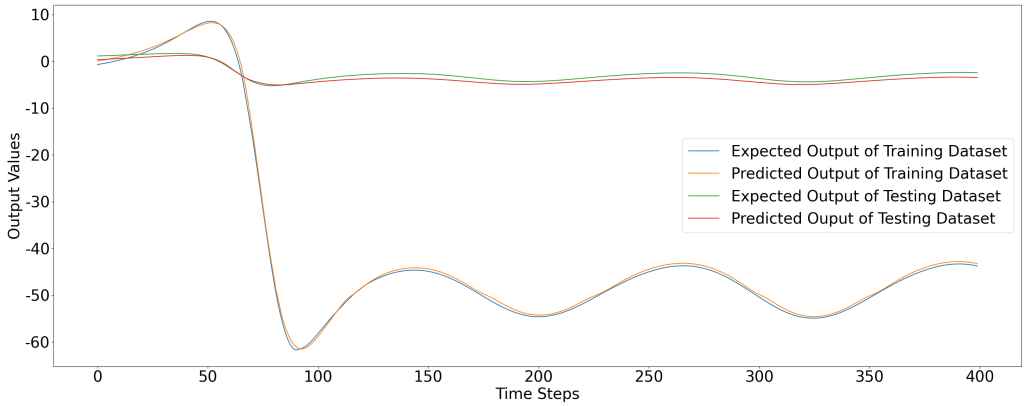



FIGURE 3.17: The Regression Performance of the Training Dataset and the Testing Dataset

### 3.5.4 The Forth Case: Changing the Parameters in the Hybrid Model

Through the aforementioned experiment, it could be easily concluded that if we only change the parameters in the hybrid model, $\alpha$, $\zeta$, and $\gamma$, the model could still be trained in the training dataset and achieve similar performance on the testing dataset, even if the output of both datasets is quite different from each other. It indicates that recurrent neural network in the hybrid model could learn the dynamical and nonlinear features in the system, and the mechanistic system in the hybrid model could adapt to different environments and situations through changing the parameters. In this simple case, we fixed the training centers to $[100, -100, 100]$ and the testing center to $[0.01, 0.01, -0.01]$. The output of the training dataset and the testing dataset is shown in Figure 3.18.

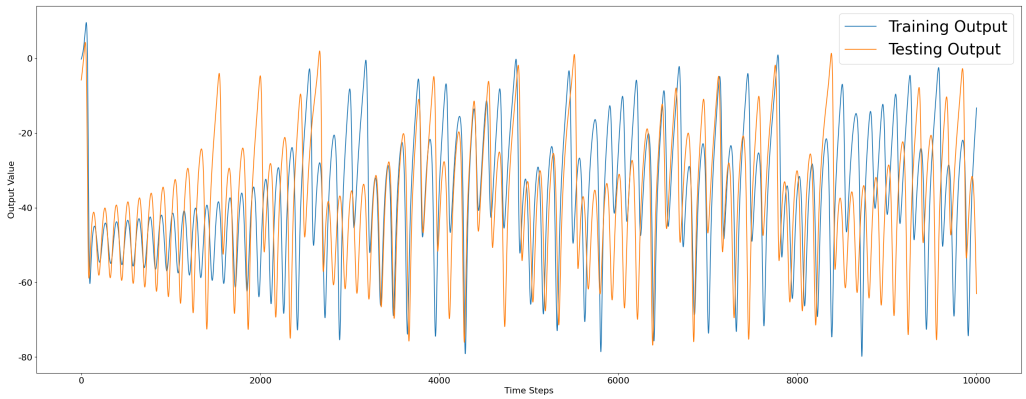



FIGURE 3.18: The Outputs of the Training Dataset and the Testing Dataset

The corresponding regression performance of the training dataset and testing

dataset using different parameters, here, it means, training centers and testing centers, is shown in Figure 3.19.

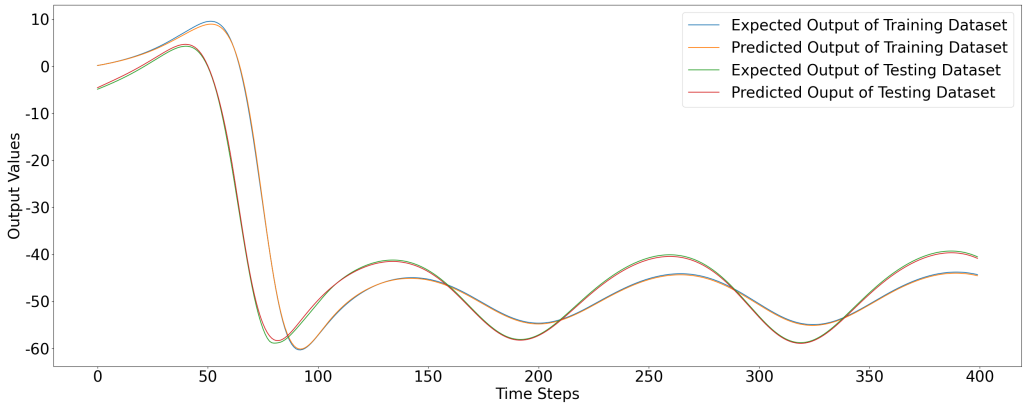



FIGURE 3.19: The Regression Performance of the Training Dataset and the Testing Dataset

The hybrid model could perform similar accuracy in both the training dataset and the testing dataset. This kind of regression effect could not be achieved by the normal deep learning model only.

### 3.5.5 The Fifth Case: Scaled Input and Output Data in the range of [-1, 1] or [0, 1]

Most of the time in practice, it is challenging to determine a suitable range for the parameters. Consequently, at the interface of the RNN and the mechanistic model, we cannot employ the data transformation Equation 3.14 due to the lack of knowledge about parameter ranges. In light of this, a crucial question emerges: Can the proposed architecture still function effectively under these circumstances? Based on observations, the mechanistic model demonstrates continuity within the range of [-1, 1] or [0, 1]. This continuity allows us to utilize its structural information and train the model using scaled data, effectively achieving our objectives. The underlying mathematical mechanism is detailed in Section 3.3.1.

The $[\alpha, \zeta, \gamma]$ values are varied in the training phase and the testing phase. For example, the $[\alpha, \zeta, \gamma]$ values for the training dataset are $[-1.8645, -1.8585, 23.9183]$, while for the testing dataset, they are $[-2.6575, -2.6880, 23.7362]$. The corresponding regression performance of the training dataset and testing dataset using different parameters is shown in Figure 3.20. It is convenient to observe, even in this case, that the output of the testing data could fit the expected value well.

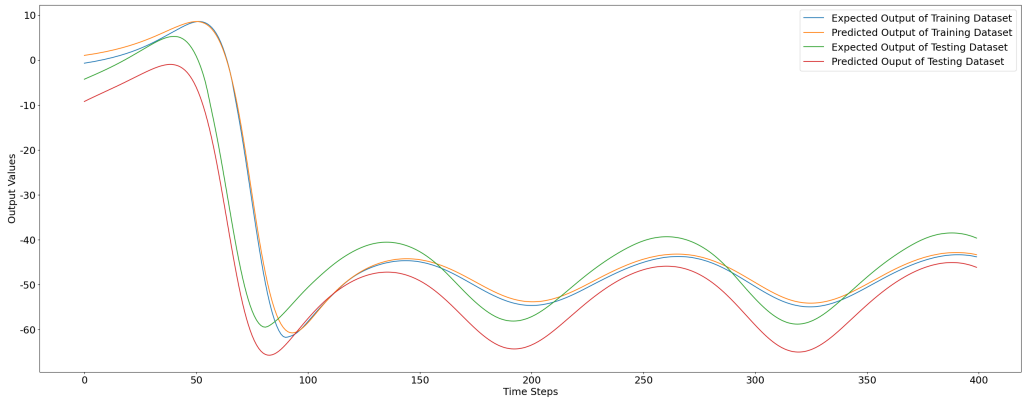



FIGURE 3.20: The Regression Performance of the Training Dataset and the Testing Dataset

However, if the $[\alpha, \zeta, \gamma]$ values are fixed to [1, 1, 1], the corresponding regression performance of the training dataset and testing dataset using the same parameters is shown in Figure 3.21. The expected curve and the predicted curve fit perfectly.

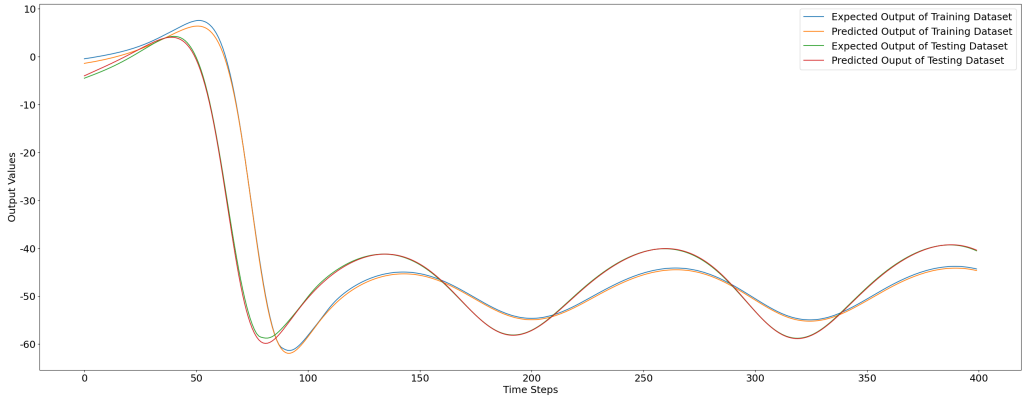



FIGURE 3.21: The Regression Performance of the Training Dataset and the Testing Dataset

Therefore, if the mechanistic model is continuous in the range [-1, 1] or [0, 1], in the situation when the accurate parameters in the mechanistic model could not be obtained, scaling all the data in the range [-1, 1] or [0, 1] did not change the dynamical behavior of the hybrid model.

### 3.5.6 Experiment Analysis

In this section, we will discuss why, in this particular case, the model remains invariant to the shift and the entire dataset's scaling. The hybrid model comprises two components: the linear component and the nonlinear component. The linear component and the nonlinear component are illustrated below:

$$\begin{aligned} &0.5 \times x + 0.8 \times y - 1.5 \times z \\ &\frac{1}{1+e^{x-\bar{X}}} + \frac{1}{1+e^{y-\bar{Y}}} + \frac{1}{1+e^{z-\bar{Z}}} \end{aligned} \tag{3.15}$$

As we can observe, the linear component contains an invariant part when the dataset is shifted or scaled as a whole. Let's assume that we want to shift the x, y, and z dimensions by a, b, and c units, respectively. In that case, we have the following equations:

$$\begin{aligned} &0.5 \times (x-a) + 0.8 \times (y-b) - 1.5 \times (z-c) \\ &\longrightarrow 0.5 \times x + 0.8 \times y - 1.5 \times z - (0.5a - 0.8b + 1.5c) \end{aligned} \tag{3.16}$$

So, it consists of an invariant part plus a constant term when shifting the x, y, and z dimensions by a certain number of units. As a result, it does not alter the behavior of the hybrid model.

While for the nonlinear component, we have:

$$\begin{aligned} &\frac{1}{1+e^{x-a-(\bar{X}-a)}} + \frac{1}{1+e^{(y-b)-(\bar{Y}-b)}} + \frac{1}{1+e^{z-c-(\bar{Z}-c)}} \\ &\longrightarrow \frac{1}{1+e^{x-\bar{X}}} + \frac{1}{1+e^{y-\bar{Y}}} + \frac{1}{1+e^{z-\bar{Z}}} \end{aligned} \tag{3.17}$$

So the nonlinear component does not change under the shifting operation.

If the entire dataset is scaled by a factor of $\zeta$, we have:

$$\begin{aligned} &0.5 \times \zeta x + 0.8 \times \zeta y - 1.5 \times \zeta z \\ &\longrightarrow \zeta(0.5 \times x + 0.8 \times y - 1.5 \times z) \end{aligned} \tag{3.18}$$

So it is a scaled factor times an invariant part. Thus, the behavior of the hybrid model could also still remain.

For the nonlinear component, we have:

$$\begin{aligned} &\frac{1}{1+e^{\zeta x-\zeta \bar{X}}} + \frac{1}{1+e^{\zeta y-\zeta \bar{Y}}} + \frac{1}{1+e^{\zeta z-\zeta \bar{Z}}} \\ &\longrightarrow (\frac{1}{1+e^{\zeta} \times e^{x-\bar{X}}} + \frac{1}{1+e^{\zeta} \times e^{y-\bar{Y}}} + \frac{1}{1+e^{\zeta} \times e^{z-\bar{Z}}}) \end{aligned} \tag{3.19}$$

So the basic form of the nonlinear component is not changed. Thus, the behavior of the hybrid model does not change in the situation where the entire dataset is scaled by a factor.

## 3.6 Conclusion Remarks

In this chapter, the limitations of current research on multivariate time series regression are discussed. In order to explore innovative approaches to analyzing multivariate time series data, a hybrid model consisting of LSTM and a mechanistic model is proposed to address the intrinsic issues observed in current research.

Based on the discussion above, it is evident that the hybrid model effectively simulates the dynamical behavior of the system. Moreover, if the range of the state variables in the mechanistic model is known in advance, it would benefit us in building a flexible hybrid model capable of tuning the state variables according to different situations. Even in situations where the range of the state variables is unknown or difficult to determine, if the mechanistic model is continuous in the space of $[0,1]^n$ or $[-1,1]^n$, we can scale the dataset within these ranges.

# Chapter 4

# Agriculture-informed Neural Networks for Predicting Nitrous Oxide Emissions

## 4.1 Introduction

In recent decades, the ultimate goal in the agriculture field has been to maximize crop yield in response to rapid population increase worldwide. Therefore, nitrogen fertilizers have been intensively applied to increase crop yields [195, 196, 197], leading to severe environmental degradation [198], which is closely linked to present ecological and climate changes because fertilized croplands are significant sources or sinks of greenhouse gases, such as carbon dioxide ($CO_2$), methane ($CH_4$), and nitrous oxide ($N_2O$) [199, 200]. As a significant greenhouse gas, $N_2O$ has a 300 times higher heat absorption potential than $CO_2$ [201] and is also the primary cause of stratospheric ozone depletion [202, 46]. To produce food in a highly efficient manner with the lowest possible environmental hazards, contemporary agriculture must shift from a single-goal to a multi-goal strategy [203].

Previously, statistical extrapolation approaches, which require collecting large amounts of sampling data, were used to estimate $N_2O$ emissions under natural conditions [204, 205]. However, such approaches become inaccurate or fail at finer spatial or temporal scales because of data limitations and an inability to describe highly complex and interactive microbial processes [206].

Alternatively, several process-based mechanistic models have been developed to approximate $N_2O$ emissions, including the Community Land Model with coupled Carbon and Nitrogen cycles (CLM-CN) [53], Dynamic Land Ecosystem Model (DLEM) [54, 55], and dynamic Land Model (LM3V-N) [56]. However, due to the large variability in forecasting $N_2O$ emissions, these models cannot completely and

effectively reveal the mechanisms of terrestrial $N_2O$ emissions. For example, estimating global terrestrial $N_2O$ emissions from natural sources is almost impossible, with estimates ranging from 3.3 to 9.0 $TgNyr^{-1}$ [207]. Furthermore, the biogenic $N_2O$ emissions from human activities on the land biosphere have not yet been thoroughly explored [67].

With the emergence of machine learning, researchers have sought to improve predictions of $N_2O$ emissions using artificial intelligence techniques. Initial attempts utilized various machine learning models [208], such as Multiple Layer Perceptron (MLP) [209, 210], Random Forest (RF) [211], and Long Short-Term Memory (LSTM) [18].

### 4.1.1 Motivations and Contributions

This research aims to address the environmental and climate challenges with a commitment to sustainable agriculture. Agriculture and Agri-Food Canada has initiated a program to reduce fertilizer emissions by 30 % from 2020 to 2030 [212]. Accurate prediction and monitoring of agricultural $N_2O$ emissions are crucial for understanding their environmental impact and implementing effective mitigation strategies [18]. This study aims to predict $N_2O$ emissions from agricultural fields using multivariate time series data, including the amount of nitrogen fertilizers in the soil, precipitation, air humidity, air temperature, soil moisture, and soil temperature. Therefore, the field experiments are conducted at Area X.O. Ottawa, as shown in Figure 4.1. However, predicting $N_2O$ emissions poses an inherent challenge [213].


(a) LI-COR Chamber in the Field (b) Overview of Area X.O

FIGURE 4.1: Images taken from Area X.O, Ottawa, Ontario, Canada

The emission of $N_2O$ is primarily related to two biological phenomena, nitrification and denitrification [214]. It can be considered a time-varying dynamic system that differential equations could describe, but obtaining these equations is difficult due to the high variability of soil $N_2O$ emissions [215]. To overcome this limitation,

inspired by the physics-informed neural network [216, 217] and chemistry-informed neural network [218], this study proposes a novel agriculture-informed neural network (AINN) model, which takes advantage of the deep learning model Recurrent Neural Network (RNN) [18] and DLEM from agricultural fields [54, 55].

Our contributions to this work are as follows:

- Combining a deep learning model and a process-based ecosystem model to form a novel AINN for accurately and efficiently predicting the emission of $N_2O$ in agricultural fields.
- Formulating the mechanism of AINN as a constrained optimization problem, which could improve the generalization of the model, with the DLEM component in AINN acting as a regularizer.
- Improving the explainability of the neural network model since the output obtained at the interface of the RNN component and DLEM component implicitly reveals the dynamic properties of the process of $N_2O$ emission.
- Exploring various neural networks and providing an in-depth understanding of how each model handles multivariate time series data.

### 4.1.2 Organization

The rest of the paper is structured as follows: Section 4.2 discusses the collection and preprocessing of the data and performs the Granger Causality Test on the data. Section 4.3 describes the conceptual design of the AINN in detail, presenting the underlying mathematical formulation and architecture. In Section 4.4, the simulation results obtained from the NN and the corresponding AINN are compared in terms of the five metrics, namely: root mean square error (RMSE), mean square error (MSE), mean absolute percentage error (MAPE), coefficient of determination ($R^2$), and percentage error (PE). Finally, Section 4.5 concludes the paper by highlighting that the AINN outperforms the corresponding NN, as the DLEM regulates the training path of AINN and ensures convergence in a specific basin of attraction.

## 4.2 Data Description and Analysis

### 4.2.1 Data Description

For the field experiments conducted during the 2021 and 2022 growing seasons at Area X.O, Ottawa, Ontario, Canada, data were collected using the LI-COR automated gas chamber [219] to measure $N_2O$ emission flux, $H_2O$ flux, air temperature, and air humidity. Additionally, the Pessl iMetos ECO D3 was used to measure soil moisture and soil temperature. To demonstrate that nitrate quantities are essential for predicting $N_2O$ emissions and to validate our assumptions regarding nitrate levels in the field, as shown in Section 4.2.2, a complementary indoor experiment under controlled conditions was conducted from January to April 2024. The LI-COR automated gas chamber was again employed to collect data on $N_2O$ emission flux, $H_2O$ flux, air temperature, and air humidity. Soil moisture, soil temperature, and nitrate levels were measured using 7-in-1 RS485 sensors and a Raspberry Pi. Both the field and indoor experiments sampled data every 30 minutes[1].

The $N_2O$ flux was calculated using the following equation:

$$F_{N_2O} = \frac{VP(1 - W_0)}{RST} \frac{dN_2O}{dt} \tag{4.1}$$

and the variables were defined as follows:

- $F_{N_2O}$: soil $N_2O$ flux (nmol $m^{-2}$ $s^{-1}$)
- $V$: chamber volume ($m^3$)
- $P$: atmospheric pressure (Pa)
- $R$: gas constant (Pa $m^3$ $mol^{-1}$ $k^{-1}$)
- $S$: soil area ($m^2$)
- $T$: temperature (Kelvin)
- $\frac{dN_2O}{dt}$: rate of $N_2O$ change (nmol$m^{-2}s^{-1}$ ) during the chamber closure

As shown in Figure 4.2, qualitative analysis demonstrated that in the 2022 growing season, due to several reasons, the crop is planted one and a half months later than in the 2021 growing season. Therefore, the soil temperature in the 2022 growing season (average soil temperature = 14 °C) is generally lower than that in the

[1]Here is the link to the dataset: https://github.com/Developer2046/n2o_emission/tree/main

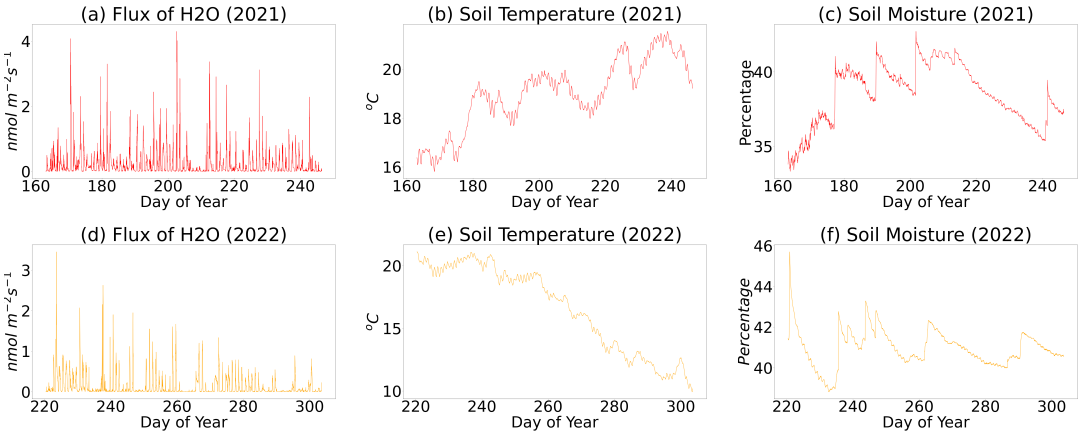



FIGURE 4.2: Data were collected during the 2021 and 2022 growing seasons. The data were collected every 30 minutes per time step. For the 2021 growing season, data collection started on day 163 of the year, while for the 2022 growing season, it started on day 220 of the year.

2021 growing season (average soil temperature = 20 °C). With lower soil temperature and higher precipitation, the soil moisture in the 2022 growing season (average soil moisture = 41 %) is higher than in the 2021 growing season (average soil moisture = 38 %). The higher soil moisture is the primary reason for the more intensive emission of $N_2O$ in the 2022 growing season. We will conduct a detailed analysis of this discrepancy in the following section.

### 4.2.2 Data Analysis

Qualitative analysis shows that soil temperature and moisture are directly related to the emission of $N_2O$ from the soil since they impact the nitrification and denitrification process. However, soil temperature and soil moisture vary with the depth of the soil. Therefore, in this research, we calculate the average values of soil temperature and soil moisture measured at 10, 20, 30, 40, 50, and 60 centimeters. At the same time, air temperature, air humidity, and water flux are also included in predicting the emission of $N_2O$ from farming for two significant reasons. Firstly, they significantly influence soil moisture and soil temperature. Secondly, nitrification and denitrification are very active in the soil subsurface, around 0 to 5 centimeters in depth [220, 221], where soil temperature and soil moisture are closely related to air temperature, air humidity, and water flux. In contrast, direct measurements of soil temperature and moisture are not available.

Therefore, after careful consideration and selection, the multivariate time series includes the water flux, air temperature, air humidity, average soil temperature, and

average soil moisture. Since nitrate content is challenging to measure in the soil, we must make a simple assumption about the amount of nitrate in the soil. Since nitrification and denitrification could be considered as enzyme-catalyzed reactions or first-order reactions [222], we have:

$$\frac{dN}{dt} = -\lambda N \tag{4.2}$$

where $N$ is the nitrate content in the soil; solving this equation, we can understand that the amount of nitrate in the soil exponentially decays over time.

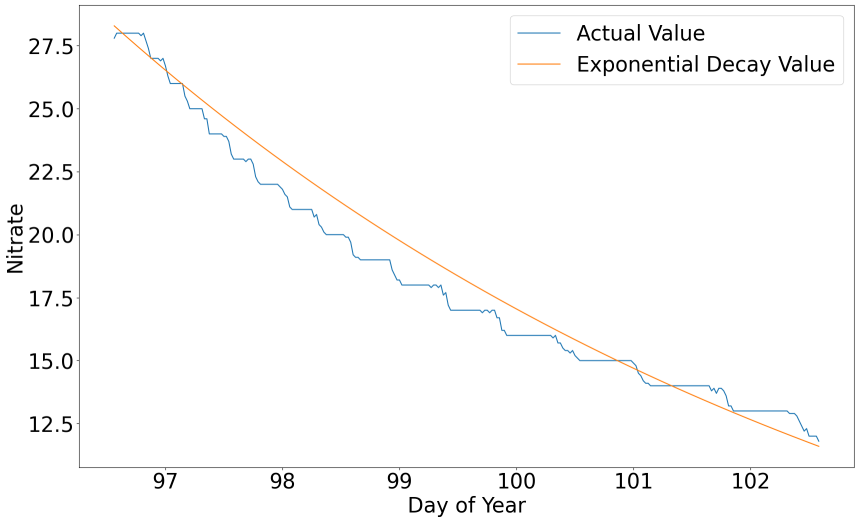



FIGURE 4.3: The amount of nitrate exponentially decays over time in-between the watering periods. The data were collected from an indoor experiment in 2024.

To verify that this assumption is reasonable and to demonstrate that the quantity of nitrate in the soil is vital for predicting the emission of $N_2O$ from the soil, an indoor experiment was conducted from January to April 2024. As shown in Figure 4.3, the blue curve represents the actual amount of nitrate measured by the 7-in-1 sensor, while the yellow curve is generated from computer simulation, exhibiting exponential decay over time. It is observed that the yellow curve fits the actual values well. Thus, we can conclude that the amount of nitrate in the soil exponentially decays over time in-between watering periods. Detailed information on the performance of the AINN is discussed in Section 4.4.3.

Why are the average soil temperature and moisture used as features rather than the soil temperatures and soil moisture at different depths? Because we could not estimate the amount of nitrate at various depth levels and could only estimate the expected amount of nitrate available for nitrification and denitrification. Therefore, according to the superposition principle for the measurement differential equation

[223], we use the average soil temperature and soil moisture as features. When we input all soil temperature and soil moisture data into the model, the model would not learn anything from the data.

#### Granger Causality Test

Initially proposed by Granger in 1969 [224], the Granger causality test is used to evaluate whether one time series helps predict another. According to previous exploration, the amount of $N_2O$ is produced from the temperature, moisture, carbon, nitrogen, and oxygen contents [225]. In determining which features we should use in multivariate time series analysis, we feed the time series of water flux, air temperature, air humidity, soil temperature, and soil moisture into the Granger causality test program with $N_2O$ flux, and the test result demonstrates that all these features help forecast the emission of $N_2O$.

More factors, including soil texture, pH, oxygen accessibility, microbial activity, environmental conditions, and the type and amount of applied nitrogen fertilizer, could control and limit the process of $N_2O$ emissions [225]. However, in our exploration, we only study the major features that could have a significant influence on the emission of $N_2O$ since, in our case, soil texture, pH, oxygen accessibility, and environmental conditions could be seen as permanent properties that rarely change in the process of nitrification and denitrification and have fewer contributions to the instant emission of $N_2O$ from the soil. Therefore, we did not include all these factors in our multivariate time series analysis when predicting the emission of $N_2O$.

### 4.2.3 Data Pre-processing

The raw data is full of noise, outliers, and missing values. Therefore, before utilizing the raw data, we need to clean it, spot the outliers, and fill in the missing data. When all these processes are finished, the data must be scaled to the range [0, 1] before being fed into the neural network model. Conventionally, researchers use the min-max or robust scaling methods to transform the data into the range [0, 1] [226]. However, in practice, we could not know the data beforehand when predicting the emission of $N_2O$. Therefore, in this exploration, we will scale the data based on our expert knowledge. In practice, we use the min-max scaler, where the minimal and maximum values are determined by previous experience, not the collected data. As shown in Table 4.1, these are the minimum and maximum values utilized in this exploration.

The Equation 4.3 is used for scaling data.

TABLE 4.1: Data Range Used in Scaling Original Data

| [1] | $FN_2O$ | $FH_2O$ | Temp (Air) | Humidity (Air) | Temp (Soil) | Moisture (Soil) |
|---|---|---|---|---|---|---|
| maximum | 260 | 5 | 40 | 50 | 25 | 50 |
| minimum | 0 | 0 | 10 | 0 | 10 | 30 |

[1] In this context, the unit of $N_2O$ flux is ppb, the unit of $H_2O$ flux is mm, the unit of air humidity and soil moisture is %, the unit of air temperature and soil temperature is °C.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.3}$$

where $X'$ is the transformed value, which is normally in the range of [0, 1], and $X$ is the original data. $X_{max}$ and $X_{min}$ are the maximum and minimum values shown in Table 4.1. The transformed data $X'$ could be represented by a vector at one time step, as shown in the following:

$$\begin{bmatrix} \mathbf{N}_t & \mathbf{Temp}_t & \mathbf{W}_t & \mathbf{H}_t & \mathbf{SM}_t & \mathbf{ST}_t \end{bmatrix}$$

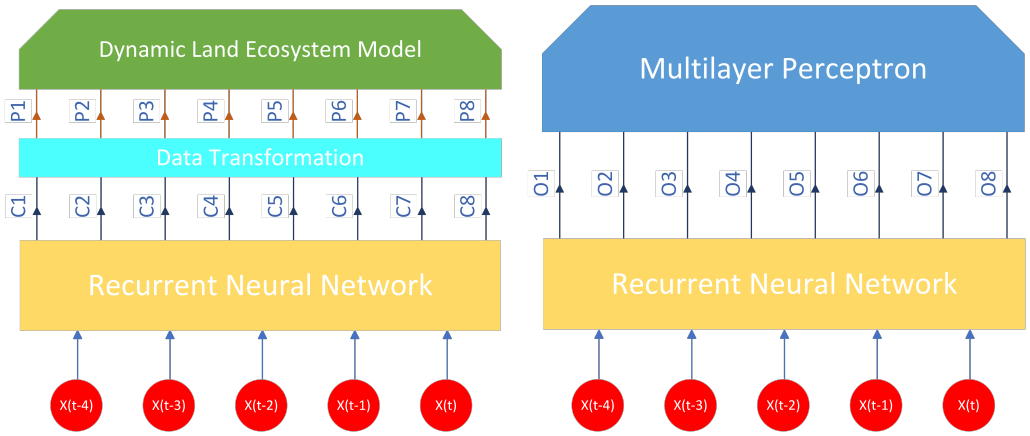
where $\mathbf{N}_t$, $\mathbf{Temp}_t$, $\mathbf{W}_t$, $\mathbf{H}_t$, $\mathbf{SM}_t$, and $\mathbf{ST}_t$ represent the amount of available nitrogen, air temperature, water flux, air humidity, soil moisture, and soil temperature, respectively, at time step $t$. Since the study did not measure the concentration of available nitrogen in the soil directly, the fertilizer was assumed to exponentially decrease in our experiment, as discussed in Section 4.2.2. To predict the emission of $N_2O$ at time step $t$, information from the previous $n$ time steps is required. Therefore, the input data can be characterized as a data matrix, as shown below:

$$\begin{bmatrix} \mathbf{N}_{t-n:t-1}^T & \mathbf{Temp}_{t-n:t-1}^T & \mathbf{W}_{t-n:t-1}^T & \mathbf{H}_{t-n:t-1}^T & \mathbf{SM}_{t-n:t-1}^T & \mathbf{ST}_{t-n:t-1}^T \end{bmatrix}$$

where $\mathbf{N}_{t-n:t-1}$, $\mathbf{Temp}_{t-n:t-1}$, $\mathbf{W}_{t-n:t-1}$, $\mathbf{H}_{t-n:t-1}$, $\mathbf{SM}_{t-n:t-1}$, $\mathbf{ST}_{t-n:t-1}$ are row vectors consisting of the values of **N**, **Temp**, **W**, **H**, **SM**, and **ST** from time step $t-n$ to $t-1$. $T$ is the transpose operation.

## 4.3 Conceptual Design of Agriculture-informed Neural Network

### 4.3.1 Architecture of Agriculture-informed Neural Network




(a) Agriculture-informed Neural Network (b) Neural Network

FIGURE 4.4: Conceptual Design of Agriculture-informed Neural Network and Neural Network

Figure 4.4(a) shows that the AINN can be divided into the RNN and DLEM components. In this exploration, we use MLP, Convolutional Neural Network (CNN), LSTM, and Transformer as RNN components to capture the dynamic properties of the system. Although MLP is not typically used for multivariate time series analysis, MLP is included in our exploration for comparison purposes. However, for CNN, LSTM, and Transformer, all these architectures can be used to address multivariate time series data, but each has its own characteristics.

In the AINN, the RNN component will output a vector of eight dimensions, representing eight adjusted factors passed to the DLEM for calculating the eight parameters in the DLEM, namely soil temperature ($T_{soil}$), water-filled porosity ($wfp$), daily maximum fraction of $NH_4^+$ that is converted into $NO_3^-$ and nitrogen gases ($k_{nit}$), potential rate of denitrification ($N_{pot_{denit}}$), percentage of clay content in soil ($P_{clay}$), soil respiration rate ($R_h$), parameter depending on plant functional type $k_{den}$, and the effect of $NO_3^-$ concentration ($gNg^{-1}soil$), which could be calculated using the $NO_3^-$ content in the soil per unit area ($av_{NO_3}$) divided by soil bulk density ($BD_{soil}$).

For comparison purposes, a particular layer with eight neural nodes is designed in the neural network model without information from the agriculture field, corresponding to the eight adjusted factors in the AINN. In other words, in a pure neural

network model, a three-layer MLP replaces the DLEM in the AINN, as shown in Figure 4.4(b).

**Case Study of the LSTM and its Corresponding AINN**

To help readers better understand the design of the AINN, we take the LSTM and its corresponding AINN, the LSTM-DLEM, as examples. The data collected for this model are organized into tensors with 15-time steps and six features and are fed into the model in batches of size 20. Detailed information about the input and output of each layer in the model is provided in Table 4.2 and Table 4.3. We must design a slightly more complex RNN component in the AINN than its counterpart in the RNN because the AINN is a constrained optimization problem, and the DLEM acts as a regularizer for the entire model. The AINN probably needs more parameters to converge successfully.

For comparison purposes, we design a hidden layer of eight nodes in the NN, as shown in Table 4.3, highlighted by apple green. The first part of the LSTM and LSTM-DLEM is the same. In the LSTM-DLEM, the RNN component will input eight adjusted factors to the DLEM component, while the LSTM will input eight nodes to the MLP component.

TABLE 4.2: Detailed Description of the LSTM-DLEM

| Layer | Input Shape | Output Shape | Parameters[1] |
|---|---|---|---|
| BDLSTM[2] | (Batch Size, 15, 6) | (Batch Size, 15, 100) | 23200 |
| BatchNorm[3] | (Batch Size, 15, 100) | (Batch Size, 15, 100) | 30 |
| Flatten | (Batch Size, 15, 100) | (Batch Size, 1500) | 75000 |
| FCL[4] | (Batch Size, 50) | (Batch Size, 8) | 400 |
| DLEM | (Batch Size, 8) | (Batch Size, 1) | 0 |

[1] Total parameters: 98630.
[2] Bidirectional Long Short-Term Memory.
[3] Batch Normalization.
[4] Full Connected Layer.

### 4.3.2 Mathematical Formulation

Assuming the emission of $N_2O$ is governed by a dynamical system represented by a system of differential equations, as shown in Equation 4.4 below.

$$\frac{d\mathbf{\Theta}}{dt} = \mathbf{f}(t, \mathbf{X}, \mathbf{\Theta}) + \mathbf{H} \tag{4.4}$$

TABLE 4.3: Detailed Description of the LSTM[1]

| Layer | Input Shape | Output Shape | Parameters[2] |
|---|---|---|---|
| BDLSTM | (Batch Size, 15, 6) | (Batch Size, 15, 100) | 23200 |
| BatchNorm | (Batch Size, 15, 100) | (Batch Size, 15, 100) | 30 |
| Flatten | (Batch Size, 15, 100) | (Batch Size, 1500) | 75000 |
| FCL | (Batch Size, 50) | (Batch Size, 8) | 400 |
| FCL | (Batch Size, 8) | (Batch Size, 50) | 400 |
| FCL | (Batch Size, 50) | (Batch Size, 1) | 50 |

[1] The highlighted rows (with green apple) are used to replace the DLEM in LSTM-DLEM.
[2] Total parameters: 99080.

where

$$\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_8]^T, \mathbf{X} = [\mathbf{N}^T, \mathbf{Temp}^T, \mathbf{W}^T, \mathbf{H}^T, \mathbf{ST}^T, \mathbf{SM}^T], \mathbf{H} = [\eta_1, \eta_2, \ldots, \eta_8]^T.$$

In this context, **N**, **Temp**, **W**, **H**, **ST**, and **SM** are the independent variables, which are the amount of nitrate, air temperature, air humidity, flux of water, soil temperature and soil moisture, respectively. $\theta_1$, $\theta_2$, ..., and $\theta_8$ represent the state variables that control the emission of $N_2O$, which are soil temperature ($T_{soil}$), water-filled porosity ($wfp$), daily maximum fraction of $NH_4^+$ that is converted into $NO_3^-$ and nitrogen gases ($k_{nit}$), potential rate of denitrification ($N_{pot_{denit}}$), percentage of clay content in soil ($P_{clay}$), soil respiration rate ($R_h$), factor depending on plant functional type $k_{den}$, $NO_3^-$ content ($D_{NO_3^-}$). At the same time, the white noise terms **H** are added to imply that the dynamical system we handled is stochastic.

However, we could not find the specified differential equation to describe the process of $N_2O$ emission. The experts from the agriculture field proposed a process-based model (DLEM) to estimate the expected value of the emission of $N_2O$ over a long period and in a large area [43]. In this section, the DLEM is illustrated in abstract form to make the discussion more straightforward and convenient, as shown in Equation 4.5.

$$E(FN_2O) \doteq DLEM(E(\theta_1), E(\theta_2), \ldots, E(\theta_8)) \tag{4.5}$$

where $E(\theta_1)$, $E(\theta_2)$, ..., and $E(\theta_8)$ represent the eight state variables in the DLEM, $FN_2O$ is the flux of $N_2O$, and $E$ is expectation operator.

Generally speaking, we could consider DLEM an average model to compute the expected value of emission of $N_2O$ daily. How can we use the information supplied by the DLEM to improve the performance of our model? We could integrate the

DLEM component with the RNN component to build an AINN, reducing the complexity while increasing the regression performance. Therefore, the RNN is utilized to simulate the solutions of the eight state variables represented by seven independent variables, as shown in Equation 4.4, and the solutions could be described in Equation 4.6.

$$\mathbf{\Theta} = \mathbf{g}(t, \mathbf{X}) + \mathbf{H} \tag{4.6}$$

where

$$\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_8]^T, \mathbf{X} = [\mathbf{N}^T, \mathbf{Temp}^T, \mathbf{W}^T, \mathbf{H}^T, \mathbf{ST}^T, \mathbf{SM}^T], \mathbf{H} = [\eta_1, \eta_2, \ldots, \eta_8]^T.$$

The RNN component in the AINN supplies the adjusted factors (in the range of [0, 1]) of the eight state variables to the DLEM component. Then, we need to transform these adjusted factors into parameters that could be fed into the DLEM. Therefore, a data transformation needs to be executed at the interface of the RNN and DLEM components in the AINN.

As shown in Table 4.4, this is the parameter range we supplied to the AINN, and at the interface of the RNN component and DLEM component, we use Equation 4.7 to transform the data.

$$P = C \times (P_{max} - P_{min}) + P_{min} \tag{4.7}$$

where $C$ is the adjusted factor supplied by the RNN component, and $P_{max}$ and $P_{min}$ are the values provided in Table 4.4, according to the descriptions from [54, 227]. However, the sigmoid activation in the neural network could never reach 0 and 1. It is convenient for the AINN to converge when the parameter range is slightly more extensive than the actual one.

TABLE 4.4: Parameter Range Used in the AINN

| [1] | $D_{NO_3}$ | $D_{NH_4}$ | Temp | wfp | Rh | $P_{clay}$ | $k_{den}$ | $k_{nit}$ |
|---|---|---|---|---|---|---|---|---|
| maximum | 400 | 50 | 35 | 1 | 1.1 | 0.5 | 0.5 | 1.1 |
| minimum | 0 | 0 | 10 | 0.3 | 0 | 0 | 0 | 0.9 |

[1] The unit of $D_{NO_3}$ and $D_{NH_4}$ is g N $g^{-1}$ soil, the unit of temperature is °C, the unit of wfp and $P_{clay}$ is %, the unit of Rh is g C $m^{-2}$ $day^{-1}$, the unit of $k_{den}$ is g N $m^{-2}$ $day^{-1}$, and $k_{nit}$ is unitless.

## 4.4 Simulation and Analysis

The MLP, MLP-DLEM, CNN, CNN-DLEM, LSTM, LSTM-DLEM, CNN-Transformer, and CNN-Transformer-DLEM are implemented using PyTorch 2.0.1. To ensure comparability of results, all models utilize batch training with mean square error (MSE) as the loss function. During training, the Adam optimizer [112] is employed for optimal convergence performance, and the early stopping method [228] is utilized to prevent overfitting. With early stopping, if the loss value stops decreasing after 20 epochs, the training is terminated, and the best weights are restored. Each model is run ten times to reduce stochastic bias in evaluating specific model architectures. In the subsequent discussion, referring to NN includes MLP, CNN, LSTM, and CNN-Transformer. When mentioning AINN, it encompasses MLP-DLEM, CNN-DLEM, LSTM-DLEM, and CNN-Transformer-DLEM.

MLPs are feed-forward networks with at least two fully connected layers that process samples in a fixed order [229]. A significant limitation of MLPs for multivariate time series analysis is their inability to process data hierarchically or in a multiscale manner, and they struggle to capture temporal relationships in the data. Despite this limitation, researchers utilize MLPs for regression problems in agriculture; thus, they are included for comparison purposes.

CNNs, initially designed for detecting local patterns within data using convolutional filters in computer vision, can automatically learn and extract essential features such as spikes, trends, or seasonal variations in time series data. Adapted to process one-dimensional (1D) input data, CNNs are suitable for time series analysis by effectively capturing temporal patterns and dependencies by applying 1D convolutional filters. Therefore, researchers adopt CNN architectures for multivariate time series analysis [230, 231].

Originally designed to address the common vanishing/exploding gradient issue in vanilla RNNs, LSTMs integrate memory cells with gate control into their state dynamics [232]. Due to their design goal, they are suitable for solving problems involving sequence data, including language translation [233], video representation learning [234], and image caption generation [235]. In multivariate time series analysis, LSTMs can understand complex relationships and dependencies between variables across different time steps.

Transformers employ self-attention mechanisms to capture dependencies between different time steps in a sequence, effectively modeling long-range dependencies in time series data where distant observations may influence each other. However, processing all time steps in parallel may limit their ability to model the sequential nature

of time series data explicitly. Unlike LSTMs, transformers lack recurrent connections that maintain an internal state representing the sequence history, potentially hindering their performance on tasks requiring modeling temporal dynamics over long sequences [236]. Multivariate time series regression transformers typically utilize a simple encoder structure comprising attention and feed-forward layers [237].

An embedding layer is included before the transformer encoder for the Transformer-based model, which we initially designed for natural language processing. Although natural language can be considered a time series, it fundamentally differs from a time series derived from dynamic systems. After experimenting multiple times, we replaced the position encoding layer with a convolutional layer, which can capture complex temporal dependencies in time series data [238].

### 4.4.1 Comparison between NN and its corresponding AINN

In this experiment, five indicators are used to evaluate the performance of models, namely: root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), coefficient of determination ($R^2$), and percentage error (PE).

TABLE 4.5: MLP-based Models with Different Time Steps

| Model[1] | Time Step | Mode | RMSE | MAE | MAPE | $R^2$ | PE |
|---|---|---|---|---|---|---|---|
| MLP | 5 | Train | 8.33±0.24 | 6.11±0.18 | 618±88.3 | 0.71±0.02 | 0.11±0.04 |
| MLP-DLEM | 5 | Train | 6.87±0.25 | 4.62±0.20 | 293±16.3 | 0.81±0.01 | -0.06±0.03 |
| MLP | 5 | Test | 36.62±0.73 | 25.81±0.52 | 5.45±0.93 | 0.50±0.02 | -0.21±0.02 |
| MLP-DLEM | 5 | Test | 32.04±0.70 | 21.13±0.83 | 3.69±0.23 | 0.61±0.02 | 0.19±0.05 |
| MLP | 15 | Train | 8.45±0.35 | 5.94±0.35 | 359±120 | 0.71±0.02 | 0.09±0.07 |
| MLP-DLEM | 15 | Train | 6.43±0.22 | 4.29±0.16 | 217±15.4 | 0.83±0.01 | -0.08±0.02 |
| MLP | 15 | Test | 38.00±0.85 | 24.78±0.36 | 3.50±0.51 | 0.46±0.02 | -0.21±0.02 |
| MLP-DLEM | 15 | Test | 31.66±0.51 | 21.03±0.50 | 3.56±0.26 | 0.62±0.01 | 0.15±0.02 |
| MLP | 45 | Train | 8.44±0.23 | 5.46±0.10 | 252±77 | 0.71±0.01 | 0.04±0.04 |
| MLP-DLEM | 45 | Train | 9.50±0.92 | 6.47±1.00 | 1012±257 | 0.63±0.07 | 0.12±0.07 |
| MLP | 45 | Test | 40.03±1.52 | 26.75±1.61 | 4.08±0.83 | 0.40±0.05 | -0.19±0.02 |
| MLP-DLEM | 45 | Test | 46.40±1.67 | 35.30±1.31 | 9.34±0.61 | 0.20±0.06 | -0.10±0.04 |

[1] In order to make the table more readable, the performance of the MLP is highlighted using different colors. In this table, we focus on the training performance, as there is an exception with the model trained using 45-time steps.

The data collected from the 2021 growing season are taken as the training dataset, and the data collected from the 2022 growing season are taken as the testing dataset. Since the 2021 growing season was drier than that in the 2022 growing season, and in the 2022 growing season, the planting date was one and a half months later than that in the 2021 growing season, the total emission of $N_2O$ in the 2022 growing season ($157\,015\,\mathrm{nmolm^{-2}s^{-1}}$) is much larger than that in the 2021 growing season ($58\,137\,\mathrm{nmolm^{-2}s^{-1}}$). Therefore, we use $PE$ to describe the percentage error of the model for the corresponding growing season.

TABLE 4.6: CNN-based Models with Different Time Steps

| Model[1] | Time Step | RMSE | MAE | MAPE | $R^2$ | PE |
|---|---|---|---|---|---|---|
| CNN | 5 | 34.27±0.60 | 22.17±0.63 | 3.09±0.68 | 0.56±0.02 | -0.12±0.06 |
| CNN-DLEM | 5 | 32.06±1.30 | 20.77±1.18 | 3.10±0.63 | 0.61±0.03 | -0.05±0.08 |
| CNN | 15 | 34.59±0.83 | 22.06±0.71 | 3.08±0.52 | 0.55±0.02 | -0.14±0.07 |
| CNN-DLEM | 15 | 31.91±0.93 | 20.54±0.62 | 2.89±0.15 | 0.62±0.02 | -0.02±0.03 |
| CNN | 45 | 33.05±1.07 | 20.07±0.90 | 2.27±0.45 | 0.59±0.03 | -0.15±0.07 |
| CNN-DLEM | 45 | 32.23±0.87 | 21.79±0.91 | 3.41±0.26 | 0.61±0.02 | 0.01±0.04 |

[1] In order to make the table more readable, the performance of the CNN is highlighted by different colors.

For multivariate time series analysis, the number of time steps is one of the most essential characteristics to investigate. To explore the dynamic properties of our system, we set the number of time steps as 5, 15, and 45. In this section, three MLPs, CNNs, LSTMs, and CNN-Transformers, are built and trained on the datasets of 5, 15, and 45-time steps, respectively. Correspondingly, three MLP-DLEMs, CNN-DLEMs, LSTM-DLEMs, and CNN-Transformer-DLEMs are constructed and trained on the datasets of 5, 15, and 45-time steps, respectively.

We will first discuss the overall behavior of all the models and then describe the specific behavior of each model in detail in the following sections. Please note that Table 4.5 shows both training and testing performance, as there is an exception with the model trained using 45-time steps. Tables 4.6, 4.7, and 4.8 show only testing performance.

It is easily observed that the testing performance for the AINN is better than that of the corresponding NN, as shown in Table 4.5, 4.6, 4.7, and 4.8. However, there is one exception: the MLP-DLEM trained with the dataset of 45-time steps, where its

TABLE 4.7: LSTM-based Models with Different Time Steps

| Model[1] | Time Step | RMSE | MAE | MAPE | $R^2$ | PE |
|---|---|---|---|---|---|---|
| LSTM | 5 | 36.61±0.72 | 24.94±0.84 | 3.47±0.25 | 0.50±0.02 | -0.05±0.02 |
| LSTM-DLEM | 5 | 30.44±1.23 | 19.74±1.19 | 3.56±0.39 | 0.65±0.03 | 0.06±0.08 |
| LSTM | 15 | 36.23±1.24 | 25.49±1.47 | 4.17±0.99 | 0.51±0.03 | -0.02±0.06 |
| LSTM-DLEM | 15 | 30.25±0.47 | 19.72±0.51 | 3.58±0.25 | 0.66±0.01 | 0.08±0.03 |
| LSTM | 45 | 34.20±1.93 | 23.23±1.38 | 3.58±0.65 | 0.56±0.05 | 0.01±0.07 |
| LSTM-DLEM | 45 | 30.66±0.44 | 20.97±0.99 | 4.16±0.70 | 0.64±0.01 | 0.12±0.04 |

[1] In order to make the table more readable, the performance of the LSTM is highlighted by different colors.

training and testing performance is far worse than its corresponding MLP, as shown in Table 4.5. This is because MLP is not inherently prepared for multivariate time series analysis. When the time steps are too large, it cannot learn the temporal dynamics inherent in the data. Therefore, it is convenient for us to conclude that the AINN, particularly the models prepared for multivariate time series data, generally reduces overfitting compared to their corresponding NN.

For AINN, the models trained with the dataset of 15-time steps could achieve the best testing performance among the same type of models trained with different time steps. It suggests that the time step of 15 could be appropriate for training the model, and it indicates that data from the 15-time steps already include primary information to determine the emission of $N_2O$. Among all the AINN, the LSTM-DLEM trained with the dataset of 15-time steps achieved the best testing performance, even better than the CNN-Transformer-DLEM. This is probably due to the complexity of the CNN-Transformer-DLEM being too high, and our data is insufficient to converge optimally. The complexity of the LSTM-DLEM is suitable for this kind of data size. In the future, if we collect more data, the Transformer-based AINN can be an appropriate candidate for analyzing multivariate time series data.

The summary of testing performance for each model is presented in Table 4.9. Among the models, LSTM-DLEM exhibits the best performance in terms of RMSE and $R^2$, the CNN-Transformer-DLEM performs the best in terms of MAE, and CNN-DLEM performs the best in terms of PE. Regarding MAPE, the NN outperforms the corresponding AINN, while CNN-Transformer performs best. Overall, the AINN tends to outperform NN across most metrics. MLP shows the poorest performance,

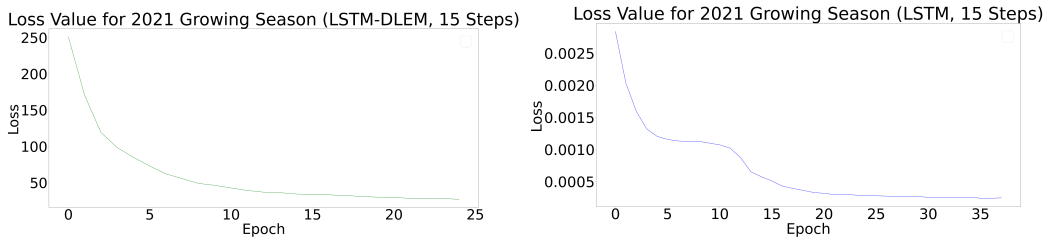
TABLE 4.8: Transformer-based Models with Different Time Steps

| Model[1] | Time Step | RMSE | MAE | MAPE | $R^2$ | PE |
|---|---|---|---|---|---|---|
| CNN-Transformer | 5 | 36.56±1.24 | 20.72±0.89 | 2.43±0.42 | 0.50±0.03 | -0.38±0.04 |
| CNN-Transformer-DLEM | 5 | 31.60±2.70 | 21.29±2.48 | 3.13±0.74 | 0.62±0.06 | -0.05±0.08 |
| CNN-Transformer | 15 | 37.60±2.12 | 22.15±1.47 | 2.64±0.80 | 0.46±0.06 | -0.34±0.06 |
| CNN-Transformer-DLEM | 15 | 30.78±1.16 | 19.10±0.88 | 2.39±0.30 | 0.64±0.03 | -0.13±0.05 |
| CNN-Transformer | 45 | 33.38±1.21 | 20.03±0.57 | 2.80±0.37 | 0.58±0.03 | -0.20±0.06 |
| CNN-Transformer-DLEM | 45 | 31.45±1.74 | 19.62±1.18 | 2.55±0.31 | 0.63±0.04 | -0.09±0.07 |

[1] In order to make the table more readable, the performance of the CNN-Transformer is highlighted by different colors.

but LSTM-DLEM and CNN-Transformer-DLEM are comparable in solving multivariate time series regression issues.

### 4.4.2 Interpretation for AINN




(a) Training Loss of LSTM-DLEM (b) Training Loss of LSTM

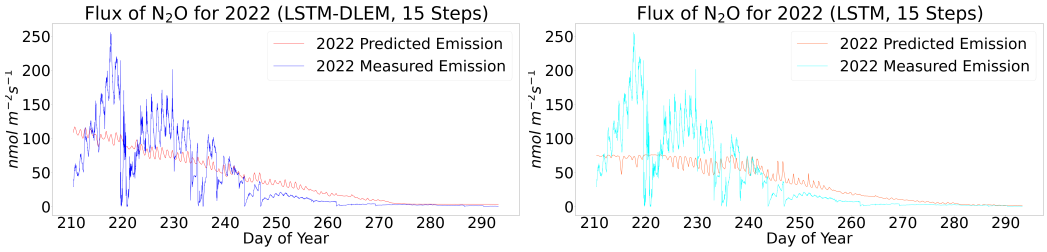
FIGURE 4.5: Training Loss Comparison between LSTM-DLEM and LSTM

As depicted in Figure 4.5, the loss values for the LSTM-DLEM and LSTM are decreasing with increasing epochs of the training process. However, the loss value for the LSTM-DLEM represents the MSE between the actual measured $N_2O$ and the predicted $N_2O$, while the loss value for the LSTM represents the MSE between the

TABLE 4.9: Rank of Testing Performance for Each Model

| Model[1] | RMSE | MAE | MAPE | $R^2$ | PE |
|---|---|---|---|---|---|
| MLP | 8 | 7 | 7 | 8 | 7 |
| MLP-DLEM | 7 | 8 | 8 | 7 | 4 |
| CNN | 4 | 5 | 3 | 4 | 6 |
| CNN-DLEM | 3 | 4 | 4 | 3 | 1 |
| LSTM | 5 | 6 | 5 | 5 | 1 |
| LSTM-DLEM | 1 | 2 | 6 | 1 | 3 |
| CNN-Transformer | 6 | 3 | 1 | 6 | 8 |
| CNN-Transformer-DLEM | 2 | 1 | 2 | 2 | 5 |

[1] In order to make the Table more readable, the performance of the NN is highlighted by different colors.




(a) Testing Performance of LSTM-DLEM (b) Testing Performance of LSTM

FIGURE 4.6: Testing Performance Comparison between LSTM-DLEM and LSTM

measured $N_2O$ (scaled) and the predicted $N_2O$, both of which fall within the range of [0, 1]. Therefore, the loss value for the LSTM-DLEM is more significant than that of the LSTM in terms of quantity. Nonetheless, both indicate that the models are well-trained and have converged.

Corresponding to the training loss for the LSTM-DLEM and LSTM, the test performance for the LSTM-DLEM and LSTM is depicted in Figure 4.6. When the models are well-trained, the LSTM-DLEM outperforms the LSTM for the testing dataset. The reasons for the superior performance of the LSTM-DLEM over the LSTM are discussed in Section 4.3.2.

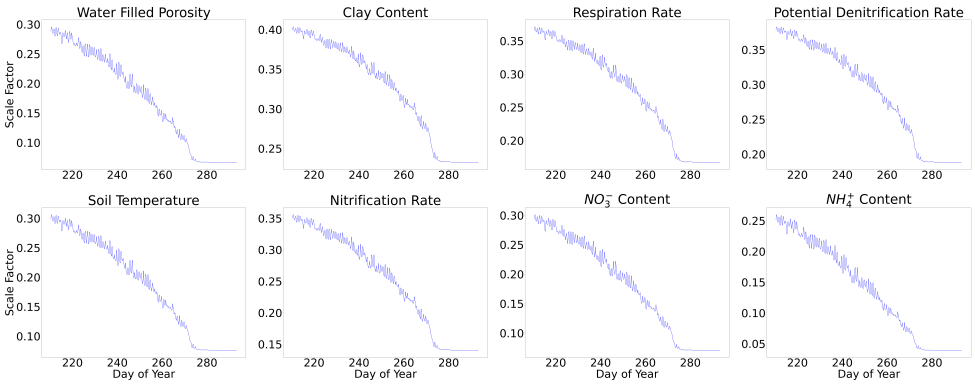



FIGURE 4.7: Adjusted Factors from RNN component in LSTM-DLEM

In this context, the dynamic behaviors of the scale factors for eight parameters in the DLEM are shown in Figure 4.7. The DLEM plays a regulatory role in training AINN, leading the entire model to converge to a specific basin of attraction. When the AINN was run multiple times with different initialization, it was observed that the dynamic behaviors of the scale factors consistently fell into a particular basin of attraction; in other words, the adjustment factors for the DLEM show almost similar

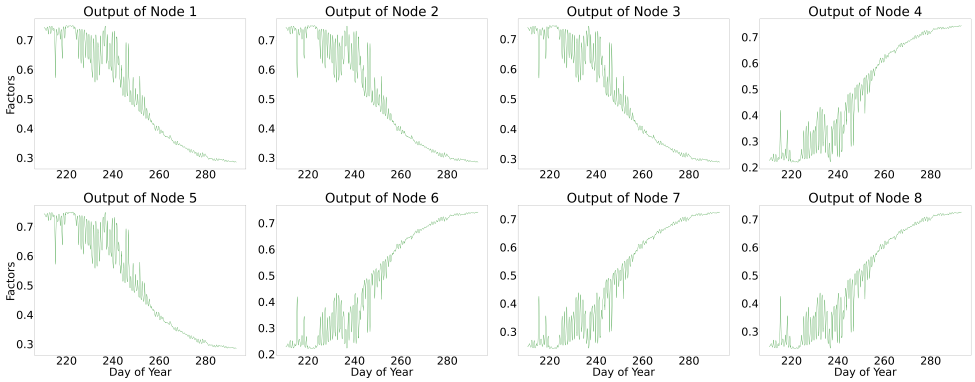


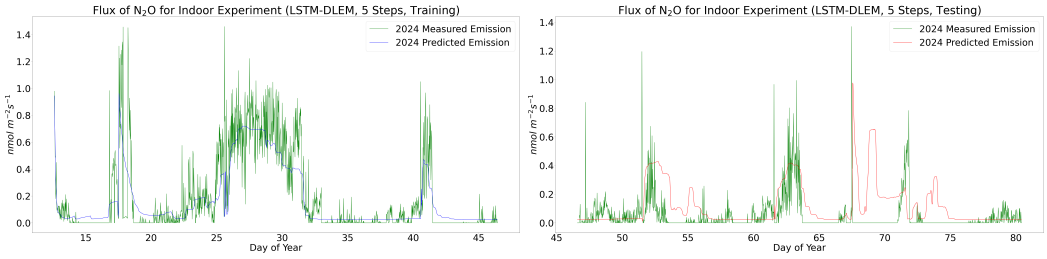

FIGURE 4.8: Output of eight Nodes from RNN component in LSTM

patterns. This property of the AINN is distinct from that of the NN, which can have an unlimited number of basins of attraction because it is a universal approximator.

As shown in Figure 4.7, this is one specific pattern generated by the LSTM-DLEM when the model achieves optimal performance. Through this, we can observe that almost all the factors decrease with time. This phenomenon is consistent with what we observed in reality. However, since the DLEM is not an accurate model that could predict the emission of $N_2O$, we could not obtain more qualitative information from Figure 4.7. Meanwhile, for the single NN, the output of the eight nodes, as shown in Figure 4.8, that we designed specifically in the hidden layer to make a comparison, could not supply any information for interpreting the behavior of model. Therefore, it is convenient to conclude that the AINN have a better interpretation than the NN.

### 4.4.3 Indoor Experiment Under Control Environment

To demonstrate the necessity of measuring the nitrate quantity for predicting the emission of $N_2O$, an additional indoor experiment in a controlled environment was conducted from January to April 2024. We then used soil temperature, soil moisture, and the amount of nitrate to train our models. As shown in Figure 4.9, we could successfully predict the emission of $N_2O$ by including the amount of nitrate in the dataset using LSTM-DLEM.




(a) Training Performance for Indoor Experiment (b) Testing Performance for Indoor Experiment

FIGURE 4.9: LSTM-DLEM Trained on the Data Collected from Indoor Experiment

## 4.5 Conclusion Remarks

This study presents the development of the AINN, which combines an RNN component with the process-based ecosystem model, particularly DLEM, for predicting $N_2O$ emissions. We aimed to design an AINN that could perform better than the individual NN alone. The data collected during the growing seasons of 2021 and 2022

(by LI-COR soil-gas measurement equipment and Pessl iMetos ECO in Area X.O, Ottawa, Ontario, Canada) were used to train and test NN (including MLP, CNN, LSTM, and CNN-Transformer) and AINN (including MLP-DLEM, CNN-DLEM, LSTM-DLEM, and CNN-Transformer-DLEM), respectively. The models were evaluated using five metrics: RMSE, MAE, MAPE, $R^2$, and PE.

Both models were trained using the early stopping technique to avoid overfitting. The experimental results align with our objective of designing the AINN. By combining the strengths of the RNN and DLEM, the AINN effectively regulates the training process, resulting in better performance on the testing datasets with smaller generalized errors in terms of MAE, RMSE, $R^2$, and PE. However, regarding MAPE, the NN are slightly better than the AINN.

Since the DLEM exists within the AINN, it could be considered a constrained optimization problem. The training process of the AINN not only minimizes the mean square error (the loss function) between the predicted $N_2O$ flux and the measured $N_2O$ flux but also needs to satisfy the constraints of the parameters fed into the DLEM. It is observed that the AINN could converge to a specific basin of attraction. This is consistent with Bayesian theory, which suggests that overall regression accuracy could be improved if prior information about the system is obtained. Among all the four AINN, MLP-DLEM, CNN-DLEM, LSTM-DLEM, and CNN-Transformer-DLEM, the general performance of the LSTM-DLEM and CNN-Transformer-DLEM is comparable, and both models are better than CNN-DLEM. MLP-DLEM has the worst performance since MLP is not inherently designed to address time series data.

In the field experiment, collecting the nitrate quantity in the soil presents a challenge. Therefore, we assume that the nitrate quantity available in the soil decays exponentially over time. An indoor experiment in a controlled environment was conducted to verify this assumption. We inputted the soil temperature, soil moisture, and the amount of nitrate collected from the indoor experiment into the model, and the model could successfully predict the emission of $N_2O$. Thus, we conclude that the assumption of the nitrate quantity in the soil decaying exponentially is reasonable for predicting $N_2O$ with the field data. In this exploration, we demonstrated only four basic AINN: MLP-DLEM, CNN-DLEM, LSTM-DLEM, and CNN-Transformer-DLEM. We do not intend to create complex architectures of AINN in this context, as our focus is on understanding the underlying mechanisms of AINN. In the future, we could explore a variety of AINN such as CNN-LSTM-DLEM, LSTM-CNN-DLEM, and LSTM-Transformer-DLEM. All these models could be modeled as a constrained optimization problem, potentially achieving better generalization capabilities than

their neural network counterparts.

# Chapter 5

# Preserving Temporal Dynamics in Synthetic Multivariate Time Series Using Generative Neural Networks and Monte Carlo Markov Chain

## 5.1 Introduction

Time series analysis is widely employed in various fields, including stock market price prediction, monthly water consumption tracking, electrical signal analysis, weather forecasting, and $N_2O$ emission prediction [10, 239, 13, 18]. However, with the advent of deep learning in time series analysis, the challenge of dealing with small datasets has emerged as a significant impediment to enhancing model performance. For most time series applications, the cost associated with collecting sufficient data to train deep learning models is prohibitive [240]. Consequently, the utilization of augmented data to complement real training data is regarded as a practical approach for enhancing model performance [241].

### 5.1.1 $N_2O$ Emission from Farming

$N_2O$ is a potent greenhouse gas, with 300 times the heat absorption potential of $CO_2$ [201]. It also serves as the primary contributor to stratospheric ozone depletion [202]. In the realm of agriculture, synthetic nitrogen fertilizer has played a significant role in boosting harvests in recent decades, leading farmers to increasingly employ nitrogen fertilizer to maximize yields. Consequently, $N_2O$ emissions have seen a dramatic increase. In response to environmental and climate challenges, while promoting sustainable agriculture [212], Agriculture and Agri-Food Canada has initiated a program with the aim of reducing fertilizer emissions by 30% from 2020 levels by 2030.

To effectively regulate $N_2O$ emissions, it is crucial to predict emissions from agricultural fields using the flux of water, humidity, nitrate concentration, and temperature data. Traditional mechanistic models such as the CLM-CN [53], the DLEM [54, 55], and the EcoSys [242] have been developed to estimate $N_2O$ emissions. However, due to the considerable variability in soil $N_2O$ emissions, these models encounter challenges in accurately representing $N_2O$ emissions.

As interest in $N_2O$ emissions from agricultural fields has grown, researchers have turned to data-driven models to enhance predictions. Early efforts have involved a variety of machine learning models [208], including the MLP [209], SVM [243], RF [211], RBFNN [244], DBN [245], and LSTM [18], to establish connections between input variables and $N_2O$ emissions.

### 5.1.2 Motivations and Contributions

In order to study the dynamical system of the emission of $N_2O$ with a data-driven approach, an observation needs to be noticed: the multivariate time series representations do not expose the full information of the underlying dynamical system in a way that a deep learning model can easily learn [246]. Fundamentally speaking, the time series dataset we collected is just a finite-dimensional projection of a hypersurface of data called the phase space of a dynamical system. This projection results in a loss of information regarding the dynamics of the system. However, we can still make inferences about the dynamical system using a data-driven method. If sufficient time series datasets are presented to the model, the generalized error would be reduced greatly. While a generalized proof establishing a direct relationship between the expected generalization error and number of augmented data points has yet to be theoretically formulated, numerous studies empirically demonstrate that using augmented data can boost both classification and regression accuracy comparable to the addition of real data [247, 118].

In this exploration, there are two major reasons for gathering sufficient and balanced time series data to fit data-driven models. Firstly, the LI-COR system used to collect the data is expensive, costing around 200,000 Canadian dollars per system. Secondly, we could collect only one sequence of time series data under specific conditions in one growing season. To be specific, we could collect 48 data points per day and 4,500 data points per growing season. Therefore, the lack of a sufficiently large dataset results in the inaccurate forecasting of $N_2O$ emissions with larger generalized error.

Our objective is to increase the dataset size using an innovative data augmentation approach that integrates GANs with sampling theory. This novel approach

is designed to generate multivariate time series data based on the original dataset. Simulation results illustrate that augmented data can effectively maintain the data distribution and the dynamic behavior of the original data.

Our contributions in this work are as follows:

1. A detailed analysis of time series data points generated by GAN models is provided, visually demonstrating that GAN-generated data preserves the data distribution but struggles to maintain the temporal dynamic behavior present in the original time series dataset.

2. A modified Metropolis-Hastings Algorithm to sample time series data from the data source generated by GAN models is introduced. The time series data generated by this algorithm retains most of the dynamic behavior of the original time series data.

3. The exponential moving average method is applied to smooth the time series dataset. Since the synthetic dataset retains the Markov Chain property (sometimes characterized as "memorylessness"), it lacks the long-term relationships present in the original time series dataset.

4. Several new evaluation methods to measure synthetic time series data are proposed, including the Kolmogorov-Smirnov Test (KS-Test), and the Hilbert-Schmidt Independence Criterion (HSIC), among others.

The remainder of the paper is structured as follows. Section 5.2 offers detailed information on the entire process of data augmentation, including the GAN module, sampling module, smoothing module, and evaluation module. In Section 5.3, we apply the synthetic time series data generated by different models to train the LSTM for predicting $N_2O$ emissions [248]. Section 5.4 summarizes the study's findings and discusses the potential for improving the regression performance of LSTM with augmented data.

## 5.2 Data Augmentation Process

As depicted in Figure 5.1, the process of data augmentation consists of four modules: the GAN module, sampling module, smoothing module, and evaluation module. The GAN module is utilized to generate the source data points for the time series. Subsequently, the time series data points are sorted in ascending order based on their timestamps and fed into the sampling module to create the raw time series data. This

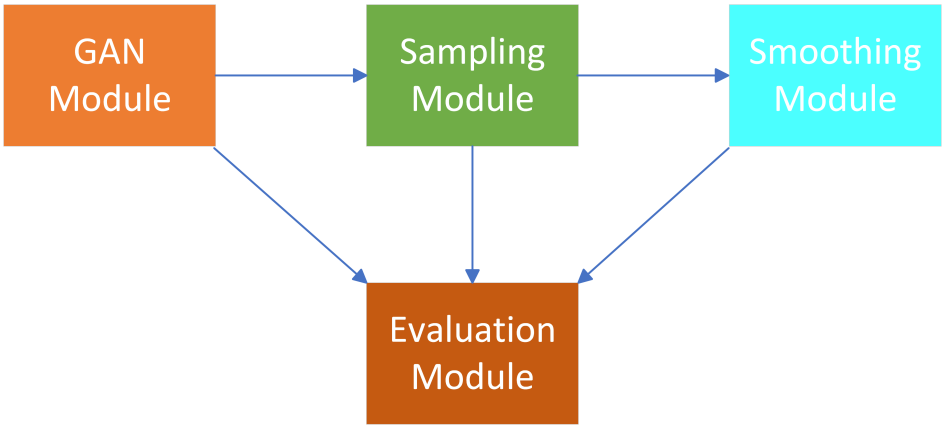



FIGURE 5.1: Process of Data Augmentation

approach ensures that the joint distribution of the first-order differences of the generated time series remains the same to the original data. Since the generated time series data from the sampling module are independent and identically distributed, the synthetic data are then passed through a smoothing module to make the time series data more continuous. Finally, the quality of the generated time series data is assessed using various metrics, including HSIC, autocorrelation function (ACF), KS-test, and the train-synthesis-and-test-real (TSTR) method.

The detailed information of each module is shown in the following:

1. In the process of generating time series data points, a column of timestamp will be added into the original time series dataset. Then the original time series data with the timestamp will be scaled into the range of [0.05, 0.95] (since 0 and 1 is impossible to reach with the sigmoid function as the activation function, the dataset is scaled in a slightly narrower range than the range of [0, 1]). Then the processed dataset is divided into blocks with certain length, for instance, 32 time steps. All these blocks (including the timestamps) are fed into the selected GAN model as the real data to train the GAN with the latent space of 100 dimensions. In this exploration, we generally use the WGAN model with a gradient penalty to generate the source of time series data.

2. When the GAN model is converged and well-trained, the generator of the GAN model will generate synthetic time series data points with a size of $1,920,000$. These synthetic data points will be ordered in a time-ascending manner and fed into the sampling module to generate a raw time series dataset, which has the same length as the original dataset. In this case, a modified Metropolis–Hastings algorithm is used as the sampling algorithm. Although generating more data point could be advantageous, if the additional data is diverse,

and computational resources allow for efficient training. However, if the new points are mostly repetitive or if resources are limited, focusing on quality and diversity in a well-selected $1,920,000$ points might yield similar performance gains.

3. The synthetic time series data are then fed into the smoothing module, which uses the EMA operator to make the time series smooth.

4. During the entire process, the synthetic data generated by every module will be evaluated with different metrics, including qualitative and quantitative approaches, such as HSIC and the TSTR method.

### 5.2.1 Data Description

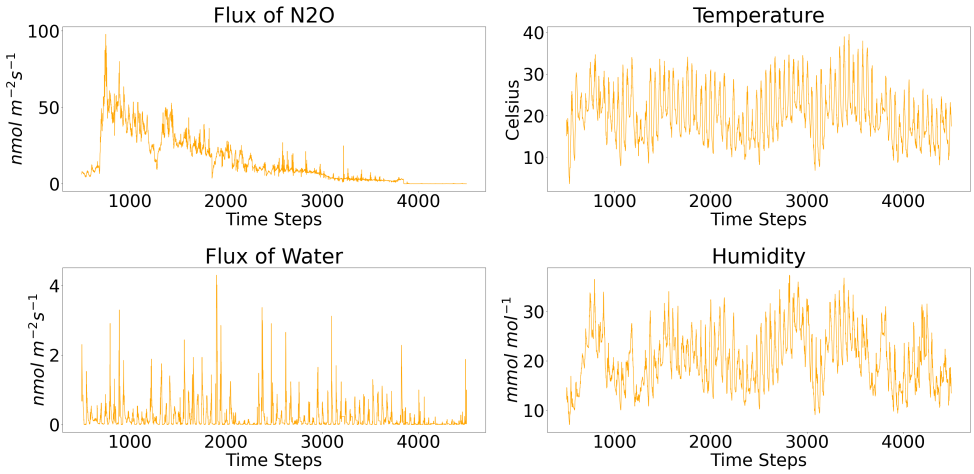



FIGURE 5.2: Original Time Series Data from the 2021 Growing Season

Figure 5.2 shows the multi-dimensional time series data we used in this exploration. The time series data were collected at a smart farm named Area X.O located in Ottawa, Ontario, Canada (45.3189° N, 75.7562° W), from in-field sensors nodes, namely, $N_2O$ emission sensors, and weather sensors, every 30 minutes [219].

### 5.2.2 GAN Module

In this exploration, for the architecture of the generator G and discriminator D, we adopt architectures from the DCGAN model [249], with D being slightly more complex than G. As for the training algorithm, we apply the WGAN-GP algorithm [151]. In the following section, unless specified otherwise, the default GAN model is a combination of WGAN with a DCGAN-like architecture, and for convenience, we refer

to it as the "WGAN" model. Please note in Figure 5.11, the training dataset needs to be scaled to the range of [0.05, 0.95] before fitting it into the neural network. We have not transferred it back to the normal units; therefore, the y label is in the range of [0.05, 0.95].

**Training Process of WGAN model**

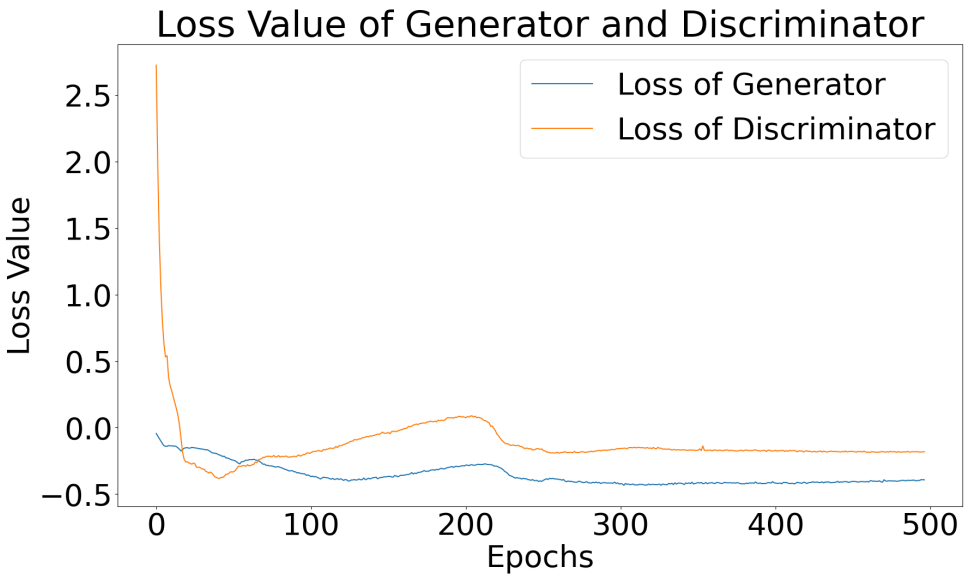



FIGURE 5.3: Loss Value for the Discriminator and Generator in WGAN

As depicted in Figure 5.3, we can see the loss curves for both the discriminator and generator. Since we change the sign for the original loss value, both the loss of discriminator and generator could be decreased over time, in general. However, it is a challenge for us to determine when to terminate the training through the loss value of the discriminator and generator only. It is easy to see that after 250 epochs, the loss value did not supply enough information for us to determine the training status of the WGAN. However, the purpose of the WGAN is to learn the underlying distribution of the data, and we can use this as our criteria to determine whether we need to terminate the training of WGAN.

As shown in Figure 5.4, we collected $1,920,000$ data points generated by the WGAN model at epoch: 100, 200, 300, 400, and 500. As the number of training epochs increases, the distribution of generated data gradually approaches that of the real data. The distributions of data generated at epoch 400 and epoch 500 did not show significant differences. This indicates that it is an appropriate time to terminate the training of the WGAN at epoch 500, as further training would not yield any benefit or improvement in the generated data.

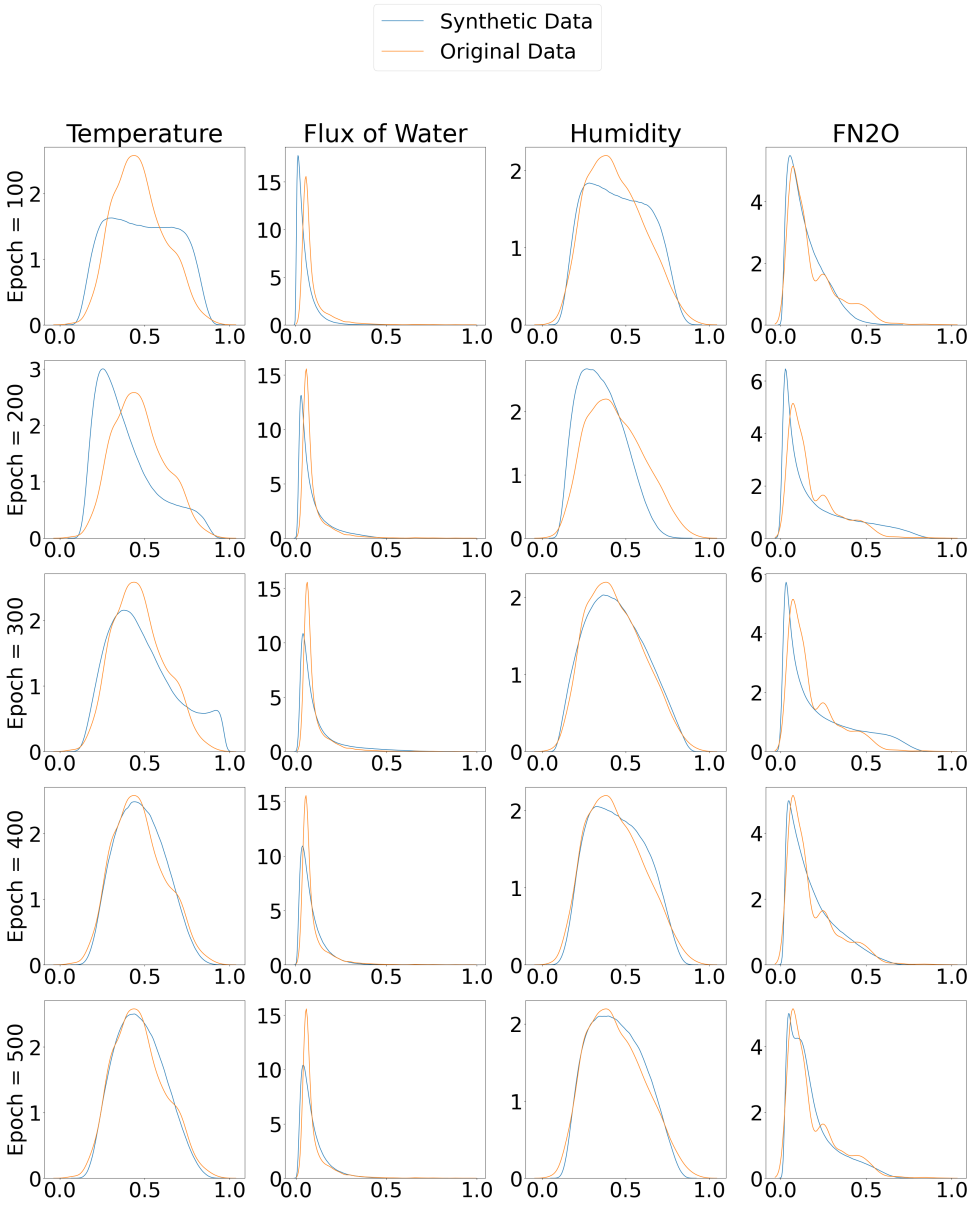



FIGURE 5.4: Distribution of Real Data and Synthetic Data

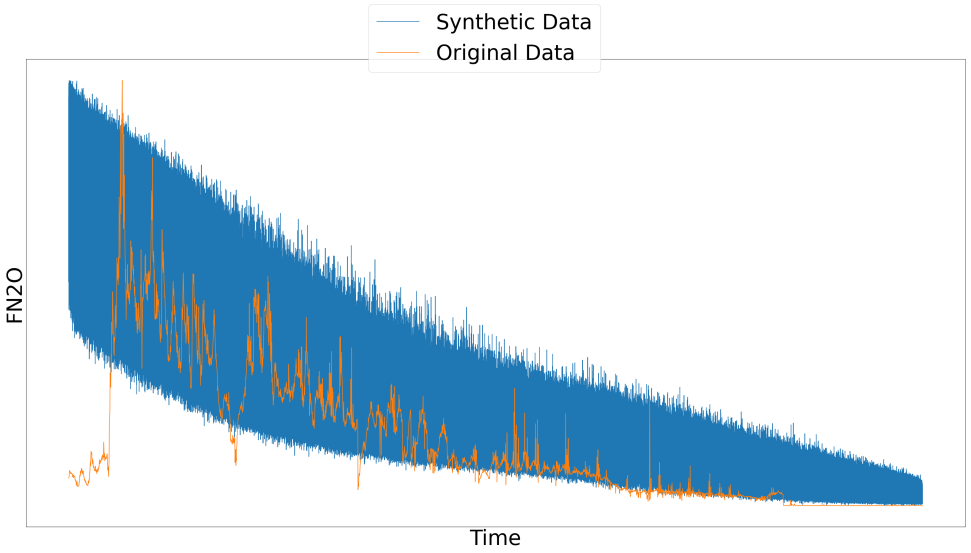



FIGURE 5.5: Synthetic and Original Flux of $N_2O$

Now, let us examine the data in ascending time order. As shown in Figure 5.5, the blue curve displays all the $1,920,000$ data points of the flux of $N_2O$ generated by WGAN, arranged in ascending order of time, while the yellow curve shows the original data for the flux of $N_2O$, which is also arranged in ascending order of time. It is evident that the trend in the synthetic data broadly follows that of the original data. Now, let us focus on the "local" data, by which we mean data at specific time points. Although we could not generate data at the exact same time points multiple times, we extracted and observed data generated within very small time intervals, as illustrated in Figure 5.6. Through observation, it can be generally concluded that data generated at the same time points tend to follow a Gaussian distribution. The underlying mathematical reasons are quite intriguing and might be related to the central limit theorem or ergodic hypothesis. Further investigation and discussion of this phenomenon will be addressed in another exploration. However, this characteristic of synthetic data lays the foundation for the sampling procedure.

### 5.2.3 Sampling Module

How can we generate time series data that faithfully preserves the dynamical behavior of the system? In essence, where does this dynamical behavior manifest itself? To gain insights, let us explore the representation of the dynamical system in the form of differential equations, as illustrated below:

$$\frac{\partial x}{\partial t} = f(x,t) \tag{5.1}$$

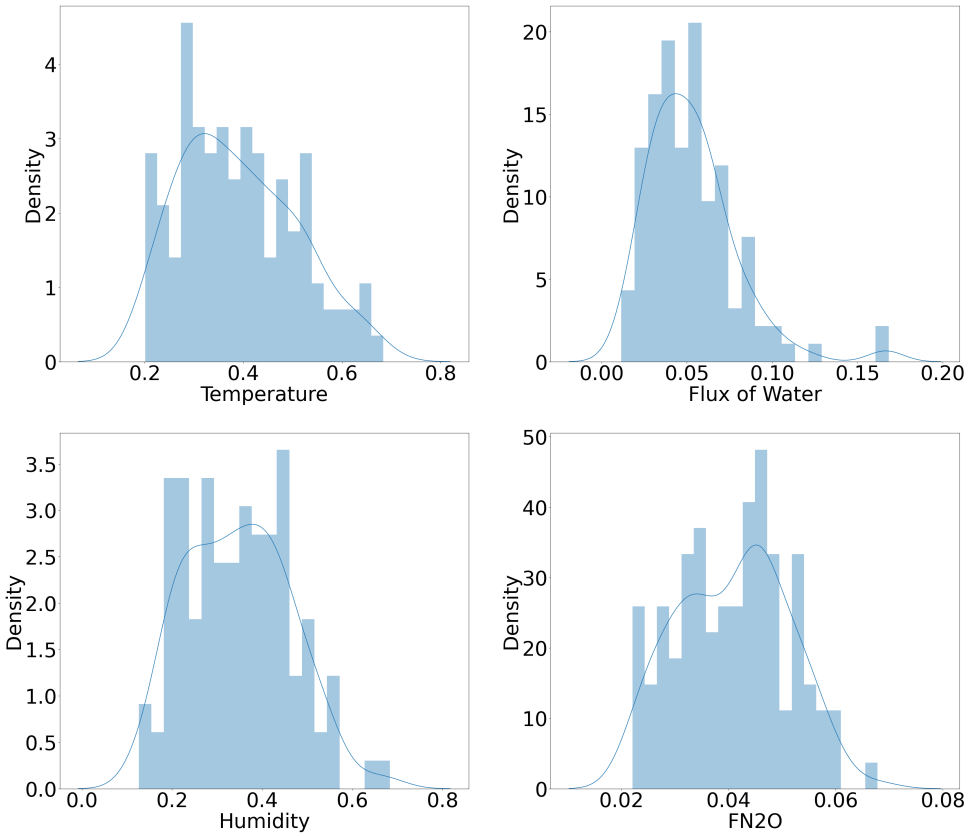



FIGURE 5.6: Distribution of Data at One Specific Time

where $x$ is the variable we care about and $t$ is the time.

It is quite convenient for us to conclude that if the synthetic time series can preserve the distribution of the first-order differences in a discrete dynamical system or the derivatives in a continuous dynamical system, then the dynamical properties of the synthetic time series can be largely preserved. Therefore, we have proposed a sampling algorithm based on MCMC, which will be discussed in detail in the following section.

After the WGAN model converges, $1,920,000$ data points can be generated, with each point having a timestamp. After arranging the $1,920,000$ data points in ascending order based on their timestamps, we utilize and modify the Metropolis-Hastings Algorithm to generate the multivariate time series, as illustrated in Algorithm 2.

### Modified Metropolis–Hastings Algorithm

Algorithm 2 receives the time series source $S$, which is organized in ascending time order, from the WGAN model. Typically, the time range of $S$ should be slightly wider than the original time series. The prior distribution $\pi(\theta)$ is derived from the

**Algorithm 2** Modified Metropolis–Hastings algorithm

**Output**: A time series $tlst$
**Input**: A time series source $S$ with ascending timestamp, a prior distribution $\pi(\theta)$, an original time series dataset $OS$, and a starting point $spv$
**Initialization**: Initialized $\theta$; an empty time series list $tlst$; an initialized vector $v$; and an $\beta$ factor
**for** i $\leftarrow$ 1 to M **do**
  Draw the timestamp $st$ and the vector $opv$ from the original time series dataset $OS$ .
  **for** j $\leftarrow$ 1 to N **do**
    Draw the timestamp $gst$ and the corresponding vector $spv$ from the time series source $S$ .
    **if** abs($st$ - $gst$) $\leq \delta$ **then**
      **if** The length of time series $tlst$ is zero **then**
        $\theta^{'} = v$ - $opv$,
      **else**
        $\theta^{'} = (1 - \beta)(v - spv) + \beta(v - opv)$,
      **end if**
      $\gamma = \min\{\frac{\pi(\theta^{'})}{\pi(\theta)+\epsilon}, 1\}$,
      Draw $u \sim$ Uniform(0, 1),
      **if** $u \leq \gamma$ **then**
        $\theta = \theta^{'}$,
        $spv = v$,
        Add random vector $v$ into the time series list $tlst$,
        Break the inside loop.
      **end if**
    **end if**
  **end for**
**end for**
**return** a time series $tlst$.

original dataset. In the outer 'for' loop, we draw the timestamp $st$ and the corresponding vector $opv$ from the original dataset $OS$ for reference. In the inner 'for' loop, we draw the timestamp $gst$ and the corresponding vector $v$. If the timestamp $gst$ falls within a tolerance of $\delta$ of the timestamp $st$, then $\theta^{'}$ is calculated based on the information from the previously selected vector and the original time series data.

In the initialization phase, the time series $tlst$ has not yet been generated. Thus, we take the first element of the original dataset as the reference and calculate the first-order difference $\theta^{'}$ by subtracting $v$ from $opv$. Initially, $\theta$ is set to the original point in the multi-dimensional space. Once the time series $tlst$ begins to have generated elements, we update the first-order difference $\theta^{'}$ to consider both the generated elements and the original elements. As outlined in the algorithm, a factor $\beta$ is introduced to control the balance between information derived from the original dataset and information from the generated elements. The value of $\beta$ significantly influences the degree of similarity between the generated dataset and the original dataset, and we will delve into its impact in detail in the following section.

Next, we calculate $\alpha$ by comparing the probabilities of $\pi(\theta^{'})$ and $\pi(\theta)$. If $\pi(\theta^{'})$ is greater than $\pi(\theta)$, then this vector is selected for inclusion in the generated time series. However, if $\pi(\theta^{'})$ is smaller than $\pi(\theta)$, indicating that $\gamma$ is less than 1, we proceed with a probability test. We generate a random number $u$ from a uniform distribution to determine whether we accept this vector. If $u$ is less than or equal to $\gamma$, we accept the element and assign $v$ to $spv$. If $u$ is greater than $\gamma$, we take no action. As the generated time series data reaches the same length as the original time series data, the algorithm will output the generated time series data.

### The effect of $\beta$

Figures 5.7 and 5.8 depict the ACF and PACF of time series data generated with different $\beta$ values. In both figures, each row corresponds to a different $\beta$ value: original data in the first row, $\beta$ = 0 in the second row, $\beta$ = 0.10 in the third row, and $\beta$ = 0.50 in the fourth row. When $\beta = 0$, the generated time series data does not exhibit any discernible periodic patterns. However, as we increase $\beta$ from 0 to 0.10, the generated data show a periodic pattern to some extent. Upon further increases in $\beta$ to 0.50, the synthetic data have completely captured the periodic information that exists in the original data. Interestingly, variations in $\beta$ do not substantially alter the shape of the PACF for the generated data, as shown in Figure 5.8.

Figure 5.9 illustrates the joint distribution of first-order differences in time series data generated with varying $\beta$ values akin to Figures 5.7 and 5.8. In each figure, rows correspond to different $\beta$ values: original data in the first row, $\beta$ = 0 in the

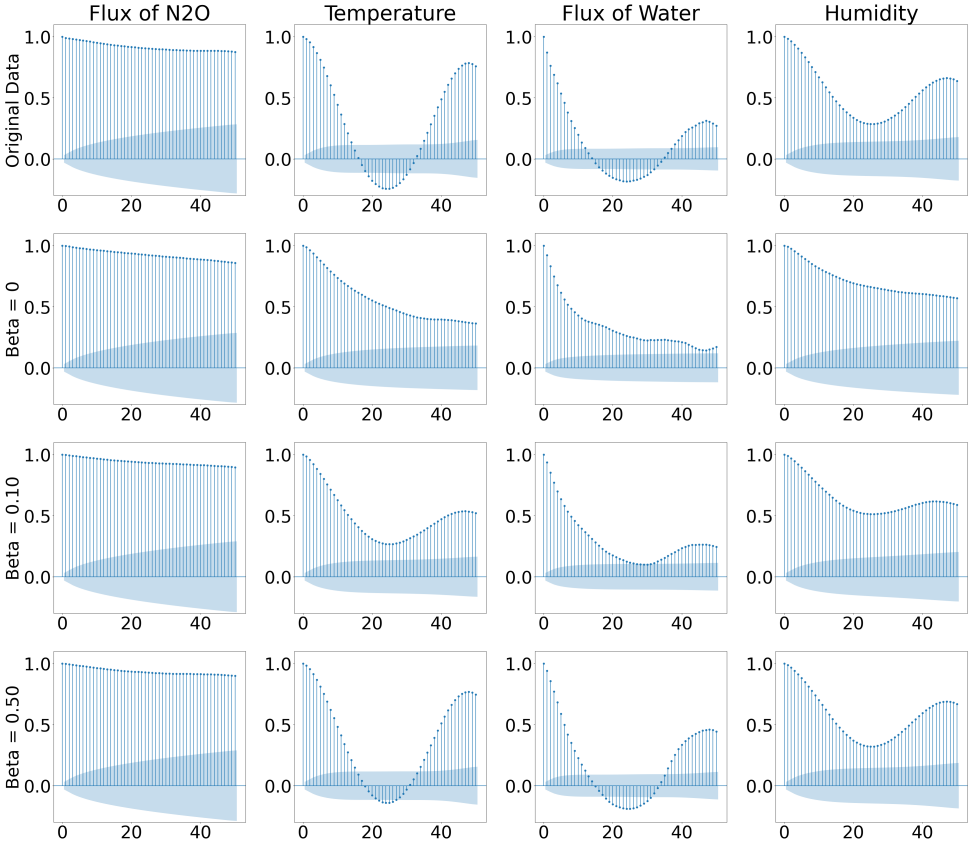



FIGURE 5.7: Autocorrelation Analysis for $N_2O$ Flux, Temperature, Flux of Water, and Humidity with Different $\beta$ Values

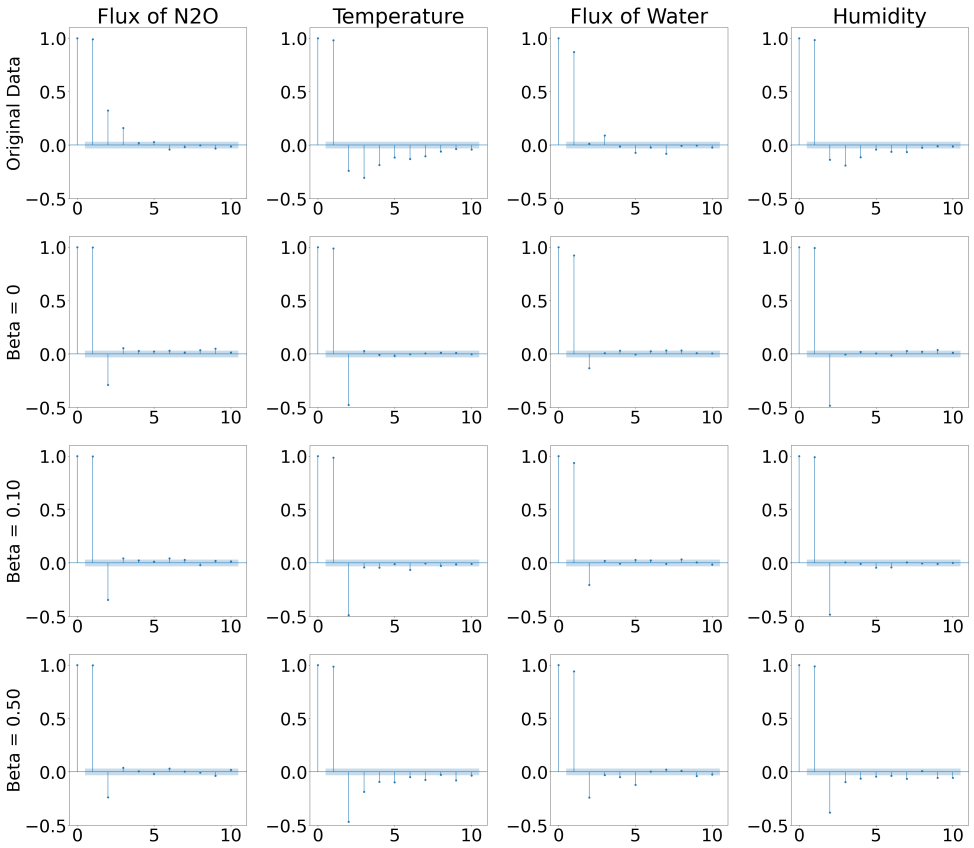



FIGURE 5.8: Partial Autocorrelation Analysis for $N_2O$ Flux, Temperature, Flux of Water, and Humidity with Different $\beta$ Values

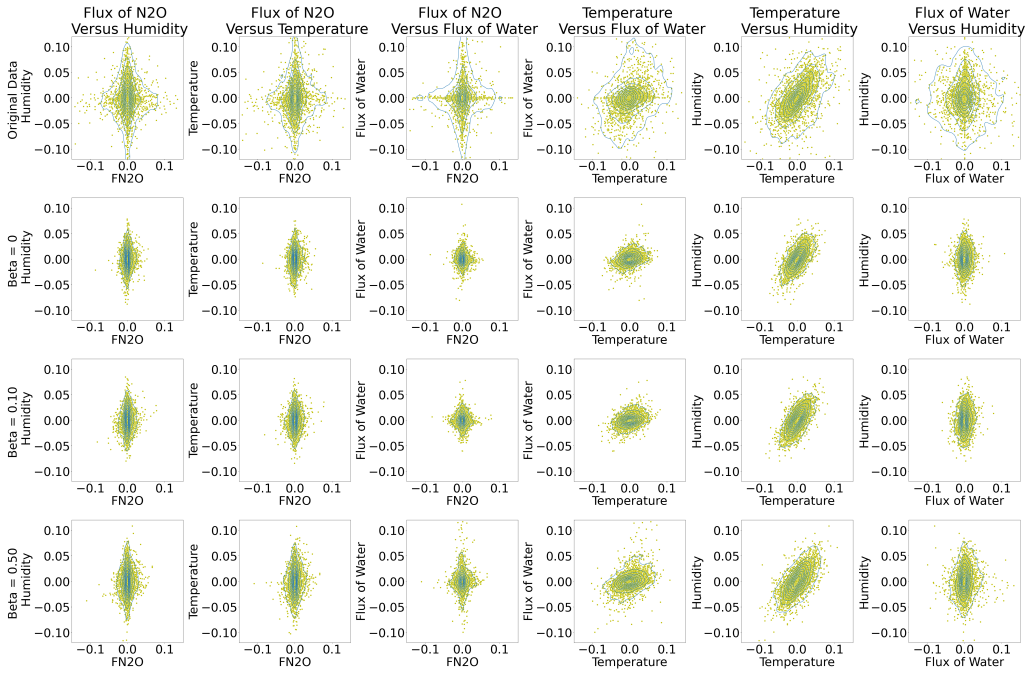



FIGURE 5.9: Joint Distribution Among Temperature, Flux of Water, Humidity, and Flux of $N_2O$ with Different $\beta$ Values

second row, $\beta = 0.10$ in the third row, and $\beta = 0.50$ in the fourth row. It is evident that the $\beta$ value has an impact on the joint distribution of first-order differences in the generated time series data. Within the data generated using the same $\beta$ value, the joint distribution of first-order differences remains consistent, as confirmed by the KS-test, as shown in Table 5.2. Furthermore, regardless of the chosen $\beta$ value, the principal component of the first-order differences in the generated time series data remains consistent with that of the original time series data.

Table 5.1 shows the HSIC values between the generated data and the original data, which increase as the $\beta$ value increases. This suggests that a larger $\beta$ value leads to greater similarity between the generated data and the original data. However, even when $\beta = 0$, the HSIC value for the generated data still reaches approximately 0.7954. This indicates that the generated data captures the major properties and patterns of the original data. Please note that the HSIC values before and after scaling differ. Generally, the HSIC value after scaling is higher than the value before scaling. The HSIC value here is calculated after data scaling.

TABLE 5.1: HSIC Values of Generated Data with Different $\beta$ Values

| $\beta$ | 0 | 0.05 | 0.1 | 0.5 | 1 |
|---|---|---|---|---|---|
| HSIC | 0.7954±0.0119 | 0.8629±0.0047 | 0.8815±0.0027 | 0.9459±0.0009 | 0.9636±0.0004 |

### 5.2.4 Smoothing Module

The data points generated by WGAN are independent and identically distributed, and the MCMC process does not establish long-term relationships among data points. Meanwhile, the white noise effect is unavoidable in synthetic time series data. Therefore, it is necessary to smooth the synthetic time series data using information from the previous data points. In this study, we adopt EMA for this purpose. The simulation results demonstrate that with EMA, the behavior of the synthetic data closely resembles that of the original data.

To quantitatively evaluate the quality of synthetic data, we calculate the HSIC and the p-value of the KS-test between the generated data and the original data. As shown in Table 5.2, as $\alpha$ decreases (indicating smoother curves), the HSIC value and the p-value of the KS-test between the generated data and the original data initially increase and then decrease. The optimal performance is typically achieved around $\alpha = 0.4$ or $\alpha = 0.5$ depending on the characteristics of the synthetic data.

TABLE 5.2: The P-Value of the KS-test and the HSIC Value Between Synthetic Data Sample ($\beta = 0.5$) and Original Dataset

| P[1] \ [2] / $\alpha$ | Temp | $FH_2O$ | Humi | $FN_2O$ | HSIC |
|---|---|---|---|---|---|
| 1 | 9e-54 | 5e-95 | 4e-59 | 1e-39 | 0.9810 |
| 0.9 | 4e-40 | 3e-81 | 1e-44 | 1e-32 | 0.9835 |
| 0.8 | 2e-29 | 3e-69 | 7e-34 | 1e-28 | 0.9854 |
| 0.7 | 5e-19 | 1e-69 | 4e-24 | 4e-24 | 0.9867 |
| 0.6 | 1e-13 | 4e-67 | 1e-13 | 3e-17 | 0.9873 |
| 0.5 | 4e-7 | 2e-64 | 4e-6 | 6e-15 | 0.9869 |
| 0.4 | 0.0045 | 1e-62 | 0.0615 | 3e-16 | 0.9847 |
| 0.3 | 5e-5 | 1e-64 | 4e-17 | 9e-18 | 0.9788 |
| 0.2 | 1e-19 | 1e-76 | 2e-23 | 6e-20 | 0.9640 |
| 0.1 | 1e-73 | 1e-89 | 2e-76 | 6e-54 | 0.9283 |

[1] In this table, P stands for p-value. If p-value is not below the threshold of 0.05, it indicates two sequences of data share the same distribution.

[2] In the header, "temp" corresponds to "temperature," "$FH_2O$" to "flux of water," "humi" to "humidity," and "$FN_2O$" denotes flux of $N_2O$.

### 5.2.5 Evaluation Module

As illustrated in Figure 5.10, we generated time series data using various combinations of $\beta$ (the coefficient in the Modified Metropolis–Hastings algorithm) and $\alpha$ (the coefficient in the EMA function). The first four plots display the generated time series with $\beta = 0$. The second and third set of four plots showcase the generated time series with $\beta = 0.10$ and $\beta = 0.50$, respectively. In all cases, $\alpha$ is consistently set to 0.4. It is evident that as $\beta$ increases, the generated time series increasingly resembles the original time series, as shown in Figure 5.2. This observation aligns with the findings presented in Table 5.1.

TABLE 5.3: The P-Value of the KS-test and the HSIC Value Between Two Synthetic Time Series with the Same $\beta$

| P[1] [2] / $\alpha$ | Temp | $FH_2O$ | Humi | $FN_2O$ | HSIC |
|---|---|---|---|---|---|
| 1 | 0.3849 | 0.9689 | 0.8279 | 0.9011 | 0.9671 |
| 0.9 | 0.5542 | 0.8593 | 0.7943 | 0.9250 | 0.9708 |
| 0.8 | 0.4657 | 0.8114 | 0.7225 | 0.9356 | 0.9741 |
| 0.7 | 0.5180 | 0.9011 | 0.6286 | 0.7769 | 0.9772 |
| 0.6 | 0.4488 | 0.7769 | 0.5911 | 0.8739 | 0.9802 |
| 0.5 | 0.3270 | 0.7409 | 0.7409 | 0.7943 | 0.9832 |
| 0.4 | 0.3270 | 0.5726 | 0.7225 | 0.5726 | 0.9862 |
| 0.3 | 0.3849 | 0.3849 | 0.5911 | 0.5726 | 0.9894 |
| 0.2 | 0.4323 | 0.3552 | 0.4004 | 0.6851 | 0.9929 |
| 0.1 | 0.3552 | 0.9541 | 0.3552 | 0.7590 | 0.9967 |

[1] In this table, P stands for p-value. If p-value is not below the threshold of 0.05, it indicates two sequences of data share the same distribution.
[2] In the header, "temp" corresponds to "temperature," "$FH_2O$" to "flux of water," "humi" to "humidity," and "$FN_2O$" denotes flux of $N_2O$.

To further explore the impact of the values of $\beta$ and $\alpha$ on the properties of the generated time series, we calculate the p-value of the KS-test and the HSIC value between two generated time series (with the same $\beta$ or with different $\beta$). As shown in Table 5.3 and Table 5.4, the results are quite revealing. Table 5.3 demonstrates that time series generated with the same $\beta$ share a similar distribution for the first-order difference, as evidenced by the p-values of the KS-test exceeding the threshold value of 0.05. Additionally, as the $\alpha$ value decreases from 1 to 0.1, the HSIC value increases from 0.9671 to 0.9967. This suggests that as the curve becomes smoother,

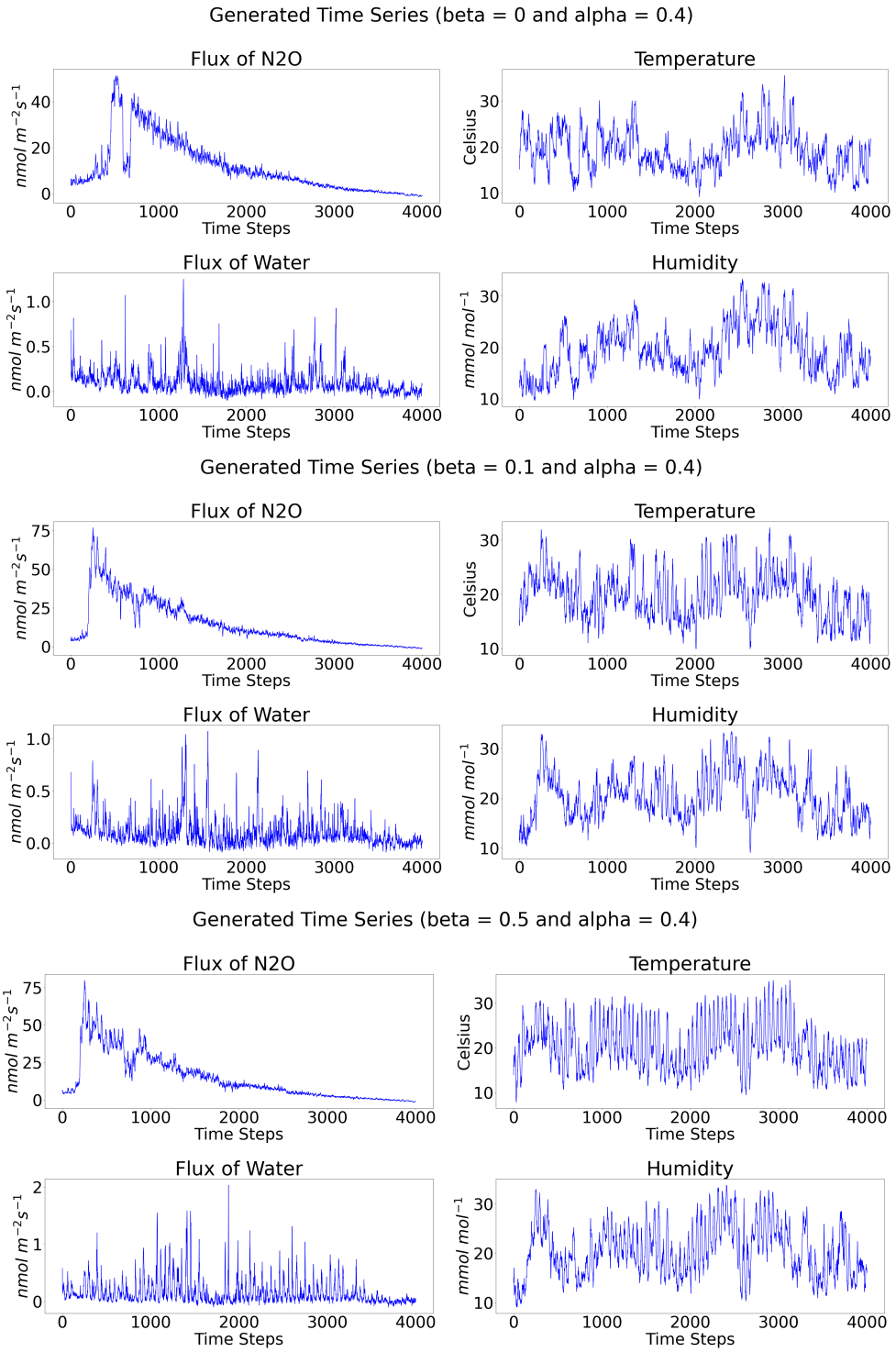



FIGURE 5.10: Synthetic Data Generated with Different $\beta$ Values

TABLE 5.4: The P-Value of the KS-test and the HSIC Value Between Two Synthetic Time Series with $\beta = 0.05$ and $\beta = 0.10$

| P[1] \ [2] / $\alpha$ | Temp | $FH_2O$ | Humi | $FN_2O$ | HSIC |
|---|---|---|---|---|---|
| 1 | 6e-6 | 0.0002 | 0.0776 | 6e-7 | 0.9275 |
| 0.9 | 1e-5 | 0.0001 | 0.0455 | 1e-5 | 0.9279 |
| 0.8 | 5e-5 | 0.0001 | 0.0615 | 6e-5 | 0.9300 |
| 0.7 | 0.0002 | 6e-5 | 0.0615 | 0.0001 | 0.9323 |
| 0.6 | 0.0004 | 3e-5 | 0.0332 | 0.0001 | 0.9349 |
| 0.5 | 0.0007 | 4e-5 | 0.0274 | 0.0003 | 0.9379 |
| 0.4 | 0.0020 | 1e-5 | 0.0546 | 0.0033 | 0.9418 |
| 0.3 | 0.0067 | 3e-6 | 0.0210 | 0.0053 | 0.9472 |
| 0.2 | 0.0240 | 3e-7 | 0.0112 | 0.0332 | 0.9555 |
| 0.1 | 0.0402 | 3e-13 | 0.0031 | 0.0652 | 0.9705 |

[1] In this table, P stands for p-value. If p-value is not below the threshold of 0.05, it indicates two sequences of data share the same distribution.
[2] In the header, "temp" corresponds to "temperature," "$FH_2O$" to "flux of water," "humi" to "humidity," and "$FN_2O$" denotes flux of $N_2O$.

the generated time series become more similar to each other. In other words, the average behavior of the generated time series becomes nearly identical.

As demonstrated in Table 5.4, it is generally challenging to conclusively state that time series generated with different $\beta$ values share the same distribution for the first-order difference. This is because most of the p-values from the KS-test fall below the threshold value of 0.05. However, when examining the humidity term and the flux of $N_2O$ term, it becomes apparent that if the $\alpha$ value is appropriately selected, the KS-test indicates that the first-order difference between these two generated time series shares the same distribution. It is worth noting that due to the limited number of data points in this investigation (only 4,000), the confidence in the KS-test results may not be very robust. Therefore, we utilize the p-value from the KS-test as a reference to gauge the similarity between the generated time series. Once again, as the $\alpha$ value decreases from 1 to 0.1, the HSIC value increases from 0.9275 to 0.9705. This implies that as the time series becomes smoother, the generated time series become more alike.

## 5.3 Experiment and Simulation

### 5.3.1 Measure Metrics

To measure the difference between two time series data, four performance indicators are utilized in this experiment, namely: mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and coefficient of determination ($R^2$) as shown in the following:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{5.2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \tag{5.3}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{n}} \tag{5.4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{5.5}$$

where $n$ is the number of data points, $x_i$ represents the observed value, $\hat{x}_i$ represents the predicted value, and $\overline{x}$ is the mean of the variable.

### 5.3.2 Train on Synthetic Data and Test on Real Data

Numerous GANs exist in the research field for generating time series data, including RGAN, WGAN, and TimeGAN. All of these models claim to preserve the temporal dynamics in time series datasets. However, in practice, they often struggle to explicitly capture the dynamic behavior inherent in time series data. To illustrate this point, our exploration includes an approach for training the GAN in the feature space and generating feature data from the original time series. To facilitate an easy comparison between the GAN-MCMC approach and the GANs, including RGAN, TimeGAN, and WGAN, LSTMs trained on synthetic data will be evaluated on original data, as shown in Figure 5.11. It is observed that the GAN can partially learn the dynamics present in the original time series data. However, all of them perform worse than

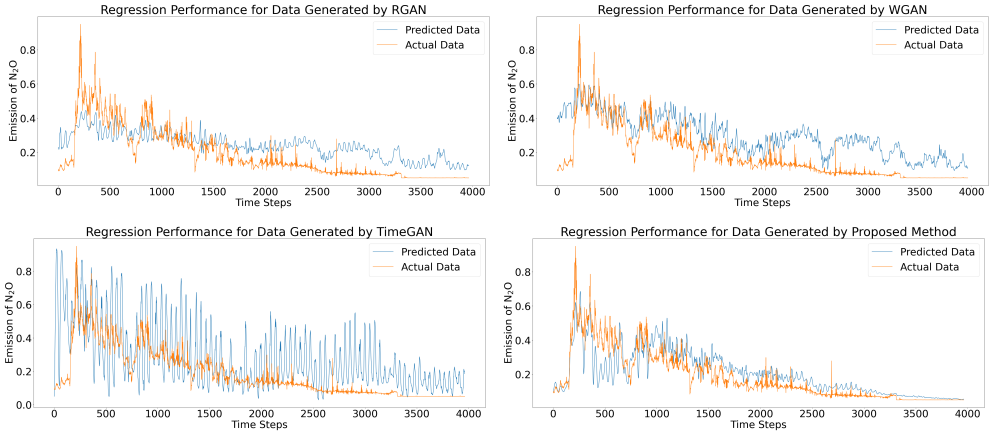



FIGURE 5.11: The neural network models are trained by the feature blocks generated by RGAN, WGAN, TimeGAN, and GAN-MCMC ($\beta = 0.1$).

our newly proposed method, as shown in the first chart in Figure 5.11. Furthermore, using a GAN alone to generate time series data presents two challenges:

1. In general, GANs learn the underlying distribution present in the original dataset. It is a challenge for GANs to learn the temporal dynamics from the time series data.

2. In most cases, GANs can only generate feature blocks. It is a difficulty for GANs to generate entire pieces of time series data with timestamps attached to them unless we add conditions to the GANs.

To further demonstrate the performance of each GAN and our proposed GAN-MCMC approach, RMSE, MAE, MAPE, and $R^2$ are used to evaluate the performance of the different synthetic data generated by the various approaches. Please note that, in order to measure how much temporal dynamics exist in the synthetic data, large and complex testing models are developed, and the terminating condition is that the training loss of the model should be under a threshold value; in this case, it is $1 \times 10^{-5}$. The stochastic effect in generating the data and in evaluating the model should not be ignored. The testing results demonstrated here are general conclusions. We run the evaluating models ten times to reduce the stochastic effect. To demonstrate how much temporal dynamics are preserved in the synthetic data, 20,000 data blocks generated by RGAN, WGAN, TimeGAN, and the multivariate time series generated by the GAN-MCMC approach with different $\beta = 0, 0.05, 0.1$ are used to train LSTMs for predicting the emission of $N_2O$ from farming. Then we use the original data to test the LSTMs. As shown in Table 5.5, it is observed that synthetic data generated by GAN-MCMC outperform the data produced by RGAN, WGAN,

and TimeGAN. In general, the performance of WGAN is worse than that of RGAN in preserving temporal dynamics. TimeGAN performed better than RGAN in terms of MAPE, but in terms of RMSE, MAE, and $R^2$, it performed slightly worse. However, from the perspective of the standard deviation of RMSE, MAE, MAPE, and $R^2$, its performance is the most stable among WGAN, RGAN, and TimeGAN. However, the synthetic data generated by GANs is a stochastic process; therefore, it is quite unpredictable how much temporal dynamics exist in the synthetic data generated by GANs, even if the distribution of synthetic data is appropriate compared to the original data. It is relatively straightforward to understand why RGAN performed best among the three GANs, because we use LSTM as the testing model. However, no matter how realistic the synthetic data generated by GANs is, it is quite difficult for them to compete with the synthetic data generated by the GAN-MCMC approach, since MCMC is a process that preserves the temporal dynamics present in the original data. Furthermore, with increasing $\beta$, more temporal dynamics are retained in the synthetic data. Therefore, from Table 5.5, it is straightforward to observe that as the $\beta$ value increases from 0 to 0.1, all RMSE, MAE, MAPE, and $R^2$ values improve gradually.

TABLE 5.5: Performance Comparison among Different Models on Generating Agriculture Data

| | RMSE | MAE | MAPE | $R^2$ |
|---|---|---|---|---|
| RGAN | 0.141±0.045 | 0.123±0.041 | 1.127±0.399 | -0.052±0.664 |
| WGAN | 0.250±0.088 | 0.218±0.081 | 1.892±0.625 | -2.364±2.263 |
| TimeGAN | 0.196±0.011 | 0.144±0.010 | 0.739±0.069 | -0.801±0.200 |
| GAN-MCMC ($\beta$=0) | 0.117±0.015 | 0.064±0.010 | 0.252±0.069 | 0.332±0.171 |
| GAN-MCMC ($\beta$=0.05) | 0.095±0.017 | 0.076±0.018 | 0.583±0.184 | 0.553±0.143 |
| GAN-MCMC ($\beta$=0.1) | 0.089±0.018 | 0.068±0.017 | 0.450±0.165 | 0.608±0.154 |

## 5.4 Conclusion Remarks

Traditional time series data augmentation methods primarily focus on GANs, including RGAN, WGAN, and TimeGAN. These GANs aim to replicate data distributions closely resembling the original time series data. However, they struggle to maintain the temporal dynamics of the system, potentially missing crucial information regarding the first-order differences present in the original time series.

In this exploration, the original data were collected from Area X.O and were used to predict $N_2O$ emissions from farming [248, 18]. It is an ideal dataset for us to use as an example, since the data were generated from a complex dynamic system, and the equipment used to collect the data is too expensive to be widely utilized. Therefore, using data augmentation techniques to generate synthetic agricultural data is both urgent and beneficial for improving the performance of the predictor developed. The proposed GAN-MCMC approach, along with other GANs, including RGAN, WGAN, and TimeGAN, is used to generate synthetic multivariate time series data for predicting $N_2O$ emissions. Our investigation provides evidence that the time series data generated using the GAN-MCMC approach preserves most of the temporal dynamics and the characteristics of the first-order distribution observed in the original time series data.

In particular, we have thoroughly explored the role of the $\beta$ factor within the modified Metropolis-Hastings algorithm. This factor controls the degree of information preservation from the original time series. Our experiments have shown that with small $\beta$ values (around 0.50), it remains possible to effectively retain periodic information. Furthermore, the joint distribution of the first-order difference in synthetic time series data remains consistent when the same $\beta$ value is applied in the modified Metropolis-Hastings algorithm. Finally, we use the synthetic time series data to train LSTMs. Our results show that LSTMs trained on synthetic agricultural data generated by the GAN-MCMC framework outperform those trained on synthetic data produced by other GAN models, including RGAN, TimeGAN, and WGAN, in terms of RMSE, MAE, MAPE, and $R^2$.

# Chapter 6

# Toward A Quantum LSTM-DLEM Model for Predicting Agricultural $N_2O$ Emissions in Various Bifurcation Branches

## 6.1 Introduction

Traditionally, Long Short-Term Memory (LSTM) networks, renowned for their capacity to capture and preserve sequential dependencies, have played a pivotal role in analyzing time series data. This includes applications in natural language processing, speech recognition, electrical signal analysis, weather forecasting, and the study of nitrous oxide ($N_2O$) emissions [10, 11, 12, 13, 18].

With the emergence of quantum computing as a revolutionary paradigm promising to reshape the landscape of information processing, there is an anticipation that quantum computing could achieve exponential speed-up compared to classical algorithms. In this rapidly advancing field, the integration of quantum principles into machine learning models has garnered significant attention, paving the way for enhanced computational capabilities [250]. The ability of quantum states to exist in superposition can lead to a substantial speedup of computation in terms of complexity, as operations can be executed on many states simultaneously [251]. This quantum advantage is particularly pronounced in domains where classical neural networks face computational bottlenecks or grapple with combinatorial explosion [252].

### 6.1.1 $N_2O$ Emission from Farming

$N_2O$ stands out as a potent greenhouse gas, possessing 300 times the heat absorption potential of carbon dioxide ($CO_2$) [201]. Additionally, it serves as a primary

contributor to stratospheric ozone depletion [202]. Within the realm of agriculture, synthetic nitrogen fertilizer has played a crucial role in augmenting harvests in recent decades, prompting farmers to increasingly employ nitrogen fertilizer to maximize yields. Consequently, $N_2O$ emissions have experienced a significant surge. In response to environmental and climate challenges, while championing sustainable agriculture [212], Agriculture and Agri-Food Canada has initiated a program with the goal of reducing fertilizer emissions by 30% from 2020 levels by 2030.

Effectively regulating $N_2O$ emissions necessitates the accurate prediction of emissions from agricultural fields using data on water flux, humidity, nitrate concentration, and temperature. Traditional mechanistic models such as the Community Land Model with coupled Carbon and Nitrogen cycles (CLM-CN) [53], the Dynamic Land Ecosystem Model (DLEM) [54, 55], and the EcoSys [242] have been developed to estimate $N_2O$ emissions. However, due to the considerable variability in soil $N_2O$ emissions, these models encounter challenges in accurately representing $N_2O$ emissions.

As interest in $N_2O$ emissions from agricultural fields has grown, researchers have turned to data-driven models to enhance predictions. Early efforts have involved a variety of machine learning models [208], including the Multilayer Perceptron (MLP) [209], Support Vector Machine (SVM) [243], Random Forest (RF) [211], Radial Basis Function Neural Network (RBFNN) [244], Deep Belief Network (DBN) [245], and Long Short-Term Memory (LSTM) [18], to establish connections between input variables and $N_2O$ emissions.

### 6.1.2 Motivations and Contributions

The motivation behind this research is rooted at the intersection of two critical domains—quantum computing and agricultural practices. $N_2O$ emissions in the agricultural field pose a significant environmental challenge, contributing to global warming. Addressing this issue requires innovative approaches that leverage cutting-edge technologies. However, the first crucial step is to accurately predict the emission of $N_2O$ in farming. Quantum computing, with its potential for exponential speedup and improved handling of complex systems, presents a unique opportunity to enhance predictive models [253].

While traditional machine learning models, including classical LSTM, have made valuable contributions to $N_2O$ emission predictions [248], the limitations of classical computing become apparent as we grapple with the complexities of climate dynamics and gas emissions. Classical methods often struggle with the high-dimensional data and intricate relationships inherent in agricultural systems, limiting their predictive

power. This research aims to explore the transformative potential of quantum computing, specifically in the context of the Quantum LSTM-DLEM (QLSTM-DLEM) model, to revolutionize our ability to forecast $N_2O$ emissions with high accuracy.

Furthermore, integrating quantum computing into agricultural practices could facilitate precision farming by enabling real-time data analysis and decision-making, ultimately leading to more sustainable farming practices. The capacity of quantum algorithms to solve complex optimization problems can assist in resource allocation, crop management, and environmental monitoring, thereby improving overall agricultural efficiency and productivity [254]. By advancing our understanding of $N_2O$ emissions through the QLSTM-DLEM model, this research not only aims to provide a robust predictive framework but also contributes to the broader goal of utilizing quantum computing for sustainable agricultural innovation .

Our contributions in this work are as follows:

1. The time series data collected from different bifurcation branches undergo analysis using the Autocorrelation Function (ACF), and joint distribution of first-order differences. Signatures are incorporated into the time series from various bifurcation branches to distinguish between time series under different initial conditions and boundary conditions. Therefore, the dataset from different bifurcation branches could feed into one single model.

2. We propose a new approach, the QLSTM-DLEM, in sequence modeling tasks, demonstrating its superior performance over the classical LSTM-DLEM in capturing the temporal dependencies inherent in $N_2O$ emission data.

3. Our research highlights that the QLSTM-DLEM model, harnessing quantum parallelism and superposition, exhibits improved generalization and stability in predicting $N_2O$ emissions compared to its classical counterparts.

### 6.1.3 Organization

This paper is organized as follows. Section 6.1 provides an introduction to $N_2O$ emissions and traditional methodologies for analyzing sequential data and outlines the motivation and contributions of our exploration. Section 6.2 discusses the collection, statistical analysis, and preprocessing methods applied to the data. Section 6.3 details the components of the QLSTM-DLEM model, including QVC, QLSTM, and DLEM. It further presents the formulation and architecture of the proposed QLSTM-DLEM model designed for real-time prediction of $N_2O$ emissions at specific locations. In Section 6.4, we introduce four metrics for model evaluation, comparing simulation

results obtained from the QLSTM-DLEM and LSTM-DLEM models. Finally, Section 6.5 concludes the paper by highlighting that the QLSTM-DLEM model achieves comparable performance with its classical counterpart, with the added advantage of improved generalization and stability.

## 6.2 Data Collection and Description

### 6.2.1 Data Description

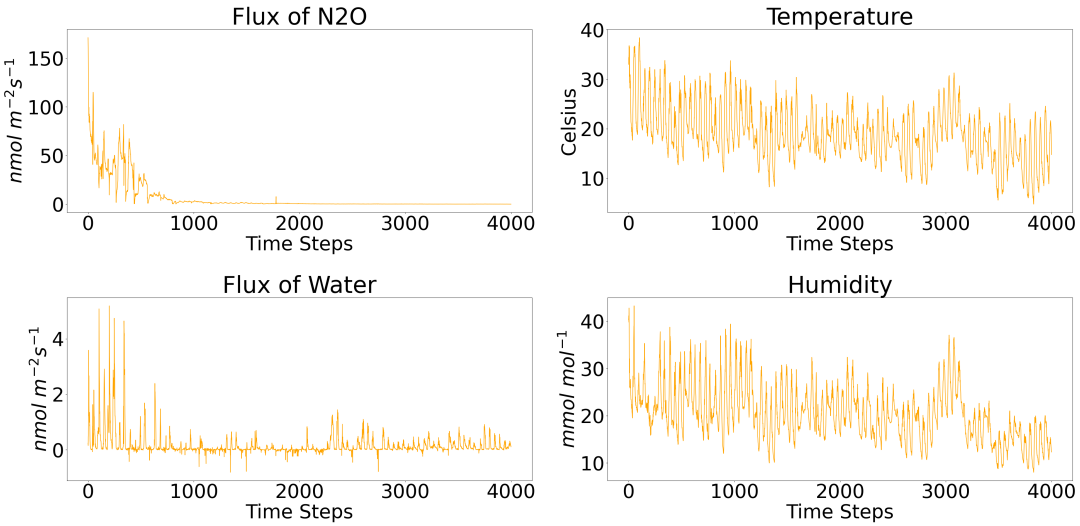



FIGURE 6.1: Data Collected in the 2023 Growing Season

The data used in this exploration was collected using LI-COR soil-gas measurement equipment, which consists of three main components [219]:

1. One LI-780 $N_2O/H_2O$ Trace Gas Analyzer
2. One LI-8250 Multiplexer
3. Four LI-8200-104 Opaque Long-Term Gas Chambers

The $N_2O$ flux was measured every 30 minutes during the growing seasons of 2021, 2022, and 2023 in Area X.O, Ottawa, Ontario, Canada. This study incorporates four time series datasets. These datasets are categorized based on distinct farming practices, growing seasons, and fertilization conditions, with their names outlined in Table 6.1.

Figure 6.1 illustrates the dataset of BM2023C. The $N_2O$ flux was calculated using the following equation:

TABLE 6.1: Four Different Time Series Dataset Used in this Exploration

| | Farming Practice | Position | Amount | Plants | Year |
|---|---|---|---|---|---|
| BS2021C | Broadcast[1] | Side | 225kg/ha | No Crop | 2021 |
| BS2022C | Broadcast | Side | 225kg/ha | No Crop | 2022 |
| BM2023C | Broadcast | Middle | 150kg/ha | Crop | 2023 |
| BM2023G | Broadcast | Middle | 150kg/ha | No Crop | 2023 |

[1] In the context of fertilizing, broadcasting means spreading fertilizer uniformly across the surface of a field or garden, rather than applying it in specific rows or spots. This method allows for a wide, even application of nutrients, making it suitable for large-scale fields where a consistent nutrient supply across the entire area is desired.

$$F_{N_2O} = \frac{VP(1-W_0)}{RST}\frac{dN_2O}{dt} \tag{6.1}$$

and the variables were defined as follows:

- $F_{N_2O}$: soil $N_2O$ flux ($nmol\ m^{-2}\ s^{-1}$)
- $V$: chamber volume ($m^3$)
- $P$: atmospheric pressure ($Pa$)
- $R$: gas constant ($Pa\ m^3\ mol^{-1}\ k^{-1}$)
- $S$: soil area ($m^2$)
- $T$: temperature ($Kelvin$)
- $\frac{dN_2O}{dt}$: rate of $N_2O$ change ($nmolm^{-2}s^{-1}$ ) during the chamber closure

### 6.2.2 Data Analysis

In the following analysis, there are three datasets being used, namely, BS2021C, BS2022C, and BM2023C.

#### Distribution of Time Series Data

Figure 6.2 illustrates significant quantitative differences in the distribution of time series data from the 2021, 2022, and 2023 growing seasons. However, fundamentally, the distribution shapes are similar to each other, with most peak values occurring around the same point. It is important to note that for the 2022 growing season, the

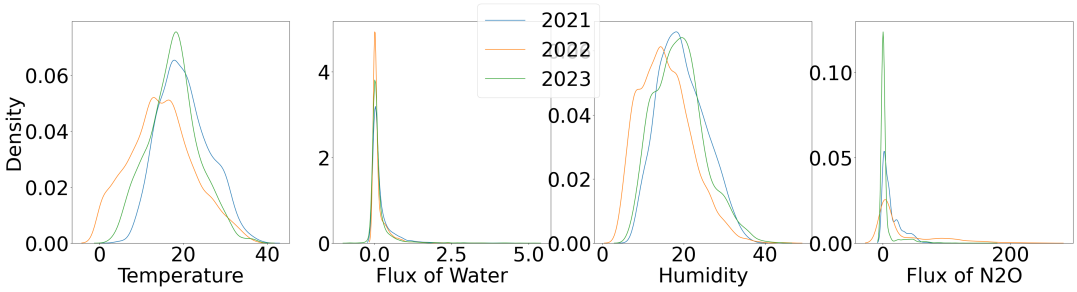



FIGURE 6.2: Distribution Analysis of $N_2O$, Temperature, Humidity, and Water Flux for the 2021, 2022, and 2023 Growing Seasons with Varied Farming Practices

temperature is lower than the other two growing seasons because this season started one month later than the 2021 growing season and half a month later than the 2023 growing season. In the 2023 growing season, the $N_2O$ flux is generally smaller than in the other growing seasons, likely due to the application of less fertilizer in the field. These factors pose a challenge for analyzing the time series data with a single model.

### Autocorrelation Function (ACF)

Autocorrelation, as defined in statistical literature and signal processing, captures the correlation between a signal and its delayed version across varying time intervals [255]. Specifically in statistics, autocorrelation involves assessing the Pearson correlation between values at different time points within a real or complex random process. The formulas of calculating the ACF is shown in Equation 6.2.

$$\hat{\gamma} = \frac{\sum_{t=k+1}^{n-k}(x_{t-k} - \bar{x})(x_t - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2} \tag{6.2}$$

where $\hat{\gamma}$ denotes the autocorrelation coefficient and $\bar{x}$ is the mean of the random variable x. The summation runs from $k+1$ to $n-k$, reflecting the lag $k$ between time points $t_{t-k}$ and $t_t$. The numerator calculates the cross-product of deviations from the mean, while the denominator computes the sum of squared deviations, providing a standardized measure of autocorrelation.

In Figure 6.3, the ACFs of $N_2O$, temperature, humidity, and water flux are presented over the 2021, 2022, and 2023 growing seasons. Evidently, the time series for temperature, humidity, and water flux display consistent periodicity across all three growing seasons. However, slight variations exist among the 2021, 2022, and 2023 growing seasons due to factors such as wind speed, precipitation, and human activity affecting the time series data. Consequently, when considering that all the

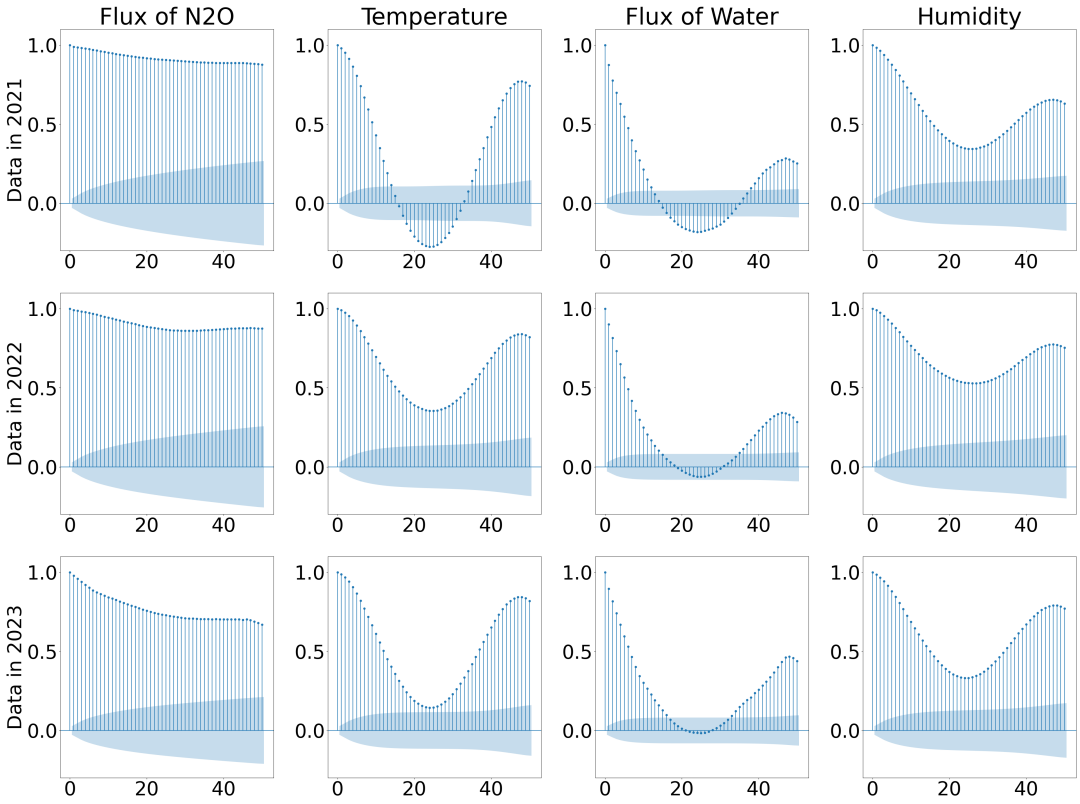



FIGURE 6.3: Autocorrelation Analysis of $N_2O$ Flux, Temperature, Humidity, and Water Flux Across Diverse Growing Seasons and Farming Practices

time series originate from the same dynamical system with slightly different initial and boundary conditions, analyzing these multidimensional time series data with a single model presents a significant challenge.

### Joint Distribution in First-Order Differences

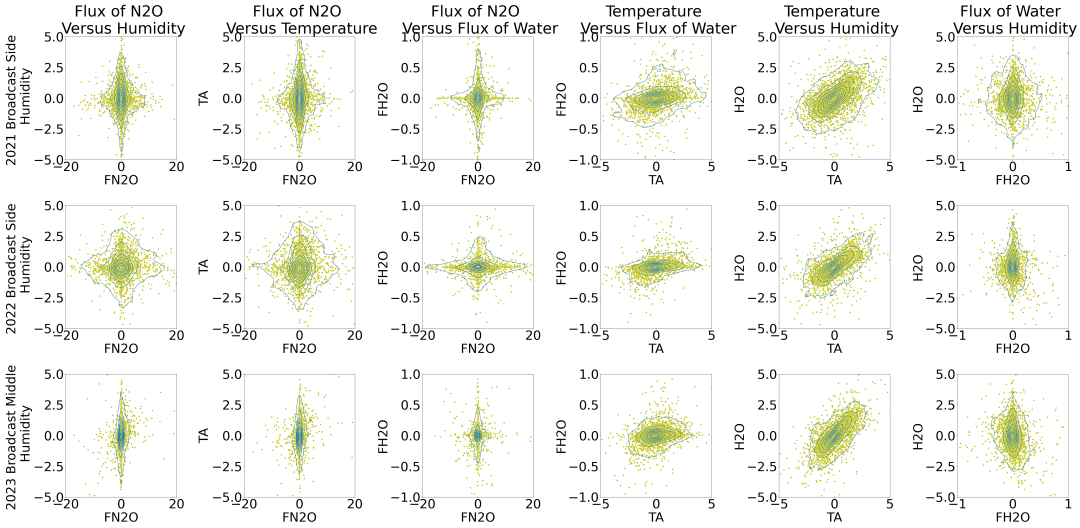



FIGURE 6.4: Joint Distributions Analysis in First-Order Differences Across Various Growing Seasons and Farming Practices

Figure 6.2 illustrates the joint distribution in the first-order difference of time series data across the 2021, 2022, and 2023 growing seasons with different farming practices. In [256], the temporal variability of $N_2O$ emissions from fertilized agricultural soil is discussed. It is observed that the emission of $N_2O$ changes depending on the day of the year when the fertilizer is applied, even if the amount of fertilizer applied remains the same. Consequently, the joint distributions differ for the three growing seasons, although the general shape is nearly identical. This indicates that the dynamical behavior of $N_2O$ emission varies when the initial conditions are slightly altered.

### Proposed Solution for Bifurcation Issue

Based on the data analysis of the data distribution, ACF, distribution, and joint distribution in first-order difference, it becomes evident that the dynamical system of $N_2O$ emission can undergo bifurcation branches due to different initial conditions and a variety of parameters. Accommodating all these bifurcation branches in a single neural network model poses a significant challenge. In this exploration, we address this challenge by incorporating time series data with signatures, enabling the distinction of data from different bifurcation branches.

### 6.2.3 Data Pre-Processing

As mentioned above, the collected raw data have different units and different scales. It will cause difficulties for us to model the emission of $N_2O$. It is necessary for us to scale the feature of input data. In this study, the data are scaled in the range [a, b] (a=0.05, b=0.95 in our case) using the customized normalization approach with specific minimum values and maximum values, as shown in the following [257]:

$$x' = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)} \tag{6.3}$$

where $x'$ is the normalized values and x is the original value, $a$ and $b$ are the max and min values after scaling, respectively. The normalized data could be represented by a vector at one time step, as shown in the following:

$$[N_t, T_t, W_t, H_t]$$

where $N_t$, $T_t$, $W_t$, and $H_t$ are the amounts of available N, temperature, water flux, and humidity, respectively, at time step $t$. Since the study did not actually measure the concentration of available N in the soil, the fertilizer was assumed to be linearly decreasing in our experiment. In order to predict the emission of the $N_2O$ at time step $t$, information from the previous $n$ time steps is needed, so the input could be characterized as a data matrix, as shown below:

$$\begin{bmatrix} X_{t-n} \\ X_{t-n+1} \\ X_{t-n+2} \\ \vdots \\ X_{t-1} \end{bmatrix} = \begin{bmatrix} N_{t-n} & T_{t-n} & W_{t-n} & H_{t-n} \\ N_{t-n+1} & T_{t-n+1} & W_{t-n+1} & H_{t-n+1} \\ N_{t-n+2} & T_{t-n+2} & W_{t-n+2} & H_{t-n+2} \\ \cdots & & & \\ N_{t-1} & T_{t-1} & W_{t-1} & H_{t-1} \end{bmatrix} \tag{6.4}$$

where $X_{t-i}$ is a 4-tuple that includes the $N_{t-i}$, $T_{t-i}$, $W_{t-i}$, and $H_{t-i}$.

Then to distinguish different conditions for a variety of time series, we add the signature bits to the data blocks:

$$\begin{bmatrix} X_{t-n} \\ X_{t-n+1} \\ X_{t-n+2} \\ \vdots \\ X_{t-1} \end{bmatrix} = \begin{bmatrix} S_1 & S_2 & N_{t-n} & T_{t-n} & W_{t-n} & H_{t-n} \\ S_1 & S_2 & N_{t-n+1} & T_{t-n+1} & W_{t-n+1} & H_{t-n+1} \\ S_1 & S_2 & N_{t-n+2} & T_{t-n+2} & W_{t-n+2} & H_{t-n+2} \\ \cdots & & & & & \\ S_1 & S_2 & N_{t-1} & T_{t-1} & W_{t-1} & H_{t-1} \end{bmatrix} \tag{6.5}$$

where $S_1$ and $S_2$ are signature bits.

## 6.3 Detailed Description of the Quantum LSTM-DLEM Model

### 6.3.1 Variational Quantum Circuits

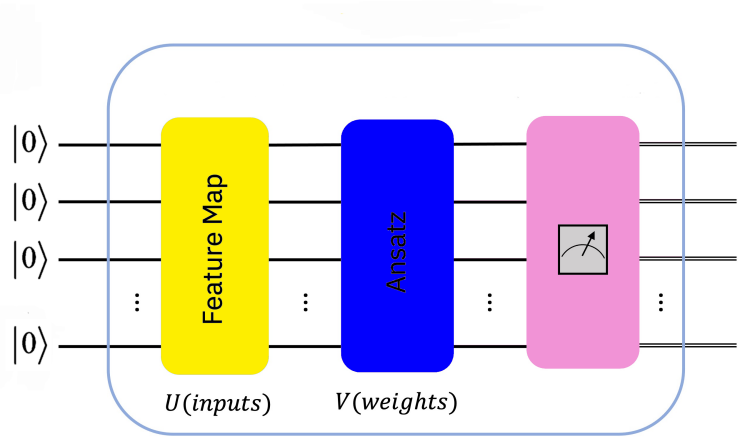



FIGURE 6.5: Generic Variational Quantum Circuits

In a Variational Quantum Circuit (VQC), illustrated in Figure 6.5, three major components constitute its structure: data encoding, variational layer, and quantum measurements. In the data encoding part, the $U(\text{input})$ block is dedicated to state preparation, encoding classical data into the quantum state of the circuit. In the variational layer component, the $V(\text{weight})$ block embodies the variational segment with learnable parameters, optimized through gradient methods. The final component involves measuring a subset (or all) of the qubits to retrieve a classical bit string.

Research has demonstrated that VQCs exhibit greater expressiveness compared to classical neural networks [258, 259]. They have shown promise in solving problems in quantum chemistry, optimization, and machine learning with an accuracy that surpasses classical methods [260, 261]. This heightened expressiveness implies an enhanced capacity to represent specific functions or distributions with a constrained number of parameters.

### 6.3.2 Quantum Long Short-Term Memory

In this paper, we elevate the classical LSTM architecture to the quantum realm by replacing the traditional neural network nodes in the LSTM cells with VQCs. These

VQCs serve a dual role, functioning as both feature extractors and data compressors. Refer to the schematic of our proposed QLSTM architecture in Figure 6.6. The VQC components utilized in QLSTM are elaborated in Figure 6.5. The $\sigma$ and $tanh$ blocks represent the sigmoid and hyperbolic tangent activation functions, respectively. Here, $x_t$ denotes the input at time $t$, $h_t$ corresponds to the hidden state, $c_t$ represents the cell state, and $y_t$ signifies the output. The symbols $\otimes$ and $\oplus$ denote element-wise multiplication and addition, respectively. For a deeper understanding of the QLSTM setup, refer to Equation 6.6, which provides the mathematical foundation for QLSTM.

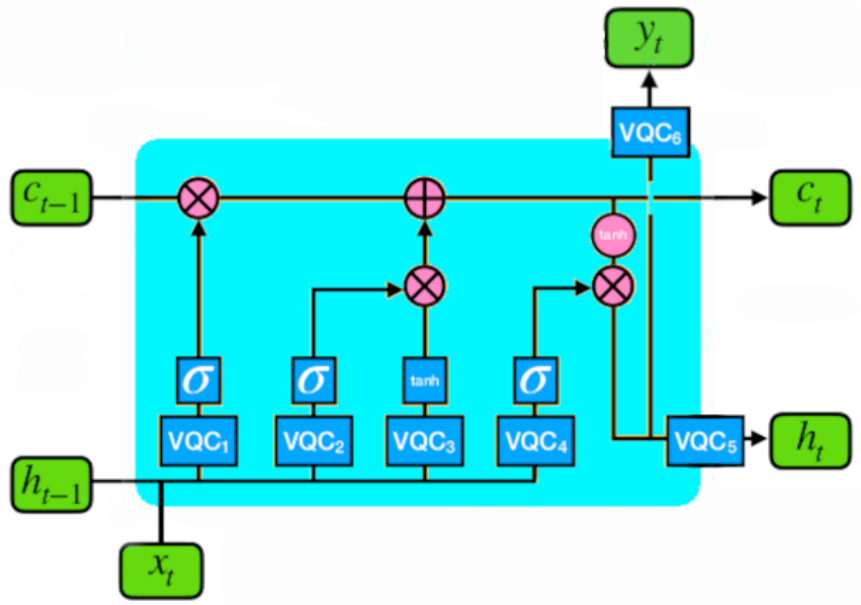



FIGURE 6.6: The Proposed QLSTM Architecture

$$
\begin{aligned}
f_t &= \sigma(VQC_1(v_t)) \\
i_t &= \sigma(VQC_2(v_t)) \\
\tilde{C}_t &= tanh(VQC_3(v_t)) \\
c_t &= f_t \bigotimes c_{t-1} \bigoplus i_t \bigotimes \tilde{C}_t \\
o_t &= \sigma(VQC_4(v_t)) \\
h_t &= VQC_5(o_t \bigotimes tanh(c_t)) \\
y_t &= VQC_6(o_t \bigotimes tanh(c_t))
\end{aligned}
\tag{6.6}
$$

### 6.3.3 Architecture of QLSTM-DLEM Model

As shown in Figure 6.7, the entire architecture of QLSTM-DLEM model is majorly consisted of two parts: the QLSTM part and the DLEM part. The original data is

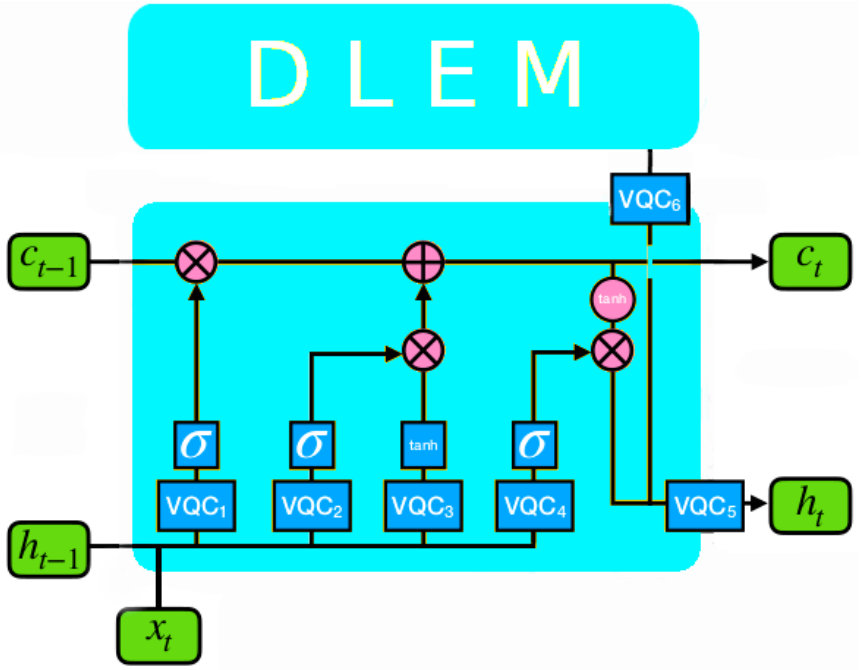



FIGURE 6.7: Architecture of the QLSTM-DLEM Model

passed through the QLSTM to capture the temporal sequential feature existed in the original data and then the processed data is fed into the DLEM model to predict the emission of $N_2O$.

For the DLEM components, it is a mechanistic model obtained from the agriculture field of study, and it needs 8 parameters, including soil temperature ($T_{soil}$), water-filled porosity ($wfp$), daily maximum fraction of $NH_4^+$ that is converted into $NO_3^-$ and nitrogen gases ($k_{nit}$), potential rate of denitrification ($N_{pot_{denit}}$), percentage of clay content in soil ($P_{clay}$), soil respiration rate ($R_h$), parameter depending on plant functional type $k_{den}$, $NO_3^-$ content in the soil per unit area ($av_{NO_3}$), and soil bulk density ($BD_{soil}$). All these parameters will be tuned by the factors passed from the FCL component, to predict the emission of $N_2O$.

In the QLSTM component, we utilize the parameter-shift method [262] to calculate the analytical gradient of the quantum circuits. To illustrate, consider the scenario where we have the expected value of an observable $\hat{B}$.

$$f(x;\theta_i) = \langle 0|U_0^+(x)U_i^+(\theta_i)\hat{B}U_i(\theta_i)U_0(x)|0\rangle = \langle x|U_i^+(\theta_i)\hat{B}U_i(\theta_i)|x\rangle \quad (6.7)$$

where $x$ represents the input value, $U_0(x)$ as the state preparation routine encoding $x$ into the quantum state, $i$ as the circuit parameter index for which the gradient is sought, and $U_i(\theta_i)$ as the single-qubit rotation generated by the Pauli operators, it has

been demonstrated that the gradient of $f$ with respect to the parameter $\theta_i$ is expressed as [261]:

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}[f(x;\theta_i + \frac{\pi}{2} - f(x;\theta_i - \frac{\pi}{2}] \tag{6.8}$$

This enables us to analytically compute the gradients of the expectation values and apply gradient descent optimization, a technique commonly used in classical machine learning, to VQC-based machine learning models.

## 6.4 Simulation and Analysis

In this study, both the classical LSTM-DLEM and QLSTM-DLEM models are implemented using PyTorch. To ensure comparability, all models adopt a batch training strategy with the mean square error (MSE) serving as the loss function. Throughout the training process, the Adam optimizer [112] is employed to achieve optimal convergence performance. To mitigate stochastic bias in evaluating the performance of a specific model architecture, each model is run multiple times.

In general, three types of experiments are designed to illustrate the advantages and disadvantages of the QLSTM-DLEM and the LSTM-DLEM:

1. Training on BS2021C, Testing on BS2022C for the QLSTM-DLEM and the LSTM-DLEM.

2. Training on BM2023C, Testing on BM2023 for the QLSTM-DLEM and the LSTM-DLEM.

3. Training on BS2021C and BM2023C with Signatures, Testing on BS022C and BM2023G with Signatures for the QLSTM-DLEM and the LSTM-DLEM.

### 6.4.1 Experiments One and Two: Training Different Bifurcation Branches in Distinct Models

As depicted in Figure 6.8, the following are the loss curves corresponding to Experiments One and Two. In the left chart of Figure 6.8, the training and testing loss for the BS branch is displayed. It is evident from this chart that, on the training dataset, both the QLSTM-DLEM and the LSTM-DLEM converge to nearly the same points, with the LSTM-DLEM slightly outperforming the QLSTM-DLEM. However, on the testing dataset, the QLSTM-DLEM achieves a superior loss value compared to the LSTM-DLEM. This observation suggests that the QLSTM-DLEM exhibits better

generalization capability than the LSTM-DLEM. Moving to the right chart of Figure 6.8, the training and testing loss for the BM branch is presented. On the training dataset, the LSTM-DLEM performs better than the QLSTM-DLEM. However, on the testing dataset, both models exhibit nearly the same loss value. This still indicates that the QLSTM-DLEM demonstrates better generalization performance than the LSTM-DLEM.

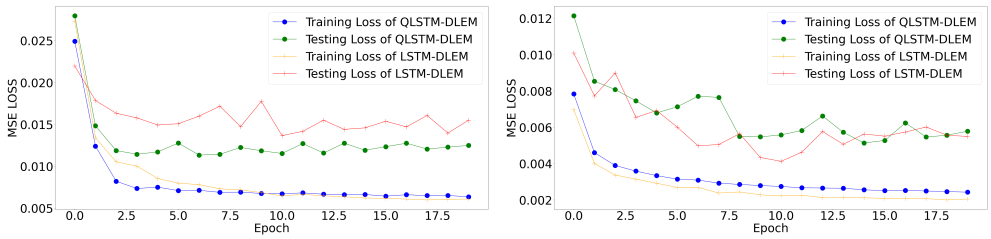



FIGURE 6.8: Loss Values for Two Distinct Bifurcation Branches

In Table 6.2, the performance of the BS branch for the QLSTM-DLEM and the LSTM-DLEM models is presented. While the LSTM-DLEM outperformed the QLSTM-DLEM in all metrics on the training dataset, it is worth noting that the QLSTM-DLEM exhibits smaller standard deviation values across all metrics. This suggests that the QLSTM-DLEM is more stable than the LSTM-DLEM during the training phase. Similar observations hold true for the testing dataset.

TABLE 6.2: Performance Comparison Between the QLSTM-DLEM and the LSTM-DLEM

| | RMSE | MAE | MAPE | $R^2$ | Mean[3] | SUM[4] |
|---|---|---|---|---|---|---|
| 1a | 8.46±0.27 | 5.38±0.24 | 187±59 | 0.71±0.02 | -0.49±0.63 | 60315±2499 |
| 1b | 32.13±1.30 | 20.44±0.58 | 3.13±0.26 | 0.61±0.03 | 3.94±2.73 | 143090±10882 |
| 2a | 8.18±1.08 | 5.16±0.77 | 165±36 | 0.72±0.06 | 0.25±1.24 | 57380±4934 |
| 2b | 33.67±2.52 | 20.56±0.55 | 2.99±0.39 | 0.57±0.06 | 7.08±4.20 | 130553±16777 |

[1a] Training Performance of the QLSTM-DLEM on BS2021C: the sum value of $N_2O$ emission in this dataset is 58368.
[1b] Testing Performance of the QLSTM-DLEM on BS2022C: the sum value of $N_2O$ emission in this dataset is 158797.
[2a] Training Performance of the LSTM-DLEM on BS2021C
[2b] Testing Performance of the LSTM-DLEM on BS2022C
[3] This is the mean of the errors between the expected value and actual value.
[4] This is the sum of the emission of $N_2O$ in the growing seasons.

In Table 6.3, the performance of the BM branch for the QLSTM-DLEM and the LSTM-DLEM models is outlined. Unlike the BS branch, in this scenario, the QLSTM-DLEM exhibits slightly better performance on both the training and testing datasets. Moreover, considering the standard deviation values across all metrics, the QLSTM-DLEM consistently demonstrates smaller values. Hence, it is convenient to

conclude that the QLSTM-DLEM is not only more stable but also more generalized compared to the LSTM-DLEM model in this context.

TABLE 6.3: Performance Comparison Between the QLSTM-DLEM and the LSTM-DLEM

| | RMSE | MAE | MAPE | $R^2$ | Mean[3] | SUM[4] |
|---|---|---|---|---|---|---|
| 1a | 8.60±0.75 | 5.77±0.34 | 16.57±2.82 | 0.68±0.05 | 0.55±0.65 | 22786±2579 |
| 1b | 10.33±1.04 | 6.10±0.57 | 93.41±21.60 | 0.62±0.08 | 4.86±0.72 | 13648±2890 |
| 2a | 8.66±0.69 | 6.14±0.70 | 18.96±3.84 | 0.67±0.05 | 0.74±1.17 | 22016±4669 |
| 2b | 10.64±0.90 | 6.58±0.35 | 113.6±33.53 | 0.60±0.07 | 5.33±1.36 | 11778±5433 |

[1a] Training Performance of the QLSTM-DLEM on BM2023C: the sum value of $N_2O$ emission in this dataset is 24964.
[1b] Testing Performance of the QLSTM-DLEM on BM2023G: the sum value of $N_2O$ emission in this dataset is 33059.
[2a] Training Performance of the LSTM-DLEM on BM2023C
[2b] Testing Performance of the LSTM-DLEM on BM2023G
[3] This is the mean of the errors between the expected value and actual value.
[4] This is the sum of the emission of $N_2O$ in the specific growing seasons.

### 6.4.2 Experiment Three: Training Both Bifurcation Branches in a Unified Model

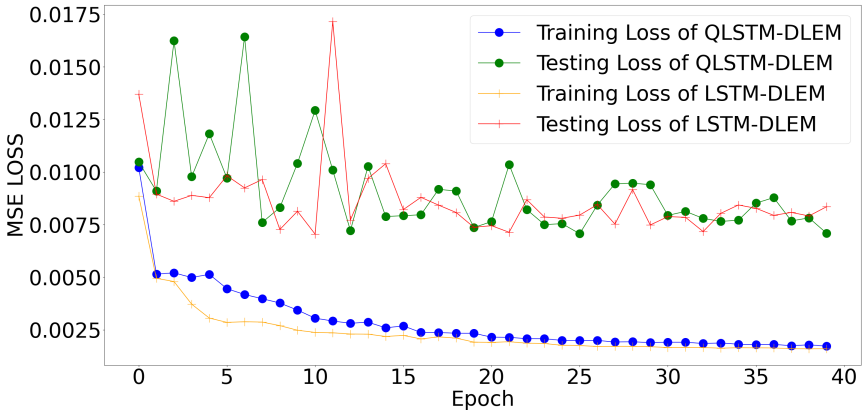



FIGURE 6.9: Loss Value of the QLSTM-DLEM and the LSTM-DLEM

The loss curves for the QLSTM-DLEM and the LSTM-DLEM on both the training and testing datasets are illustrated in Figure 6.9. In this case, the Adam optimizer is initialized with a learning rate of $1e-3$, decayed by $\gamma = 0.95$ every epoch. Both models undergo 40 fixed training epochs. Given the significant changes in the dynamical behavior of $N_2O$ emission with slightly different initial values and boundary conditions, and to showcase the learning capabilities of the QLSTM-DLEM and the LSTM-DLEM, the early stopping technique is not employed.

Observing the convergence of both the training and test losses, it is evident that they reach nearly the same points. However, upon quantitative analysis, it is discovered that the QLSTM-DLEM generally exhibits more stability than the LSTM-DLEM. This will be demonstrated in Table 6.4.

Since the same model is trained on two datasets with different initial conditions, direct determination of the overall performance is challenging. As presented in Table 6.4, for datasets BS2021C and BS2022C, the QLSTM-DLEM slightly outperformed the LSTM-DLEM model in terms of RMSE, MAE, MAPE, and $R^2$. Additionally, for the sum value of $N_2O$ emissions, both models produced values close to the actual values. However, the standard deviation values indicate that the QLSTM-DLEM is more stable than the LSTM-DLEM model under the same training conditions.

Conversely, as detailed in Table 6.4, for datasets BM2023C and BM2023G, the QLSTM-DLEM exhibited slightly worse performance than the LSTM-DLEM model in RMSE, MAE, MAPE, and $R^2$. Nonetheless, for the sum value of $N_2O$ emissions, both models generated values close to the actual values. Notably, the standard deviation values still emphasize that the QLSTM-DLEM is more stable than the LSTM-DLEM model under the same training conditions.

TABLE 6.4: Performance Comparison Between the QLSTM-DLEM and the LSTM-DLEM

| | RMSE | MAE | MAPE | $R^2$ | Mean[5] | SUM[6] |
|---|---|---|---|---|---|---|
| 1a | 5.12±0.06 | 3.24±0.10 | 168±101 | 0.89±0.00 | 0.28±0.54 | 57255±2145 |
| 1b | 29.79±1.42 | 18.33±1.65 | 2.93±0.36 | 0.67±0.03 | 3.59±1.29 | 144455±5171 |
| 2a | 5.44±0.40 | 3.45±0.26 | 205±36 | 0.87±0.02 | −0.29±1.17 | 59505±4678 |
| 2b | 30.31±0.88 | 18.04±0.67 | 2.81±0.17 | 0.65±0.02 | 2.88±3.16 | 147286±12619 |
| 3a | 6.07±0.43 | 2.89±0.58 | 3.75±2.98 | 0.84±0.02 | -0.01±0.51 | 24988±2027 |
| 3b | 9.64±0.38 | 4.71±0.46 | 39.07±22.18 | 0.67±0.03 | 4.55±0.50 | 14913±2007 |
| 4a | 5.91±0.36 | 2.68±0.20 | 2.71±0.41 | 0.85±0.02 | −0.20±1.08 | 25763±4315 |
| 4b | 9.85±1.43 | 4.73±0.85 | 25.03±15.33 | 0.65±0.10 | 4.58±0.94 | 14773±3735 |

[1a] Training Performance of Quantum LSTM-DLEM on BS2021C: the sum value of $N_2O$ emission in this dataset is 58368.
[1b] Testing Performance of Quantum LSTM-DLEM on BS2022C: the sum value of $N_2O$ emission in this dataset is 158797.
[2a] Training Performance of Classical LSTM-DLEM on BS2021C
[2b] Testing Performance of Classical LSTM-DLEM on BS2022C
[3a] Training Performance of Quantum LSTM-DLEM on BM2023C: the sum value of $N_2O$ emission in this dataset is 24964.
[3b] Testing Performance of Quantum LSTM-DLEM on BM2023G: the sum value of $N_2O$ emission in this dataset is 33059.
[4a] Training Performance of Classical LSTM-DLEM on BM2023C
[4b] Testing Performance of Classical LSTM-DLEM on BM2023G
[5] This is the mean of the errors between the expected value and actual value.
[6] This is the sum of the emission of $N_2O$ in the growing seasons.

As depicted in Figure 6.10, it is evident that for both training datasets, the predicted curves from the QLSTM-DLEM and the LSTM-DLEM effectively capture the trend of the actual curve. This observation is consistent when evaluating the testing dataset as well.

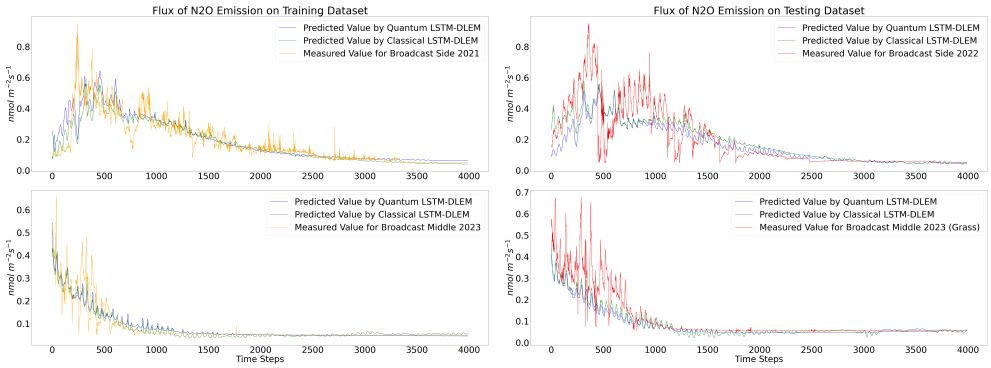



FIGURE 6.10: Training and Testing Performance of quantum and classical LSTM-DLEM

## 6.5 Conclusion Remarks

This study introduces the development of the QLSTM-DLEM model, a fusion of the QLSTM and the DLEM designed for predicting $N_2O$ emissions. Our objective was to create a quantum neural network model that could outperform its classical counterpart. Data collected during the growing seasons of 2021, 2022, and 2023 from different farming practices were utilized to train and test both the QLSTM-DLEM and the LSTM-DLEM. The models were assessed using four metrics: MAE, MAPE, RMSE, $R^2$, Error Mean, and Sum.

Data analysis revealed slight variations in the data distribution, ACF, and joint distribution in first-order difference for data from different branches, although they shared some similar characteristics. Integrating data from different branches into a single model posed a challenge, leading us to incorporate signatures into the data to distinguish time series from various bifurcation branches.

To scrutinize the QLSTM-DLEM model, three experiments were designed to evaluate its learning capability and expressive power. The simulation results align with our objective, demonstrating that the QLSTM-DLEM model effectively avoids overfitting and exhibits superior performance on both training and testing datasets, characterized by minimal generalized errors in terms of mean errors. Additionally, the QLSTM-DLEM is found to be more stable than the LSTM-DLEM, showcasing robustness to different initialized parameters.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

This study first discusses various approaches to time series analysis, including statistical models, mechanistic models, and recurrent neural network models, and weighs the pros and cons of each approach. Then an innovative type of hybrid model, which combines recurrent neural network models and mechanistic models as dynamical systems, is proposed to solve the practical issues, such as predicting the emission of $N_2O$.

Before applying the hybrid model to the practical utilization, a brief investigation on the mathematical mechanism of the hybrid models is discussed. It is observed that the recurrent neural network in the hybrid model could learn the dynamical characteristic of the time series data successfully and the mechanistic model in the hybrid model could adapt to different cases through changing the parameters. This feature makes the hybrid model a universal model that could be trained with a dataset with limited size and tested using a variety of datasets.

The AINN could take advantage of the benefits of both the recurrent neural network models and the mechanistic models. In this investigation, we proposed a hybrid model, which consists of the RNNs and DLEM to predict the emission of $N_2O$ effectively and accurately. The simulation results show that the AINN performs better than single pure models (RNN and DLEM).

At the same time, since it is challenging to collect sufficient time series data, the data augmentation approach for time series data is developed to supply enough data to train the hybrid model and reduce the generalized error. Since the time series is not identically independently distributed, it makes the traditional sampling method fail in this case. Therefore, inspiring from the deep learning model, the WGAN-GP is used to generated identically independently distributed time series vectors, and then the MCMC approach is adopted to sampling the data along the time axis while keeping the distribution of the generated data aligned with the original data. Finally, the

generated dataset is passed into the smoothing filter to simulate the time-dependent effect of the generated data points. The experiment results show that the augmented dataset could achieve the same performance on training the hybrid model.

Through the emergence of the quantum computing, the QLSTM is developed and combined with the DLEM model to form a new type of quantum model, which uses quantum bits to predict the emission of $N_2O$ from the agricultural field. The experiment results demonstrated that the QLSTM-DLEM has better generalization capability than the traditional LSTM-DLEM model and also is more robust to a variety of weight initialization.

## 7.2 Future Work

While the current study has focused on the hybrid model that combines RNN models with mechanistic models, there are several other promising directions for future research that warrant exploration.

1. Further exploration is needed to deepen the theoretical background of combining RNN models with mechanistic models.

2. The data augmentation approach used in this investigation needs to be polished further. There still exist several drawback on the current approach; for example, it is difficult to generate the peak value of the flux of $N_2O$, since it rarely occurs in one growing season.

3. The mechanistic model utilized in this investigation is just an empirical system. It is necessary for us to do more research to improve the accuracy of the mechanistic model and understand the essence of $N_2O$ emissions from farming.

4. Sparsity is an interesting property to explore when developing this type of model.

5. The discussion on the hybrid model's resilience against adversarial attacks is an intriguing topic.

# Bibliography


[1] Jeffrey L Elman. “Finding structure in time”. In: *Cognitive Science* 14.2 (1990), pp. 179–211.

[2] Mike Schuster and Kuldip K Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[3] L Brown et al. “Development and application of a mechanistic model to estimate emission of nitrous oxide from UK agriculture”. In: *Atmospheric Environment* 36.6 (2002), pp. 917–928.

[4] Ruth E Baker et al. “Mechanistic models versus machine learning, a fight worth fighting for the biological community?” In: *Biology Letters* 14.5 (2018), p. 20170660.

[5] Michael Brin and Garrett Stuck. *Introduction to dynamical systems*. Cambridge University Press, 2002.

[6] James Douglas Hamilton. *Time series analysis*. Princeton University Press, 2020.

[7] Amin Ghadami and Bogdan I Epureanu. “Data-driven prediction in dynamical systems: recent developments”. In: *Philosophical Transactions of the Royal Society A* 380.2229 (2022), p. 20210213.

[8] Steven L Brunton and J Nathan Kutz. *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2022.

[9] Luca Faes et al. “Detecting nonlinear causal interactions between dynamical systems by non-uniform embedding of multiple time series”. In: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. 2010, pp. 102–105. DOI: 10.1109/IEMBS.2010.5626127.

[10] Abdusselam Altunkaynak and Tewodros Assefa Nigussie. “Monthly water consumption prediction using season algorithm and wavelet transform–based models”. In: *Journal of Water Resources Planning and Management* 143.6 (2017), p. 04017011.

[11] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. "Evaluation of bidirectional LSTM for short-and long-term stock market prediction". In: *2018 9th International Conference on Information and Communication Systems (ICICS)*. IEEE. 2018, pp. 151–156.

[12] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. "Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU)". In: *2018 21st Saudi Computer Society National Computer Conference (NCC)*. IEEE. 2018, pp. 1–7.

[13] Zhiyong Cui et al. "Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values". In: *Transportation Research Part C: Emerging Technologies* 118 (2020), p. 102674.

[14] Deniz Kenan Kılıç and Ömür Uğur. "Multiresolution analysis of S&P500 time series". In: *Annals of Operations Research* 260.1 (2018), pp. 197–216.

[15] Peihao Li et al. "Autoregressive moving average modeling in the financial sector". In: *2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. IEEE. 2015, pp. 68–71.

[16] Guisheng Zhang, Xindong Zhang, and Hongyinping Feng. "Forecasting financial time series using a methodology based on autoregressive integrated moving average and Taylor expansion". In: *Expert Systems* 33.5 (2016), pp. 501–516.

[17] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 1394–1401.

[18] Ci Lin, Tet Hin Yeap, and Iluju Kiringa. "Stacked Bidirectional LSTM for Predicting Emission of Nitrous Oxide". In: *35th Canadian Conference on Artificial Intelligence, Toronto, Ontario, Canada, May 30 - June 3, 2022*. Ed. by Iluju Kiringa and Sébastien Gambs. Canadian Artificial Intelligence Association, 2022. DOI: 10.21428/594757db.daad1be1. URL: https://doi.org/10.21428/594757db.daad1be1.

[19] Tet Yeap. "Implementation of an associative memory using a restricted hopfield network". In: *Global Journal of Research In Engineering* (2021).

[20] Faezeh Halabian. "An Enhanced Learning for Restricted Hopfield Networks". MA thesis. Université d'Ottawa/University of Ottawa, 2021.

[21] Futong Li. "Global Optimization Techniques Based on Swarm-intelligent and Gradient-free Algorithms". MA thesis. Université d'Ottawa/University of Ottawa, 2021.

[22] Yong-gang Zhang et al. "A novel displacement prediction method using gated recurrent unit model with time series analysis in the Erdaohe landslide". In: *Natural Hazards* 105 (2021), pp. 783–813.

[23] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[24] Bethany A Bradley et al. "A curve fitting procedure to derive inter-annual phenologies from time series of noisy satellite NDVI data". In: *Remote Sensing of Environment* 106.2 (2007), pp. 137–145.

[25] Krishnan Bhaskaran et al. "Time series regression studies in environmental epidemiology". In: *International Journal of Epidemiology* 42.4 (2013), pp. 1187–1195.

[26] James EH Davidson et al. "Econometric modelling of the aggregate time-series relationship between consumers' expenditure and income in the United Kingdom". In: *The Economic Journal* 88.352 (1978), pp. 661–692.

[27] Jian Zhao et al. "Exploratory analysis of time-series with chronolenses". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2422–2431.

[28] Rob J Hyndman and Yeasmin Khandakar. "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 27 (2008), pp. 1–22.

[29] Paul W Rasmussen et al. "Time-series intervention analysis: unreplicated large-scale experiments". In: *Design and Analysis of Ecological Experiments*. Chapman and Hall/CRC, 2020, pp. 138–158.

[30] Miodrag Lovrić, Marina Milanović, and Milan Stamenković. "Algoritmic methods for segmentation of time series: An overview". In: *Journal of Contemporary Economic and Business Issues* 1.1 (2014), pp. 31–53.

[31] George Udny Yule. "VII. On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 226.636-646 (1927), pp. 267–298.

[32] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[33] Nick Heard et al. *An Introduction to Bayesian Inference, Methods and Computation*. Springer, 2021.

[34] Steven L Scott and Hal R Varian. "Predicting the present with Bayesian structural time series". In: *International Journal of Mathematical Modelling and Numerical Optimisation* 5.1-2 (2014), pp. 4–23.

[35] Kay H. Brodersen et al. "Inferring causal impact using Bayesian structural time-series models". In: *Annals of Applied Statistics* 9 (2015), pp. 247–274.

[36] William M Bolstad and James M Curran. *Introduction to Bayesian statistics*. John Wiley & Sons, 2016.

[37] Dirk P. Kroese et al. "Why the Monte Carlo method is so important today". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6 (2014).

[38] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[39] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[40] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[41] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM networks". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 4. IEEE. 2005, pp. 2047–2052.

[42] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. "Hybrid speech recognition with deep bidirectional LSTM". In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 273–278.

[43] Hanqin Tian et al. "Century-scale responses of ecosystem carbon storage and flux to multiple environmental changes in the southern United States". In: *Ecosystems* 15.4 (2012), pp. 674–694.

[44] Hanqin Tian et al. "The global N 2 O model intercomparison project". In: *Bulletin of the American Meteorological Society* 99.6 (2018), pp. 1231–1251.

[45] Rajendra K Pachauri et al. *Climate change 2014: synthesis report. Contribution of Working Groups I, II and III to the fifth assessment report of the Intergovernmental Panel on Climate Change*. Ipcc, 2014.

[46] Trevor D Rapson and Helen Dacres. "Analytical techniques for measuring nitrous oxide". In: *TrAC Trends in Analytical Chemistry* 54 (2014), pp. 65–74.

[47] David Fowler et al. "Effects of global change during the $21^{st}$ century on the nitrogen cycle". In: *Atmospheric Chemistry and Physics* 15.24 (2015), pp. 13849–13893.

[48] James N Galloway et al. "Transformation of the nitrogen cycle: recent trends, questions, and potential solutions". In: *Science* 320.5878 (2008), pp. 889–892.

[49] Nicolas Gruber and James N Galloway. "An Earth-system perspective of the global nitrogen cycle". In: *Nature* 451.7176 (2008), pp. 293–296.

[50] Cecelia Macfarling Meure et al. "Law Dome $CO_2$, $CH_4$ and $N_2O$ ice core records extended to 2000 years BP". In: *Geophysical Research Letters* 33.14 (2006).

[51] Michael J Prather, Christopher D Holmes, and Juno Hsu. "Reactive greenhouse gas scenarios: Systematic exploration of uncertainties and the role of atmospheric chemistry". In: *Geophysical Research Letters* 39.9 (2012).

[52] Rona Louise Thompson et al. "TransCom $N_2O$ model intercomparison–Part 2: Atmospheric inversion estimates of $N_2O$ emissions". In: *Atmospheric Chemistry and Physics* 14.12 (2014), pp. 6177–6194.

[53] E Saikawa, CA Schlosser, and RG Prinn. "Global modeling of soil nitrous oxide emissions from natural processes". In: *Global Biogeochemical Cycles* 27.3 (2013), pp. 972–989.

[54] Hanqin Tian et al. "Spatial and temporal patterns of CH 4 and N 2 O fluxes in terrestrial ecosystems of North America during 1979–2008: application of a global biogeochemistry model". In: *Biogeosciences* 7.9 (2010), pp. 2673–2694.

[55] Rongting Xu et al. "Preindustrial nitrous oxide emissions from the land biosphere estimated by using a global biogeochemistry model". In: *Climate of the Past* 13.7 (2017), pp. 977–990.

[56] Yuanyuan Huang and Stefen Gerber. "Global soil nitrous oxide emissions in a dynamic carbon-nitrogen model". In: *Biogeosciences* 12.21 (2015), pp. 6405–6427.

[57] IC Prentice. "Terrestrial nitrogen cycle simulation with a dynamic global vegetation model". In: *Global Change Biology* 14.8 (2008), pp. 1745–1764.

[58] Stefan Olin et al. "Soil carbon management in large-scale Earth system modelling: implications for crop yields and nitrogen leaching". In: *Earth System Dynamics* 6.2 (2015), pp. 745–768.

[59] Benjamin D Stocker et al. "Multiple greenhouse-gas feedbacks from the land biosphere under future climate change scenarios". In: *Nature Climate Change* 3.7 (2013), pp. 666–672.

[60] Sönke Zaehle and AD Friend. "Carbon and nitrogen cycle dynamics in the O-CN land surface model: 1. Model description, site-scale evaluation, and sensitivity to parameter estimates". In: *Global Biogeochemical Cycles* 24.1 (2010).

[61] Nicolas Vuichard et al. "Accounting for carbon and nitrogen interactions in the global terrestrial ecosystem model ORCHIDEE: multi-scale evaluation of gross primary production". In: *Geoscientific Model Development* 12.11 (2019), pp. 4751–4779.

[62] Daniel S Goll et al. "A representation of the phosphorus cycle for ORCHIDEE". In: *Geoscientific Model Development* 10.10 (2017), pp. 3745–3770.

[63] Qing Zhu et al. "Modelling methane emissions from natural wetlands by development and application of the TRIPLEX-GHG model". In: *Geoscientific Model Development* 7.3 (2014), pp. 981–999.

[64] Kerou Zhang et al. "Process-based TRIPLEX-GHG model for simulating $N_2O$ emissions from global forests and grasslands: Model development and evaluation". In: *Journal of Advances in Modeling Earth Systems* 9.5 (2017), pp. 2079–2102.

[65] Motoko Inatomi et al. "Greenhouse gas budget of a cool-temperate deciduous broad-leaved forest in Japan estimated using a process-based model". In: *Ecosystems* 13 (2010), pp. 472–483.

[66] A Ito and M Inatomi. "Use of a process-based model for assessing the methane budgets of global terrestrial ecosystems and evaluation of uncertainty". In: *Biogeosciences* 9.2 (2012), pp. 759–773.

[67] Hanqin Tian et al. "Contemporary and projected biogenic fluxes of methane and nitrous oxide in North American terrestrial ecosystems". In: *Frontiers in Ecology and the Environment* 10.10 (2012), pp. 528–536.

[68] Abderrachid Hamrani, Abdolhamid Akbarzadeh, and Chandra A Madramootoo. "Machine learning for predicting greenhouse gas emissions from agricultural soils". In: *Science of the Total Environment* 741 (2020), p. 140338.

[69] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and Computing* 14 (2004), pp. 199–222.

[70] Vrushali Y Kulkarni and Pradeep K Sinha. "Pruning of random forest classifiers: A survey and future directions". In: *2012 International Conference on Data Science & Engineering (ICDSE)*. IEEE. 2012, pp. 64–68.

[71] Robert Tibshirani. "Regression shrinkage and selection via the lasso: a retrospective". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 73.3 (2011), pp. 273–282.

[72] Andres Schmidt, Whitney Creason, and Beverly E Law. "Estimating regional effects of climate change and altered land use on biosphere carbon fluxes using distributed time delay neural networks with Bayesian regularized learning". In: *Neural Networks* 108 (2018), pp. 97–113.

[73] Hao Yu et al. "Advantages of radial basis function networks for dynamic system design". In: *IEEE Transactions on Industrial Electronics* 58.12 (2011), pp. 5438–5450.

[74] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507.

[75] Nigel W Daw. "The foundations of development and deprivation in the visual system". In: *The Journal of Physiology* 587.12 (2009), pp. 2769–2773.

[76] Naftali Tishby, Fernando C Pereira, and William Bialek. "The information bottleneck method". In: *arXiv Preprint physics/0004057* (2000).

[77] Konstantinos I Diamantaras and Sun Yuan Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., 1996.

[78] Erkki Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15 (1982), pp. 267–273.

[79] Erkki Oja and Juha Karhunen. "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix". In: *Journal of Mathematical Analysis and Applications* 106.1 (1985), pp. 69–84.

[80] Liang-Hwa Chen and Shyang Chang. "An adaptive learning algorithm for principal component analysis". In: *IEEE Transactions on Neural Networks* 6.5 (1995), pp. 1255–1263.

[81] Terence D Sanger. "Optimal unsupervised learning in a single-layer linear feedforward neural network". In: *Neural networks* 2.6 (1989), pp. 459–473.

[82] Hong Chen and Ruey-Wen Lin. “An on-line unsupervised learning machine for adaptive feature extraction”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 41.2 (1994), pp. 87–98.

[83] Ferdinand Peper and Hideki Noda. “A symmetric linear neural network that learns principal components and their variances”. In: *IEEE Transactions on Neural Networks* 7.4 (1996), pp. 1042–1047.

[84] Ke-Lin Du and Madisetti NS Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013, pp. 203–204.

[85] Jeanette Rubner and Klaus Schulten. “Development of feature detectors by self-organization: A network model”. In: *Biological Cybernetics* 62.3 (1990), pp. 193–199.

[86] Sun-Yuan Kung and Kostas I Diamantaras. “A neural network learning algorithm for adaptive principal component extraction (APEX)”. In: *International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 1990, pp. 861–864.

[87] Lei Xu. “Least mean square error reconstruction principle for self-organizing neural-nets”. In: *Neural Networks* 6.5 (1993), pp. 627–648.

[88] Bin Yang. “Projection approximation subspace tracking”. In: *IEEE Transactions on Signal processing* 43.1 (1995), pp. 95–107.

[89] Shan Ouyang, Zheng Bao, and Gui-Sheng Liao. “Robust recursive least squares learning algorithm for principal component analysis”. In: *IEEE Transactions on Neural Networks* 11.1 (2000), pp. 215–221.

[90] Yongfeng Miao and Yingbo Hua. “Fast subspace tracking and neural network learning by a novel information criterion”. In: *IEEE Transactions on Signal Processing* 46.7 (1998), pp. 1967–1979.

[91] Ralf Moller and Axel Konies. “Coupled principal component analysis”. In: *IEEE Transactions on Neural Networks* 15.1 (2004), pp. 214–222.

[92] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. In: *Neural Computation* 10.5 (1998), pp. 1299–1319.

[93] B Sch6lkopf et al. “Input space vs. feature space in kernel-based methods”. In: *IEEE-NN, 10 (5): 1000* 1017 (1999).

[94] Lei Xu and Alan L Yuille. “Robust principal component analysis by self-organizing rules based on statistical physics approach”. In: *IEEE Transactions on Neural Networks* 6.1 (1995), pp. 131–143.

[95] Hervé Bourlard and Yves Kamp. "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological Cybernetics* 59.4-5 (1988), pp. 291–294.

[96] Isabelle Guyon et al. *Feature extraction: foundations and applications*. Vol. 207. Springer, 2008.

[97] Ryotaro Kamimura. "Conditional information and information loss for flexible feature extraction". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 2074–2083.

[98] Pedro Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[99] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828.

[100] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), pp. 433–459.

[101] Xiangyu Kong, Changhua Hu, and Zhansheng Duan. *Principal component analysis networks and algorithms*. Springer, 2017.

[102] Simon Du et al. "Gradient descent finds global minima of deep neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1675–1685.

[103] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv Preprint arXiv:1609.04747* (2016).

[104] David Saad. "Online algorithms and stochastic approximations". In: *Online Learning* 5.3 (1998), p. 6.

[105] Jason Brownlee. *Better deep learning: train faster, reduce overfitting, and make better predictions*. Machine Learning Mastery, 2018.

[106] Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. "Mini-batch gradient descent: Faster convergence under data sparsity". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 2880–2887.

[107] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1 (1999), pp. 145–151.

[108] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence o($1/k^2$)". In: 1983. URL: https://api.semanticscholar.org/CorpusID:202149403.

[109] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of Machine Learning Research* 12.7 (2011).

[110] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv Preprint arXiv:1212.5701* (2012).

[111] Fangyu Zou et al. "A sufficient condition for convergences of Adam and RMSprop". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11127–11135.

[112] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv Preprint arXiv:1412.6980* (2014).

[113] M Lavanya Dr R Parameswari. "Adamax Crop Prediction Neural Network Classification Model Using Iot Data For Green House". In: *Solid State Technology* 63.6 (2020), pp. 3528–3543.

[114] Qikun Zhang et al. "Boosting adversarial attacks with nadam optimizer". In: *Electronics* 12.6 (2023), p. 1464.

[115] Bernardetta Addis, Marco Locatelli, and Fabio Schoen. "Local optima smoothing for global optimization". In: *Optimization Methods and Software* 20.4-5 (2005), pp. 417–437.

[116] Boris Hanin. "Which neural net architectures give rise to exploding and vanishing gradients?" In: *Advances in Neural Information Processing Systems* 31 (2018).

[117] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. "An Efficient Approach for Assessing Hyperparameter Importance". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 2014, pp. 754–762. URL: https://proceedings.mlr.press/v32/hutter14.html.

[118] Terry T Um et al. "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks". In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. 2017, pp. 216–220.

[119] Chris M Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural Computation* 7.1 (1995), pp. 108–116.

[120] Guozhong An. "The effects of adding noise during backpropagation training on a generalization performance". In: *Neural Computation* 8.3 (1996), pp. 643–674.

[121] Khandakar M Rashid and Joseph Louis. "Window-warping: a time series data augmentation of IMU data for construction equipment activity identification". In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 36. IAARC Publications. 2019, pp. 651–657.

[122] Jordan Yeomans et al. "Simulating time-series data for improved deep neural network performance". In: *IEEE Access* 7 (2019), pp. 131248–131255.

[123] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[124] François Petitjean, Alain Ketterlin, and Pierre Gançarski. "A global averaging method for dynamic time warping, with applications to clustering". In: *Pattern Recognition* 44.3 (2011), pp. 678–693.

[125] Brian Kenji Iwana and Seiichi Uchida. "Time series data augmentation for neural networks by time warping with a discriminative teacher". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 3558–3565.

[126] Naoya Takahashi, Michael Gygli, and Luc Van Gool. "Aenet: Learning deep audio features for video analysis". In: *IEEE Transactions on Multimedia* 20.3 (2017), pp. 513–524.

[127] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. "Data augmentation for deep neural network acoustic modeling". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.9 (2015), pp. 1469–1477.

[128] Christoph Bergmeir, Rob J Hyndman, and José M Benítez. "Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation". In: *International Journal of Forecasting* 32.2 (2016), pp. 303–312.

[129] Pierre Comon. "Independent component analysis, a new concept?" In: *Signal Processing* 36.3 (1994), pp. 287–314.

[130] Norden E Huang et al. "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1971 (1998), pp. 903–995.

[131] Hong Cao, Vincent YF Tan, and John ZF Pang. "A parsimonious mixture of Gaussian trees model for oversampling in imbalanced and multimodal time-series classification". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.12 (2014), pp. 2226–2239.

[132] Sylvia Frühwirth-Schnatter. "Data augmentation and dynamic linear models". In: *Journal of Time Series Analysis* 15.2 (1994), pp. 183–202.

[133] Xiao-Li Meng and David A Van Dyk. "Seeking efficient data augmentation schemes via conditional and marginal augmentation". In: *Biometrika* 86.2 (1999), pp. 301–320.

[134] Yutai Hou et al. "Sequence-to-sequence data augmentation for dialogue language understanding". In: *arXiv Preprint arXiv:1807.01554* (2018).

[135] Slawek Smyl and Karthik Kuber. "Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks". In: *36th International Symposium on Forecasting*. 2016.

[136] Yanfei Kang, Rob J Hyndman, and Feng Li. "GRATIS: GeneRAting TIme Series with diverse and controllable characteristics". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 13.4 (2020), pp. 354–376.

[137] Aswathy Madhu and Suresh Kumaraswamy. "Data augmentation using generative adversarial network for environmental sound classification". In: *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE. 2019, pp. 1–5.

[138] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. "Time-series generative adversarial networks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[139] Jisung Wang, Sangki Kim, and Yeha Lee. "Speech augmentation using wavenet in speech recognition". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6770–6774.

[140] Juanhui Tu et al. “Spatial-temporal data augmentation based on LSTM autoencoder network for skeleton-based human action recognition”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 3478–3482.

[141] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. “Data augmentation using random image cropping and patching for deep CNNs”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.9 (2019), pp. 2917–2931.

[142] Emmanuel Okafor, Lambert Schomaker, and Marco A Wiering. “An analysis of rotation matrix and colour constancy data augmentation in classifying images of animals”. In: *Journal of Information and Telecommunication* 2.4 (2018), pp. 465–491.

[143] Torbjorn Eltoft. “Data augmentation using a combination of independent component analysis and non-linear time-series prediction”. In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No. 02CH37290)*. Vol. 1. IEEE. 2002, pp. 448–453.

[144] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems* 27 (2014).

[145] Yize Chen et al. “Model-free renewable scenario generation using generative adversarial networks”. In: *IEEE Transactions on Power Systems* 33.3 (2018), pp. 3265–3275.

[146] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. “Sensegen: A deep learning architecture for synthetic sensor data generation”. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE. 2017, pp. 188–193.

[147] Luca Simonetto. “Generating spiking time series with generative adversarial networks: an application on banking transactions”. In: *MS thesis-University of Amsterdam* (2018).

[148] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. “Biosignal data augmentation based on generative adversarial networks”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2018, pp. 368–371.

[149] Chi Zhang et al. “Generative adversarial network for synthetic time series data generation in smart grids”. In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE. 2018, pp. 1–6.

[150] Yizhe Zhang, Zhe Gan, and Lawrence Carin. “Generating text via adversarial training”. In: *NIPS workshop on Adversarial Training*. Vol. 21. academia. edu. 2016, pp. 21–32.

[151] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 214–223.

[152] Ishaan Gulrajani et al. “Improved training of wasserstein GANs”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[153] Olof Mogren. “C-RNN-GAN: Continuous recurrent neural networks with adversarial training”. In: *arXiv Preprint arXiv:1611.09904* (2016).

[154] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. “Real-valued (medical) time series generation with recurrent conditional gans”. In: *arXiv Preprint arXiv:1706.02633* (2017).

[155] Chris Donahue, Julian McAuley, and Miller Puckette. “Adversarial audio synthesis”. In: *arXiv Preprint arXiv:1802.04208* (2018).

[156] Jinsung Jeon et al. “GT-GAN: General Purpose Time Series Synthesis with Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 36999–37010.

[157] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv Preprint arXiv:1411.1784* (2014).

[158] Mohammad H Amin et al. “Quantum boltzmann machine”. In: *Physical Review X* 8.2 (2018), p. 021050.

[159] Kurowski Krzysztof et al. “Applying a quantum annealing based restricted Boltzmann machine for MNIST handwritten digit classification”. In: *CMST* 27.3 (2021), pp. 99–107.

[160] Lorenzo Moro and Enrico Prati. “Anomaly detection speed-up by quantum restricted Boltzmann machines”. In: *Communications Physics* 6.1 (2023), p. 269.

[161] Gong Chen et al. “Learning music emotions via quantum convolutional neural network”. In: *Brain Informatics: International Conference, BI 2017, Beijing, China, November 16-18, 2017, Proceedings*. Springer. 2017, pp. 49–58.

[162] Tak Hur, Leeseok Kim, and Daniel K Park. “Quantum convolutional neural network for classical data classification”. In: *Quantum Machine Intelligence* 4.1 (2022), p. 3.

[163] Samuel Yen-Chi Chen et al. “Quantum convolutional neural networks for high energy physics data analysis”. In: *Physical Review Research* 4.1 (2022), p. 013231.

[164] Seth Lloyd and Christian Weedbrook. “Quantum generative adversarial learning”. In: *Physical Review Letters* 121.4 (2018), p. 040502.

[165] Pierre-Luc Dallaire-Demers and Nathan Killoran. “Quantum generative adversarial networks”. In: *Physical Review A* 98.1 (2018), p. 012324.

[166] He-Liang Huang et al. “Experimental quantum generative adversarial networks for image generation”. In: *Physical Review Applied* 16.2 (2021), p. 024051.

[167] Haozhen Situ et al. “Quantum generative adversarial network for generating discrete distribution”. In: *Information Sciences* 538 (2020), pp. 193–208.

[168] Lu Bai et al. “Graph Convolutional Neural Networks based on Quantum Vertex Saliency”. In: *arXiv Preprint arXiv:1809.01090* (2018).

[169] Cenk Tüysüz et al. “A quantum graph neural network approach to particle track reconstruction”. In: *arXiv Preprint arXiv:2007.06868* (2020).

[170] Ju-Young Ryu, Eyuel Elala, and June-Koo Kevin Rhee. “Quantum Graph Neural Network Models for Materials Search”. In: *Materials* 16.12 (2023), p. 4300.

[171] Nouhaila Innan et al. “Financial fraud detection using quantum graph neural networks”. In: *arXiv Preprint arXiv:2309.01127* (2023).

[172] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L Fang. “Quantum long short-term memory”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 8622–8626.

[173] Yunjun Yu et al. “Prediction of solar irradiance one hour ahead based on quantum long short-term memory network”. In: *IEEE Transactions on Quantum Engineering* (2023).

[174] Mina Abbaszade et al. “Application of quantum natural language processing for language translation”. In: *IEEE Access* 9 (2021), pp. 130434–130448.

[175] Feng Li et al. “Quantum weighted long short-term memory neural network and its application in state degradation trend prediction of rotating machinery”. In: *Neural Networks* 106 (2018), pp. 237–248.

[176] William Huggins et al. “Towards quantum machine learning with tensor networks”. In: *Quantum Science and technology* 4.2 (2019), p. 024001.

[177] Fanjie Kong, Xiao-Yang Liu, and Ricardo Henao. “Quantum tensor network in machine learning: An application to tiny object classification”. In: *arXiv Preprint arXiv:2101.03154* (2021).

[178] Rui Huang, Xiaoqing Tan, and Qingshan Xu. “Variational quantum tensor networks classifiers”. In: *Neurocomputing* 452 (2021), pp. 89–98.

[179] Francesco Tacchino et al. “An artificial neuron implemented on an actual quantum processor”. In: *Quantum Information* 5.1 (2019), p. 26.

[180] Erik Torrontegui and Juan José García-Ripoll. “Unitary quantum perceptron as efficient universal approximator”. In: *Europhysics Letters* 125.3 (2019), p. 30004.

[181] Alaa Sagheer, Mohammed Zidan, and Mohammed M Abdelsamea. “A novel autonomous perceptron model for pattern classification applications”. In: *Entropy* 21.8 (2019), p. 763.

[182] Richard P Feynman et al. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6/7 (2018).

[183] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510.

[184] Subhash C Kak. “Quantum Neural Computing”. In: *Advances in Imaging and Electron Physics* 94 (1995), pp. 259–313.

[185] Rishab Parthasarathy and Rohan T Bhowmik. “Quantum optical convolutional neural network: a novel image recognition framework for quantum computing”. In: *IEEE Access* 9 (2021), pp. 103337–103346.

[186] Dong Yumin, Mingqiu Wu, and Jinlei Zhang. “Recognition of pneumonia image based on improved quantum neural network”. In: *IEEE Access* 8 (2020), pp. 224500–224512.

[187] Ge Liu et al. “A quantum Hopfield neural network model and image recognition”. In: *Laser Physics Letters* 17.4 (2020), p. 045201.

[188] Lihui Fu and Junfeng Dai. "A speech recognition based on quantum neural networks trained by IPSO". In: *2009 International Conference on Artificial Intelligence and Computational Intelligence*. Vol. 2. IEEE. 2009, pp. 477–481.

[189] Chao-Han Huck Yang et al. "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6523–6527.

[190] Pranav Kairon and Siddhartha Bhattacharyya. "COVID-19 outbreak prediction using quantum neural networks". In: *Intelligence Enabled Research: DoSIER 2020* (2021), pp. 113–123.

[191] Engy El-Shafeiy et al. "Approach for training quantum neural network to predict severity of COVID-19 in patients". In: *Computers, Materials, & Continua* 66.2 (2020), pp. 1745–1755.

[192] Xiuqiong Chen et al. "Recurrent neural networks are universal approximators with stochastic inputs". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.10 (2022), pp. 7992–8006.

[193] D Anderson and K Burnham. "Model selection and multi-model inference". In: *Second. NY: Springer-Verlag* 63.2020 (2004), p. 10.

[194] Edward N Lorenz. "Deterministic nonperiodic flow". In: *Journal of Atmospheric Sciences* 20.2 (1963), pp. 130–141.

[195] Xiao-Tang Ju et al. "Reducing environmental risk by improving N management in intensive Chinese agricultural systems". In: *Proceedings of the National Academy of Sciences* 106.9 (2009), pp. 3041–3046.

[196] Patrick Killeen et al. "Corn grain yield prediction using UAV-based high spatiotemporal resolution imagery, machine learning, and spatial cross-validation". In: *Remote Sensing* 16.4 (2024), p. 683.

[197] Tet Yeap et al. "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.

[198] Jianguo Liu and Jared Diamond. "China's environment in a globalizing world". In: *Nature* 435.7046 (2005), pp. 1179–1186.

[199] Feng Cui et al. "Assessing biogeochemical effects and best management practice for a wheat–maize cropping system using the DNDC model". In: *Biogeosciences* 11.1 (2014), pp. 91–107.

[200] David Ussiri and Rattan Lal. "The role of nitrous oxide on climate change". In: *Soil Emission of Nitrous Oxide and its Mitigation*. Springer, 2013, pp. 1–28.

[201] Dennis L Hartmann et al. "Observations: atmosphere and surface". In: *Climate change 2013 the physical science basis: Working group I contribution to the fifth assessment report of the intergovernmental panel on climate change*. Cambridge University Press, 2013, pp. 159–254.

[202] AR Ravishankara, John S Daniel, and Robert W Portmann. "Nitrous oxide (N2O): the dominant ozone-depleting substance emitted in the 21st century". In: *Science* 326.5949 (2009), pp. 123–125.

[203] Ravi P Sripada et al. "Achieving Sustainable Agriculture: Overview of Current and Future Agronomic Best Practices". In: *Convergence of Food Security, Energy Security and Sustainable Agriculture* (2014), pp. 173–196.

[204] Annette Freibauer and Martin Kaltschmitt. "Controls and models for estimating direct nitrous oxide emissions from temperate and sub-boreal agricultural mineral soils in Europe". In: *Biogeochemistry* 63 (2003), pp. 93–115.

[205] T Leppelt et al. "Nitrous oxide emission budgets and land-use-driven hotspots for organic soils in Europe". In: *Biogeosciences* 11.23 (2014), pp. 6595–6612.

[206] Klaus Butterbach-Bahl et al. "Nitrous oxide emissions from soils: how well do we understand the processes and their controls?" In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 368.1621 (2013), p. 20130122.

[207] Philippe Ciais et al. "Carbon and other biogeochemical cycles". In: *Climate Change 2013: the Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2014, pp. 465–570.

[208] Deepak R Joshi et al. "Quantification and Machine learning based $N_2O$-N and $CO_2$-C emissions predictions from a decomposing rye cover crop". In: *Agronomy Journal* (2022).

[209] Laurent Bigaignon et al. "Combination of two methodologies, artificial neural network and linear interpolation, to gap-fill daily nitrous oxide flux measurements". In: *Agricultural and Forest Meteorology* 291 (2020), p. 108037.

[210] Rezvan Taki et al. “Comparison of two gap-filling techniques for nitrous oxide fluxes from agricultural soil”. In: *Canadian Journal of Soil Science* 99.1 (2018), pp. 12–24.

[211] Debasish Saha, Bruno Basso, and G Philip Robertson. “Machine learning improves predictions of agricultural nitrous oxide ($N_2O$) emissions from intensively managed cropping systems”. In: *Environmental Research Letters* 16.2 (2021), p. 024004.

[212] Sylvia Xiaojing Deng. “Assessment of Greenhouse Gas Emissions from Agriculture in Canada: A Comparative Analysis”. In: *Office of Audit and Evaluation* (2022).

[213] Dave S Reay et al. “Global agriculture and nitrous oxide emissions”. In: *Nature climate change* 2.6 (2012), pp. 410–416.

[214] M Maag and Finn Pilgaard Vinther. “Nitrous oxide emission by nitrification and denitrification in different soil types and at different soil moisture contents and temperatures”. In: *Applied Soil Ecology* 4.1 (1996), pp. 5–14.

[215] Yuk-Faat Wu et al. “Diurnal variability in soil nitrous oxide emissions is a widespread phenomenon”. In: *Global Change Biology* 27.20 (2021), pp. 4950–4966.

[216] Rana A Genedy et al. “A physics-informed long short-term memory model for estimating ammonia emissions from dairy manure during storage”. In: *Science of The Total Environment* 912 (2024), p. 168885.

[217] Haotian Chen, Enno Katelhon, and Richard G Compton. “Predicting voltammetry using physics-informed neural networks”. In: *The Journal of Physical Chemistry Letters* 13.2 (2022), pp. 536–543.

[218] Gabriel Bradford et al. “Chemistry-Informed Machine Learning for Polymer Electrolyte Discovery”. In: *ACS Central Science* 9.2 (2023), pp. 206–216.

[219] Johannes Wilhelmus Maria Pullens et al. “Identifying criteria for greenhouse gas flux estimation with automatic and manual chambers: A case study for $N_2O$”. In: *European Journal of Soil Science* 74.1 (2023), e13340.

[220] J Luo et al. “Variation in denitrification activity with soil depth under pasture”. In: *Soil Biology and Biochemistry* 30.7 (1998), pp. 897–903.

[221] Peter M Groffman. “Nitrification and denitrification in conventional and no-tillage soils”. In: *Soil Science Society of America Journal* 49.2 (1985), pp. 329–334.

[222] C Müller, RR Sherlock, and PH Williams. "Field method to determine $N_2O$ emission from nitrification and denitrification". In: *Biology and Fertility of Soils* 28 (1998), pp. 51–55.

[223] Fabio Camilli et al. "Superposition principle and schemes for measure differential equations". In: *arXiv preprint arXiv:1902.05619* (2019).

[224] Clive WJ Granger. "Investigating causal relations by econometric models and cross-spectral methods". In: *Econometrica: journal of the Econometric Society* (1969), pp. 424–438.

[225] R Ruser et al. "Emission of $N_2O$, $N_2$ and $CO_2$ from soil fertilized with nitrate: effect of compaction, soil moisture and rewetting". In: *Soil Biology and Biochemistry* 38.2 (2006), pp. 263–274.

[226] J Brownlee. "How to Use StandardScaler and MinMaxScaler Transforms in Python". In: ().

[227] Chaoqun Lu et al. "Century-long changes and drivers of soil nitrous oxide ($N_2O$) emissions across the contiguous United States". In: *Global Change Biology* 28.7 (2022), pp. 2505–2524.

[228] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On early stopping in gradient descent learning". In: *Constructive Approximation* 26.2 (2007), pp. 289–315.

[229] Navid Mohammadi Foumani et al. "Deep learning for time series classification and extrinsic regression: A current survey". In: *arXiv:2302.02515* (2023).

[230] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *Pattern recognition* 77 (2018), pp. 354–377.

[231] Yi Zheng et al. "Time series classification using multi-channels deep convolutional neural networks". In: *International conference on web-age information management*. Springer. 2014, pp. 298–310.

[232] Long Short-Term Memory. "Long short-term memory". In: *Neural computation* 9.8 (2010), pp. 1735–1780.

[233] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[234] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[235] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3128–3137.

[236] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[237] George Zerveas et al. "A transformer-based framework for multivariate time series representation learning". In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2114–2124.

[238] Wensi Tang et al. "Rethinking 1D-CNN for time series classification: A stronger baseline". In: *arXiv preprint arXiv:2002.10061* (2020), pp. 1–7.

[239] Sheikh Mohammad Idrees, M Afshar Alam, and Parul Agarwal. "A prediction approach for stock market volatility based on time series data". In: *IEEE Access* 7 (2019), pp. 17287–17298.

[240] Huu-Thanh Duong and Tram-Anh Nguyen-Thi. "A review: preprocessing techniques and data augmentation for sentiment analysis". In: *Computational Social Networks* 8.1 (2021), pp. 1–16.

[241] Sumeyra Demir et al. "Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting". In: *Applied Energy* 304 (2021), p. 117695.

[242] RF Grant and E Pattey. "Temperature sensitivity of $N_2O$ emissions from fertilized agricultural soils: Mathematical modeling in ecosys". In: *Global biogeochemical cycles* 22.4 (2008).

[243] Chairul Saleh, Nur Rachman Dzakiyullah, and Jonathan Bayu Nugroho. "Carbon dioxide emission prediction using support vector machine". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 114. 1. IOP Publishing. 2016, p. 012148.

[244] Jun Wang et al. "NOx Prediction by Cylinder Pressure Based on RBF Neural Network in Diesel Engine". In: *2010 International Conference on Measuring Technology and Mechatronics Automation*. Vol. 2. 2010, pp. 792–795. DOI: 10.1109/ICMTMA.2010.621.

[245] Xiaochen Hao et al. "Prediction of nitrogen oxide emission concentration in cement production process: a method of deep belief network with clustering and time series". In: *Environmental Science and Pollution Research International* 28.24 (2021), pp. 31689–31703.

[246] Raquel Prado. *Latent structure in non-stationary time series*. Duke University, 1998.

[247] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. "Data augmentation for time series classification using convolutional neural networks". In: *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2016.

[248] Ci Lin et al. "Agriculture-informed Neural Networks for Predicting Nitrous Oxide Emissions". In: *ACM Transactions on Internet of Things* (2024).

[249] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv Preprint arXiv:1511.06434* (2015).

[250] Yao Zhang and Qiang Ni. "Recent advances in quantum machine learning". In: *Quantum Engineering* 2.1 (2020), e34.

[251] Fevrier Valdez and Patricia Melin. "A review on quantum computing and deep learning algorithms and their applications". In: *Soft Computing* 27.18 (2023), pp. 13217–13236.

[252] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. "Quantum computing methods for supervised learning". In: *Quantum Machine Intelligence* 3.2 (2021), p. 23.

[253] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202.

[254] Chrysanthos Maraveas et al. "Harnessing quantum computing for smart agriculture: Empowering sustainable crop management and yield optimization". In: *Computers and Electronics in Agriculture* 218 (2024), p. 108680.

[255] John A Gubner. *Probability and random processes for electrical and computer engineers*. Cambridge University Press, 2006.

[256] KA Metivier, E Pattey, and RF Grant. "Using the ecosys mathematical model to simulate temporal variability of nitrous oxide emissions from a fertilized agricultural soil". In: *Soil Biology and Biochemistry* 41.12 (2009), pp. 2370–2386.

[257] Xi Hang Cao, Ivan Stojkovic, and Zoran Obradovic. "A robust data scaling algorithm to improve classification accuracies in biomedical data". In: *BMC bioinformatics* 17 (2016), pp. 1–10.

[258] Trevor Lanting et al. "Entanglement in a quantum annealing processor". In: *Physical Review X* 4.2 (2014), p. 021041.

[259] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms". In: *Advanced Quantum Technologies* 2.12 (2019), p. 1900070.

[260] Samuel Yen-Chi Chen et al. "Variational quantum circuits for deep reinforcement learning". In: *IEEE Access* 8 (2020), pp. 141007–141024.

[261] Kosuke Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (2018), p. 032309.

[262] Maria Schuld et al. "Evaluating analytic gradients on quantum hardware". In: *Physical Review A* 99.3 (2019), p. 032331.