# Distributed Intelligence and Generative AI to Enhance the Performance of Next-Generation Wireless Networks

by

Han Zhang

A thesis submitted to the University of Ottawa
in partial fulfillment of the thesis requirement for the degree of

**Doctor of Philosophy in Electrical and Computer Engineering**

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Han Zhang, Ottawa, Ontario, Canada, 2025

**Acknowledgment**

First and foremost, I want to thank my supervisor, Dr. Melike Erol-Kantarci. She has always been responsible and supportive throughout my Ph.D. journey. She not only led my research with insight but also taught me how to think critically and independently as a researcher. Beyond academics, her thoughtful advice on life, career, and personal development has meant a great deal to me. I feel truly fortunate to have had her as my supervisor.

I would also like to thank all my collaborators from the NETCORE lab and the Ericsson team. I truly appreciate the help and time from Dr. Hao Zhou, Dr. Medhat Elsayed, Dr. Pedro Enrique Iturria-Rivera, Dr. Shahram Mollahasani, Dr. Majid Bavand, Dr. Raimundas Gaigalas, Dr. Yigit Ozcan, Dr. Akram Bin Sediq, Dr. Ali Afana, Mohammad Farzanullah, and Mohammad Ghassemi. Special thanks to Dr. Hao Zhou, who helped me a lot at the beginning of my research. He is always patient, and I learned many research skills from him. I also want to thank my labmates Shavbo Salehi, Md Arafat Habib, Ycaro Dantas, and others for their support.

I am deeply grateful to my parents, Qiuhong Wang and Wei Zhang, for being my unwavering source of strength and encouragement. Their belief in me has carried me through challenging times. I also want to thank my aunt Linda Wang, her husband Allan Qu and my cousins for their kindness and generous help when I first arrived in this new country.

To my friends, thank you for your company and encouragement during this journey. Special thanks to Xinnan Zheng, Shixi Jiang, Yan Liu, Wenlin Zhang, Yeshan Sun, and Linke Zhou, for being with me during my study. And most of all, I want to express my deepest appreciation to my love, Junran Chen. He has always been my best friend and my strongest support. His understanding and encouragement helped me face many challenges, and his presence has meant everything to me.

Finally, I want to thank myself for working hard during the past five years. I am really proud of what I have done in my Ph.D. and grateful to all the kind people I met along the way. I hope we all have a bright future ahead.

## Abstract

In recent years, the demand for mobile communications has steadily increased, and the volume of wireless traffic data has reached an unprecedented level. The growing capabilities of the next-generation wireless networks offer the potential for revolutionary applications and use cases, including massive machine-type communications, immersive gaming and virtual reality, edge computing and preventive maintenance through Internet of Things (IoT) sensors. These emerging use cases impose stringent requirements on communication systems, demanding greater agility, intelligence, and adaptability.

To meet these rising demands while controlling capital and operational costs, artificial intelligence (AI) techniques have been increasingly integrated into mobile networks, particularly the radio access network (RAN). AI accelerates the automation of network tasks across various domains. For example, reinforcement learning (RL) has shown promising performance in dynamic control applications such as resource allocation and traffic steering. Similarly, supervised learning methods are effective for prediction tasks like traffic volume forecasting and classification tasks like modulation classification.

However, the integration of AI into mobile networks also introduces new challenges, including high computational demands, security vulnerabilities, privacy concerns, and increased system overhead. How to effectively incorporate advanced AI techniques such as transfer learning (TL), knowledge distillation (KD), and model compression to mitigate these concerns and optimize their performance in mobile networks remains an open problem. There is still a big gap between existing AI methodologies and their practical, accessible applications in mobile communication systems.

This thesis focuses on two major research directions that aim to bridge this gap. First, we investigate distributed AI frameworks for mobile networks. By decentralizing intelligence to various nodes, such as base stations (BSs) and user equipment (UE), distributed learning enhances scalability, reduces service latency, and supports user-centric network management. Nevertheless, it introduces challenges including conflicting local decisions, limited node resources, and the presence of adversarial agents, motivating the need for robust and efficient distributed solutions.

Second, we explore the application of generative AI (GAI) and foundation models (FMs) in mobile communications. GAI is an emerging AI technology that has recently achieved groundbreaking advancements in content generation. FMs refer to large-scale machine learning (ML) models trained on a broad data set that can be adapted and fine-tuned for various applications and downstream tasks. As a combination of GAI and FMs, generative FMs (GFMs), such as large language models (LLMs), are equipped with

emergent abilities that can deliver unprecedented benefits. Their deployment opens new possibilities for automating network management and realizing intrinsic intelligence in mobile communication systems.

The findings of this research demonstrate the effectiveness of applying both distributed AI techniques and GFMs in mobile communication systems. Distributed learning enables adaptive, decentralized network management across various network nodes and unlocks the potential of on-device intelligence. GFMs introduce a higher level of cognitive capability, assisting in complex or unforeseen network scenarios. By integrating these AI approaches, this work envisions a future mobile communication system that is AI-enhanced, fully automated, and continuously self-evolving.

# Table of Contents

**List of Tables** **xi**

**List of Figures** **xiii**

**List of Abbreviations** **xviii**

**List of Symbols** **xxi**

**1 Introduction** **1**

1.1 Motivations . . . . . 1

1.1.1 Collaborative Distributed AI Frameworks for Mobile Networks . . . 2

1.1.2 Enhance intelligence and adaptability in wireless networks with GFM Integration . . . . . 4

1.2 Thesis Contributions and Organization . . . . . 5

1.2.1 Collaborative distributed AI-related contributions . . . . . 5

1.2.2 GFM-related contributions . . . . . 7

1.2.3 Summary . . . . . 8

1.3 Publications . . . . . 9

1.3.1 Journal Publications . . . . . 9

1.3.2 Conference Publications . . . . . 9

1.3.3 Patents and Invention Disclosures . . . . . 10

**2 Background and Literature Review** **11**

2.1 Background . . . . . 11

2.1.1 Reinforcement learning and deep reinforcement learning . . . . . 11

2.1.2 Federated learning . . . . . 14

2.1.3 Transfer learning and transfer reinforcement learning . . . . . 16

2.1.4 Generative AI and foundation models . . . . . 18

2.2 Literature Review . . . . . 21

2.2.1 AI-empowered network management applications . . . . . 22

2.2.2 Distributed ML in mobile networks and security concerns . . . . . 25

2.2.3 Enhanced intelligence in wireless networks with GFMs . . . . . 30

**3 Deep reinforcement learning for resource allocation in O-RAN** **37**

3.1 Resource Allocation System Model . . . . . 37

3.1.1 Communication Channel Model . . . . . 38

3.1.2 QoS Definition . . . . . 39

3.1.3 Problem Formulation . . . . . 40

3.2 DQN based Power Control and Resource Allocation xAPPs . . . . . 41

3.2.1 MDP of Power Control xAPP and Resource Allocation Control xAPP Agent for a Single Slice Setup . . . . . 41

3.2.2 MDP of Power Control xAPP and Resource Allocation Control xAPP Agent for multiple slices setup . . . . . 43

3.3 Team Learning and Federated Learning-based xAPPs Conflicts mitigation 44

3.3.1 The conflicts of xAPPs . . . . . 44

3.3.2 Team learning-based Mitigation . . . . . 45

3.3.3 Federated Learning-based Mitigation . . . . . 47

3.4 Simulation Result . . . . . 48

3.4.1 Parameter Settings . . . . . 48

3.4.2 Simulation Results of Single Slice Setup . . . . . 51

3.4.3 Simulation Results of Multiple Slices Setup . . . . . 52

**4 Knowledge Transfer and Federated Learning Enabled UE-Centric Traffic Steering** **55**

4.1 UE-centric System Model for traffic steering . . . . . 56

4.1.1 Architecture . . . . . 56

4.1.2 Communication Model . . . . . 57

4.1.3 Problem Formulation . . . . . 58

4.2 Proposed traffic steering Scheme . . . . . 59

4.2.1 Algorithm Framework . . . . . 59

4.2.2 Federated Reinforcement Learning based UE-centric traffic steering 61

4.2.3 Growing and Pruning-based Model Compression . . . . . 65

4.2.4 Q-value-based Knowledge Transfer . . . . . 66

4.3 Simulation Results . . . . . 69

4.3.1 Baselines . . . . . 69

4.3.2 Simulation Settings . . . . . 69

4.3.3 Performance Analysis between UE-centric traffic steering and Cell-centric traffic steering . . . . . 70

4.3.4 Performance Analysis between Four UE-centric Frameworks . . . . 73

**5 Attacks and defense over federated DRL-enabled cell sleep control scenario** **77**

5.1 System Model and Problem Formulation . . . . . 77

5.1.1 System Model . . . . . 79

5.1.2 Problem Formulation . . . . . 79

5.1.3 Federated Learning for Cell Sleep Control . . . . . 80

5.2 Intelligent Attacks: GAN-enhanced and Regularization-based Attacks. . . . 82

5.2.1 GAN-enhanced Model Poisoning Attack . . . . . 82

5.2.2 Regularization-based Model Poisoning Attack . . . . . 84

5.2.3 Baseline Method: Data Poisoning Attack . . . . . 85

5.2.4 Computational complexity and overhead analysis . . . . . 85

5.3 Intelligent Defense Methods: Autoencoder and KD-based Defense . . . . . 86
5.3.1 Autoencoder-based Two-step Defense . . . . . 86
5.3.2 Knowledge Distillation-based Defense . . . . . 88
5.3.3 Baseline Defense: Krum . . . . . 92
5.3.4 Computational complexity and overhead analysis . . . . . 93
5.3.5 Attackers and defenders analysis . . . . . 93
5.4 Numeric Results . . . . . 94
5.4.1 Simulation Settings . . . . . 94
5.4.2 Model distribution analyses . . . . . 95
5.4.3 Performance of Attacks . . . . . 96
5.4.4 Performance of Defense Methods . . . . . 99

**6 Dual attention model-weighted robust FL in mobile communication systems 104**
6.1 System Model and Problem Formulation . . . . . 104
6.2 Dual attention model-weighted FL for radio signal classification . . . . . 106
6.2.1 Attention model . . . . . 107
6.2.2 Dual attention model-weighted FL. . . . . 107
6.3 Numeric Results . . . . . 110
6.3.1 Simulation settings . . . . . 110
6.3.2 Simulation results . . . . . 111

**7 GAI-in-the-loop in mobile communication systems 114**
7.1 GFM-in-the-loop . . . . . 116
7.2 Enhance traditional ML models with GFM/GPTs . . . . . 117
7.2.1 Requirement stage . . . . . 118
7.2.2 Data processing stage . . . . . 118
7.2.3 Model development stage . . . . . 119

7.2.4 Operation stage . . . . . 120
7.3 Deployments of ML models and GFM in mobile networks . . . . . 120
7.4 Pre-trained LLM-empowered framework for automatic network intrusion detection: A case study . . . . . 122
7.4.1 LLM-empowered network intrusion detection . . . . . 123
7.4.2 In-context learning-enhanced detection . . . . . 125
7.4.3 Experimental settings and Results . . . . . 127

**8 Pre-trained LLM-empowered mobile traffic prediction 132**
8.1 System Design . . . . . 132
8.1.1 Problem Formulation . . . . . 133
8.1.2 Data Preprocessing and Prompt design . . . . . 135
8.1.3 Deployment Considerations for LLM-based Predictions . . . . . 136
8.2 Two-step efficient demonstration selection . . . . . 137
8.2.1 Finding Suitable Demonstrations for ICL . . . . . 137
8.2.2 First step: Evaluate the effectiveness of ICL demonstrations . . . . 139
8.2.3 Second step: Evaluate the informativeness of ICL demonstrations . 142
8.2.4 Computational complexity of proposed method . . . . . 145
8.2.5 Batch calibration . . . . . 146
8.3 Numerical Results . . . . . 147
8.3.1 Experimental Settings and Evaluation Metrics . . . . . 147
8.3.2 Performance Analysis of ICL . . . . . 149
8.3.3 Performance Analysis under Different Predicted Error Rate Thresholds 153
8.3.4 Performance Analysis of LLM-based Prediction and LSTM-based Prediction . . . . . 153
8.3.5 Numerical Results with PHI-3-mini model . . . . . 156

**9 Supervised Contrastive Learning for Uncertainty-aware Wireless Signal Analysis 158**

9.1 System Design . . . 158

9.1.1 Signal Model . . . 159

9.1.2 Problem Formulation . . . 161

9.2 Supervised Contrastive Learning for Latent Representation Generation . . 162

9.2.1 Supervised Contrastive Learning . . . 162

9.2.2 Data Pre-processing . . . 164

9.2.3 Partial Hard Positive Mining . . . 165

9.3 Uncertainty Estimation and Rejection . . . 166

9.4 Numerical Results . . . 168

9.4.1 Experimental Settings . . . 168

9.4.2 Evaluation of ResNet Backbone . . . 168

9.4.3 Performance of Representation Learning . . . 169

9.4.4 Uncertainty Estimation and Rejection . . . 172

**10 Conclusion 176**

10.1 Mitigate conflicts between distributed AI-based applications in mobile networks . . . 176

10.2 Knowledge Transfer and Federated Learning Enabled UE-Centric Traffic Steering . . . 177

10.3 Attacks and defense over federated DRL-enabled cell sleep control scenario 177

10.4 Dual attention model-weighted robust FL in mobile communication systems 178

10.5 GAI-in-the-loop in mobile communication systems . . . 178

10.6 Pre-trained LLM-empowered mobile traffic prediction . . . 179

10.7 Supervised Contrastive Learning for Uncertainty-aware Wireless Signal Analysis . . . 179

10.8 Challenges and Future Directions . . . 180

# List of Tables


2.1 A comparison between the definition and capabilities of LLM, Generative AI, and FM. . . . 18

2.2 AI-empowered other prediction tasks in mobile networks . . . . 24

2.3 Summary of PFL algorithms with different personalization strategies. . . . 28

2.4 A summary of prominent single-modality FMs . . . . 32

2.5 A summary of prominent multi-modal FMs . . . . 33

3.1 Simulation settings for the single slice scenario . . . . 49

3.2 Simulation settings. . . . 50

4.1 Simulation settings. . . . 71

4.2 Running time of knowledge transfer and federated learning-enabled UE-centric (KT-FLUC) and FLI. . . . 75

5.1 Simulation settings. . . . 95

8.1 MAE, RMSE and $R^2$-Score of mobile traffic prediction under different scenarios. . . . 149

8.2 The ICL ratio, MAE, RMSE and $R^2$-Score under different predicted error rate threshold settings. . . . 152

8.3 MAE, RMSE and $R^2$-Score of mobile traffic prediction of LLM-based method and LSTM-based method. . . . 154

8.4 MAE, RMSE and $R^2$-Score of LSTM-based mobile traffic prediction and LLM-based prediction while using training data from different scenarios . . 155

8.5 MAE, RMSE and $R^2$-Score of mobile traffic prediction using PHI-3-mini. . 157

9.1 Simulation Settings . . . . . 167

# List of Figures


2.1 An illustration of reinforcement learning (RL). . . . . 12
2.2 The structure of Deep Q-Network (DQN). . . . . 14
2.3 The structure of federated learning (FL). . . . . 15
2.4 An illustration of transfer reinforcement learning (TRL). . . . . 17
2.5 The transformer model architecture. . . . . 20
2.6 open radio access network (O-RAN) architecture and x-application (xAPP)s. 25
2.7 FM-enhanced visual data analysis and applications in wireless networks . . 35

3.1 System model of resource allocation in O-RAN. . . . . 38
3.2 xAPPs interactions with team learning. . . . . 46
3.3 Process of federated deep reinforcement learning for xAPPs conflicts mitigation. . . . . 47
3.4 System throughput (traffic load = 4 Mbps, user speed = 20 m/s). . . . . 51
3.5 System throughput with different traffic load (user speed = 20 m/s). . . . 51
3.6 packet drop rate (PDR) with different traffic load (user speed = 20 m/s). . 52
3.7 System throughput with different user moving speed (traffic load = 4 Mbps). 52
3.8 ultra reliable and low latency Communications (URLLC) slices delay under various enhanced mobile broadband (eMBB) traffic load. . . . . 53
3.9 empirical complementary cumulative distribution function (ECCDF) of the URLLC slices delay distributions. . . . . 53
3.10 Throughput of eMBB slice under various eMBB traffic load. . . . . 53

3.11 Convergence comparison. . . . . 53

4.1 Dynamic wireless networks with one macro base station (MBS) and several small base stations (SBSs). . . . . 56
4.2 The framework of proposed user equipment (UE)-centric traffic steering method. . . . . 59
4.3 Federated learning process. . . . . 63
4.4 Two phases of growing and pruning. . . . . 66
4.5 Knowledge transfer process. . . . . 67
4.6 The convergence performance of system reward. . . . . 70
4.7 The average guaranteed bit rate traffic (GBR) traffic delay with different UE numbers. . . . . 72
4.8 The average non-GBR traffic throughput. . . . . 72
4.9 The average traffic blocking of UEs of cell-centric traffic steering and UE-centric traffic steering. . . . . 74
4.10 The average quality of service (QoS) violation rate of UEs of cell-centric traffic steering and UE-centric traffic steering. . . . . 74
4.11 The average transmission cycle of UEs of cell-centric traffic steering and UE-centric traffic steering. . . . . 74
4.12 The convergence performance of system reward. . . . . 75
4.13 The average GBR traffic delay with different UE numbers. . . . . 76
4.14 The average non-GBR traffic throughput. . . . . 76

5.1 Attacks in federated deep reinforcement learning-enabled cell sleep control scenario. . . . . 78
5.2 The procedure of GAN-enhanced model poisoning attack. . . . . 82
5.3 The procedure of autoencoder-based two-step defense. . . . . 86
5.4 The structure of KD-based defense against malicious participants in FL. . 88
5.5 A comparison between IID and non-IID data. . . . . 96
5.6 The system performance under three different attacks. . . . . 97

5.7 The convergence curves of the system reward without attacks, under GAN-based attacks, under regularization-based attacks, and proposed defense. . 98
5.8 The system performance under data poisoning attacks with different defense schemes. . . . . 99
5.9 The system performance under GAN-enhanced model attacks with different defense schemes. . . . . 100
5.10 The system performance under regularization-based model attacks with different defense schemes. . . . . 101
5.11 The dynamic change of energy efficiency after attacks and defense. . . . . 102
5.12 The dynamic change of system throughput after attacks and defense. . . . 103

6.1 System model of FL-enabled radio signal classification. . . . . 105
6.2 The proposed dual attention model for FL. . . . . 108
6.3 The convergence curves of FedAvg and proposed dual attention model-weighted FL in the secure system. . . . . 111
6.4 The accuracy of modulation type classification under different signal noise ratios (SNRs) in the secure system. . . . . 111
6.5 The convergence curves of Multi-Krum and the proposed dual attention model-weighted FL in the presence of one attacker. . . . . 112
6.6 The accuracy of modulation type classification under different SNRs in the presence of one attacker. . . . . 112
6.7 The convergence curves of Multi-Krum and the proposed dual attention model-weighted FL in the presence of two attackers. . . . . 113
6.8 The accuracy of modulation type classification under different SNRs in the presence of two attackers. . . . . 113

7.1 Illustration of the evolution from human-in-the-loop to GFM-in-the-loop in next-generation networks. In the GFM-in-the-loop framework, GFMs act as intermediaries between human-level management and traditional machine learning and optimization algorithms by automating network control based on semantic intentions, generating semantic-based explanations, and performing model and network management. . . . . 115

7.2 Different ways to enhance machine learning (ML) models with generative foundation modelss (GFMs). The life cycle of ML models includes four stages: requirement stage, data processing stage, operation stage, and model development stage. GFMs can be integrated into each stage. . . . . 117

7.3 Three ways to deploy GFMs and ML models. (a) Both GFMs and ML models are deployed at the center and combined for network management. (b) Both GFMs and ML models are deployed in a distributed manner for Multi-Agent interaction. (c) GFMs are deployed at the center for management, and ML models are deployed in a distributed manner for local training and inference. . . . . 121

7.4 System model of pre-trained large language model (LLM)-empowered network intrusion detection. . . . . 123

7.5 Three different methods to implement in-context learning with labelled examples. . . . . 125

7.6 The feature selection and importance ranking results from three different LLMs, generative pre-trained transformer (GPT)-3.5, GPT-4, and LLAMA. 128

7.7 Comparison of intrusion detection testing accuracy under three different in-context learning methods with GPT-3.5 while increasing the number of in-context learning examples. . . . . 129

7.8 Comparison of intrusion detection testing F1-Score under three different in-context learning methods with GPT-3.5 while increasing the number of in-context learning examples. . . . . 129

7.9 Comparison of intrusion detection testing accuracy under three different in-context learning methods with GPT-4 while increasing the number of in-context learning examples. . . . . 129

7.10 Comparison of intrusion detection testing F1-Score under three different in-context learning methods with GPT-4 while increasing the number of in-context learning examples. . . . . 129

7.11 Comparison of intrusion detection testing accuracy with GPT-4 and convolutional neural network (CNN)-based method. . . . . 130

7.12 Comparison of intrusion detection testing F1-Score with GPT-4 and CNN-based method. . . . . 130

8.1 System design of LLM-based mobile traffic prediction. . . . . 133

8.2 The workflow of the two-step in-context learning (ICL) demonstration selection method. . . . . 136
8.3 A comparison of traffic prediction with the proposed selected ICL demonstrations and zero-shot traffic prediction in four scenarios using PHI-3-medium. 148
8.4 MAE and $R^2$-Score of mobile traffic prediction under different ICL settings. 151
8.5 The ICL ratio and the $R^2$-Score under different predicted error rate threshold settings. . . . . 154
8.6 Traffic prediction with selected ICL demonstrations using PHI-3-mini. . . . 156

9.1 System design of residual network (ResNet)-based automatic modulation classification and representation generation. . . . . 159
9.2 Modulation classification accuracy of ResNet, CNN, and LSTM models across different SNR levels. . . . . 169
9.3 Convergence curve of the supervised contrastive learning (SupCon) model. 169
9.4 Visualization of the learned embeddings from the SupCon model and the supervised learning-based baseline model. . . . . 170
9.5 A comparison of the silhouette score of the representations. . . . . 171
9.6 A comparison of uncertainty scores for in distribution (ID) and out of distribution (OOD) predictions. . . . . 171
9.7 Acceptance rate of correct examples, false examples, and OOD examples under different uncertainty thresholds. . . . . 172
9.8 Uncertainty and recall of correct predictions under different uncertainty rejection thresholds. . . . . 173
9.9 Modulation classification accuracy with uncertainty-aware rejection strategy. 174

# List of Abbreviations

| Acronym | Meaning |
|---|---|
| 3D | three-dimensional |
| 3GPP | 3rd Generation Partnership Project |
| 5G | fifth-generation |
| 6G | sixth-Generation |
| AI | artificial intelligence |
| AMC | automatic modulation classification |
| APT | advanced persistent threat |
| ARIMA | autoregressive integrated moving average |
| BCE | binary cross-entropy |
| BS | base station |
| CE | cross-entropy |
| CNN | convolutional neural network |
| CRL | centralized reinforcement learning |
| CSI | channel state information |
| DC | dual connectivity |
| DL | deep learning |
| DNN | Deep Neural Network |
| DQN | Deep Q-Network |
| DRL | deep reinforcement learning |
| ECCDF | empirical complementary cumulative distribution function |
| EE | energy efficiency |
| eMBB | enhanced mobile broadband |
| FedAvg | Federated Averaging |
| FedProx | Federated Proximal |
| FL | federated learning |
| FM | foundation model |

| | |
|---|---|
| FRL | federated reinforcement learning |
| GAI | generative artificial intelligence |
| GAN | generative adversarial networks |
| GBR | guaranteed bit rate traffic |
| GFM | generative foundation models |
| GNN | graph neural network |
| GPS | global positioning system |
| GPT | generative pre-trained transformer |
| GPU | graphics processing unit |
| GRU | gated recurrent unit |
| HFRL | horizontal federated reinforcement learning |
| HRL | hierarchical reinforcement learning |
| ICL | in-context learning |
| ID | in distribution |
| IDL | independent deep Q-learning |
| IID | independent and identically distributed |
| IRL | independent reinforcement learning |
| KD | knowledge distillation |
| KL | Kullback–Leibler |
| KT-FLUC | knowledge transfer and federated learning-enabled UE-centric |
| LiDAR | light detection and ranging |
| LLM | large language model |
| LoRA | Low-Rank Adaptation |
| LSTM | long short-term memory |
| LTE | Long-Term Evolution |
| MAE | mean absolute error |
| MARL | multi-agent reinforcement learning |
| MBS | macro base station |
| MDP | Markov decision process |
| ML | machine learning |
| mmWave | millimeter wave |
| NN | neural network |
| non-IID | non-independent and identically distributed |
| OOD | out of distribution |
| O-RAN | open radio access network |
| PCA | principal component analysis |
| PDR | packet drop rate |
| PFL | personalized federated learning |

| | |
|---|---|
| PoZ | percentage of zero |
| PPF | proportional fair algorithm |
| PRB | physical resource block |
| QoS | quality of service |
| RADAR | radio detection and ranging |
| RAG | Retrieval augmented generation |
| RAN | radio access network |
| rAPP | RAN Application |
| RAT | radio access technology |
| RBG | resource block group |
| ReLU | Rectified Linear Unit |
| ResNet | residual network |
| RF | radio frequency |
| RIC | radio intelligent controllers |
| RL | reinforcement learning |
| RMSE | root mean squared error |
| RSSI | received signal strength indicator |
| SBS | small base station |
| SINR | signal interference noise ratio |
| SMA | simple moving average |
| SNR | signal noise ratio |
| SupCon | supervised contrastive learning |
| SVM | support vector machine |
| TDL | team deep Q-learning |
| TL | transfer learning |
| TRL | transfer reinforcement learning |
| TS | traffic steering |
| TTI | Transmission Time Interval |
| UAV | unmanned aerial vehicle |
| UE | user equipment |
| UMAP | uniform manifold approximation and projection |
| URLLC | ultra reliable and low latency Communications |
| VFRL | vertical federated reinforcement learning |
| ViT | vision transformer |
| WMA | weighted moving average |
| xAPP | x-application |

# List of Symbols

| | |
|---|---|
| $\alpha$ | Learning rate/ Binary indicator |
| $\delta_n$ | whether the SBS is sleeping or not |
| $\epsilon$ | Exploration rate |
| $\epsilon_{LLM}$ | LLM prediction error for the test sample |
| $\Gamma$ | Received signal strength indicator |
| $\gamma$ | The discount factor |
| $\gamma_1$ | Ratio of energy consumption in the sleep mode |
| $\gamma_2$ | Ratio of energy consumption in the deep sleep mode |
| $\lambda$ | The mean arrival rate |
| $\omega_k$ | Parameters of the local model |
| $\Phi$ | Achievable rate |
| $\phi(t)$ | Instantaneous phase of the signal |
| $\phi_0$ | Initial phase offset |
| $\Phi_k$ | The frequency that the device achieves a specific QoS requirement |
| $\Pi$ | The policy function |
| $\Psi$ | Training data size |
| $\theta$ | Model parameter |

| | |
|---|---|
| $\theta_k$ | Parameters of the local model |
| $a$ | Action |
| $a(t)$ | Instantaneous amplitude of the signal |
| $B$ | Bandwidth |
| $B_{max}$ | Estimated maximum throughput |
| $C$ | Link capacity |
| $d$ | Transmission delay |
| $D_{max}$ | Estimated maximum delay |
| $E[.]$ | Expectation |
| $Enc$ | Encoder layers |
| $Er$ | Predicted error rate |
| $f(t)$ | Instantaneous frequency of the signal |
| $f_c$ | Carrier frequency |
| $f_s$ | Sampling frequency |
| $G$ | The discounted cumulative reward |
| $h(t)$ | Channel impulse response function |
| $K$ | The key vector |
| $L$ | The length of queued data |
| $Loss_k^{local}$ | Local loss function |
| $N_0$ | Noise power density of noise |
| $N_{ID}$ | Number of ID classes |
| $N_{OOD}$ | Number of OOD classes |
| $O(.)$ | Asymptotic upper bound |

$P$ Transmission power

$P(i)$ Set of positive examples

$P_n^{AM}$ Energy consumption of the power amplifier

$P_n^{FPGA}$ Energy consumption of FPGA

$P_n^{MP}$ Energy consumption of microprocessor

$PoZ_i^j$ Average PoZ of the $j^{th}$ neuron in the $i^{th}$ layer

$Proj$ Projection layers

$Q$ Q-values/ The query vector/ Quadrature-phase component

$Q$ Quadrature-phase component

$R$ Set of resource blocks

$r$ Reward

$r_I[l]$ In-phase component

$r_Q[l]$ Quadrature-phase component

$s$ State

$s_k$ $k^{th}$ discrete complex symbol

$SINR_{n,k,m}$ signal interference noise ratio

$T$ Transition probability

$t$ Time slot

$T_k$ Traffic type of devices

$V$ The value vector

$w_k$ Parameters of the global model

$x_n$ Input of in-context learning example

$y_n$ Label of in-context learning example

| | |
|---|---|
| $z_i$ | Latent representation |
| $z_k$ | Input of the generator |

# Chapter 1

# Introduction

## 1.1 Motivations

The design of next-generation mobile communication systems is expected to accommodate increasingly diverse use cases and an exponential surge in traffic demand. This evolution necessitates consideration of a broader range of performance metrics, including transmission rate, latency, mobility, energy efficiency, reliability, and security [1]. Mobile network operators are advancing network management strategies to meet these demands.

Nowadays, artificial intelligence (AI) has emerged as a transformative tool in the design, optimization, and management of mobile networks. With its ability to process large-scale data and solve complex decision-making problems, AI is enabling advanced, data-driven network functions. However, the widespread adoption of AI in mobile communication systems also brings new challenges, such as high computational overhead, privacy risks, and security vulnerabilities. As a result, a significant gap remains between the capabilities of current AI techniques and their practical deployment in real-world mobile networks.

Recent breakthroughs in AI, including more efficient architectures and novel training paradigms, are addressing many of these limitations. To fully exploit their potential, it is essential to align emerging AI technologies with the specific demands and constraints of mobile networks.

In this thesis, we explore two major directions for integrating AI into mobile communication systems. The first direction focuses on the design of collaborative AI frameworks with distributed network nodes. This direction is motivated by the inherently decentralized nature of wireless networks, where data and control are often distributed across numerous

devices and access points. Distributed AI provides a good fit for such settings, but also introduces new challenges, including coordination among nodes, resource competition, and security concerns. The second direction investigates the application of generative artificial intelligence (GAI) and foundation models (FMs) to enable more intelligent and adaptive mobile network functions. Pretrained FMs such as LLMs have the ability to operate with limited labeled data. Their emergence suggests new opportunities for building more intelligent and generalized mobile network functions, potentially mitigating some of the limitations in distributed AI systems. The following subsections will detail the motivations and challenges associated with these two directions.

### 1.1.1 Collaborative Distributed AI Frameworks for Mobile Networks

Mobile communication systems consist of various nodes, including base stations (BSs), UEs, edge nodes, and cloud nodes. In early mobile communication networks, the architecture was mainly centralized decision-making, where control operations were executed by central nodes, and others merely followed instructions. Although central nodes had access to global network information and could make optimized decisions, this structure suffered from limited scalability, high service delay, and potential privacy leakage.

With the evolution of mobile communication systems, distributed and hybrid network management approaches have become increasingly popular. These methods distribute tasks across multiple nodes, offering improved scalability, fault tolerance, and responsiveness. Future networks are expected to achieve full autonomy by integrating advanced AI technologies, enabling each node to make independent decisions, collaborate dynamically, and adapt to environmental changes. These limitations motivate a transition from centralized AI-based systems to distributed frameworks that better leverage the decentralized nature of modern networks for enhanced efficiency, reliability, and adaptability.

Despite their benefits, distributed AI-enabled systems face several challenges. Distributed agents may produce conflicting decisions, degrading performance when operating independently. Moreover, they often suffer from limited local resources, such as data, computation, and storage, which constrain their ability to train high-quality models. These issues necessitate research into more collaborative and efficient distributed AI frameworks.

In this thesis, we aim to design AI-based network control applications and combine them with collaborative distributed AI frameworks. This allows distributed nodes to work together, enhancing QoS, considering metrics such as throughput, delay, energy efficiency (EE), and PDR. For instance, we develop a radio resource allocation application that

dynamically assigns limited radio frequency (RF) spectrum among UEs to balance diverse transmission demands. Similarly, we propose a power allocation scheme to improve energy efficiency and mitigate intra-cell interference by adjusting UE transmission power. A BS sleep control algorithm is also designed to reduce energy consumption by switching idle BSs to sleep mode during low-traffic periods. In addition, we introduce a traffic steering strategy to minimize latency by redistributing traffic load and dynamically selecting the best-serving BS for each UE.

These applications are mostly designed with deep reinforcement learning (DRL). Compared to traditional threshold-based or optimization-based methods, AI-enabled solutions deliver superior performance by learning from data and adapting to dynamic environments. These models collect environmental feedback, identify patterns, and make real-time decisions, allowing them to sustain optimal operation under varying network conditions.

Building on this foundation, our distributed implementations emphasize three key aspects. First, we use team learning and FL techniques to mitigate inter-agent conflicts and coordinate different network management applications. Second, we apply neural network compression and transfer learning (TL) methods to overcome resource limitations and enable on-device intelligence. Third, we address security risks, particularly those arising from malicious attacks in FL-enabled systems. Due to its decentralized nature, FL is more susceptible to attacks than centralized AI. Adversaries may manipulate local participants to submit poisoned model updates during global aggregation. These manipulations will be propagated to the global model and affect all nodes. For example, in a federated DRL-based BS sleep control scenario, an attacker could make the model select sleep actions during high-traffic periods, leading to increased delay, higher PDR, and lower EE. Thus, security concerns must be addressed to ensure FL only delivers benefits to the network performance. Motivated by these concerns, this thesis investigates both attacks and defenses in federated wireless systems. Studying potential vulnerabilities helps identify weak points in system design, and exploring robust defense mechanisms can safeguard both model integrity and end-to-end QoS.

To this end, Chapter 3 introduces two collaborative distributed AI frameworks for O-RAN resource allocation: one based on team learning [2], and the other on federated DRL under network slicing [3]. In Chapter 4, we introduce a knowledge transfer and FL-enabled UE-centric traffic steering framework for highly dynamic fifth-generation (5G) radio access network (RAN) [4]. In Chapter 5, we introduce our research on attacks and defense over federated DRL-enabled cell sleep control scenario in [5]. In Chapter 6, we discuss how to build more robust FL in mobile communication systems through dual-attention mechanisms [6].

### 1.1.2 Enhance intelligence and adaptability in wireless networks with GFM Integration

GAI and FMs are emerging AI technologies that have recently achieved groundbreaking advancements and attracted significant attention across various domains. GAI refers to a class of models and algorithms that are capable of generating new content or meaningful latent representations from complex input patterns. FMs, on the other hand, are large-scale ML models pre-trained on broad datasets and are designed to be adaptable and fine-tunable for a wide range of downstream tasks through fine-tuning or prompting.

As a combination of GAI and FMs, GFMs, such as LLMs, have demonstrated powerful emergent capabilities that enable unprecedented benefits for intelligent decision-making and automation. Compared with traditional AI techniques, GFMs offer several appealing advantages for mobile communication systems. First, LLMs are typically pre-trained with vast knowledge, enabling task adaptation via parameter-efficient fine-tuning or prompt-based methods such as ICL, without the need for training from scratch. Second, while traditional models depend heavily on high-quality and task-specific data, LLMs can often perform effectively with only a few examples, owing to their general intelligence obtained during pre-training. Third, LLMs provide improved explainability, as they are capable of generating semantic explanations for their decisions.

Despite these promising benefits, integrating LLMs into mobile communication systems remains in its early stages. Most pre-trained LLMs lack exposure to wireless signal data or communication protocols, limiting their effectiveness in domain-specific tasks. Moreover, their size and inference cost often exceed what can be supported on typical wireless hardware.

To address these challenges and bridge the domain gap, there is increasing interest in building AI models that follow the design principles of FMs, but are trained specifically for wireless communication tasks. One promising direction is to leverage representation learning strategies to train models that can extract general, transferable, and structurally meaningful representations from radio signals. In this way, the trained feature extractors can be reused across tasks and enable quick adaptation to new data types, such as by updating the downstream classifiers or fine-tuning part of the model, without retraining the entire network.

To this end, Chapter 7 introduces our proposed GAI-in-the-loop framework, which explores how GAI and LLMs can be effectively integrated into mobile communication systems [7]. We further present our research on ICL-enhanced automatic network intrusion detection, demonstrating that with ICL, pre-trained LLMs are equipped with relevant com-

munication knowledge and can enhance the performance and robustness of communication system functions such as intrusion detection [7]. In Chapter 8, we introduce a pre-trained LLM-empowered wireless traffic prediction framework and discuss how to select effective ICL examples to enhance the performance of LLM. Finally, Chapter 9 further delves into signal-level representation learning and introduces a SupCon framework that can extract class-discriminative features from raw I/Q signals and support uncertainty-aware modulation classification. These studies demonstrate how GAI models and techniques can be adapted and integrated into mobile communication tasks, laying the groundwork for more intelligent and generalized AI-enhanced communication systems.

## 1.2 Thesis Contributions and Organization

This thesis addresses key challenges in designing intelligent, adaptable, and secure next-generation mobile communication systems. We focus on enabling AI-driven applications for 5G and sixth-Generation (6G) networks, targeting the problems of decentralized network coordination and security concerns. To this end, we propose collaborative distributed AI frameworks that support learning and decision-making across decentralized nodes and decision agents. In addition, we study the vulnerabilities of distributed frameworks by examining potential adversarial attacks and proposing effective defense mechanisms.

Beyond conventional AI methods, we extend our research to cutting-edge AI techniques, including GFMs and LLMs, and demonstrate their applicability in wireless networking tasks. By systematically addressing the key challenges associated with these technologies, this work contributes both theoretical insights and practical solutions toward building more intelligent, automated, and adaptive mobile networks.

The major contributions of this thesis are summarized as follows.

### 1.2.1 Collaborative distributed AI-related contributions

**1) Mitigate conflicts between distributed AI-based applications in mobile networks**

In Chapter 3, we present our work [2,3] on designing DRL-based resource allocation strategies in O-RAN. Specifically, we develop two xAPPs: one for power control and another for radio resource allocation. To address potential conflicts arising from their coexistence, we

propose two collaborative frameworks: a team learning-based approach and a FL-based approach.

In the team learning framework, different xAPPs are treated as cooperative agents pursuing a common goal—maximizing system throughput. By enabling inter-agent information exchange, the framework mitigates control conflicts and improves coordination. In the FL-based framework, local DRL models are aggregated into a joint Q-table, which is then decomposed into local Q-tables for each xAPP, enabling decentralized but aligned decision-making.

These approaches enhance coordination among network functions. However, challenges such as limited local data and resource constraints persist, which are further addressed in Chapter 4.

### 2) Knowledge transfer and FL-enabled UE-centric traffic steering for 5G RAN

In Chapter 4, we propose a knowledge transfer and FL-enabled UE-centric traffic steering framework for 5G RAN [4]. This framework offloads part of the control intelligence to the UE side, enhancing scalability compared to traditional cell-centric schemes.

We use several innovative techniques to solve the challenges of implementing UE-centric traffic steering. We introduce a novel TL strategy using local expert models and expert Q-values. Additionally, we integrate grouping mechanisms and attention-based FL techniques to improve convergence and robustness in the distributed environment.

The proposed method effectively improves the service quality of communication. However, its distributed nature introduces new security vulnerabilities, which are analyzed and mitigated in Chapters 5 and 6.

### 3) Attacks and defense over federated DRL-enabled cell sleep control scenario

In Chapter 5, we investigate adversarial threats in a federated DRL-based cell sleep control scenario [8]. In this setting, BSs collaboratively learn sleep control and energy-saving strategies using FL, while some participants are assumed to be malicious.

We examine both regular and intelligent poisoning attacks and analyze their impact on key QoS metrics. To mitigate these attacks, we propose two defense mechanisms: an autoencoder-based anomaly detector that can identify the malicious attackers, and a knowledge distillation (KD)-based defense mechanism that controls the knowledge transferred between the global model and local models.

Inspired by this study, we observe that certain types of attackers have similarities with system stragglers. Thus, the defense strategies not only improve security but also accelerate convergence. This topic is further explored in Chapter 6.

#### 4) Dual attention model-weighted robust FL in mobile communication systems

In Chapter 6, we propose a robust FL framework incorporating dual attention mechanisms to enhance model performance both in a secure system and under attacks [6]. The framework assigns dynamic aggregation weights to each participant using two attention models. The first performs self-attention across participants to capture inter-model relationships, while the second evaluates temporal alignment between global and local models to assess training progression.

We validate the approach on a modulation classification task, demonstrating improved accuracy, faster convergence, and greater robustness compared to conventional FL frameworks. The methodology is generalizable to a variety of wireless communication applications.

### 1.2.2 GFM-related contributions

#### 5) GAI-in-the-loop in mobile communication systems

In Chapter 7, we introduce the concept of "GAI-in-the-loop", which leverages the semantic understanding, contextual reasoning, and generalization capabilities of LLMs to support complex and unforeseen tasks in mobile communication systems. This design enables LLMs to assist in decision-making scenarios that are typically beyond the reach of conventional ML models.

We show that integrating LLMs with traditional ML techniques results in performance gains by combining the advanced capabilities of LLMs with the task-specific accuracy of ML models. To support this idea, we explore how to effectively combine ML and LLM-based modules in communication systems and explain several potential use cases aligned with the requirements of next-generation networks.

Furthermore, we propose a pre-trained LLM-based intrusion detection framework that implements a fully automated pipeline [9]. This includes LLM-based input information selection, prompt building, in-context learning, and output conversion. The framework demonstrates the feasibility and effectiveness of deploying LLMs in wireless network-related tasks.

**6) Pre-trained LLM-empowered mobile traffic prediction**

In Chapter 8, we continue to focus on integrating ICL techniques into wireless system design and propose an LLM-based framework for mobile traffic prediction [10]. A key contribution in this chapter is to propose a novel two-step demonstration selection strategy that can improve ICL effectiveness.

Our findings show that ICL enables accurate traffic prediction without requiring model fine-tuning. The pre-trained LLMs are capable of generalizing across diverse traffic scenarios, effectively utilizing demonstrations even when the testing context differs from the training data. This generalizability reduces reliance on scenario-specific data of traditional data-driven AI models.

**7) Supervised Contrastive Learning for Uncertainty-aware Wireless Signal Analysis**

In Chapter 9, we delve into signal-level representation learning and propose an uncertainty-aware automatic modulation classification (AMC) framework based on SupCon. We first train a ResNet-based SupCon model to learn class-discriminative representations from raw I/Q signals. These embeddings are then used to train class-wise binary classifiers that not only perform classification but also estimate sample-wise uncertainty.

To enhance the reliability of AI-based predictions, we incorporate a rejection mechanism based on the output uncertainty scores that allows the model to reject uncertain decisions. This approach improves robustness under noisy conditions and for OOD data, and helps ensure trustworthy decision-making in wireless environments.

### 1.2.3 Summary

The contributions above reflect a progressive advancement in AI integration within mobile communication systems. Chapters 3–6 focus on AI-enabled control tasks, distributed collaboration, and security in FL. Chapters 7–9 advance into the generative AI models and techniques, demonstrating how pre-trained GFMs and GAI techniques can be integrated into wireless network applications for enhanced intelligence and adaptability.

## 1.3 Publications

### 1.3.1 Journal Publications

**[J01]** **H. Zhang**, H. Zhou, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and Melike Erol-Kantarci, "On-Device Intelligence for 5G RAN: Knowledge Transfer and Federated Learning Enabled UE-Centric Traffic Steering," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 2, pp. 689-705, Dec 2023.

**[J02]** **H. Zhang**, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "Cooperation and Personalization on a Seesaw: Choice-based FL for Safe Cooperation in Wireless Networks," *IEEE Wireless Communications Magazine* vol. 32, no. 2, pp. 172-179, Apr. 2025.

**[J03]** **H. Zhang**, H. Zhou, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "Intelligent Attacks and Defense Methods in Federated Learning-enabled Energy-Efficient Wireless Networks," *IEEE Transactions on Wireless Communication* (Early Access), pp. 1-16, Apr. 2025.

**[J04]** **H. Zhang**, A. Bin Sediq, A. Afana and M. Erol-Kantarci, "Mobile Traffic Prediction using LLMs with Efficient In-context Demonstration Selection", *IEEE Transactions on Communication*(Early Access), pp. 1-16, Jun. 2025.

### 1.3.2 Conference Publications

**[C01]** **H. Zhang**, A. Bin Sediq, A. Afana and M. Erol-Kantarci, "Large Language Models in Wireless Application Design: In-Context Learning-enhanced LLM-based Automatic Network Intrusion Detection," (Accepted to 2024 IEEE Globecom), pp.1-6, Dec. 2024.

**[C02]** **H. Zhang**, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and Melike Erol-Kantarci, "Federated Learning with Dual Attention for Robust Modulation Classification under Attacks," in Proceedings of IEEE ICC, pp.5238-5243, Jun. 2023.

**[C03]** **H. Zhang**, H. Zhou, Melike Erol-Kantarci, "Federated Deep Reinforcement Learning for Resource Allocation in O-RAN Slicing," in Proceedings of IEEE Globecom, pp. 958-963, Dec. 2022.

**[C04]** **H. Zhang**, H. Zhou, Melike Erol-Kantarci, "Team learning-based resource allocation for open radio access network (O-RAN)," in Proceedings of IEEE ICC, pp. 4938-4943, Jun. 2022.

**[C05]** **H. Zhang**, H. Zhou, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and Melike Erol-Kantarci, "Distributed Attacks over Federated Reinforcement Learning-enabled Cell Sleep Control," in Proceedings of IEEE Globecom Workshops, pp. 485-490, Dec. 2023.

### 1.3.3 Patents and Invention Disclosures

**[P01]** **H. Zhang**, A. Bin Sediq, A. Afana and M. Erol-Kantarci, "In-Context Learning-enhanced LLM-based Automatic Network Intrusion Detection", US provisional patent filed on 1 April 2024.

**[P02]** **H. Zhang**, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "Choice-based FL for Safe Cooperation in wireless networks", US provisional patent filed on 22 March 2024.

**[P03]** **H. Zhang**, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "Dual Attention model-based Federated Learning", US provisional patent filed on 9 November 2023.

**[P04]** **H. Zhang**, H. Zhou, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "Attacks and defense in federated reinforcement learning-based wireless networks", US provisional patent filed on 5 May 2023.

**[P05]** **H. Zhang**, H. Zhou, M. Erol-Kantarci, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, "System and Method for UE-centric Traffic Steering for 5G RAN", US provisional patent filed on 18 April 2023.

# Chapter 2

# Background and Literature Review

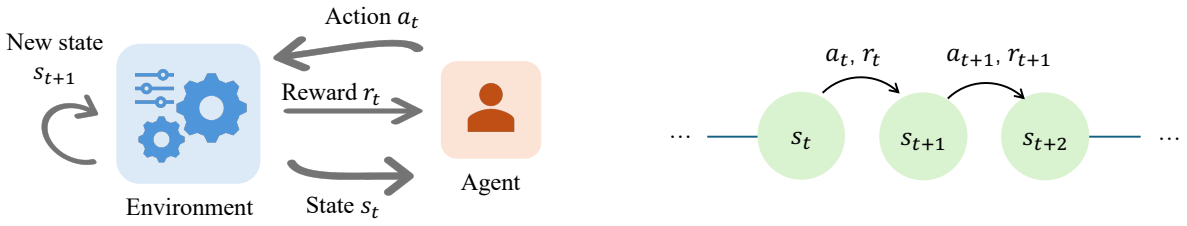
In this section, we first introduce the background of this research. We start with an introduction to AI techniques that have been used in our research, such as DRL, FL, TL, GAI, LLMs and so on. Next, we investigate existing studies on how AI techniques accelerate the next-generation networks and summarize related works in the literature review.

## 2.1 Background

### 2.1.1 Reinforcement learning and deep reinforcement learning

This section introduces the fundamentals of RL and DRL, with a focus on their application to sequential decision-making tasks.

RL is a class of AI algorithms that solve the problem of how agents perform actions in an environment, which is shown in Fig. 2.1a. The RL algorithms usually contain the following elements: agent, action, state, environment, and reward. The agent is the decision-making entity that interacts with the environment by taking actions. Action refers to the behaviours that an agent can perform on the environment in a given state. The state describes the information that the environment currently contains. Rewards provide scalar feedback indicating the immediate utility of an action taken in a given state. During the RL, agents will interact with the environment, choosing actions and receiving the corresponding immediate rewards. For a given task, the agent will develop




(a) A schematic of the RL process.

(b) Model RL as Markov decision process (MDP).

Figure 2.1: An illustration of RL.

an action selection strategy based on its experience of interactions and learn to choose optimal actions that can maximize the long-term cumulative rewards.

The RL framework can be modeled as a MDP, which is shown in Fig. 2.1b. At each time $t$, the agent will select an action $a_t$ based on the current state $s_t$. Then the state will change to $s_{t+1}$ and the agent will receive a reward $r_t$. According to the Markov property, the selection of $s_{t+1}$ only depends on the previous state $s_t$ and the action $a_t$ and is irrelevant to earlier actions and states. The process can be described with the state-transition function $P(s_{t+1}, r_t|s_t, a_t)$, which stands for the probability of getting the next state $s_{t+1}$ and reward $r_t$ under the given current state $s_t$ and action $a_t$. The policy function $\pi$ describes the probability that the agent chooses action $a$ in state $s$, which can be formulated as:

$$\pi(a|s) = P(a_t = a, s_t = s) \tag{2.1}$$

Two functions are defined to help the agent update the model: the state value function and the action value function. The state value function can be formulated as:

$$V_\pi(s) = E_\pi[G_t|s_t = s] \tag{2.2}$$

where $E[.]$ denotes the expectation. $G_t$ denotes the discounted cumulative reward, which can be formulated as:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{n} \gamma^k r_{t+k}, \tag{2.3}$$

where $\gamma$ denotes the discount factor. The state value function shows the cumulative future reward the agent can obtain from the current state $s_t = s$ following the policy $\pi$. Similarly, the action value function can be formulated as:

$$Q_\pi(s, a) = E_\pi[G_t|s_t = s, a_t = a] \tag{2.4}$$

This function shows the cumulative future reward the agent can obtain if action $a_t = a$ is selected under state $s_t = s$. The relationship between the action value function and the reward value function can be written as:

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) * Q_\pi(s, a) \tag{2.5}$$

$$Q_\pi(s, a) = r(s, a) + \gamma * \sum_{s' \in S} P(s_{t+1} = s'|s_t = s, a_t = a) V_\pi(s') \tag{2.6}$$

With these two functions, the Bellman Equation can be formulated as:

$$V_\pi(s) = E_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})|s_t = s] \tag{2.7}$$

$$Q_\pi(s, a) = E_\pi[r_t + \gamma max Q(s_{t+1}, a_{t+1})|s_t = s, a_t = a] \tag{2.8}$$

These equations reflect the relationship between the value of a state or state-action pair and the expected rewards and serve as a theoretical basis for the RL algorithm design.

There are two kinds of RL models, value-based and policy-based. Value-based RL methods, like Q-learning, estimate values for different states and actions, and use these values to guide policy. In contrast, policy-based RL methods, like policy gradient, directly optimize the policy by maximizing the reward. Considering the application scenario, in our research, we mainly focus on value-based RL methods.

Q-learning is a common value-based RL method that constructs a Q-table to store the expected long-term reward of different states and actions. The iterative training of Q-learning models consists of the following steps: choosing an action $a_t$, applying the action to the environment, measuring immediate reward, and updating the Q-table. Q-values are iteratively updated using the Bellman optimality equation as shown in Eq. 2.8. On this basis, a hyperparameter, learning rate $\alpha$, is introduced into the Q-learning algorithm to control the step size of parameter updates during learning, which is formulated as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.9}$$

Another hyperparameter $\epsilon$ is introduced to balance the exploration and the exploitation of the Q-learning algorithm. During the iterative training, the agent performs random action selection at a probability of $\epsilon$ for exploration and chooses the action with the highest Q-value at a probability of $1 - \epsilon$ to maximize the reward.

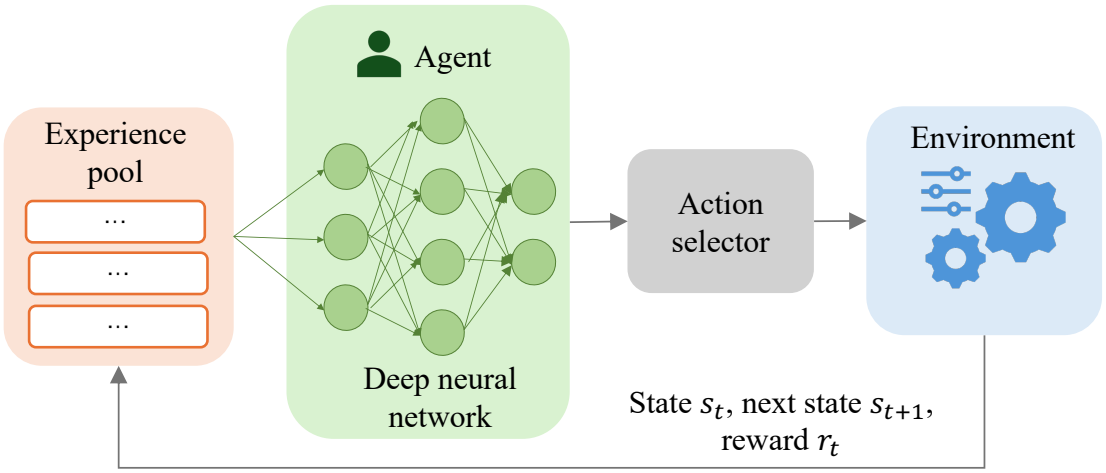



Figure 2.2: The structure of DQN.

Despite its effectiveness, Q-learning suffers from a long convergence time when faced with a huge state-action space, and consequently, DQN is proposed. The structure of DQN is shown in Fig. 2.2.

DQN uses Deep Deep Neural Network (DNN) to estimate the Q-values and updates the parameters of DNNs with a stochastic gradient descent algorithm. The parameters of the DNN can be updated as:

$$\theta_{t+1} = \theta_t + \alpha[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_t) - Q(s_t, a_t; \theta_t)]\nabla Q(s_t, a_t; \theta_t) \tag{2.10}$$

where $\theta_t$ denotes the parameters of DQN at time $t$. $\nabla Q(s_t, a_t; \theta_t)$ is the gradient. During the simulation, the current state, action, next state and reward are recorded as $< s_t, a_t, s_{t+1}, r_t >$. The recorded data is used for the experience pool of DQN training.

### 2.1.2 Federated learning

In this section, we introduce the FL technique and the federated RL technique. FL is an emerging approach in ML, which refers to multiple partners training models separately in a decentralized but collaborative manner to build a shared global model while maintaining data privacy. We first introduce the most common Federated Averaging (FedAvg)-based FL algorithms [11] and subsequently introduce FL algorithms in variant forms. A generalized structure of FL is shown in Fig. 2.3 [12].

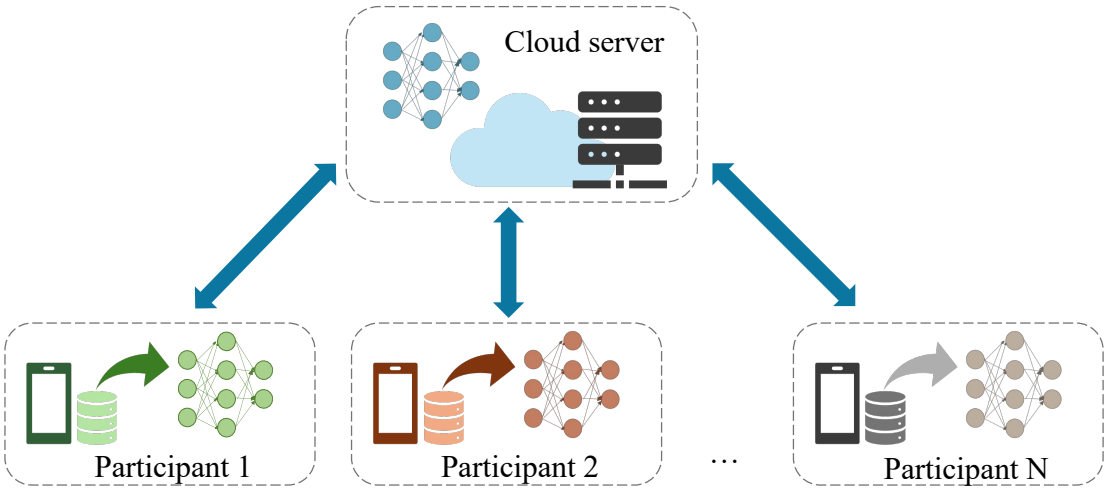



Figure 2.3: The structure of FL.

FL usually contains two parts, a global cloud server and several distributed clients. The global server holds a global model while each of the local clients holds a local model. The FL process can be concluded as two main steps. The first step is the local training and updates. In this step, each UE trains a local model with its private dataset by minimizing the local loss function. The local model update can be formulated as:

$$\theta_{k,t+1} = \theta_{k,t} - \alpha \frac{\partial L_k^{Local}(\theta_k)}{\partial \theta_k} \tag{2.11}$$

where $\theta_{k,t+1}$ denotes the local model parameters of the $k^{th}$ client at the $t+1$ iteration and $\alpha$ denotes the learning rate.

In the second step, the local models parameters are sent from distributed clients to the global server, and the global model is updated by the local models parameters. The optimization goal of the global model can be formulated as:

$$\min_{\omega} \{L^{Global}(\omega) = \sum_{k=1}^{K} L_k^{Local}(\omega_k)\} \tag{2.12}$$

In FedAvg, the global model parameters can be calculated through model averaging, which can be formulated as:

$$\omega_{G,t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{k,t} \tag{2.13}$$

where $\omega_{G,t+1}$ denotes the global model parameters of the $k^{th}$ client at the $t+1$ iteration. $n_k$ denotes the number of samples in the local dataset of the first client. $n$ denotes the number of samples in the local datasets of all the clients. After the global aggregation, the parameters of the global model are sent back to the clients to accelerate the local training.

On the basis of FedAvg, some other FL algorithms are proposed to improve the performance on clients with a non-independent and identically distributed (non-IID) data distribution. For instance, the Federated Proximal (FedProx) algorithm balances the impact of personalized local model parameters and shared global model parameters during local model updates by adding a proximal term to the local loss function [13]. This mitigates the negative impact of non-IID data and device heterogeneity, and improves model performance.

Federated RL is proposed as a combination of FL and RL, aiming to apply the idea of FL to distributed RL models to enhance the collaboration among multiple RL agents [14]. According to environment partitioning, federated reinforcement learning can be divided into two types, horizontal federated reinforcement learning (HFRL) and vertical federated reinforcement learning (VFRL) [15]. In HFRL, different agents have similar action and state spaces, and they act independently in various environments. The purpose of federating is to accelerate the exploration of the environment by experience sharing. In VFRL, all the agents act in the same environment. The state and action space of agents can be heterogeneous, and their actions collaboratively change the environment and influence the reward.

### 2.1.3 Transfer learning and transfer reinforcement learning

TL, or knowledge transfer, is a ML technique that can transfer the knowledge from a pre-trained model to a new model to solve similar tasks. The task performed by the pre-trained model is called the source task, and the new task is called the target task. Typically, training a new ML model is a time-consuming and complex process that requires a large amount of data and a long training time to ensure that the model is well-trained and ready for use. Instead of training from scratch, the knowledge transfer technique allows the training of new models to reuse knowledge learned from a source task [16].

When TL is applied in reinforcement learning settings, it gives rise to the field of transfer reinforcement learning (TRL). As illustrated in Fig. 2.4, TRL addresses key challenges in RL, such as high sample complexity, delayed or sparse rewards, and inefficient learning caused by repeated exploration. However, TL in a RL setting is more challenging than TL in a supervised learning setting because knowledge needs to be transferred with the MDP.

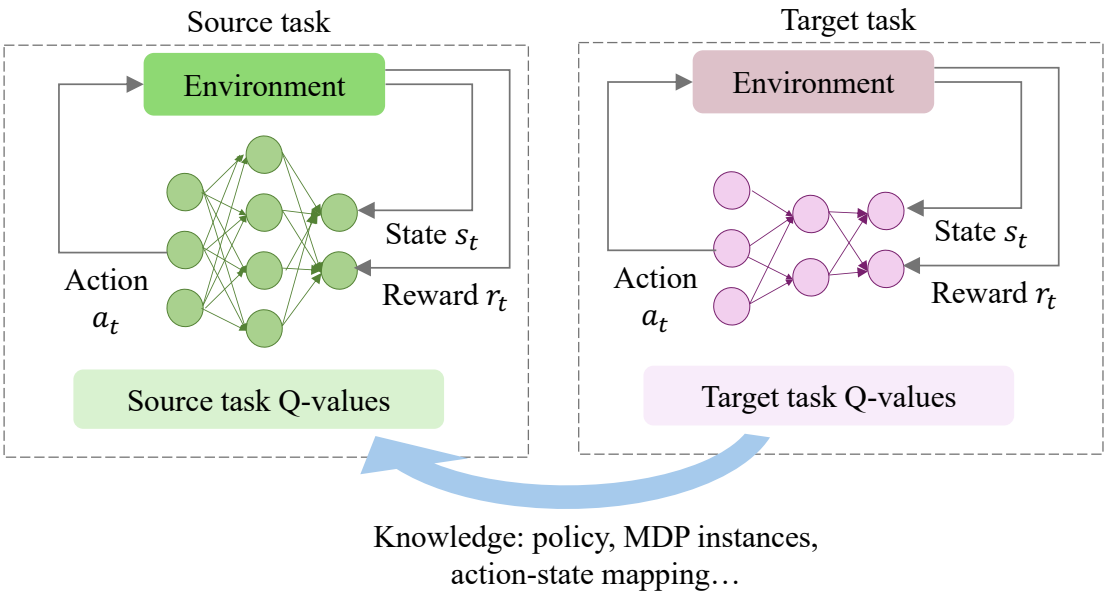



Figure 2.4: An illustration of TRL.

Considering the different components in the MDP, knowledge can be presented in different forms and needs to be transferred in different ways [17]. The transferable Knowledge includes low-level knowledge, such as the MDP instances $< s_t, a_t, s_{t+1}, r_t >$, the action value function and the policy. It also includes high-level knowledge, such as action selection experience in the given situations and important features for learning and shaping rewards [18]. Other than that, TRL algorithms can also be divided into three situations: knowledge transfer in the same state action space, knowledge transfer in a different state action space and knowledge transfer in multi-agent reinforcement learning (MARL). For different situations, the TRL can be performed in different ways.

Other than TRL, some other related AI techniques can play a similar role, including lifelong learning [19], imitation learning [20], and hierarchical reinforcement learning (HRL) [21]. Lifelong learning means using only one network structure trained separately on different tasks to make the network capable of all the tasks. It can be seen as a special form of transfer learning, where knowledge is only transferred from past observed tasks to future tasks. Meanwhile, it is also more challenging and requires the model to learn new knowledge without forgetting previous knowledge. Imitation learning means agents can learn to perform a task by watching another agent or human with similar abilities. HRL groups the action space at different levels to develop high-level actions and low-level

actions. With the high-level abstraction on tasks, actions, and state spaces, HRL can facilitate knowledge transfer across similar domains.

### 2.1.4 Generative AI and foundation models

Table 2.1: A comparison between the definition and capabilities of LLM, Generative AI, and FM.

| Concepts | Definition | Capabilities | Examples |
|---|---|---|---|
| LLM | Transformer-based machine learning models that can comprehend and generate human language text [22]. | Multi-modal information understanding and generation, emergent capabilities, reasoning and so on | Generative pre-training transformer (GPT) [23, 24], Pathways language model (PaLM) [25], Large language model Meta AI (LLAMA) [26] |
| Generative AI | A class of algorithms and models within AI and NLP designed to generate new, previously unseen data rather than distinguishing between existing categories as discriminative models [27]. | Multi-model content generation, fraud detection, recommendations | Generative adversarial network (GAN) [28], Variational autoencoder (VAE) [29], Autoregressive Models, Diffusion Models |
| FM | A series of models trained on broad data that can be adapted to a wide range of downstream tasks, making them different from task-agnostic models [30]. | Emergence, homogenization, and domain adaption, serves as the common basis | GPT [23, 24], Bidirectional encoder representations from transformers (BERT) [31], DALL-E [32] |

GAI, FMs, and LLMs are three related but distinct concepts, as summarized in Table 2.1. GAI refers to a class of AI algorithms designed to generate various forms of content such as images, videos, and text [33]. FMs are large-scale models pre-trained on broad datasets and designed to be fine-tuned or prompted for various downstream tasks. LLMs are transformer-based models specialized in understanding and generating human language text, and they represent the intersection of GAI and FMs.

The emergence of transformer architecture has dramatically accelerated the progress of GFMs and GAI [34]. Transformer is a neural network architecture based on the multi-head attention mechanism [35]. It usually consists of alternating attention and feed-forward layers to perform repeated transformations on the vector representations and extract latent information. The introduction of the attention mechanism in the transformer architecture allows elements in the input sequence to be weighted based on their importance and enables the model to capture long-range dependencies.

In addition, beneficial from this architecture, transformers exhibit scaling laws [36], showing that the model performance can expect a predictable increase when the model size, dataset size, and compute budget increase. The scaling law of the transformer ensures its ability to expand the size of the model to allow generalization over ultra-large

datasets. This insight motivates the development of large-scale transformer-based models pre-trained on massive datasets to acquire emergent generalization capabilities on a series of downstream tasks, which motivated the emergence and popularity of GFMs. In the following part of this section, we first introduce the attention mechanism and then introduce how the transformer architecture is designed.

The attention mechanism usually includes three vectors: the key vector $K$, the query vector $Q$ and the value vector $V$. In the transformer architecture, the scaled dot-product attention is adopted, and it can be calculated as:

$$Attention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d_k}}) \cdot V, \tag{2.14}$$

where $d_k$ denotes the dimensionality of the key vector. In this formulation, the dot product of the key vector $K$ and the query vector $Q$ is first calculated to measure the similarity between these two vectors as the attention scores. Next, the square root of the dimension of the key vector $K$ is used to scale and control the range of the attention scores. Then, a SoftMax function is used for normalization, and it turns the attention scores into attention weights. Finally, the value vector $V$ is weighted and summed using the attention weights to get the final output.

The transformer architecture is shown in Fig. 2.5. It includes two parts: the encoder part and the decoder part. The whole architecture is built by stacking attention layers and feed-forward layers. Both the encoder and decoder consist of six identical layers, with the decoder including an additional encoder-decoder attention sub-layer.

Moreover, the multi-head attention mechanism is introduced to allow the model to process information represented at different positions. In this mechanism, the key vector $K$, the query vector $Q$ and the value vector $V$ are projected $h$ times in different dimensions with different parameter matrices $W_i^Q$, $W_i^K$ and $W_i^V$. After projections, new vectors are generated and used to calculate multiple attention weights through single self-attention models. These attention weights are then concatenated and projected to the final attention weight values with a parameter matrix $W^o$. This process can be described as:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^o, \tag{2.15}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V). \tag{2.16}$$

The multi-head attention mechanism effectively improves the ability of models to capture information from different locations over long distances and learn global information.

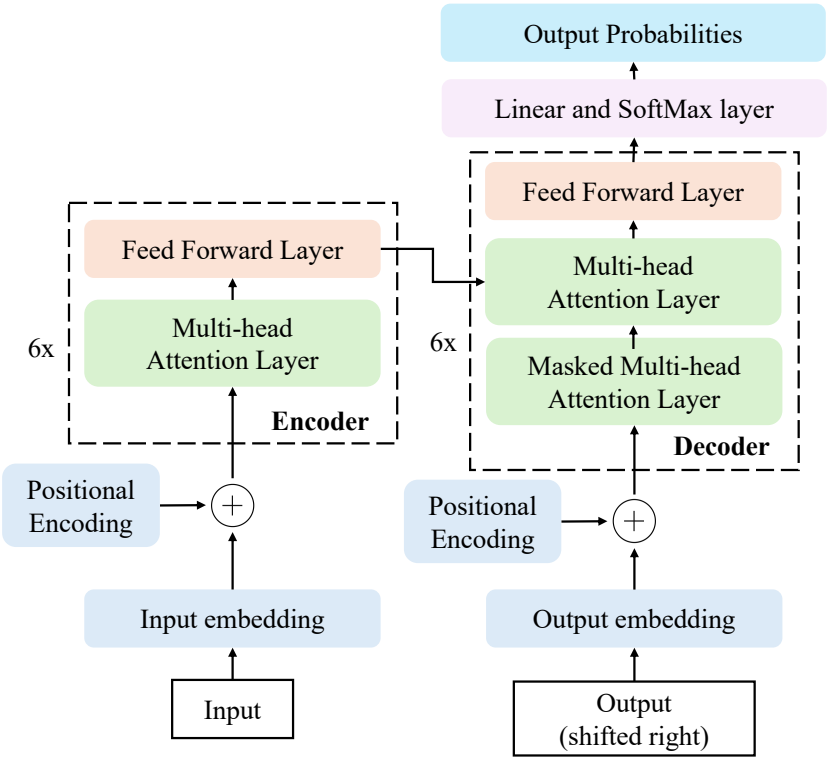



Figure 2.5: The transformer model architecture.

The feed-forward layers in the transformer architecture are composed of the fully connected layers and the Rectified Linear Unit (ReLU) activation, which can be given as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2. \tag{2.17}$$

In addition, positional embedding is introduced to capture positional information of the sequence at the bottoms of the encoder and decoder. The positional embedding can be performed in different ways, either learned or fixed [37].

Built on the transformer structure, GFMs shows two major characteristics: emergence and homogenization [30]. Emergence means that the model behavior is induced from training on diverse and extensive data rather than being explicitly constructed. It makes GFMs adaptable task-solvers that can perform tasks beyond the scope of training objectives. Homogenization means a GFM is a single generic learning algorithm that can handle a wide range of applications. The same model can be repeatedly used for fine-tuning and training for downstream tasks, as they are effective across many tasks and too cost-intensive

to train individually. Given the emergence characteristic, some GFMs exhibit numerous important emergent capabilities that are absent in smaller AI models.

LLMs are a type of transformer-based GFMs that have significant size and are pre-trained on vast amounts of data [22]. Given the extremely large size, LLMs demonstrate some capabilities that are not seen in traditional machine learning models, including basic abilities like world knowledge, coding, multi-lingual understanding, the emerging abilities like instruction following, ICL and reasoning, and the augmented abilities like tool utilization and self-improvement [38]. With these capabilities, LLMs have demonstrated near-human-level performance in cognitive tasks across various fields, but they also bring some challenges and concerns.

Building a LLMs includes several steps. First, a dataset needs to be prepared and filtered for model training. Next, the tokenization method and the positional encoding method are designed to break down text into smaller sub-word units and capture the positional information. After that, the LLMs architecture is designed. It can be decoder-only, encoder-only or encoder-decoder-based architectures. Training the LLMs includes two steps: a pre-training step and a fine-tuning step. Next, an alignment step is performed to enhance the safety and reliability of the LLMs. Finally, the decoding strategies are designed to help LLMs generate desired outputs.

In some use cases, domain adaptation needs to be performed on pre-trained LLMs to increase its knowledge and capabilities for a specific area. Some cost-effective techniques have been designed for domain adaptation [39], including prompt tuning, Retrieval augmented generation (RAG) and Low-Rank Adaptation (LoRA). Prompt tuning means the domain adaption can be realized by including proper instructions or examples in the prompt to guide LLMs. RAG means adding additional steps to retrieve relevant content from the prompt as additional knowledge for LLMs. LoRA means freezing most of the model weights and applying fine-tuning to only a small separate set of weights. These techniques help LLMs meet specific needs and empower them to evolve and address diverse challenges.

## 2.2 Literature Review

In this section, we investigate related works on how AI techniques accelerate the next-generation model communication system. Specifically, we studied how to design AI techniques-empowered xAPPs, how to mitigate the security concerns of FL techniques, and how to integrate the latest AI techniques, GAI, and LLMs into wireless networks.

### 2.2.1 AI-empowered network management applications

In existing studies, AI techniques have been widely used to improve the performance of wireless network management applications. There are usually two important types of tasks in wireless network management: control tasks and prediction tasks. Control tasks involve dynamic resource management and configuration optimization to achieve reliable connectivity, high QoS, and energy efficiency [40]. Prediction refers to the tasks that use data-driven techniques to forecast future information on wireless networks. The predicted network conditions, user behaviors, and resource demands can be used to improve the network performance [41]. AI techniques can be applied to both types of tasks to develop intelligent prediction and decision-making schemes for wireless network management.

#### 1) Control tasks

For control tasks, RL techniques are widely used to select optimal actions under given states. Intelligent agents can learn strategies through RL, and the process includes four key components: observation, exploration, learning, and interaction with the environment. Critical control tasks in wireless networks include resource allocation, traffic steering and cell sleep control.

RL techniques have been widely applied to resource allocation problems involving power, radio, cache, and computation resources, which can be managed independently or jointly optimized to achieve global performance objectives [42–46]. Commonly used RL algorithms include Q-learning, correlated Q-learning and DQN. In these designs, the state usually includes network information such as the traffic load and the channel state information, and the action is usually set as allocating resources among different UE or network slices. With RL, the network controller can adapt to the dynamic wireless communication environment and be competent in complicated tasks.

Traffic steering enhances transmission reliability by distributing network loads to appropriate BSs. Existing works largely focus on cell-centric frameworks, where BSs centrally make decisions for all UEs, often overlooking individual user needs. In [47], a centralized DRL framework is designed to decide the individual cell offset to control the handover of UEs. [48] studied how to offload a mobile UE from macro cells to small cells and derived optimal offloading strategies. Meanwhile, some other existing works investigate UE-centric frameworks with ML techniques. [49] studies how to design a UE-centric network selection framework in ultra-dense heterogeneous networks by choosing the optimal radio access technology (RAT) according to the link state prediction. [50] provides a fully distributed traffic steering approach running on distributed devices and optimizes each UE's received

signal strength and energy consumption. In [51], distributed UEs choose their cooperative BS according to the received signal interference noise ratio (SINR) and predicted load of BS. [52] designed a UE-based distributed traffic steering mechanism between Wi-Fi networks and Long-Term Evolution (LTE) with Q-learning.

Cell sleep control aims to save the energy costs of the communication system by adopting different sleeping modes for BS cells. The objective of the sleep control application is usually to achieve high energy efficiency. For example, in [53], an online RL-based traffic adaptive sleep mode control algorithm is proposed to decide which sleep mode to choose given time and BS load. Compared to traditional optimization-based sleep control methods, AI-empowered cell sleep control can adapt online by updating policy from experience and execute quickly during inference.

### 2) Prediction tasks

Prediction tasks aim to forecast future network states, such as traffic load, channel conditions, or user behavior, based on historical data. A critical focus of these tasks lies in how to effectively handle the time-series data. Wireless traffic prediction is a fundamental prediction task in wireless networks. Accurate traffic prediction enhances the performance of core network functions such as resource allocation, traffic steering, and load balancing. Some studies have used AI techniques to learn traffic patterns from historical data and developed AI-based wireless traffic prediction applications. For instance, [54] developed a deep learning (DL) model for real-time downlink bandwidth prediction and handover prediction in a 5G communication network. In [55], a meta-learning scheme was proposed to perform short-term predictions of user traffic intensity. [56] proposed a graph-based temporal convolutional network to predict the future traffic of each network unit in a wireless network. In [57], the long short-term memory (LSTM) model was applied to the one-step wireless traffic prediction. While earlier works mainly relied on LSTM models and convolutional networks, transformer-based architectures have recently gained attention for their ability to model long-range temporal dependencies. For instance, [58] used a transformer architecture to predict wireless network traffic in concise temporal intervals for open radio access networks. [59] designed transformer blocks to model the temporal and spatial features of traffic flows in communication networks.

Other than wireless traffic prediction, there are also some other prediction tasks in mobile networks, such as CSI prediction, radio link failure prediction, blockage prediction and resource usage prediction for network slices. The objectives and AI methods being used are summarized in Table 2.2.

Table 2.2: AI-empowered other prediction tasks in mobile networks

| Task | AI Methods | Objective |
|---|---|---|
| Channel state information (CSI) Prediction | CNN [60], LSTM [61], Transformer [62] | Predict future CSI to enable accurate beamforming and reduce feedback overhead. |
| Radio Link Failure Prediction | Graph neural network (GNN) and Time-Series Transformer [63], LSTM and Support vector machine (SVM) [64] | Detect potential radio link failures in advance to improve network robustness and reduce service interruption. |
| Blockage Prediction | Gated recurrent unit (GRU) [65], Vision transformer (ViT) [66] | Predict communication link blockages to enable proactive handover or beam switching. |
| Network Slice Resource Usage Prediction | Autoregressive integrated moving average (ARIMA) [67], LSTM [68] | Predict resource usage within network slices to proactive allocate resources. |

Although these applications demonstrate good performance in their respective tasks, they are often assumed to run independently, and the conflicts between different control agents are not taken into consideration. In practical network environments, however, multiple AI-driven control applications are frequently deployed in parallel, sharing the same network infrastructure and resources. This co-existence can lead to unexpected conflicts that degrade overall system performance and hinder the reliability of AI-enabled network automation.

On the other hand, the O-RAN architecture introduces a modular and open framework for deploying intelligent control applications in mobile networks [69]. It has become a promising technology that provides an open environment for network vendors and operators. An illustration of the O-RAN architecture and xAPPs under a network slicing environment is shown in Fig. 2.6. One of the main purposes of O-RAN is to improve RAN performance and supply chain by adopting open interfaces that allow multiple vendors to manage the intelligence of RAN. Specifically, the O-RAN system introduces two radio intelligent controllerss (RICs), non-real-time RIC and near real-time RIC. These two RICs serve as platforms to host AI-powered applications called xAPPs and RAN Applications (rAPPs). Each xAPP and rAPP can be considered as an independent agent to control a specific network function.

The coordination among different xAPPs and rAPPs remains a big challenge in O-RAN [70]. This motivates our work on designing collaborative AI frameworks under O-

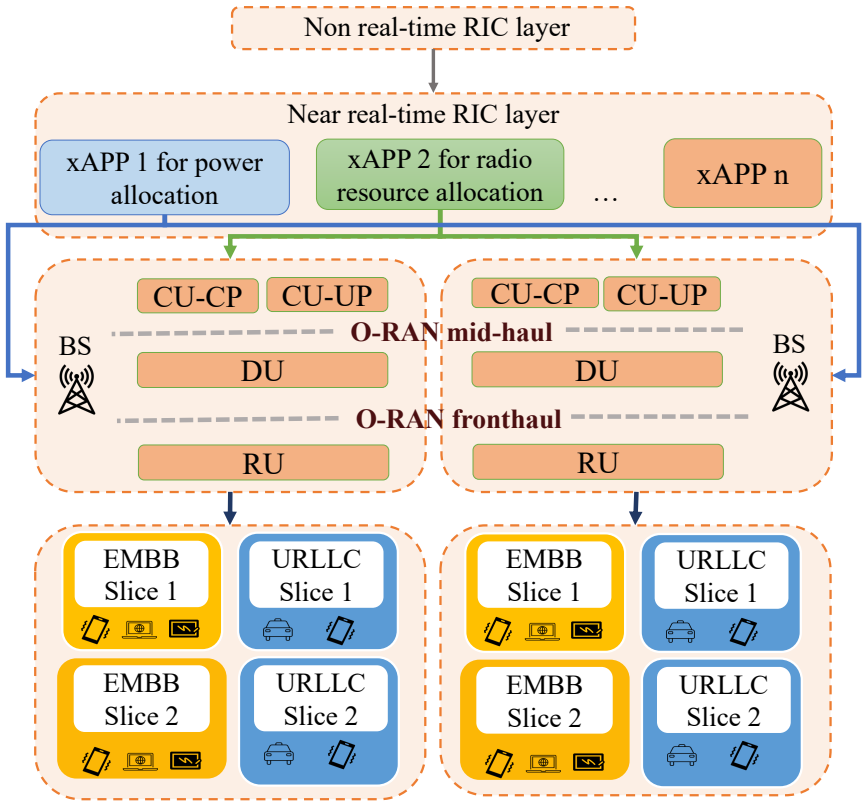



Figure 2.6: O-RAN architecture and xAPPs.

RAN architecture and mitigating conflicts among xAPPs.

### 2.2.2 Distributed ML in mobile networks and security concerns

The distributed deployment approach has been adopted for a variety of wireless network applications to handle the increasingly diversified service demands and to safeguard the privacy and data security of users in the communication system [71]. Some AI techniques, such as MARL, FL and team learning, can be leveraged to enhance the performance of distributed RAN applications.

MARL applies to the communication systems where more than one agent appears and operates controls at the same time. Each agent discovers its best strategy according to its local observations using its local RL model. MARL can be applied to scenarios such as resource allocation for unmanned aerial vehicle (UAV) networks [72], user association [73]

and anti-jamming communications [74].

Team learning is a distributed AI technique that applies to scenarios in which distributed agents in the communication system form teams that cooperate to reach a common goal of maximizing the team's total reward. It can be leveraged to mitigate the conflicts between independent decision-making by agents [2].

FL is a distributed model training technique that can make multiple decentralized participants train models separately in a collaborative manner and build shared models without collecting local data to a centralized controller.

FL has been used for collaborative training of different ML models, including supervised learning, unsupervised learning and RL. federated reinforcement learning (FRL) has been used to collaboratively train control applications in wireless networks. For instance, in [75], the authors leveraged DRL to allocate heterogeneous resources and perform computation offloading in the 5G ultra-dense network scenarios, and the RL model is trained in a decentralized way with FL. In [76], a hybrid federated reinforcement learning- based structure is proposed for device association in RAN slicing with parameter aggregation on two levels. In [77], a federated reinforcement learning algorithm is used to control the access of users in O-RAN, where RL is used for access decision making on single users and FL is used to aggregate models from different users. In these works, FL is mainly adopted as a solution for the federation between agents performing the same network functions, which often have similar state and action spaces, without considering how different network functions and agents are federated.

The benefits of applying FL to wireless networks are summarized below:

- Privacy: Different from centralized learning methods, FL participants do not upload raw data to the global servers. They only upload model parameters, thus avoiding data leakage.
- Scalability: As a distributed learning framework, FL can handle large-scale data by leveraging on-device computational and memory resources.
- Communication efficient and time-saving: FL trains models locally and skips the step of collecting and aggregating data from diverse sources. This cooperative pattern improves communication efficiency by avoiding data transfer and saves model training time.
- Robustness: The adversarial robustness in FL can be propagated from rich-resource participants to those with poor resources, thus improving the general performance of the system [78].

- Diversity and inclusiveness: FL allows the use of data that cannot be collected directly as part of the training dataset. It helps reduce bias and develop a more representative and inclusive model [79].

Despite the benefits mentioned above, FL algorithms also suffer from several concerns. Key challenges associated with FL are included as follows:

- Fairness among participants: Participants involved in FL usually carry varying amounts of data and computational power. Without well-developed incentives, it will be unfair to the resourceful participants.
- Heterogeneity: The heterogeneity of participants in FL includes the heterogeneity of data, the heterogeneity of model structure, and the heterogeneity of training methods.
- Trade-off between the security level and the system performance: There is an intrinsic conflict between the system security level and the performance of FL. Some defense methods affect model performance when they attempt to improve the security level.

Given these concerns, many variants of the traditional FL algorithms have been proposed to improve the results.

### 1) Heterogeneity problems in FL

Personalized federated learning (PFL) algorithms are effective solutions to the problem of local data and problem-setting heterogeneity in FL. This is typically achieved through the design of personalized models or customized local training for each participant. A thorough survey is presented in [80] on recent advances in personalized FL and gives a taxonomy of personalized FL techniques in terms of personalization strategies.

Some widely used personalized FL approaches include clustering, parameter decoupling, model interpolation, multi-task learning, knowledge distillation, meta-learning, and regularized local loss [80].

Table 2.3: Summary of PFL algorithms with different personalization strategies.

| Algorithms | Personalization strategies | Personalization is performed at |
|---|---|---|
| Clustering [81] | Cluster clients with similar models or distributions | Global server |
| Parameter decoupling [82] | Split the model into shared and personalized parts | Global server and local clients |
| Model interpolation [83] | Combine global and local models during inference | Global server or local clients |
| Multi-task learning [84] | Treat client models as tasks in MTL optimization | Local clients |
| Transfer learning [85] | Transfer knowledge from global model to personalized local model | Local clients |
| Knowledge distillation [86] | Distill knowledge from global ensemble to local models | Global server or local clients |
| Reward shaping [87] | Modify rewards in FL-based reinforcement learning | Local clients |

**2) Security problems in FL**

The distributed and open system architecture introduces new threats and makes FL models more vulnerable to poisoning attacks compared with centralized machine learning models. Distributed clients in FL are easier to attack and manipulate, and the parameter sharing and updating between local and global servers may expose the FL to potential risks [88]. As a result, it is crucial to investigate security issues in FL.

There are two types of attacks against FL systems that are most frequently discussed, the privacy attacks and the poisoning attacks [89]. Privacy attacks are carried out by passive or honest-but-curious attackers who try to obtain the private information of other honest clients. According to the design of the FL structure, the clients are assumed to only observe the aggregated or averaged gradient of other clients. However, the attackers can still obtain sensitive information through membership inference [90] or label inference [91].

The poisoning attacks aim to modify the behaviour of the target model in some undesirable way. It can be either random attacks or targeted attacks. Random attacks aim to reduce the overall accuracy of the FL model, while targeted attacks aim to reduce the model accuracy on some specific data or to induce the model to output a target label specified by the attacker.

According to the poisoning method, the poisoning attack can be divided into two categories, data poisoning attacks [92,93] and model poisoning attacks [94,95]. Data poisoning attackers usually pollute the local dataset by injecting poisoned data into the local memory buffer of the malicious clients or changing the labels of the valid datasets. In this way, it can make malicious clients train their local model in the wrong direction and spread the wrong model parameters to the global model and other benign clients through global aggregation.

On the other hand, for model poisoning attacks, attackers directly manipulate the local model parameters of the malicious clients to cause them to misbehave. Previous research has found model poisoning attacks to be more effective than data poisoning attacks. This is because the data poisoning attacks change model parameters indirectly by polluting the training data, whereas model poisoning attacks make more direct changes to the model parameters [96].

The defense methods against malicious clients in FL can be divided into two categories. The first method is to mitigate the attacks from a global view. This can usually be achieved by finding malicious clients and eliminating the negative impact of the malicious clients during global aggregation [97]. Malicious clients can be detected by measuring the similarity between local updates and between local model parameters [98], or by verification of the global validation dataset [99]. Or the global server can add difficulties for attackers by changing the rules of global aggregation and model distribution [100]. The second method is to mitigate the attacks from a local view. For instance, the influence of malicious attackers can be removed during the local fine-tuning using methods like matching networks [101].

While existing research has proposed various distributed learning algorithms and addressed some security challenges, most of these efforts are either theoretical or conducted in generic ML settings. Studies that discuss these topics within realistic wireless commu-

nication scenarios remain scarce and often oversimplified. In this thesis, we aim to bridge this gap by developing distributed AI frameworks and solving security concerns specific to improving the performance demands of mobile networks. Our work offers deeper insights into how distributed intelligence can be made both effective and secure in practical wireless systems.

### 2.2.3 Enhanced intelligence in wireless networks with GFMs

While existing small-scale, custom-built AI techniques have significantly accelerated the evolution of wireless networks, several opportunities remain to be seized. A primary concern is that custom-built AI techniques are usually only applicable to a single use case or a limited number of specific use cases. Once the scenario changes, the model has to be rebuilt and trained again. Future intelligent communications are expected to consider not only how AI techniques can be used in specific network optimization problems or scenarios. They are also expected to integrate AI across all layers of the wireless communication system and promote a unified design paradigm for AI-driven communications and realizing the vision of intrinsic AI [102]. Therefore, there is a need to build general-purpose AI models that apply to broader types of use cases in wireless networks [103].

The second issue is that the complexity of network management tasks in wireless communication has been continuously increasing. The next-generation network represents a highly complex and heterogeneous structure with diversified QoS requirements. Subscriber-centric service is becoming a key enabler to the 5G success, which means service providers are expected to understand customers' needs and provide tailored network capabilities and network characteristics according to the requirements of the customers [104]. This makes some of the network management tasks beyond the capabilities of regular small-scale AI models [105].

Another concern is that contextual information in wireless communication is becoming increasingly diverse. With the development of different sensors on BSs and mobile devices, various types of sensory data with different operating ranges and properties have been collected and included in wireless network management [106]. These data include, but are not limited to, RF data, radio detection and ranging (RADAR) data, light detection and ranging (LiDAR) data, red-green-blue camera data, and the global positioning system (GPS) [107].

Multi-modal sensing can be leveraged for various channel-related downstream tasks, including channel estimation, blockage prediction, beam selection, and handover control. As a result, there is a chance to leverage multi-modal learning to improve the performance

of communication systems in the 5G era. However, training multi-modal models incurs a high computational cost and typically requires a large and diverse dataset. Training custom-built AI models from scratch for each use case will impose a heavy computational burden on service providers and exacerbate data scarcity issues. More importantly, the AI models being trained across different tasks tend to have commonalities when the input information is of the same modalities. This underscores the possibility of fine-tuning a single underlying base model for multiple downstream tasks.

The discussions above highlight the need for utilizing FMs, especially multi-modal FMs in wireless networks. Throughout the evolution of FMs, several well-known models have emerged, marking significant milestones in the development of FMs. Table 2.4 and 2.5 summarizes 17 representative FMs, providing details about model architectures, sizes, functionalities, release years, and whether they are open source. Table 2.4 focuses on single-modal FMs while Table 2.4 focuses on multi-modal FMs. These models cover diverse application domains, including language processing models, vision models, code generation models, general-purpose models, and multi-modal models. For multi-modal models, we only include FMs capable of processing data across more than three modalities, distinguishing them from vision-language models. Notably, some FMs may be applicable to multiple domains. For instance, some general-purpose FMs also function as language processing models, while some multi-modal models are also pre-trained as general-purpose FMs.

Compared with previous AI models, FMs, especially LLMs, has the following benefits. First, they have the ability to handle highly dynamic environments and deal with unforeseen problems [122]. Second, the pre-trained models have less data dependency [123]. They can adapt to new tasks with just a few contextual examples and do not need comprehensive training processes. In addition, LLMs can generate explanations for outputs in human-like language forms, enhancing the explainability of the decisions [124].

Given the large size of FMs, practical deployment becomes a prerequisite for fine-tuning and inference of FMs in wireless networks. The deployment method of FMs largely depends on the hardware support of network nodes, the complexity of the FMs and application requirements. In existing works, four deployment methods have been discussed, including cloud deployment [125, 126], edge deployment [127], on-device deployment [128] and collaborative deployment [129]:

- Cloud deployment: Cloud deployment is the most straightforward method for the deployment of FMs [130]. Cloud servers typically provide a range of configurations with varying hardware capabilities to support computationally intensive FMs. Therefore,

Table 2.4: A summary of prominent single-modality FMs

| Model Name | Key Model Architecture | Model Size | Application Domain | Functionality | Year Realeased | Open Source |
|---|---|---|---|---|---|---|
| BERT [31] | Multilayer bidirectional transformer encoder | 42M/108M/ 334M | Language | Natural language understanding | 2018 | Yes |
| T5 [108] | Encoder-decoder transformer | 60M/220M/ 770M/3B/ 11B | | Perform NLP tasks in a text-to-text form, such as translation, summarization, and question-answering | 2019 | Yes |
| ViT [109] | Transformer encoder | 86M/307M /632M | Vision | Image classification | 2020 | Yes |
| CLIP [110] | ResNet-based image encoder and transformer-based text encoder | 152M | | Connect textual and visual information and perform vision-language tasks such as image classification | 2021 | Yes |
| DALL-E [32] | Decoder-only transformer | Not publicly disclosed | | Generate images from textual descriptions | 2021 | Not officially open-source |
| DALL-E 2 [111] | Diffusion-based decoder with CLIP embeddings | Not publicly disclosed | | Text-conditional image generation | 2022 | No |
| Stable Diffusion [112] | VAE and U-Net decoder | 890M | | Generate images from text prompts | 2022 | Yes |
| Codex [113] | Decoder-only transformer | 175B | Code | Code generation and assistance | 2021 | No |
| NVIDIA's Isaac GR00T N1 [114] | A dual-system architecture inspired by human cognition | Not publicly disclosed | Robotics | Generalized humanoid robot reasoning and skills | 2025 | Yes |
| GPT-3 [115] | Decoder-only transformer | 175B | General purpose | General-purpose language understanding and generation | 2020 | No |
| GPT-4 [23] | Decoder-only transformer | Not publicly disclosed | | General-purpose image and language understanding and text generation | 2023 | No |
| LLAMA [26] | Decoder-only transformer | 7B/13B/ 30B/65B | | General-purpose language understanding and generation | 2023 | Yes |
| PaLM [25] | Decoder-only transformer | 540B | | General-purpose language understanding and generation | 2023 | Not officially open-source |
| BLOOM [116] | Decoder-only transformer | 176B | | General-purpose language understanding and generation | 2022 | Yes |

Table 2.5: A summary of prominent multi-modal FMs

| Model Name | Key Model Architecture | Model Size | Functionality | Year Realeased | Open Source |
|---|---|---|---|---|---|
| GATO [117] | Decoder-only transformer with ResNet patch | 1.17B | Works as a multi-modal, multi-task generalist agent especially for robots (images, text, joint torques, button presses) | 2022 | Yes |
| ImageBind [118] | Modality-specific encoders | Not explicitly disclosed | Combine data from six modalities, including text, video, audio, depth, thermal, and inertial measurement unit (IMU), into a single embedding space | 2023 | Yes |
| GPT-4o [119] | Decoder-only transformer | Not explicitly disclosed | General-purpose agent handling multiple modalities including text, audio, video, and images | 2024 | No |
| Gemini [120] | Multimodal encoder and transformer | Varying sizes for four different versions | General-purpose model handling multiple modalities including text, code, audio, image and video | 2023 | No |
| DeepSeek R1 [121] | Decoder-only transformer | Not explicitly specified, around 671B | General-purpose agent handling multiple modalities including text, audio, and images | 2025 | Yes |

cloud deployment is suitable for scenarios where high computational power is required. When deployed in a cloud environment, FM-based services can choose configurations and distribute model operations across multiple processing units with model parallelism to improve cost-efficiency [125]. However, cloud deployment presents several challenges, particularly in terms of latency and security. Information sharing with the cloud server raises privacy concerns. Moreover, communication with the cloud server introduces extra end-to-end latency, which makes cloud deployment unsuitable for latency-critical applications.

- Edge deployment: Another deployment method is to deploy FMs at the edge of the

network and integrate them with mobile edge computing nodes. This approach brings FMs closer to the data source, resulting in reduced latency and backhaul bandwidth usage compared to cloud deployment. It is particularly effective for edge-tailored models, which use techniques such as quantization, pruning, or distillation to reduce the computation cost. These optimizations allow for faster local inference, making edge deployment more efficient than relying on cloud-based execution. However, compared to the cloud server, mobile edge servers are typically resource-constrained, which makes edge deployment challenging. Various methods have been proposed to address this issue. For example, when multiple FMs with shared parameters or blocks are deployed at an edge node, the node can load a single copy of shared parameters, which can significantly reduce memory usage and minimize data swapping time [131].

- On-device deployment: With the increasing capabilities of mobile devices, such as the integration of graphics processing units (GPUs), another deployment option has emerged as running FMs directly on devices. The on-device deployment method processes tasks locally, which can reduce the service latency. It also offers advantages, including high scalability and the ability to customize FMs based on specific user needs. However, the user devices face inherent limitations in computational and storage resources compared to cloud and edge deployment. Therefore, specialized techniques are essential to address these challenges, such as implementing cost-effective and fast inference using AI accelerators [132], optimization for large models [133] and hardware acceleration [128].

- Collaborative deployment: Collaborative deployment refers to the joint utilization of cloud servers, edge servers, and devices for fine-tuning and inference of FMs. For instance, [129] presents a novel device-edge fine-tuning framework, enabling efficient and privacy-preserving fine-tuning services at the 6G network edge. Similarly, [134] proposes a device-server co-inference method and leverages the split inference technique to offload the computation from devices to a server via layer-wise model partitioning. Such methods can simultaneously solve both the privacy concerns associated with cloud deployment and the computational limitations of on-device deployment. By integrating various network nodes, collaborative deployment balances the trade-offs between data privacy, computational resources, and real-time processing demands and offers more flexible FM-based applications to wireless networks.

With the increasing use of multi-modal FMs and the increasing diversity of data modalities in wireless networks, one primary application of FMs in this domain is the interpretation of multi-modal contextual information. Wireless networks can be considered as

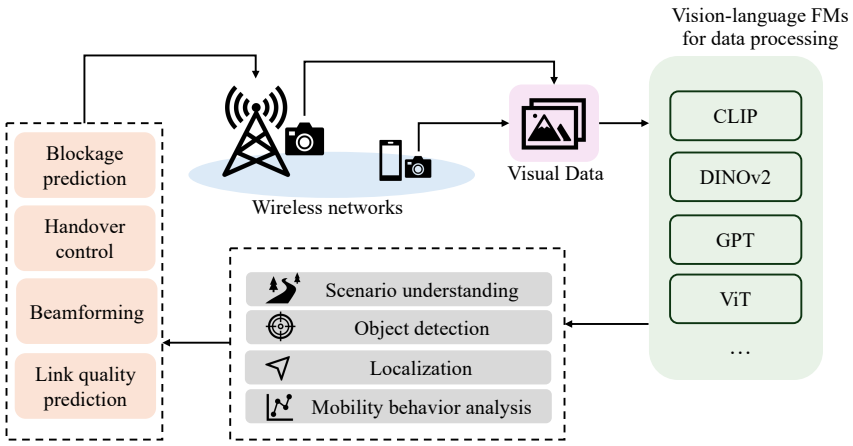



Figure 2.7: FM-enhanced visual data analysis and applications in wireless networks

sensor networks if network nodes function as sensors. By analyzing sensor data, changes in the propagation environment caused by moving objects can be captured and used to optimize wireless network management [135]. FMs can be applied to analyze and generate embeddings for various types of multi-modal contextual data in wireless networks, mainly including visual data, graph data, and point cloud data.

In Fig. 2.7, the FM-enhanced visual data analysis and applications in wireless networks are presented. Researchers have proposed incorporating cameras at millimeter wave (mmWave) BSs, highlighting potential gains in obtaining sensing information and advanced network management [136, 137]. FMs can help with visual localization in wireless networks and estimate the three-dimensional (3D) location of mobile devices and network nodes. This capability is especially important for devices lacking access to the GPS [138]. The integration of location information of mobile devices enables wireless operators to perform location-based network analysis, thereby optimizing the performance of location-dependent network management applications [139, 140].

Given the capabilities of FMs and the information FMs can extract from visual data, several applications can possibly benefit from FM-based wireless visual information analysis, including blockage prediction [141], handover control [142], beamforming [143] and link quality prediction [144].

In addition, graph-based analysis has become increasingly important for optimizing and managing wireless network operations [145, 146]. Graph FMs have emerged as a significant advancement in the graph domain. They are designed to learn transferable knowledge from large-scale graph datasets that can be applied across different wireless network sce-

narios [147,148], including mobility management [149], interference management [150], and anomaly detection and security [151].

In wireless networks, 3D point cloud data is usually gathered through advanced technologies such as LiDAR and radar. These data can provide rich spatial details and enable networks to understand the geometric and structural characteristics of the contextual environment. FMs can utilize 3D point clouds for tasks like segmentation, classification, and scenario understanding, or refine these spatial representations to better suit the needs of specific applications in wireless networks.

Some existing research has combined FMs with 3D point cloud in wireless networks. LiDAR-LLM leverages LLMs to interpret raw LiDAR data, enabling a comprehensive understanding of outdoor 3D scenes, which can be applied to tasks such as 3D captioning and question answering [152]. Additionally, the NetOrchLLM framework aims to master wireless network orchestration with LLMs, considering inputs from various sensors such as RADAR, LiDAR, and cameras [153]. It highlights the potential of LLMs in processing 3D point cloud data to enhance wireless communication systems. These advancements demonstrate the potential of further leveraging FMs to enhance the application of 3D point cloud in wireless networks.

While the integration of LLM into wireless communication systems has recently attracted much interest, the majority of existing works remain at an exploratory stage. Many research works focus on conceptual studies, without addressing the practical challenges of deploying LLMs in real-world communication scenarios. In this thesis, we further contribute to this emerging direction by investigating applications of pre-trained LLMs and GAI techniques in wireless networks.

# Chapter 3

# Deep reinforcement learning for resource allocation in O-RAN

Recently, the concept of O-RAN has been proposed, which aims to adopt intelligence and openness in the next generation RAN. It provides standardized interfaces and the ability to host network applications from third-party vendors by xAPPs, which enables higher flexibility for network management. However, this may lead to conflicts in network function implementations, especially when these functions are implemented by different vendors. To this end, in this chapter, we aim to mitigate the conflicts between xAPPs for near real-time RIC of O-RAN. In particular, we first consider the joint power allocation and radio resource problem in O-RAN with a single slice setup, and propose a team learning algorithm to mitigate the conflicts between two xAPPs. On this basis, we extend the research to multiple slices setup and propose a FL algorithm to enhance the performance of the network by increasing cooperation between xAPPs. We compare the proposed approaches with independent DQN where network functions individually optimize resources. Our simulations show that the proposed approaches has better network performance under various traffic loads. The team learning achieves 8% higher throughput and 64.8% lower PDR, and the FL algorithm. The FL algorithm achieves 11% higher throughput for eMBB slices and 33% lower delay for ultra-reliable low-latency communication URLLC slices.

## 3.1 Resource Allocation System Model

This section will introduce the communication channel model, the QoS definitions and the network slicing settings.

### 3.1.1 Communication Channel Model

We consider a downlink orthogonal frequency-division multiplexing cellular system with $N$ BSs serving $K$ UE simultaneously. The considered wireless system architecture is shown in Fig. 3.1. BSs are controlled by the near-real-time RIC and non real-time RIC of O-RAN architecture. Radio controller of O-RAN architecture is divided into two parts, near-real-time RIC and non-real-time RIC. Near-real-time RIC is used for RAN control and optimization. The role of the non-real-time layer is to provide guidance, as well as ML models to support near-real-time RIC functions [154]. Near-real-time RIC can communicate with BSs through interface E2 and non-real-time RIC can communicate with BSs by interface O1. Interface A1 is used for communications between two RICs.

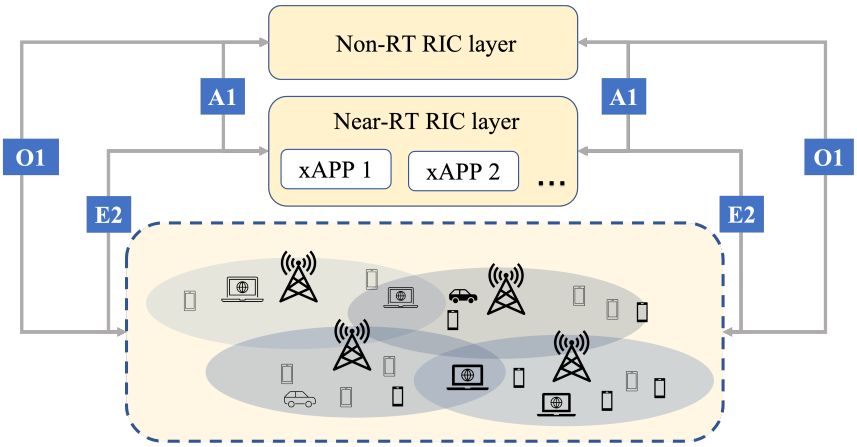



Figure 3.1: System model of resource allocation in O-RAN.

A resource block refers to the smallest unit of resources that can be allocated to a user and it is composed of 12 subcarriers. In this model, several consecutive resource blocks is bundled as a resource block group (RBG), and each BS has $M$ RBGs respectively. Here we consider the RBG as the smallest time-frequency resources to be allocated [155]. At time $t$, the list of users associated to BS $n$ can be described as $\mathcal{H}_t^n$, where $\sum_n |\mathcal{H}_t^n| = K$. The transmission power allocated to the $m^{th}$ RBG of BS $n$ can be described as $P_t^{n,m}$.

For simplicity, a flat-fading channel model is adopted in this work to reduce system complexity and to capture the key characteristics relevant to the learning task. The signal interference noise ratio (SINR) between BS $n$ and user $k$ on RBG $m$ at time $t$ is described

as $\eta_t^{n,m,k}$. It can be formulated as:

$$\eta^{n,m,k} = \frac{\alpha^{n,m,k} g^{n,k} P^{n,m}}{\sum_{\substack{n'=1 \\ n' \neq n}}^{N} \sum_{k'=1}^{K^{n'}} \alpha^{n',m,k'} g^{n',k} P^{n',m} + \sigma^2}, \tag{3.1}$$

where $\alpha^{n,m,k}$ is a binary indicator that denotes whether BS $n$ allocates RBG $m$ to user $k$. $K^n$ denotes the total number of associated users of BS $n$. $g^{n,k}$ denotes the channel coefficient between BS $n$ and user $k$. $\sigma^2$ denotes the noise power at the receiver. The transmission capacity of BS $n$ on RBG $m$ can be formulated as:

$$C^{n,m} = B_m log_2(1 + \sum_k \eta^{n,m,k}), \tag{3.2}$$

where $B_m$ denotes the bandwidth of RBG $m$. We assume user traffic follows Poisson arrivals with a mean arrival rate $\lambda$ and pending traffic is queued in a transmission buffer with limited buffer size. If the length of queuing data is longer than the buffer size, then the extra data will be discarded.

### 3.1.2 QoS Definition

The transmission rate of BS $n$ on RBG $m$ can be formulated as:

$$R^{n,m} = \begin{cases} C^{n,m}, & C^{n,m}T < \sum_k \alpha^{n,m,k} L^k \\ L^k/T, & C^{n,m}T >= \sum_k \alpha^{n,m,k} L^k \end{cases} \tag{3.3}$$

where $T$ denotes the length of time slot and $L^k$ denotes the amount of remained data of user $k$ in the transmission buffer.

The total delay of user $k$ consists of three components, the transmission delay, the queuing delay and the retransmission delay. It can be given as:

$$d_k^{total} = d_k^{tx} + d_k^{que} + d_k^{rtx}, \tag{3.4}$$

where $d_k^{tx}$, $d_k^{que}$ and $d_k^{rtx}$ denote the transmission delay, the queuing delay and the retransmission delay. The transmission delay is formulated as:

$$d_k^{\mathrm{tx}} = \frac{L_k}{\sum_n \sum_m \alpha^{n,m,k} C^{n,m}}, \tag{3.5}$$

$L_k$ is the packet size of user $k$. For eMBB slices, optimization aims to achieve maximum total throughput, and for URLLC slices, the aim is to minimize the average delay of packets.

### 3.1.3 Problem Formulation

For the single slice setup, we improve the system QoS by maximizing the system sum throughput and minimizing the PDR. The problem can be formulated as:

$$\begin{aligned} \underset{P^{m,n},\alpha^{n,m,k}}{max} \; & \sum_n \sum_m R^{n,m}, \\ s.t. \; & (3.1) - (3.3), \\ & P_{min} \leq P^{m,n} \leq P_{max}, \forall n, m, \\ & \alpha^{n,m,k} = \{0,1\}, \forall n, m, k \\ & \sum_k \alpha^{n,m,k} = 1, \forall n, m \end{aligned} \tag{3.6}$$

where $P_{min}$ and $P_{max}$ denote the maximum and minimum transmission power. The power range is set to reflect practical hardware and regulatory limits, prevent excessive interference, ensure stable optimization, and maintain efficient and reliable network performance.

For the multiple slices setup, we assume each BS includes two types of slices, namely eMBB slice, and URLLC slice, and each slice may contain several UE with similar QoS requirements. For eMBB slices, optimization aims to achieve maximum total throughput, and for URLLC slices, the aim is to minimize the average delay of packets. To improve training stability, the original throughput rewards are transformed using the $tan^{-1}$ function, following the approach in [156]. This transformation compresses extreme values into a bounded range and mitigates the impact of outliers. The problem can be formulated as:

$$\underset{P_n,\alpha_{n,m,k}}{max} \; \sum_n \sum_s w_s r_s \tag{3.7}$$

$$s.t. \; (3.1) - (3.5)$$

$$r_s^{embb} = \begin{cases} tan^{-1}(\sum_{k \in \mathcal{K}_s^{embb}} b_k), & |\mathcal{H}_s^{embb}| \neq 0 \\ 0, & else \end{cases} \tag{3.7a}$$

$$r_s^{urllc} = \begin{cases} 1 - \sum_{k \in \mathcal{K}_s^{urllc}} d_k, & |\mathcal{H}_s^{urllc}| \neq 0 \\ 0, & else \end{cases} \tag{3.7b}$$

$$P_{min} \leq P_n \leq P_{max}, \forall k \tag{3.7c}$$

$$\alpha_{n,m,k} = \{0,1\}, \forall n, m, k \tag{3.7d}$$

$$\sum_k \alpha_{n,m,k} = 1, \forall n, m \tag{3.7e}$$

where $S$ denotes the $s^{th}$ slice. $w_s$ and $r_s$ denote the priority weight and the reward of each slice, respectively. $d_k$ denotes the end-to-end delay. $b_k$ denotes the transmission throughput of the $k^{th}$ user and $d_k$ denotes the end-to-end delay. $\mathcal{K}_s^{urllc}$ denotes the UEs from the URLLC slides and $\mathcal{K}_s^{embb}$ denotes the UEs from the eMBB slides.

## 3.2 DQN based Power Control and Resource Allocation xAPPs

In this section, we introduce how to transform the power control and radio resource allocation as two independent xAPPs. For power control, we deploy a DQN to decide the transmission power level of the BS. For radio resource allocation, we use different methods for a single slice setup and multiple slices setup. For a single slice setup, each BS selects and distributes RBGs among its own associated users, and we define a BS as an agent. For multiple slices setup, we combine the DQN with a hierarchical decision-making mechanism. The inter-slice radio resource allocation is first performed to decide the amount of radio blocks that will be allocated to each slice.

### 3.2.1 MDP of Power Control xAPP and Resource Allocation Control xAPP Agent for a Single Slice Setup

Under a single slice setup, the power control xAPP will choose a power level for each RBG of each BS. The MDP of power control agent is defined as follows:

- State: State includes transmission rate, transmission power, CSI and the length of queued data in the buffer [157]. The state of an agent is given as:

$$S_t^m = \{\Gamma_t^{n,m}, R_t^{n,m}, p_t^{n,m}, L_t^{n,m} | n \in \{1, 2, ..., N\}\} \tag{3.8}$$

  The $\Gamma_{n,m}^t$ is the logarithmic normalized CSI, which is given by:

$$\Gamma_t^{n,m} = \{log_2(1 + \frac{\sum_{k\in\{1,2,...,K)} \alpha_t^{n',m,k} g_t^{n',k}}{\sum_{k\in\{1,2,...,K\}} \alpha_t^{n,m,k} g_t^{n,k}}) | n' \in \{1, 2, ..., N\}, n' \neq n\} \tag{3.9}$$

  $L_t^{n,m}$ is the length of queued data in the buffer, which can be given as:

$$L_t^{n,m} = \sum_k \alpha_t^{n,m,k} L^k \tag{3.10}$$

- Action: In our model, we divide the transmission power into $B$ levels according to the maximum and the minimum transmission power. The action is to choose a power level for each RBG of each BS, which is given as:

$$\begin{aligned} a_t^{n,m} &= \{P_{min}, P_{min} + \frac{P_{max} - P_{min}}{B-1}, ..., P_{max}\} \\ A_t^m &= \{a_t^{n,m} | n \in \{1, 2, ..., N\}\} \end{aligned} \tag{3.11}$$

- Reward: The reward is defined as the total throughput since our objective is to improve throughput:

$$r_t = \sum_m \sum_n R_t^{n,m} \tag{3.12}$$

  Where $R_t^{m,n}$ is the transmission rate of BS $n$ on RBG $m$.

In radio resource allocation, each BS selects and distributes resource blocks among its own associated users, and we define a BS as an agent. The MDP is defined by:

- State: the state of radio resource allocation agent includes transmission rate, transmission power, CSI and the length of queuing data in the buffer. The state is given as:

$$S_t^n = \{\alpha_t^{n,m,k}\Gamma_t^{n,m}, \alpha_t^{n,m,k}R_t^{n,m}, \alpha_t^{n,m,k}p_t^{n,m}, \alpha_t^{n,m,k}L_t^{n,m} | n \in \{1, 2, ..., N\}\}, \tag{3.13}$$

  Since the radio resource allocation policy of a BS is only related with its associated users, we only consider the features related to these users. So we add indicator $\alpha_t^{n,m,k}$ in the state.

- Action: The action of BS $n$ is to choose a user by:

$$\begin{aligned} a_t^{n,m} &= \{k_0, k_1, ..., k_{D-1} | k_d \in \mathcal{H}_t^n\} \\ A_t^n &= \{a_t^{n,m} | m \in \{1, 2, ..., M\}\} \end{aligned} \tag{3.14}$$

- Reward: The reward of radio resource allocation is the total throughput of each base station, which is given as:

$$r_t = \sum_m \sum_n R_t^{n,m} \tag{3.15}$$

### 3.2.2 MDP of Power Control xAPP and Resource Allocation Control xAPP Agent for multiple slices setup

For multiple slices setup, the power control xAPP will decide the transmission power level of the BS, and after that we assume the power is uniformly allocated to all radio resource blocks. The Markov decision process (MDP) of power control agent is defined as follows:

- State: The state of power control model includes the queue length of packets, the current delay and the current transmission power, which is given as:

$$s_{n,t} = \{H_s, \sum_{k=1}^{K_s} d_m, P_n | \forall s \in \{1, 2, ..., S), n \in \{1, 2, ..., N\}\}, \tag{3.16}$$

  where $H_s$ denotes the queue length of packets in the $s^{th}$ slice, $P_n$ denotes the transmission power of the $n^{th}$ BS. Here the state definition represents the traffic demand and network status, and then the agent can change the transmission power accordingly to achieve desired network performance.

- Action: The action of power control is to choose power level from:

$$a_{n,t} = \{1, 2, ..., L_{max}\}, \tag{3.17}$$

  where $L_{max}$ denotes the highest power level. Then the transmission power of the $n^{th}$ BS is given as:

$$P_{n,t} = \frac{a_{n,t} P_{max}}{L_{max}}, \tag{3.18}$$

- Reward: The reward of power control model is defined as the weighted sum reward of all the slices with a penalty of large transmission power.

$$r_{n,t} = \sum_{s} w_s r_s - \alpha a_{n,t}, \tag{3.19}$$

  The purpose of establishing penalty items is to balance the power consumption and network performance, and control the interference on neighbouring BSs.

For radio resource allocation xAPP, the inter-slice radio resource allocation is first performed to decide the amount of radio resource blocks that will be allocated to each slice. The MDP of inter-slice radio resource allocation is given as follows:

- State: The state is composed of the queue length of packets and the current delay, which is given as:

$$s_{n,t} = \{H_s, \sum_k d_k\}, \tag{3.20}$$

- Action: The action is to decide the portion of radio resource blocks allocated to each slice, which is given as:

$$a_{n,t} = \{RB_s \in \{0, 1, ..., R_{max}\} | \sum_s RB_s = R_{max}\}, \tag{3.21}$$

  where $R_{max}$ denotes the number of RBGs that can be allocated

- Reward: The reward of inter-slice radio resource allocation model is defined as the weighted sum reward of all the slices, which is given as:

$$r_{n,t} = \sum_{s \in s_n} w_s r_s, \tag{3.22}$$

Then, for the intra-slice resource allocation, we assume all the devices in the same slice have equal priority. Hence, we deploy a proportional fair algorithm (PPF) [157]. The basic idea is to determine the priority of devices for resource scheduling based on the ratio of instantaneous transmission rate and the long-term average transmission rate of a single device. The intra-slice radio resource allocation strategy based on proportional fairness scheduling can be formulated as follows:

$$\underset{k \in K_{n,s}}{argmax} \frac{C_{n,k}}{(\sum_{t-\Delta t}^{t} b_{t,k})/\Delta t} \tag{3.23}$$

where $(\sum_{t-\Delta t}^{t} b_{t,k})/\Delta t$ denotes the average transmission rate of device $k$ over the past period $\Delta t$. The radio resource blocks allocated to slice $s$ will be given to device $k$ according to the selection rule in the above formula.

## 3.3 Team Learning and Federated Learning-based xAPPs Conflicts mitigation

### 3.3.1 The conflicts of xAPPs

When we deploy two intelligent xAPPs in the near-RT RIC layer of O-RAN architecture, conflicts may arise if they have overlapping objectives. Firstly, the intelligent xAPP A

and B observe states from system and select actions accordingly. Then xAPP A and B simultaneously apply actions to the system, causing changes to the system that produces a reward. The reward is given back to xAPP A and B for the respective model updates. The experience of xAPP A and xAPP B will be recorded in their own experience relay memory and used for DQN model training. Finally, the DQN module will generate strategies about action selection for two xAPPs.

When an xAPP acts, it does not take into account the actions of other xAPPs. The selected action may be the optimal choice in its own view, but the performance of other xAPPs may be affected. For example, xAPP A may assign a high transmission power to a RBG. At the same time, xAPP B may reassign this RBG to user with a small traffic load. When xAPP A and xAPP B take actions together, the high transmission power does not bring high throughput as expected by xAPP A, but increases the power consumption and generates more interference. The goal of our team learning algorithm is to eliminate conflicts between actions from different team members.

### 3.3.2 Team learning-based Mitigation

In team learning based architecture, the xAPPs will include the action of other xAPPs in its own state, which means they can learn to avoid conflicts and cooperate better. In this paper, we propose an information exchange based team learning architecture, which is shown by Fig. 3.2. Different from the regular DQN models, xAPP A keeps two experience replay memories and trains two DQNs. The first DQN is used to select a first-round action based on its own observation from environment. This action will be shared with xAPP B, and xAPP B selects its own action based on both the observation from environment and first-round action from xAPP A. Then the action of xAPP B will feed into xAPP A and will help xAPP A make a second-round action with the second DQN.

In this case, for the power allocation xAPP, the MDP defined in the last section is used for the first DQN model to select the intended action which will be informed to xAPP radio resource allocation. The second model is also defined and used to determine the power allocation action for the second-round. This action is the actual action to be performed by power allocation. The state of this model is related to the environment and the information passed by the radio resource allocation. The MDP for the second DQN model of power allocation is given below:

- State: The sate is given as:

$$S_t^m = \{\Gamma_t^{n,m'}, R_t^{n,m'}, p_t^{n,m'}, \Gamma_t^{n',m'}, R_t^{n',m'}, p_t^{n',m'}, L_t^{n',m'} | n \in \{1, 2, ..., N\}\} \tag{3.24}$$

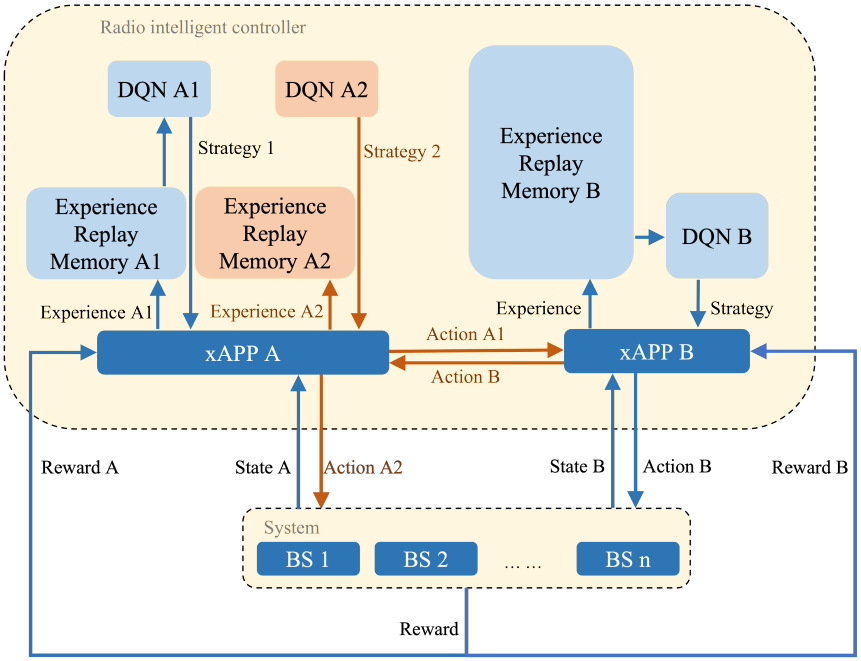



Figure 3.2: xAPPs interactions with team learning.

where $m'$, $n'$ is given as:

$$m', n' \in \{m' \in \{1, 2, ..., M\}, n' \in \{1, 2, ..., N\}, \alpha_{t+1}^{n',m',k} = 1 | k \in \{\alpha_t^{n,m,k} = 1\}\} \quad (3.25)$$

$\alpha_{t+1}^{n',m',k}$ is obtained from the intended action of radio resource allocation model.

- Action and Reward: The definition of action and reward of the second model is the same as the first model.

  We only define one DQN for radio resource allocation because it only needs one round action. Similarly, the state with team learning can be modified by including the intended action of power allocation model, which is given as:

$$S_t^n = \{\alpha_t^{n,m,k}\Gamma_t^{n,m}, \alpha_t^{n,m,k}R_t^{n,m}, \alpha_t^{n,m,k}p_t^{n,m}, \alpha_t^{n,m,k}L_t^{n,m}, p_{t+1}^{n,m}\}, \quad (3.26)$$

### 3.3.3 Federated Learning-based Mitigation

Fig. 3.3 shows the structure of our proposed VFRL algorithm. The general process of one update cycle in VFRL can be described as follows. Firstly, each agent observes their states from the environment. Next, the local models of different agents are respectively updated with local training. The intermediate results calculated by local models are then submitted to a global model for global training, and the decisions of the global model are sent back to agents and executed to change the environment. Local models are then updated based on the feedback of the global model and the reward from the environment.

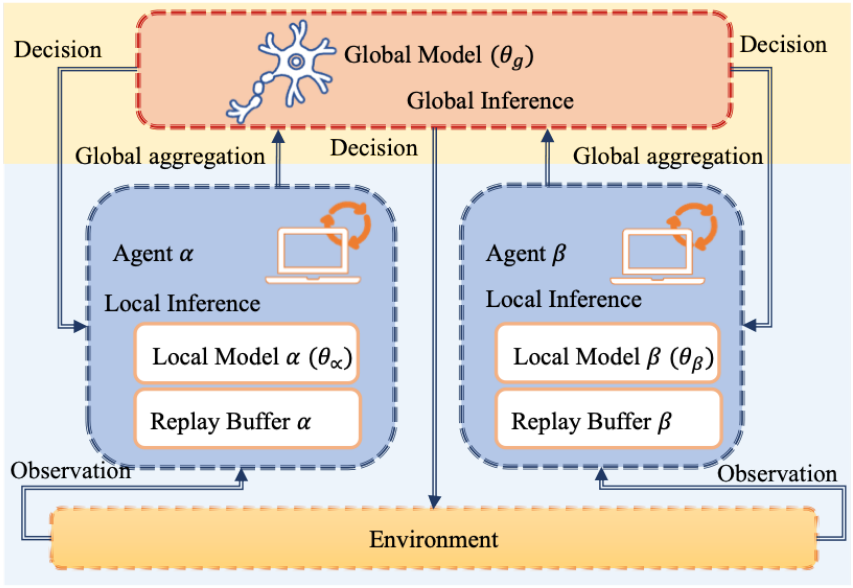



Figure 3.3: Process of federated deep reinforcement learning for xAPPs conflicts mitigation.

In Fig. 3.3, agent $\alpha$ and agent $\beta$ refer to the power control xAPP and radio resource allocation xAPP, respectively. $\theta_g$, $\theta_\alpha$ and $\theta_\beta$ respectively denote the parameters of the global model and the local models of two xAPPs. The core idea of our proposed algorithm is to use the local Q-table generated by two xAPPs to indicate the expected reward of independent actions and make a global calibration of the local models.

The federating process can be summarized as three steps:

- Local inference. During the local inference step, power control and radio resource allocation xAPPs observe state information from BSs, then they use observed state

and local DQN model to calculate the local Q-table, which can be formulated as:

$$\boldsymbol{Q_\alpha} = DQN^\alpha(s_\alpha, \boldsymbol{a_\alpha}; \theta_\alpha) \tag{3.27}$$

$$\boldsymbol{Q_\beta} = DQN^\beta(s_\beta, \boldsymbol{a_\beta}; \theta_\beta) \tag{3.28}$$

where $DQN^\alpha$ and $DQN^\beta$ denote two local DQN models of agent $\alpha$ and agent $\beta$. $s_\alpha$ and $s_\beta$ denote the locally observed states and $\theta_\alpha$ and $\theta_\beta$ denote the parameters of two DQN models.

- Global aggregation. During the global inference step, the two local Q-tables generated by global power control and radio resource allocation xAPPs are submitted and combined as the input of the global model to calculate a joint global Q-table, which can be formulated as:

$$\boldsymbol{Q_g} = [\boldsymbol{Q^g_\alpha}|\boldsymbol{Q^g_\beta}] = DNN^g([\boldsymbol{Q_\alpha}|\boldsymbol{Q_\beta}]; \theta_g) \tag{3.29}$$

where $Q^g_\alpha$ denotes the Q-table of agent $\alpha$ after global calibration and $[\boldsymbol{Q^g_\alpha}|\boldsymbol{Q^g_\beta}]$ denotes the global Q-table is a splicing of Q-tables of two agents after global calibration.

- Global inference. During the global inference step, the global Q-table is split into two calibrated Q-tables and the split Q-tables are used for action selection, which can be formulated as:

$$a_\alpha = \underset{a_\alpha}{argmax} \boldsymbol{Q^g_\alpha} \tag{3.30}$$

$$a_\beta = \underset{a_\beta}{argmax} \boldsymbol{Q^g_\beta} \tag{3.31}$$

The actions decided by global inference will be executed in the environment and the corresponding rewards will be stored in the replay buffers. In addition, an adaptive $\epsilon$-greedy learning strategy is adopted, which means the agent will randomly choose actions with a certain probability in order to explore the unknown action space.

## 3.4 Simulation Result

### 3.4.1 Parameter Settings

For the single slice scenario, we use Python simulation platform and the reinforcement learning and team learning algorithms are based on the deep-learning package TensorFlow.

Table 3.1: Simulation settings for the single slice scenario

| Parameter | Value |
|---|---|
| Networking Environment | 4 Base Stations with 1 kilometre inter-site distance, 30 users. |
| Propagation | $\beta$ = 120.9+37.6 log10(d)+10 log10(z) dB |
| Carrier configuration | 20MHz bandwidth, 100 resource blocks, 12 subcarriers per resource block, 12 RBGs. |
| PHY configuration | Maximum transmission power of 38 dBm, minimum transmission power of 1dBm Additive white Gaussian noise = -114dBm. |
| Traffic model | Poisson distribution with varying load between 3-6 Mbps. |
| Simulation time | 20,000 time slots. 1 time slot is 100ms. |

The algorithm is simulated with 4 BSs and 30 users. The amount of data generated by each user per time slot follows a Poisson distribution with the central value of the distribution varying from 3 Mbps to 6 Mbps. The users move with a constant speed during the simulation, and the probability of changing direction is 0.3 in each time slot. Table 3.1 concludes the parameter settings.

We use two four-layer neural networks in the DQN for power allocation and radio resource allocation. According to the dimension of states and actions, we set the neuron numbers of two hidden layers of power allocation model as 256 and 128 and the neuron numbers of radio resource allocation model as 512 and 256. The discount factor is set as 0.2. The initial learning rate of both power allocation and radio resource allocation is set as 0.001. We run 20000 time slots in each experiments and apply an adaptive $\epsilon$-greedy learning strategy with an initial exploration rate of 0.3 [158].

We compared our proposed team deep Q-learning (TDL) algorithm with independent deep Q-learning (IDL) algorithm. In IDL, xAPPs only observe information from the environment and do not communicate with each other. We change the traffic load of each user between 3-6 Mbps and the user moving speed between 0-30 m/s and compare the performance of TDL and IDL in different scenarios.

For the multiple slices scenario, we consider 2 BSs, and each BS has 4 slices, 2 eMBB slices and 2 URLLC slices. Each slice has 3 users. We use constant bit rate traffic for eMBB slices while the URLLC traffic follows a Poisson distribution. Simulations are implemented with Matlab 5G toolbox and run 10 simulations with 5000 Transmission Time Intervals

Table 3.2: Simulation settings.

| Parameters | Settings |
|---|---|
| Carrier configuration | Bandwidth: 20MHz,<br>Number of RBs: 100,<br>Subcarriers of each RB: 12,<br>Maximum transmission power: 30 dBm<br>Tx/Rx antenna gain: 15 dB<br>TTI size: 2 OFDM symbols |
| Propagation model | 128.1 + 37.6log(Distance),<br>Log-Normal shadowing 8 dB. |
| Retransmissions | Maximum number of retransmissions: 1,<br>Round trip delay: 4 TTIs |
| Traffic mode | eMBB: Constant bit rate traffic,<br>Packet size 32 Bytes,<br>URLLC: Poisson traffic,<br>Packet size 16 Bytes |
| Networking environment | 2 gNBs, 12 users for each BS<br>Inter-gNB distance: 500m |
| Slice settings | 2 eMBB slices and 2 URLLC slices<br>in each BS,<br>Priority:<br>URLLC 2 >URLLC 1 >eMBB 2>eMBB 1 |

(TTIs). Table 3.2 shows the settings of our simulations.

We compare three cases, namely independent reinforcement learning (IRL), centralized reinforcement learning (CRL), and FRL. IRL means the local models of agents are trained separately without the global model. CRL means putting two xAPPs into a single agent and training a single model with joint states and actions. Here we use CRL as an optimal baseline, since the centralized control with global vision will generally bring the best performance. FRL is our proposed algorithm, in which local models are federated by a coordination model. During the simulations, We fix the URLLC traffic, change the traffic load of eMBB slices.

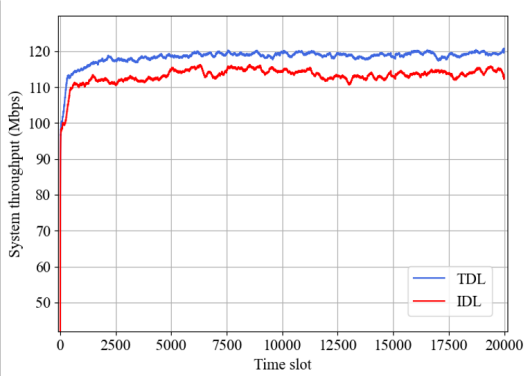



Figure 3.4: System throughput (traffic load = 4 Mbps, user speed = 20 m/s).

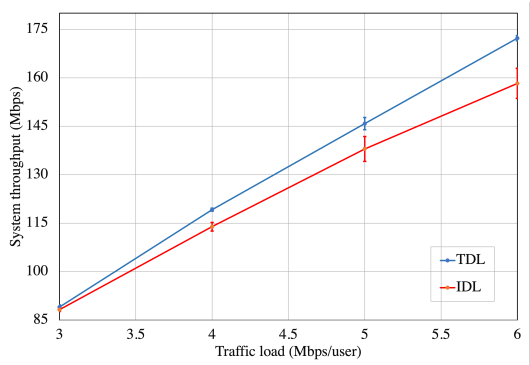



Figure 3.5: System throughput with different traffic load (user speed = 20 m/s).

### 3.4.2 Simulation Results of Single Slice Setup

We first compare the convergence as well as the throughput of all BSs and the PDR of the system for TDL and IDL.

Fig. 3.4 shows the convergence curves of TDL and IDL during the learning process and compares the system throughput with TDL and IDL at 4 Mbps traffic load when the speed of users is 20 m/s. We can observe from Fig. 3.5 that the throughput with TDL is more stable and higher than that with IDL during the whole simulation process. When the simulation performance is stabilized, the system can achieve a 4.6% higher throughput with TDL.

Fig. 3.6 and Fig. 3.7 show the system throughput and PDR when the speed of users is 20 m/s and the average traffic load of users changes from 3 Mbps to 6 Mbps. In Fig. 3.6, we observe that the system throughput with TDL is higher than that with IDL. Correspondingly, we can also see from Fig. 3.7 that the PDR with TDL is lower than that with IDL. The gap between them becomes more noticeable when the traffic load becomes higher. This is because when the traffic load is relatively low, less data needs to be transmitted, and even a poorer allocation policy can handle the transmission task. When the traffic load is high there is more data to be transmitted, hence the advantage of a good allocation policy is more significant. When the traffic load is 6 Mbps, the TDL achieves 8.8% higher throughput and 64.8% lower PDR than IDL.

Fig. 3.7 shows the system throughput when the average traffic load of users is 4 Mbps and the speed of users changes from 0 m/s to 30 m/s. Fig. 3.7 presents that the system throughput of TDL is higher than that IDL especially when the user speed is high. This

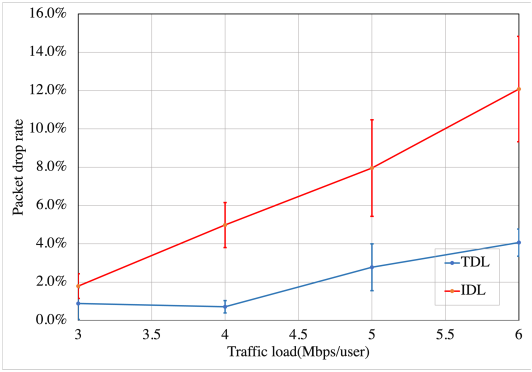



Figure 3.6: PDR with different traffic load (user speed = 20 m/s).

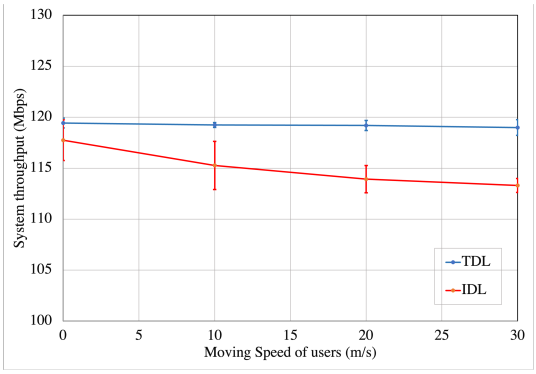



Figure 3.7: System throughput with different user moving speed (traffic load = 4 Mbps).

is because in the case of fast moving users, the allocation strategies will change more frequently, and there will be more conflicts between xAPPs. As a result, cooperation becomes more important. Note that, the mobility of users are limited in a circular region therefore mobility doesn't cause handover in our experiments. When the moving speed of users is 30 m/s, the TDL achieves 5.0% higher throughput than IDL.

The above results show that for different traffic loads and user speeds, TDL results in higher throughput and lower PDR compared to IDL.

### 3.4.3 Simulation Results of Multiple Slices Setup

Fig. 3.8 shows the delay of URLLC slices when the URLLC traffic load is 2 Mbps and the eMBB traffic load changes from 4 Mbps to 10 Mbps. Since the priority of URLLC slices is higher than that of eMBB slices and the traffic load of URLLC slices is constant, the delay of URLLC slices does not significantly increase when the eMBB traffic load changes. It can be observed that the delay of URLLC slices with the FRL algorithm is lower than that with the IRL algorithm and is very close to that with CRL. It means the federating process can help coordinate different xAPPs and achieve comparable network performance with centralized training. Specifically, when the eMBB traffic load is 10 Mbps, the delay with FRL is 33% lower than the IRL algorithm. Fig. 3.9 further shows the empirical complementary cumulative distribution function (ECCDF) of the delay of URLLC slices when the eMBB traffic load is 8 Mbps. It shows that our proposed FRL has a better delay distribution than IRL, which is indicated by a lower ECCDF curve.

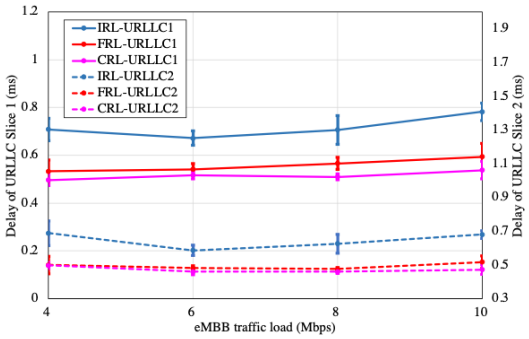



Figure 3.8: URLLC slices delay under various eMBB traffic load.

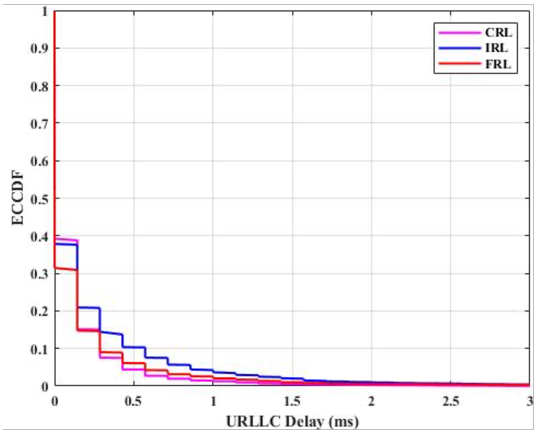



Figure 3.9: ECCDF of the URLLC slices delay distributions.

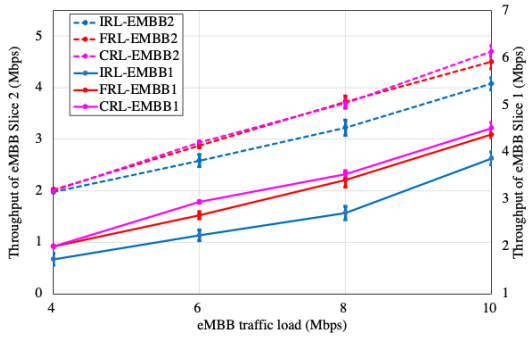



Figure 3.10: Throughput of eMBB slice under various eMBB traffic load.

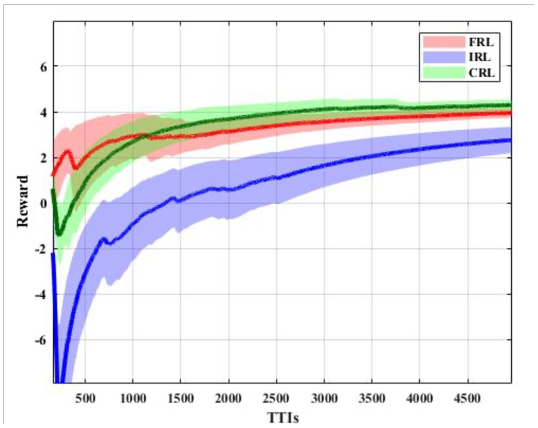



Figure 3.11: Convergence comparison.

Moreover, Fig. 3.10 shows the throughput of eMBB slices at various eMBB traffic loads. When the eMBB traffic load grows, more eMBB traffic needs to be transmitted, so the throughput of eMBB slices will also grow. We can observe that FRL can achieve higher eMBB throughput compared with IRL cases, and the network performance of FRL is very close to IRL. Specifically, when the eMBB traffic load is 10 Mbps, the eMBB throughput of FRL is 11% higher than that of IRL.

Finally, the Fig. 3.11 shows the convergence performance of IRL, CRL, and FRL when the eMBB traffic is 8 Mbps, and the URLLC traffic is 2 Mbps. We can observe that CRL can ultimately achieve the highest reward, while the converged reward of FRL is slightly lower than CRL. This is because CRL uses a central training approach, which allows for more comprehensive global information to be captured. However, the FRL has a faster convergence speed than CRL, which can be explained by the huge state and action space of CRL approach. The reward of IRL is much lower than the other two algorithms. Based on the convergence trend, FRL converges faster than CRL.

We can draw the conclusion that the proposed FRL algorithm can achieve better network performance than IRL, and the performance of FRL is very comparable with the optimal baseline CRL. Meanwhile, FRL enables faster convergence and can scale to multiple agent situations with distributed structure. It also allows information observed by different agents to be kept confidential and consumes fewer training resources.

# Chapter 4

# Knowledge Transfer and Federated Learning Enabled UE-Centric Traffic Steering

traffic steering (TS) is a promising approach to support various service requirements and enhance transmission reliability by distributing network traffic loads to appropriate BSs. In this chapter, we propose a KT-FLUC traffic steering framework for highly dynamic 5G RAN. Specifically, first, we propose an attention-weighted group federated learning scheme. It enables intelligent UEs to make traffic steering decisions autonomously using local models and observations, and a global model is defined to coordinate local traffic steering decisions and share experiences among UEs. Secondly, considering the individual UE's limited computation and energy resources, a growing and pruning-based model compression method is introduced, mitigating the computation burden of UEs and reducing the communication overhead of federated learning. In addition, we propose a Q-value-based knowledge transfer method to initialize newcomer UEs, achieving a jump start for their training efficiency. Finally, the simulations show that our proposed KT-FLUC algorithm can effectively improve the service quality, achieving 65% and 38% lower delay and 52% and 57% higher throughput compared with cell-based traffic steering and other UE-centric traffic steering strategies, respectively.

## 4.1 UE-centric System Model for traffic steering

### 4.1.1 Architecture

The proposed traffic steering system model of this work is shown in Fig. 4.1. We consider a network with two coexisting RAT, 5G and LTE, with dual connectivity (DC) for downlink transmission. We assume the system has $k$ UEs and $N$ BSs. One BS is a MBS equipped with the LTE technology, and the remaining $(N-1)$ are SBS equipped with the 5G technology. Each UE can choose a single BS at one time for better service. BSs generate interference with each other due to frequency reuse. Meanwhile, dynamic UE arrival and departure are involved, and we assume UEs arrive at the cell with a fixed file length to be transmitted [159].

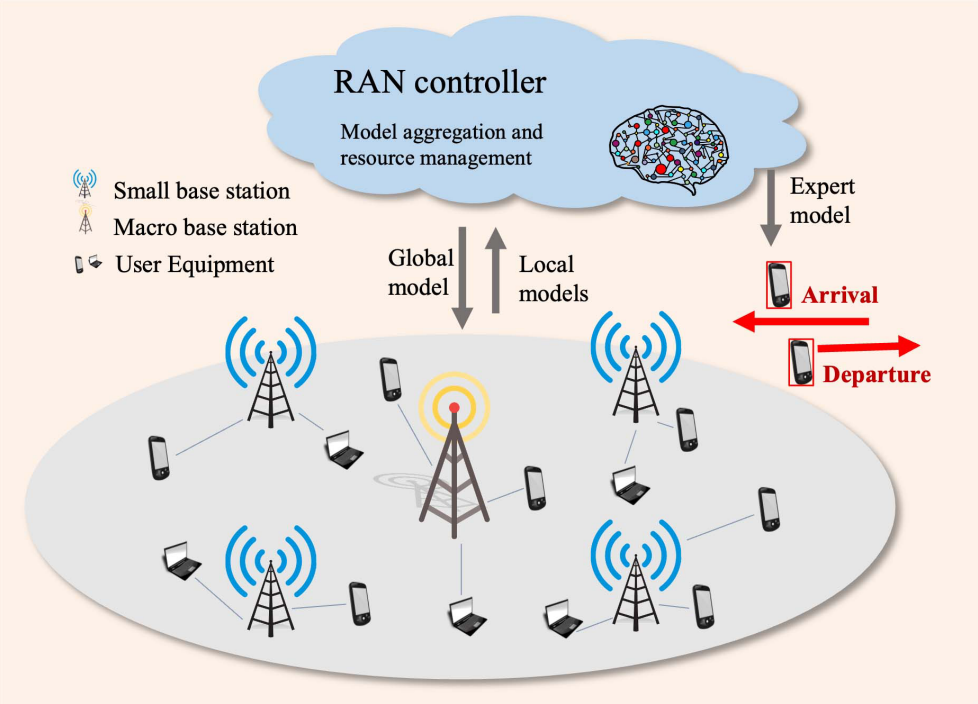



Figure 4.1: Dynamic wireless networks with one MBS and several SBSs.

We assume UEs have two types of services, GBR service and non-GBR service, which have different QoS requirements [160]. Each UE either runs GBR traffic or non-GBR traffic. Most GBR services are sensitive to transmission delay, while non-GBR services require higher throughput [161]. So, the optimization goal of this traffic steering framework

is to minimize the packet delay of GBR services and maximize the throughput of non-GBR services.

### 4.1.2 Communication Model

The communication delay consists of two components, the queuing delay and the transmission delay:

$$d_k = d_k^{que} + d_k^{tx}, \tag{4.1}$$

where $d_k^{que}$ is the queuing delay of the last packet transmitted to the user $k$. It refers to the delay incurred while the packet waits in the BS before being transmitted. $d_k^{tx}$ is the transmission delay of the packet. It refers to the time pushing the bits onto the link, which is given as:

$$d_k^{tx} = \frac{L_k}{C_{n,k}}, \tag{4.2}$$

where $L_k$ denotes the packet length and $C_{n,k}$ denotes the link capacity between the $n^{th}$ BS and the $k^{th}$ UE. According to Shannon's theorem, the link capacity is influenced by the signal interference noise ratio (SINR) between the BS and the UE, which can be formulated as:

$$C_{n,k} = \sum_{r \in \mathcal{R}_n} B_r log_2(1 + \eta_{n,r,k}), \tag{4.3}$$

where $\mathcal{R}_n$ denotes the set of all the available resource blocks of the $n^{th}$ BS and $B_r$ denotes the available bandwidth of the $r^{th}$ resource block. $\eta_{n,r,k}$ denotes the SINR between the transmission link between the $n^{th}$ BS and the $k^{th}$ UE with the $r^{th}$ resource block. It can be given as:

$$\eta_{n,r,k} = \frac{\alpha_{n,r,k} g_{n,k} P_{n,r}}{\sum_{n' \in \mathcal{N}, n' \neq n} \sum_{k' \in \mathcal{K}_{n'}} \alpha_{n',r,k'} g_{n',k} P_{n',r} + B_r N_0}, \tag{4.4}$$

where $\alpha_{n,r,k}$ is a binary indicator denoting whether the $r^{th}$ physical resource block of the $n^{th}$ BS is allocated to the $k^{th}$ UE. $\mathcal{N}$ denotes the set of BSs. $\mathcal{K}_{n'}$ denotes the set of UEs that connect to the $n'$ BS. $g_{n,k}$ is the channel gain of the transmission link between the $n^{th}$ BS and the $k^{th}$ UE which is decided by the free space propagation model. $K_n$ denotes the set of UE associated with the $n^{th}$ BS. $P_{n,r}$ denotes the transmission power and $N_0$ denotes the power density of the noise.

### 4.1.3 Problem Formulation

According to the service type of UEs, we have different optimization objectives for different UEs since they have different QoS requirements. If a UE has a GBR service, the utility of a UE is decided by the delay of the traffic, which is given by Eq (4.1). If a UE has a non-GBR service, the utility of a UE is decided by the throughput of the UE, which is decided by the packet arrival and the channel capacity given by Eq (4.3). To constrain the utility value between 0 and 1, we use the estimated maximum throughput and delay of UEs, $B_{max}$ and $D_{max}$, for normalization. Furthermore, since we want to minimize the delay time rather than maximize it, we define the utility of GBR traffic as $1 - \frac{d_k}{D_{max}}$. Therefore, for the $k^{th}$ UE, the traffic steering problem is formulated to maximize the utility:

$$\max_{\beta_{k,n}} \quad U_k, \tag{4.5}$$

$$s.t.\ (4.1) - (4.2)$$

$$U_k = \begin{cases} \frac{b_k}{B_{max}}, & if\ T_k = 0 \\ 1 - \frac{d_k}{D_{max}}, & if\ T_k = 1 \end{cases} \tag{5a}$$

$$\alpha_{n,r,k} = \{0, 1\} \forall n, r \tag{5b}$$

$$\sum_{k \in \mathcal{K}_n} \alpha_{n,r,k} = 1 \quad \forall r \in \mathcal{R}_n, n \in \mathcal{N} \tag{5c}$$

$$K_n = \{k' | \forall \beta_{k',n} = 1\} \tag{5d}$$

$$\beta_{k,n} = \{0, 1\}, \forall n \tag{5e}$$

$$\sum_{n \in N} \beta_{n,k} = 1 \tag{5f}$$

where Eq (4.1)-(4.2) defines the variable $d_k$ which used in the definition of the utility. Eq (5a) indicates that the optimization goal of each UE is related to its traffic type $T_k$. $T_k = 0$ indicates the $k^{th}$ UE holds non-GBR traffic and the utility is related to its throughput $b_k$. $T_k = 1$ indicates the $k^{th}$ UE holds a GBR traffic, and its reward is related to its delay $d_k$. $B_{max}$ and $D_{max}$ are the estimated maximum throughput and delay of UEs and they are fixed values to normalize the utility of delay and throughput. $\beta_{k,n}$ is a binary indicator denoting whether the $k^{th}$ UE chooses the $n^{th}$ BS. Eq (5b) indicates that a PRB can be assigned or unassigned to a user. Eq (5c) indicates that each physical resource block of a BS can only be allocated to one UE at the same time. Eq (5d) indicates that $\beta_{k,n}$ decides the attached list of BSs and Eq (5f) indicates that each UE can only choose one BS at one time. Given the UE-centric structure of our proposed framework, it is worth noting that the problem formulation in Eq (4.5) is defined for one single UE. According to this problem formulation, each UE makes local decisions to maximize its own utility.

As it can be observed in Eq (4.5), the proposed problem contains uncertain or stochastic parameters, and the optimal solution may depend on the probability distribution of the parameters. In addition, in the traffic steering scenario, the BS selection of a UE at a given moment affects not only the immediate QoS rewards but also the QoS rewards in the future. We want to focus more on the long-term rewards, and it is difficult to complete by optimization. Therefore, the formulated problem could not be directly solved, and an AI-based solution is proposed.

## 4.2 Proposed traffic steering Scheme

In this section, we present the proposed KT-FLUC method. First, we provide an overview of the algorithmic framework. Then, we describe the federated reinforcement learning algorithm, model compression method, and knowledge transfer technique we used in KT-FLUC, respectively.

### 4.2.1 Algorithm Framework

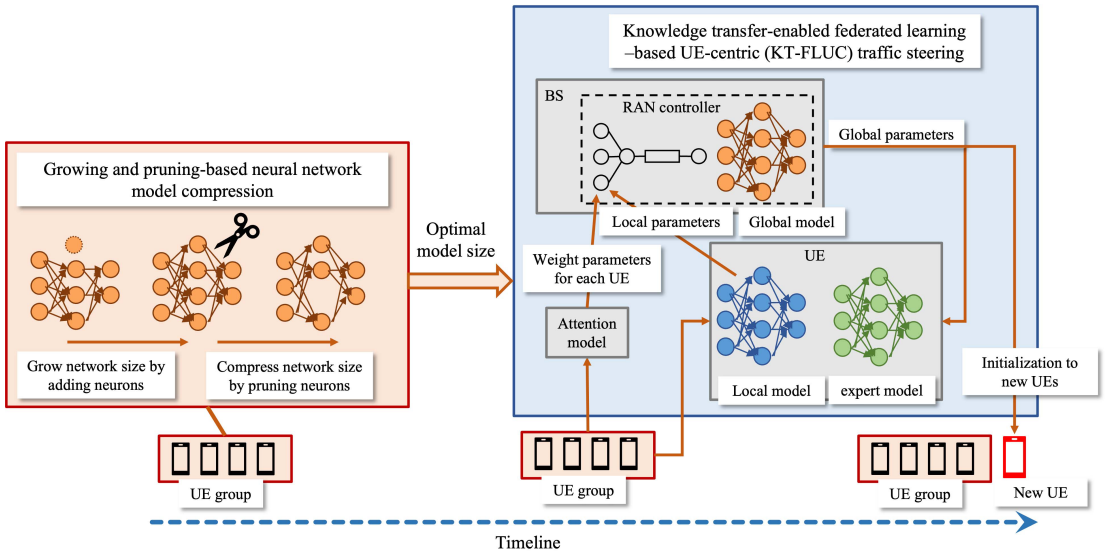



Figure 4.2: The framework of proposed UE-centric traffic steering method.

Fig. 4.2 shows the framework of the proposed UE-centric traffic steering method. Firstly, we apply the growing and pruning method to decide the optimal network size

for the UR intelligence. In particular, we first grow the neural network (NN) size by adding neurons, then compress the NN size by pruning neurons. Given the appropriate NN size, federated learning and knowledge transfer are cooperatively performed within a KT-FLUC framework. This framework uses an attention-weighted federated learning algorithm to aggregate experience from distributed UEs and generates a global model. A weight parameter is calculated for each UE by an attention model according to the performance of each UE. Then, UEs submit the parameters of their local models to the RAN controller, and the parameters of the global model are updated according to the local parameters and attention weights. After that, the knowledge transfer technique is used to transfer the aggregated experience from the global model to UEs. It is implemented by defining an expert model for each UE. The expert is a copy of the aggregated global model, which runs inside each UE. In the local training process of each UE, only local models are trained, and expert models remain unchanged. Every time the global model is updated, the parameters of expert models will also be updated by parameter replication, and the action selection is processed inside each UE according to the outputs of expert models and local models.

In addition, for realistic dynamic environments, new UEs may join the network without prior knowledge of intelligent traffic steering. Knowledge transfer will be performed between the global model and the newcomer for initialization. After that, the newcomer will perform local training and join the next round of federated learning.

This KT-FLUC framework uses federated learning to collect and aggregate experience from UEs with similar traffic types and QoS requirements. Knowledge transfer transfers the aggregated experience to existing UEs and newcomers. The expert model of knowledge transfer is obtained during federated learning, which is how these two techniques are connected [162]. The update of the global model by federated learning is non-real-time, while the traffic steering decisions made by UEs are real-time. Under this framework, UEs may leave the system, but their experience will be kept for newcomers. The shared model is constantly updated with the distributed efforts of all the UEs in this network. Despite the high dynamics of the system, the knowledge learned by UEs consistently contributes to the optimization of a shared expert model. In turn, the expert model accelerates the local models' training and enhances system performance.

In the following, we first introduce the federated reinforcement learning-based UE-centric traffic steering algorithm since it serves as the main scheme in this work. Then, we explain how to implement model compression for neural networks, and finally, transfer learning techniques are introduced.

### 4.2.2 Federated Reinforcement Learning based UE-centric traffic steering

For UE-centric traffic steering, each UE is treated as an independent agent, including a DQN for decision-making [162]. MDP of UE-centric traffic steering is defined as:

- State: The state of traffic steering of each UE is composed of the received signal strength indicator (RSSI) from each BS, the number of queued packets, the delay of the last arrived packet, and its current attached BS. The RSSI information is accessible to UEs, and it can reveal the state information of the system, including the channel quality and the distance between the UE and BSs. The number of queued packets is related to the QoS of the UE. The delay of the last arrived packet, and the current attached BS indicates whether the UE needs to switch to a new BS. These indicators are important for traffic steering decisions and can easily be observed and collected from the view of a single UE. The state at time $t$ is given as:

$$S_{k,t} = \{\Gamma_{n,t}, L_{k,t}^{queue}, d_{k,t}, \beta_{k,n,t} | n \in \mathcal{N}\}, \tag{4.6}$$

  where $\Gamma_n$ denotes the RSSI from the $n^{th}$ BS and $L_k^{queue}$ denotes the current packet queue length.

- Action: After observing states from the environment, UE will make local traffic steering decisions. Since traffic steering is to select the best serving BS for each UE, the action of traffic steering of each UE is to choose an optimal BS, which is given as:

$$A_{k,t} = \{\beta_{k,n,t} | n \in \mathcal{N}\},, \tag{4.7}$$

  where $\beta_{k,n}$ denotes whether the $k^{th}$ UE chooses the $n^{th}$ BS.

- Reward: The traffic steering reward for each UE is related to their traffic type and the delay and throughput of the traffic, which can be given as:

$$R_{k,t} = \begin{cases} \frac{b_{k,t}}{B_{max}}, & if\ T_n = 0. \\ 1 - \frac{d_{k,t}}{D_{max}}, & if\ T_n = 1. \end{cases} \tag{4.8}$$

As it can be observed from the MDP definition, the definition of reward has a similar format to the utility, which is the optimization goal of the problem formulation. The action is similar to the control variable of the problem formulation. By doing this, we convert

the problem defined in Eq (4.5) into a MDP problem, which can then be solved with the DQN algorithm. The optimal gap of the solution is decided by the training effect of the DQN. In an ideal situation, the DQN is able to develop an optimal traffic steering solution with full exploration and accurate prediction. However, it is very difficult for DQN to be trained to perfection in real cases. To reduce the optimal gap, in this work, we also use FL and transfer learning techniques to enhance the training effect of DQN. Also, to limit the number of handovers, the handover action will be selected only if the reward value differential is higher than a given threshold. In this way, the framework will avoid unnecessary handovers.

The core idea of DQN is to use a NN to predict the long-term reward, which is denoted as Q-values, of different actions under given states. The Q-values in each step can be updated according to the Bellman equation:

$$Q(s_{k,t}, a_{k,t}) = (1-\alpha)Q(s_{k,t}, a_{k,t}) + \alpha(R_{k,t} + \underset{a}{max}\gamma Q(s_{k,t+1}, a)),$$

where $Q(s_{k,t}, a_{k,t})$ denotes the long-term reward of choosing action $a$ under the observation of state $s$ at time $t$ of the $k^{th}$ UE. $\gamma$ is the discount factor denoting the importance of distant future reward compared with instant reward, and $\alpha$ is the learning rate denoting the update speed of the NN. The local update of DQN at the $k^{th}$ UE can be formulated as:

$$\begin{aligned}\theta_{k,t+1} = \theta_{k,t} + \alpha[R_{k,t} + \gamma \underset{a}{max}\, Q(s_{k,t+1}, a; \theta_{k,t}) \\ - Q(s_{k,t}, a_{k,t}; \theta_{k,t})]\nabla Q(s_{k,t}, a_{k,t}; \theta_{k,t}),\end{aligned} \tag{4.9}$$

where $\theta_{k,t}$ denotes the parameters of the DQN at time $t$.

The federated learning scheme is shown in Fig. 4.3. Here, we adopted a grouped attention-weighted federated learning approach based on FedAvg, and multi-centered federated learning [163, 164]. Considering the heterogeneity between different traffic types, we assume UEs with GBR traffic as the first group and UEs with non-GBR traffic as the second group and enable model federating and experience sharing among groups. Inside each group, an attention weight is calculated and assigned to each UE to represent the contribution of each UE to the global model and promote model aggregation.

The federated learning process inside each group can be concluded as three main steps. The first step is the local training and updates. In this step, each UE observes local states, selects actions according to the Q-values generated by local DQN, and uses local experience to train DQN by minimizing the local loss function. The local loss function is defined as

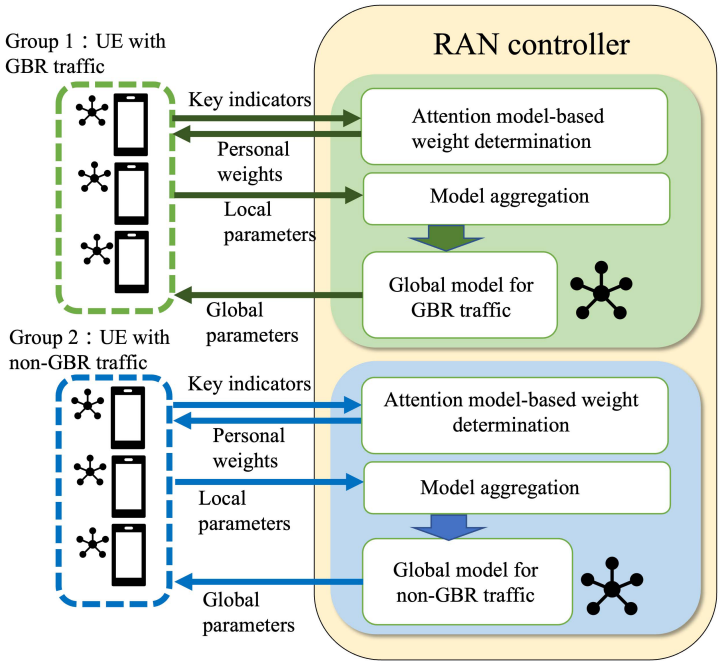



Figure 4.3: Federated learning process.

the mean squared error, which can be given as:

$$Loss_k^{local}(\theta_{k,t}) = \sum_{\Psi_k}(y_k - R_{k,t} - \gamma \underset{a}{max} Q(s_{k,t+1}, a; \theta_{k,t}))^2,$$

where $Loss_k^{local}$ denotes the local loss function of the $k^{th}$ UE and $\Psi_k$ denotes the memory buffer of the $k^{th}$ UE. $y_k$ denotes the output of the local DQN.

The second step is attention weights determination. In each federating interval, all the UEs submit the parameters of their local DQN to the RAN controller to aggregate a global model. Weight parameters are calculated and assigned for each UE as preparation for aggregation. Since all the local models are trained separately and independently, they are trained at different degrees. To generate a more accurate and generalized global model, we expect well-trained models to contribute more to the global model. In contrast, barely trained models contribute less to the global model. Given this, three key indicators are used to evaluate how well the model has been trained, the average reward, training data size and the achievement rate.

- The average reward of a UE is denoted as $\overline{R_k}$ and refers to the average value of system reward defined by Equation (4.8) of the local DQN during the interval of two

federating processes. It can be seen as an intuitive representation of the DQN model performance.

- The training data size of a UE is denoted as $|\Psi_k|$ and refers to the amount of training data. We expect UEs with richer experience to account for more weight in federating.
- The achievable rate of a UE is denoted as $\Phi_k$ and refers to the frequency that the UE achieves a specific level of latency or throughput requirement. We add this indicator to $\overline{R_k}$ to prevent coincidental large rewards from giving a false representation of model performance.

After obtaining all the key indicators from the UE side, we use a SoftMax function-based attention mechanism to combine these indicators and decide a weight for each local model accordingly:

$$w_k = softmax(\frac{QK_k}{\sqrt{n_k}}), \tag{4.10}$$

where $w_k$ denotes the weight of the $k^{th}$ UE. $n_k$ is the dimension of indicators. $K_k$ is the key indicators and can be formulated as $K_k = [\frac{\overline{R_k}}{\underset{k}{max}\overline{R_k}}, \frac{|\Psi_k|}{\underset{k}{max}|\Psi_k|}, \frac{\Phi_k}{\underset{k}{max}\Phi_k}]$. $Q$ is the query and can be formulated as $Q = [1, 1, 1]$. By defining the key and the query, we expect UEs with indicators closer to the maximum values receive more attention. For instance, if a UE has a higher average reward value and a higher achievable rate, it is considered to be better trained. If it has a larger training data size, the training may considered to be more reliable. Therefore, we firstly pre-process the indicators by finding the maximum $\overline{R_k}$, $|\Psi_k|$ and $\Phi_k$ among all the UEs and perform normalization. The weight can then be calculated as:

$$\begin{aligned} w_k &= softmax(\frac{\overline{R_k}' + |\Psi_k|' + \Phi_k'}{\sqrt{n_k}}) \\ &= \frac{\exp\left\{(\frac{\overline{R_k}' + |\Psi_k|' + \Phi_k'}{\sqrt{n_k}}\right\})}{\sum_{k' \in \mathcal{K}} \exp\left\{(\frac{\overline{R_{k'}'}' + |\Psi_{k'}'|' + \Phi_{k'}'}{\sqrt{n_k}})\right\}}, \end{aligned} \tag{4.11}$$

where $\overline{R_k}'$, $|\Psi_k|'$ and $\Phi_k'$ denotes the indicators after normalization.

Finally, the last step is model aggregation and model update. The parameter of global model can be obtained as the weighted sum of all the local models, which is given as:

$$\theta_{g,t+1} = (1 - \eta_1)\theta_{g,t} + \eta_1 w_k \sum_{k \in \mathcal{K}} \theta_{k,t}, \tag{4.12}$$

where $\theta_g$ denotes the parameters of the global model. $\eta_1$ is a parameter denoting the update speed of global model parameters. $\theta_k$ denotes the parameters of the $k^{th}$ local model. In this way, the well-trained models will be given larger weights and contribute more to the global model. The barely trained models will be given smaller weights and contribute less to the global model.

Then the local models are updated with the global model, which is given as:

$$\theta_{k,t+1} = (1 - \eta_2)\theta_{k,t} + \eta_2\theta_{g,t}, \tag{4.13}$$

where $\eta_2$ is a parameter denoting the update speed of local model parameters during federating.

The proposed federated learning framework is designed with flexibility to accommodate heterogeneous UE models. In scenarios where some local models are non-AI and incompatible with the aggregation process, the framework can selectively group clients with similar AI model architectures for aggregation, while excluding non-AI models from the global update. This design also supports partial aggregation, enabling the system to aggregate only a subset of AI models without being constrained by the presence of heterogeneous or non-AI participants.

### 4.2.3 Growing and Pruning-based Model Compression

This section will introduce the growing and pruning method for model compression. Based on biological neural theories, neurons compete with each other. The neurons with more substantial competitiveness are suitable for survival and growth while the neurons with weaker competitiveness will be eliminated [165,166]. Therefore, the model compression can be divided into two phases, the growing phase and the pruning phase, shown in Fig. 4.4.

As shown in Fig. 4.4, a growing phase is first performed to estimate how many neurons will be enough to support the network functions. It is worth explaining that although UEs are resource-constrained, the growing process is designed before our real implementation and does not have to be run online on UE devices. So the constrained resources will not be a problem. During the growing phase, the number of neurons is added one by one by splitting the neuron with the strongest competitiveness into two. If no improvement is observed in the network performance as the model size increases, the NN is supposed to be large enough to support the network functions. Then the algorithm steps into the pruning process and the neurons are pruned one by one by cutting off the neuron with the weakest competitiveness until there are no more neurons to be cut off.

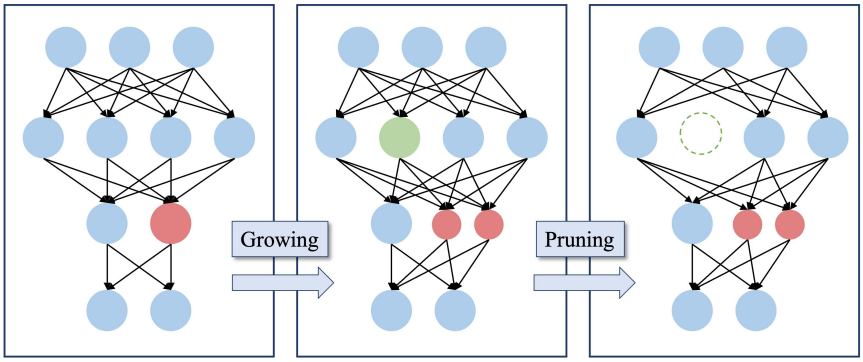



Figure 4.4: Two phases of growing and pruning.

The competitiveness of a neuron is defined according to the average percentage of zero (PoZ) by measuring the percentage of zero activations of a neuron after the ReLU mapping [167], which is given as:

$$PoZ_i^j = \frac{\sum_z^Z \sum_l^L f(o_{l,j}^{(i)}(z) == 0)}{Z * L}, \tag{4.14}$$

where $PoZ_i^j$ denotes the average PoZ of the $j^{th}$ neuron in the $i^{th}$ layer. $L$ and $Z$ denote the dimension of the neuron output and the number of validation samples. $o_{l,j}^{(i)}$ denotes the $l^{th}$ output of the $j^{th}$ neuron in the $i^{th}$ layer. $f(*)$ is a binary logic function, which can be given as:

$$f(*) = \begin{cases} 1, & if\ true. \\ 0, & if\ false. \end{cases} \tag{4.15}$$

Since the ReLU mapping will change all the negative inputs into zero, the average PoZ of ReLU mapping can reflect the percentage of discarded and reserved neuron information. The neuron with the highest average PoZ is supposed to reserve the minor information and have the weakest competitiveness. In contrast, the neuron with the lowest average PoZ keeps the most information and have the most substantial competitiveness.

### 4.2.4 Q-value-based Knowledge Transfer

This section introduces the transfer learning technique. The framework of knowledge transfer is shown in Fig. 4.5. We define two models for each UE, an expert model and a local

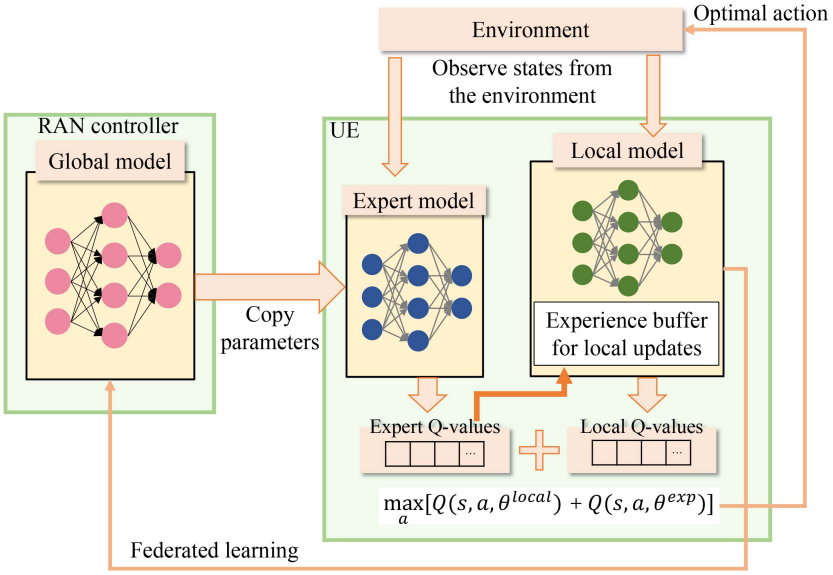



Figure 4.5: Knowledge transfer process.

model. During the traffic steering process, both the expert and local models of a UE observe the state from the environment and generate Q-vales for different actions. The expert model is initialized by the global model in the RAN controller through parameter replication. Although expert models share the same network parameters with the global model, they are different models because they are deployed at different parts of the network and play different roles. The expert model is deployed inside each UE. It takes the previously learned knowledge of the global model as expert knowledge and provides expert Q-values based on the global experience during local training process in federated learning. The local model provides local Q-values based on the local experience. The traffic steering actions are chosen by maximizing the sum of expert and local Q-values. This process can be formulated as:

$$a = \underset{a}{argmax}(Q(s_{k,t}, A; \theta_{k,t}^{local}) + Q(s_{k,t}, A; \theta_{k,t}^{exp})), \tag{4.16}$$

where $\theta^{local}$ denotes local model parameters and $\theta^{exp}$ denotes expert model parameters. In this way, the global experience from the expert model and the local experience from the local model are combined for decision-making.

When a new UE arrives at the system, the global model parameters are directly copied into the local and expert models for initialization. Next, during each federated interval,

the actions are chosen as a combined effort of both the local and expert models. We use separated local models and expert models since the local model is trained and updated during local training, while the global model remains unchanged. When federated learning is performed, the global model is updated by federating all the local models. The updated global model is used to update the expert models at each UE side. Therefore, the global experience and local experience can be simultaneously learned but separately kept at each UE. Another advantage of using the proposed knowledge transfer scheme is that it can reduce the running time by performing global model aggregation and local training simultaneously. Conventionally, the local model update is based on feedback from the global model, so local training and model aggregation should be performed sequentially. But with the introduction of the expert model, feedback is given to the expert model instead of the local model. That means UEs do not need to wait for feedback from the global model and model aggregation can be run in parallel with local training. In this way, the running time of model aggregation is saved.

Since the global model is an aggregation of local models and expert models are transferred from the global model, the local models and expert models are supposed to converge to the same value. Therefore, we redefine the loss function of local training as:

$$\begin{aligned} Loss_k =& \sum\nolimits_{\Psi_k} (y_k + Q(s_{k,t}, a_{k,t}; \theta_{k,t}^{exp}) - 2R_{k,t} \\ & - \gamma \underset{a}{max}(Q(s_{k,t+1}, a; \theta_{k,t}^{local}) + Q(s_{k,t+1}, a; \theta_{k,t}^{exp}))^2. \end{aligned} \tag{4.17}$$

Since the expert Q model is not updated during the local training, $Q(s_{k,t+1}, a; \theta_{k,t}^{exp})$ and $Q(s_m^t, a_m^t; \theta_{k,t}^{exp})$ can be seen as constants. So the update of local models at each UE can be formulated as:

$$\begin{aligned} \theta_{k,t+1} =& \theta_{k,t} + \alpha[2R_{k,t} - Q(s_{k,t}, a_{k,t}; \theta_{k,t}^{exp}) \\ & - \gamma \underset{a}{max}(Q(s_{k,t+1}, a; \theta_{k,t}^{local}) + Q(s_{k,t+1}, a; \theta_{k,t}^{exp})) \\ & - Q(s_{k,t}, a_{k,t}; \theta_{k,t}^{local})] \nabla Q(s_{k,t}, a_{k,t}; \theta_{k,t}^{local}). \end{aligned} \tag{4.18}$$

With this definition, the sum of expert Q-values and local Q-values is supposed to converge to twice the long-term accumulated reward. On the other hand, expert Q-values and local Q-values will converge to a similar value according to the federated learning process. So the final output is expected to converge to the long-term accumulated reward of given actions and states. In this way, expert models can provide a lead for UEs according to the previous experience of other UEs and UEs can make personalized modifications based on the suggestions of experts. It is worth pointing out that the expert models do not need

to be trained on the UE side. It is only used for model inference during the local training, which only requires minimum computation.

## 4.3 Simulation Results

### 4.3.1 Baselines

In this research, we proposed four baseline algorithms. Firstly, we compared our proposed KT-FLUC algorithm with a centralized cell-centric DRL-based traffic steering algorithm (denoted as CL in the simulation results), which is implemented based on [168]. In this algorithm, we only use a single agent at the RAN controller, collecting observations from distributed UEs and making a centralized decision. The action is a joint traffic steering decision of all the UEs, which will be broken into independent traffic steering actions for each UE, and the model will be trained based on the average reward of all the UEs.

Then, we compared our proposed KT-FLUC algorithm with three other UE-centric algorithms. The first one is a distributed independent DRL-based traffic steering algorithm, which is implemented based on the idea of [167]. It is denoted as DIL in the results. In this baseline, each UE acts as an independent agent, making decisions without information exchanging or experience sharing.

The third baseline is a federated DRL-based traffic steering algorithm, which is denoted as FL in the result. The federated learning algorithm used in this baseline is borrowing ideas from [169]. Compared with DIL, the federated learning steps are added, while compared with the proposed algorithm, the knowledge transfer part is removed.

The last baseline is a federated DRL-based traffic steering algorithm with direct initialization, which is denoted as FLI in the result. Direct initialization means that the global model parameters are directly copied to the local model every time a new UE enters the system.

### 4.3.2 Simulation Settings

We consider a scenario with $N = 5$ BSs, an LTE MBS in the center and 4 5G SBSs in the surrounding. We consider a dynamic system and UEs will arrive and depart during the simulation. 40% UEs have GBR service and 60% UEs have non-GBR service. The simulation is repeated five times using MATLAB, and the average outcomes are presented

along with a 95% confidence interval. The $B_{max}$ and $D_{max}$ we use during the simulation are 10 ms and 10 Mbps. The update speed of the global model $\eta_1$ is 0.5. The update speed of the local model $\eta_2$ is 1. The split interval we use during model compression is 150 TTIs. The $N_{required}$ is set as 5. During the simulation, we change the average number of UEs from $M = 25$ to $M = 65$ to see how the proposed algorithm performs under different UE numbers and traffic loads. Depending on the dimensions of the state space and action space, the number of hidden neurons is initially set to 32 and 64 for most UEs. Then we perform growth and pruning for one single UE and start with two hidden neurons in each layer. Other simulation details about network environment settings and deep learning settings are shown in Table 4.1.

### 4.3.3 Performance Analysis between UE-centric traffic steering and Cell-centric traffic steering

This section will compare the proposed KT-FLUC with cell-centric baseline CL and DIL.

Fig. 4.6 shows the convergence curves of DIL, CL and proposed KT-FLUCalgorithms when the number of UE is 45. It can be observed that the CL algorithm can achieve a higher system reward than the DIL algorithm, but a lower reward than our proposed KT-FLUC algorithm. This demonstrates that the cell-based algorithm has an advantage in system performance compared with simple UE-centric algorithms because the cell-centric algorithms hold more comprehensive information and can coordinate between UEs.

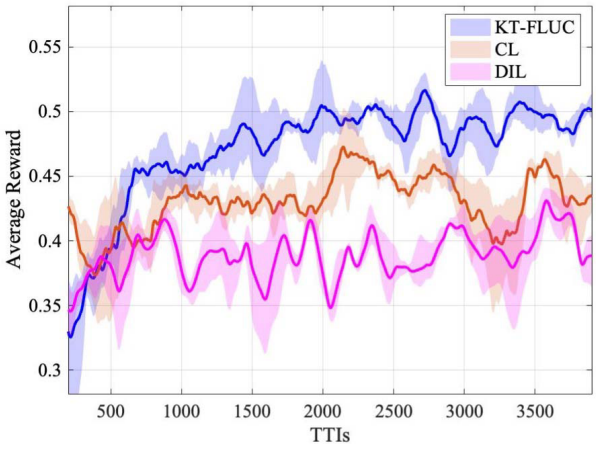



Figure 4.6: The convergence performance of system reward.

Table 4.1: Simulation settings.

| **Network environment settings** | **UE traffic settings** |
|---|---|
| Bandwidth: 20 MHz | **GBR traffic** |
| Subcarriers of each RB: 12 | QCI: 2 |
| Tx/Rx antenna gain: 15 dB | Proportion: 40% |
| Number of resource blocks: 100 | Data length: 50 KB |
| Propagation model: 128.1 + 37.6log(Distance) | Packet size: 1600 Bytes |
| Log-Normal shadowing: 8 dB | Packet arrival mode: Constant bit rate data |
| Number of UEs: 25/35/45/55/65 | Inter-arrival time between packets: 3 ms |
| Number of BSs: 5. | **Non-GBR traffic** |
| **MBS settings** | QCI: 6 |
| Radio access technology: LTE | Proportion: 60% |
| Carrier frequency: 0.8 GHz | Data length: 250 KB |
| Transmission power: 40 W | Packet size: 3200 Bytes |
| Number of MBS: 1 | Packet arrival mode: Constant bit rate data |
| Bandwidth: 10 MHz | Inter-arrival time between packets: 3 ms |
| Channel model: 3GPP urban macro | **Deep learning settings** |
| **SBS settings** | Initial learning Rate: 0.001 |
| Radio access technology: 5G NR | Discount factor: 0.5 |
| Carrier frequency: 3.5 GHz | Epsilon value: 0.05 |
| Transmission power: 20 W | Experience pool: 200 |
| Number of MBS: 4 | Batch size: 64 |
| Bandwidth: 20 MHz | Hidden layer: 2 |
| Channel model: 3GPP urban micro | Federated interval: 30 TTIs |

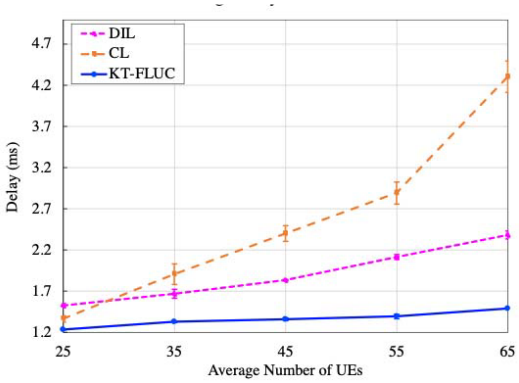



Figure 4.7: The average GBR traffic delay with different UE numbers.

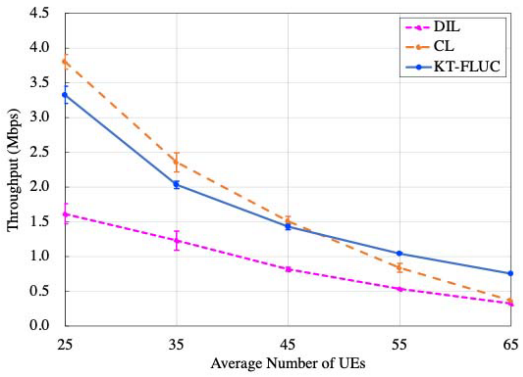



Figure 4.8: The average non-GBR traffic throughput.

In Fig. 4.7 and Fig. 4.8, the average delay of GBR traffic and throughput of non-GBR traffic under different numbers of UEs are shown. As the number of UE increases, the competition for resources among UEs becomes more intense, so the average delay will increase, and the average throughput will decrease. It can be observed that the CL algorithm performs better when there are fewer UEs. In contrast, UE-centric algorithms are influenced less by the increase in the number of UEs. This is because fewer UEs are easier to coordinate, and there is less information for the centralized model to deal with. When the average UE number is 25, the CL algorithm can achieve 14% higher throughput compared with the proposed KT-FLUC algorithm. However when the average UE number is 65, the proposed KT-FLUC algorithm can achieve 52% higher throughput than the CL algorithm. It can be observed that the delay of the CL algorithm is the highest among the three algorithms. This is because the cell-based algorithm tends to focus more on the overall performance of a cell rather than on the individual performance of UEs. Since the proportion of non-GBR traffic is much higher than GBR traffic, the CL algorithm may make decisions that are more suitable for non-GBR traffic at the expense of the benefits of GBR traffic to guarantee the overall reward. In comparison, in UE-centric algorithms, each UE fights for its own rewards, so it will choose the most appropriate actions for itself and makes the delay lower. When the average UE number is 65, the proposed KT-FLUC algorithm can achieve a 65% lower delay than the CL algorithm.

To further compare the cell-centric traffic steering and UE-centric traffic steering, we define three indicators to measure the UE's satisfaction with the traffic steering service, traffic blocking, QoS violation rate and the transmission cycle. Since GBR traffic has a higher priority than non-GBR traffic, non-GBR traffic may be blocked when two kinds of

traffic request transmission simultaneously. The traffic blocking represents the probability of GBR traffic being blocked. QoS violation rate represents the probability that the delay of GBR traffic exceeds the QoS requirements. And the last indicator, the transmission cycle, represents the average time for UEs to finish the transmission of a given file.

It can be observed from Fig. 4.9, Fig. 4.10,Fig. 4.11 that the UE-centric traffic steering can achieve a lower traffic blocking and lower QoS violation rate compared with cell-centric traffic steering algorithms. This verifies that the UE-centric traffic steering algorithm can respond more promptly to environmental changes. UE-based performance can give a more intuitive illustration of the UE's satisfaction than cell-based performance. The transmission cycle of UE-centric traffic steering is also smaller than cell-centric traffic steering, which means the traffic is transmitted faster in a UE-centric framework with more rational traffic steering strategies.

### 4.3.4 Performance Analysis between Four UE-centric Frameworks

In this section, we compare the simulation results from DIL, FL, FLI and proposed KT-FLUC algorithms to analyze the contribution of transfer learning and federated learning techniques and to verify that the combination of two algorithms can create synergy.

Fig. 4.12 shows the convergence curves of DIL, FL, FLI and proposed KT-FLUC algorithms when the number of UE is 45. It can be observed that the DIL algorithm achieves the lowest system reward among the four algorithms. Furthermore, the DIL curves are oscillating and unstable, with large fluctuation intervals. In comparison, the FL algorithms can achieve higher system reward, and the FL curves are smoother. This is because in the DIL algorithm, there is no experience sharing between UEs and some UEs need to spend more time exploring and trying non-optimal actions. UEs may choose some bad actions during the exploration process, thus causing large fluctuations in the network performance and the reward value. Compared with the FL algorithm, the FLI algorithm can achieve higher system reward. This is because the parameter initialization from the expert model to newcomers can help them start faster. Also, newcomers may contribute lousy experience to the global model which will interfere with the experience sharing if they fail to learn practical experience before the federated learning process. The initialization can prevent users from contributing lousy experience, so it will help enhance the system performance. This can demonstrate that in addition to the regular role in federated learning, the global model can also play an additional role as an expert model for transfer learning. Finally, it can be observed that our proposed KT-FLUC algorithm can achieve the highest system reward, lowest delay, and highest throughput. This verifies

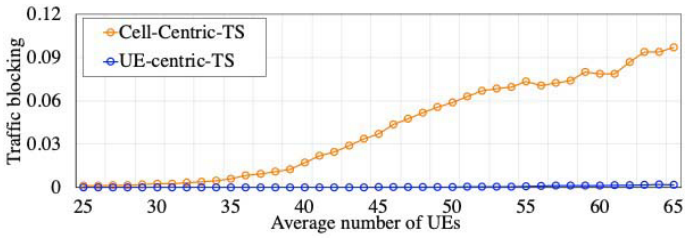



Figure 4.9: The average traffic blocking of UEs of cell-centric traffic steering and UE-centric traffic steering.

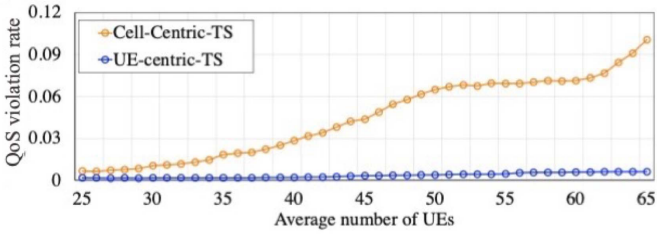



Figure 4.10: The average QoS violation rate of UEs of cell-centric traffic steering and UE-centric traffic steering.

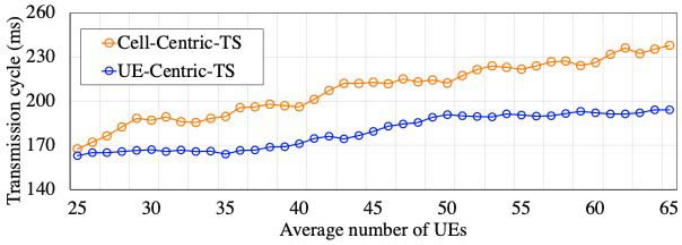



Figure 4.11: The average transmission cycle of UEs of cell-centric traffic steering and UE-centric traffic steering.

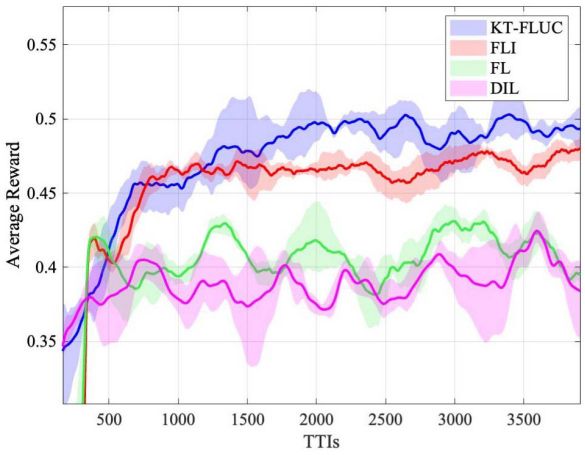



Figure 4.12: The convergence performance of system reward.

that the proposed knowledge transfer algorithm performs better than copying parameters directly. The combination of federated learning and knowledge transfer can further improve system performance compared with federated learning.

In Fig. 4.13 and Fig. 4.14, the average delay of GBR traffic and throughput of non-GBR traffic under different numbers of UEs are shown. With FLI and proposed KT-FLUC algorithms, there is no significant change in delay while the throughput will decrease when the number of UEs increases. This is because GBR traffic has a higher priority than non-GBR traffic, demonstrating that the traffic has been directed more effectively. In summary, the proposed KT-FLUC algorithm can consistently achieve the lowest packet delay and the highest throughput with different UE numbers. When the average UE number is 45, the proposed KT-FLUC algorithm can achieve 24% lower delay and 43% higher throughput compared with DIL algorithm. Also, with the proposed knowledge transfer scheme, it can achieve 10% lower delay and 10% higher throughput compared with FLI algorithm.

Table 4.2: Running time of KT-FLUC and FLI.

| | KT-FLUC | FLI |
|---|---|---|
| Federated learning time (s) | 25.0549 | 28.6939 |
| Knowledge transfer time (s) | 0.0291 | 0.0243 |
| Other time (s) | 1.9063 | 1.8299 |
| Total time (s) | 26.9903 | 30.5481 |

Table 4.2 further compares the running time of the KT-FLUC algorithm and the FLI

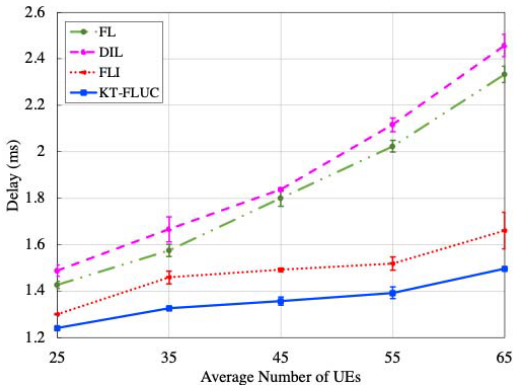



Figure 4.13: The average GBR traffic delay with different UE numbers.

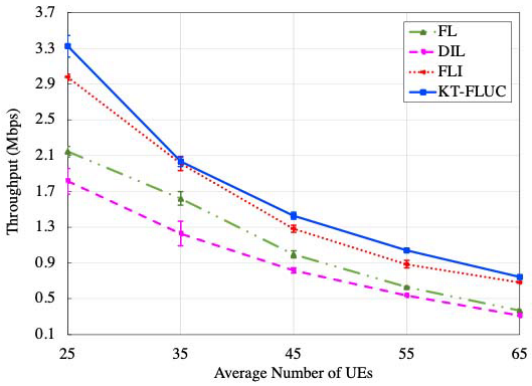



Figure 4.14: The average non-GBR traffic throughput.

algorithm. It shows the running time of different modules of the framework over 500 TTIs. In this table, the other time includes all other processes during the simulation, the resource block scheduling, and the data transmission. It can be observed that the knowledge transfer time of FLI is larger than KT-FLUC since the expert model will cause additional computational workload. However, the parallel running method used by KT-FLUC can efficiently reduce the federated learning time and total running time of the framework. Although extra calculation steps are introduced in KT-FLUC, it can still reduce the total running by 10%.

# Chapter 5

# Attacks and defense over federated DRL-enabled cell sleep control scenario

This chapter aims to investigate the attack models in a FL-enabled wireless network. Specifically, we introduce a federated DRL-based cell sleep control scenario that enhances the energy efficiency of the network. We propose multiple intelligent attacks targeting the learning-based approach and we propose defense methods to mitigate such attacks.

We designed two attack models, generative adversarial networks (GAN)-enhanced model poisoning attack and regularization-based model poisoning attack. As a counteraction, we have proposed two defense schemes, autoencoder-based defense, and KD-enabled defense. The autoencoder-based defense method leverages an autoencoder to identify the malicious participants and only aggregates the parameters of benign local models during the global aggregation, while KD-based defense protects the model from attacks by controlling the knowledge transferred between the global model and local models. The simulation results demonstrate that the proposed attacks can degrade the network performance by 34% and 77%, and lead to lower throughput and energy efficiency. On the other hand, our proposed defense schemes can effectively protect the system from attacks. The system performance can be recovered to approximately 95% of a secure system by using the proposed KD-based defense.

## 5.1 System Model and Problem Formulation

As shown in Fig. 5.1, we consider a heterogeneous network environment with one MBS and several SBSs. To save energy, we perform dynamic sleep control on each SBSs through

its local model, while the MBS will stay active to ensure coverage and control of data services. On top of the deep learning models, the FL algorithm is performed to enable a privacy-preserving collaboration between all the local models. During the FL process, the MBS serves as the global server, aggregating models from SBSs and sending feedback to SBSs, which can accelerate local training.

To investigate the intelligent attacks and defense methods in the proposed scenario, we assume that an attacker can compromise the cell sleep control application of more than one SBS [170]. It can manipulate the local model parameters of compromised SBSs during the training process. As shown in Fig. 5.1, the sleep control application in the second SBS is compromised, so it provides malicious local model parameters to the global server during a single update in FL. The malicious model parameters will then be aggregated into the global model and fed to other local models, causing SBSs to choose non-optimal actions.

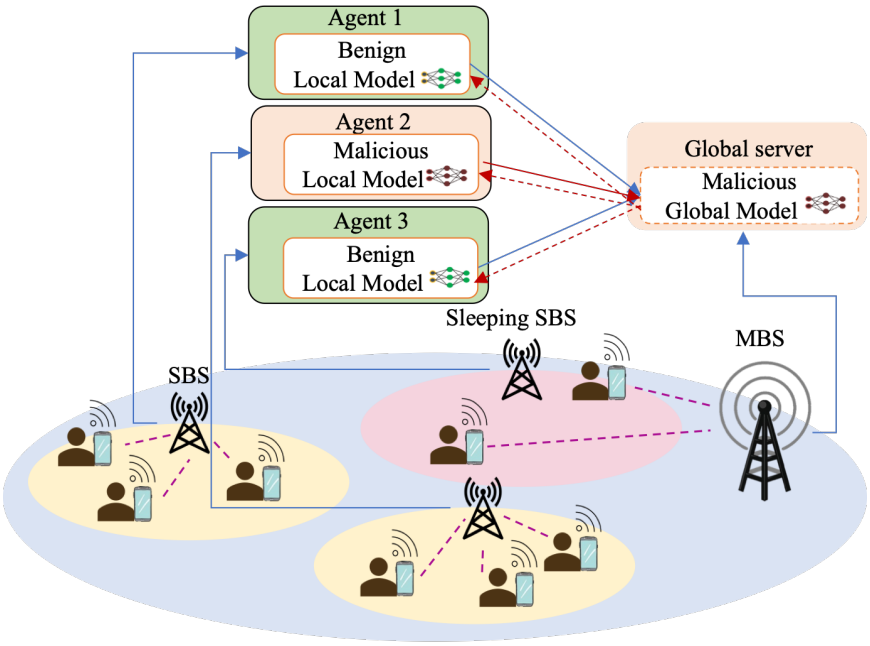



Figure 5.1: Attacks in federated deep reinforcement learning-enabled cell sleep control scenario.

### 5.1.1 System Model

We consider a downlink orthogonal frequency-division multiplexing cellular system and the link capacity between the $n^{th}$ SBS and the $k^{th}$ UE can be given as:

$$C_{n,k} = \sum_{m=1}^{R_n} B_m log_2(1 + SINR_{n,k,m}), \tag{5.1}$$

where $R_n$ is the set of physical resource blocks (PRBs) allocated to the $n^{th}$ SBS. $B_m$ is the bandwidth of the $m^{th}$ PRB and $SINR_{n,k,m}$ is the signal interference noise ratio on the $m^{th}$ resource block. It can be formulated as:

$$\text{SINR}_{n,k,m} = \frac{\beta_{n,k,m} g_{n,k} P_n^T}{\sum_{\substack{n'=1 \\ n' \neq n}}^{N} \sum_{k'=1}^{K_{n'}} \beta_{n',k',m} g_{n',k} P_{n'}^T + B_m N_0} \tag{5.2}$$

where $\beta_{n,k,m}$ denotes whether the $r^{th}$ resource block is allocated to the $k^{th}$ UE. $g_{n,k}$ is the channel gain of the transmission link. $P_n^T$ is the transmission power and $N_0$ is the noise power density.

According to [171], the energy consumption model of the $n^{th}$ SBS can be formulated as:

$$P_n = P_n^T + P_n^{AM} + P_n^{MP} + P_n^{FPGA}, \tag{5.3}$$

where $P_n^T$ is the transmission power. $P_n^{AM}$, $P_n^{MP}$, and $P_n^{FPGA}$ respectively denote the energy consumption of the power amplifier, microprocessor, and FPGA. The energy efficiency can then be calculated as:

$$EE = \frac{\sum_n \delta_n b_n + b_0}{\sum_n P_n + P_0}, \tag{5.4}$$

where $\delta_n$ is a binary variable used to indicate the sleep status of the $n^{th}$ SBS. $b_n$ denotes the throughput of the $n^{th}$ SBS and $b_0$ denotes the throughput of the MBS. In our simulation, the throughput is determined by both the traffic pattern and the link capacity between BSs and UEs. $P_0$ and $P_n$ denotes the energy consumption of the MBS and the $n^{th}$ SBS.

### 5.1.2 Problem Formulation

To save energy, we define three sleep modes for SBSs, active, sleep, and deep sleep. In active mode, SBSs are in full operation and consume the most energy. In sleep mode,

the front end of the radio frequency transmitter is shut down to save about half of the energy consumption [171]. Other hardware remains operational to ensure SBSs can be quickly woken up. In deep sleep mode, only the least amount of necessary energy will be consumed, but SBSs take longer to wake up.

The optimization objective of sleep control is to achieve high energy efficiency. The problem can be formulated as follows:

$$\max_{a_n} \ EE, \tag{5.5}$$

$$s.t. \ (5.1) \ (5.2) \ (5.3) \ and \ (5.4) \tag{5a}$$

$$a_n \in \{0, 1, 2\}, \ \forall n \in \{1, 2, ..., N\} \tag{5b}$$

$$\delta_n = \begin{cases} 1, & if \ a_n = 0, \\ 0, & else. \end{cases} \tag{5c}$$

$$P_n = \begin{cases} P_w, & if \ a_n = 0, \\ \gamma_1 P_w, & if \ a_n = 1, \\ \gamma_2 P_w, & if \ a_n = 2, \end{cases} \tag{5d}$$

where $a_n$ denotes the sleeping modes of the $n^{th}$ SBS. $P_w$ denotes the energy consumption of the SBS in full operation. $\delta_n$ is a binary variable indicating whether the $n^{th}$ SBS is sleeping or not. $\gamma_1$ and $\gamma_2$ denote the ratio of energy consumption in the sleep mode and the deep sleep mode.

### 5.1.3 Federated Learning for Cell Sleep Control

We consider each SBS as a distributed agent and set up a FL framework. It aims to make distributed agents cooperatively train a shared global model while keeping local data private. This cooperative training approach allows SBSs to train a generalized DQN model with limited local data that can adapt to the dynamic changes in the environment. The MDP of each local DQN can be given as:

- State: The state of sleep control is composed of four components, which are the sleeping status of the SBS $\delta_n$, the traffic load of the SBS and MBS in the past 5 transmission time intervals, and the current throughput of the SBS. It can be formulated as:

$$s_n = \{\delta_n, L_n, L_0, b_n\}, \forall n \in \{1, 2, ..., N\}. \tag{5.6}$$

Among these components, $L_n$ denotes the traffic load of the SBS in the past 5 transmission time intervals. It indicates the traffic pattern of the SBS. $L_0$ is the traffic load of the MBS in the past 5 transmission time intervals. It indicates the ability of the MBS to perform transmission tasks when the SBS is in sleep mode. $b_n$ is the current throughput of the SBS. The SBS can change the sleep mode accordingly to deal with the incoming traffic.

- Action: The action is to choose an optimal sleeping mode for the SBS, which can be formulated as:

$$a_n = \{0, 1, 2\}. \forall n \in \{1, 2, ..., N\}. \tag{5.7}$$

- Reward: The reward function is defined as a combination of throughput, packet loss rate, and energy cost, which can be given as:

$$r_n = \omega_1 b_n - \omega_2 \epsilon_n - \omega_3 P_n, \forall n \in \{1, 2, ..., N\}, \tag{5.8}$$

where $\epsilon_m$ is the packet loss rate of the $m^{th}$ UE. $\omega_1$, $\omega_2$ and $\omega_3$ are the coefficients to balance different rewards.

In each FL cycle, the local models at SBSs will first perform local training according to local experience. The local training process can be given as:

$$\theta_n^{t+1} = \theta_n^t + \alpha[r_n^t + \gamma \max_a Q(s_n^{t+1}, a; \theta_n^t) - Q(s_n^t, a_n^t; \theta_n^t)] \nabla Q(s_n^t, a_n^t; \theta_n^t), \tag{5.9}$$

where $\theta_n$ denotes the local model parameters of the $n_{th}$ SBS, $\alpha$ denotes the learning rate and $\gamma$ denotes the discount factor. $Q(s_n^t, a_n^t; \theta_n^t)$ denotes the long-term expected reward of the $n_{th}$ SBS choosing the action $a_n^t$ under the state $s_n^t$.

After that, local parameters will be uploaded to the global server. A global model will then be generated by aggregating the local parameters. Since we set the same size of memory buffer for all the SBSs, the aggregation weight of each local model is the same and equal to $\frac{1}{N}$ in this scenario. The global aggregation can be defined as:

$$\theta_G^{t+1} = \sum_{n=1}^{N} \frac{1}{N} \theta_n^{t+1}, \tag{5.10}$$

where $\theta_G^{t+1}$ denotes the global model parameters and $\theta_n^{t+1}$ denotes the local model parameters of the $n^{th}$ SBS.

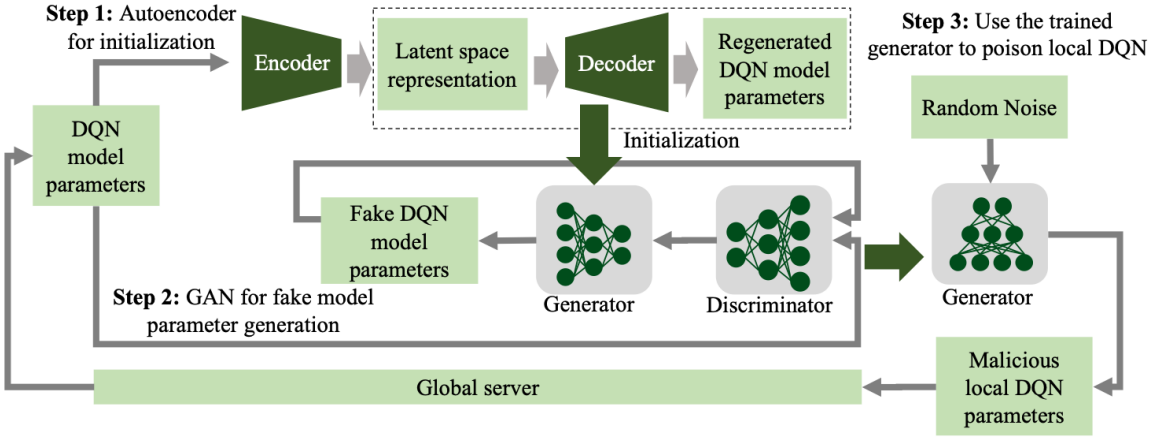



Figure 5.2: The procedure of GAN-enhanced model poisoning attack.

## 5.2 Intelligent Attacks: GAN-enhanced and Regularization-based Attacks.

This section introduces two proposed intelligent attacks in the federated deep reinforcement learning-based sleep control scenario, the GAN-enhanced model poisoning attack, and the regularization-based model poisoning attack.

### 5.2.1 GAN-enhanced Model Poisoning Attack

In regular data poisoning attacks, attackers destroy model parameters by polluting training data. These attacks can successfully attack the wireless system when there are enough compromised SBSs. However, one downside is that the distribution of malicious model parameters may differ from benign model parameters. Therefore, malicious local models can be easily detected and defended by checking the similarity of all the local models with Byzantine-robust aggregation rules [172]. In contrast, a more stealthy and effective attack strategy is to exploit the strength of the GAN technique and generate malicious fake model parameters with similar distributions to benign model parameters.

Fig. 5.2 illustrates our proposed GANs-enhanced model poisoning attack. The whole procedure can be divided into training and attack phases. The training phase includes autoencoder-based initialization and GAN training for fake model parameter generation. The attack phase is to use the trained generator to poison local DQNs.

One of the main difficulties of generating fake model parameters using GAN is the high dimensionality of the model parameters and the limited number of training samples. Since the only source of real model parameter samples for the compromised SBSs is through the global model distribution, the compromised SBSs can only obtain one training sample during each iteration. That means the GAN of attackers only has limited training samples. To handle such difficulty, GAN is only used to generate parameters of the bottom layer of the local model instead of the whole model.

In the first iterations, the compromised SBSs remain in regular training and accumulate real model parameter samples until sufficient training samples are available. Then they are trained on the autoencoder with collected training samples. An autoencoder is a self-supervised learning algorithm designed to enable neural networks to learn to compress and reconstruct high-dimensional data by using the same inputs and outputs. The problem can be formulated as:

$$\underset{\theta_1,\theta_2}{argmin}\ L(x_{input}, f_2(f_1(x_{input};\theta_1);\theta_2)), \tag{5.11}$$

where $L$ denotes the loss function. $f_1$ and $f_2$ denote the encoder and the decoder. $x_{input}$ denotes the input data. $\theta_1$ and $\theta_2$ denote the model parameters of the encoder and the decoder. After training, the decoder can reconstruct the high-dimensional data from the low-dimensional compressed representations. Therefore, it could be an initialization to the generator to ensure the GAN will start training from a good initial solution [173].

The core idea of GAN is to use two neural networks, a generator network, and a discriminator network, to perform adversarial training. The task of the generator network is to generate fake model parameters that the discriminator could not identify. Therefore, the loss of the generator network over a training set $\mathcal{K}$ can be calculated as:

$$L_G(\theta_G) = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}(\log(1 - D(G(z_k;\theta_G);\theta_D))), \tag{5.12}$$

where $\theta_G$ and $\theta_D$ denote the model parameters of the generator and the discriminator. $z_k$ denotes the input of the generator, which is randomly generated noise. On the other hand, the task of the discriminator network is to identify fake model parameters from real ones. The loss of the discriminator network can be calculated as:

$$L_D(\theta_D) = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}(\log D(x_k;\theta_D) + \log(1 - D(G(z_k;\theta_G);\theta_D))),$$

where $x_k$ denotes the real model parameter samples.

After training the GAN, we enter the attack phase and use the generator to generate malicious local model parameters. In the attack phase, the random noise generation pattern differs from the input noise in the training phase. This ensures the generator output will have a similar distribution but contain different information from the other benign local parameters. The output will then replace the parameters of the global model received from the last model aggregation and submit the new malicious local model to the global server.

By generating fake model parameters with GAN, the malicious model parameters will share a similar distribution with benign model parameters. Therefore, it is difficult for the defenders to defend against GAN-enhanced model poisoning attacks by comparing the similarity between local parameters. This makes malicious models more stealthy and more resistant to detection.

### 5.2.2 Regularization-based Model Poisoning Attack

Another promising and more concise approach to attack the FL system is to directly change the direction of gradient descent of malicious local models by changing the loss function of the compromised SBSs. On the other hand, we expect the attack to be stealthy enough and be able to still successfully attack the system under byzantine-robust FL aggregation rules [172]. Therefore, there are two criteria to follow while designing the loss function of malicious models. Firstly, the output of malicious models is expected to be different from the benign models. Secondly, the model parameters of malicious models are expected to be close to the benign models [94] [96].

The first criterion can be achieved by maximizing the loss function instead of minimizing it during local training. The second criterion can be achieved by adding a regularization term to the loss function to limit the distance between the malicious and benign model parameters. Therefore, the loss function can be re-formulated as:

$$L(\theta^t_{Local}) = (r^t + \gamma \max_a Q(s^{t+1}, a; \theta^t_{Local}) + Q(s^t, a^t; \theta^t_{Local})))^2 + \omega ||\theta^t_{Local} - \theta^t_{Global}||_2, \tag{5.13}$$

where $\theta_{Local}$ and $\theta_{Global}$ denote the local and global parameters. $r$ is the reward. $\gamma$ is the discount factor of reinforcement learning. $||.||$ denotes the L2-Squared distance. $\omega$ is a coefficient used to balance two sources of the loss. The local update of malicious models can be formulated as:

$$\begin{aligned}\theta^{t+1}_{Local} = \theta^t_{Local} - 2\alpha(r + \gamma \max_a Q(s^{t+1}, a; \theta^t_{Local}) + Q(s^t, a^t; \theta^t_{Local}))\nabla Q(s^t, a^t; \theta^t_{Local}) \\ + Q(s^t, a^t; \theta^t_{Local})) - 2\omega\alpha(\theta^t_{Local} - \theta^t_{Global}),\end{aligned}$$

where $\alpha$ denotes the learning rate. $\nabla(.)$ denotes the differential value.

To minimize the first term of the loss function, the output Q values of the network will represent the complementary values of the long-term reward for given actions in given states. By choosing the action with the maximum Q value, the malicious model will select the action that minimizes the long-term reward. In this way, the compromised SBSs will generate malicious model parameters during local training with a modified loss function, leading to attacks on the system through malicious local models.

### 5.2.3 Baseline Method: Data Poisoning Attack

To make a comparison with two proposed intelligent attacks, we use a non-intelligent data poisoning attack as a baseline. It is implemented by poisoning some of the MDP data in the local memory buffer of the compromised SBSs so that malicious local models are trained in the wrong directions [92]. According to the settings of the given scenario, high reward values are usually scarce but contain critical information. Therefore, in the sleep control application, the data poisoning attacker can select training records with high reward values from the local memory buffer of the compromised SBSs and poison the data by turning positive rewards into negative rewards. In this way, by poisoning only a small fraction of the data, the attacker can make malicious local models learn and share the opposite experience to the global model and benign local models.

### 5.2.4 Computational complexity and overhead analysis

In this subsection, we analyze the computational complexity and the overhead of two proposed attacks. For the GAN-enhanced model poisoning attack, the main computation comes from the training and inference of GAN models. The complexity of the GAN model is proportional to the model layers and the number of neurons of both the generator and the discriminator. So it can be given as $O(\sum_{i=1}^{I_G} N_{i-1}^G N_i^G + \sum_{D_i=1}^{I_D} N_{i-1}^D N_i^D)$, where $N_i^G$ denotes the number of neurons of the generator at the $i^{th}$ layer. $N_i^D$ denotes the number of neurons of the discriminator at the $i^{th}$ layer. $I_G$ and $I_D$ denote the number of layers of the generator and the discriminator.

Regularization-based model poisoning attack brings almost no extra computations. The only added step is to calculate the Euclidean norm between the local model parameters and the global model parameters and the computational complexity is proportional to the dimension of model parameters $len(\theta)$.

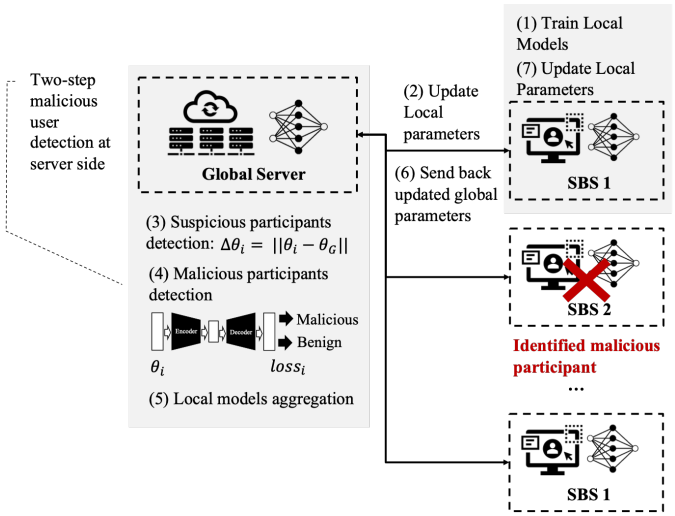



Figure 5.3: The procedure of autoencoder-based two-step defense.

Neither of the attacks introduces additional overhead to the system. The same as FedAvg, the only information exchanged between the global server and the local servers is the local model parameters and the global model parameters.

## 5.3 Intelligent Defense Methods: Autoencoder and KD-based Defense

This section introduces two proposed defense schemes in federated deep reinforcement learning-based sleep control scenario, an autoencoder-based two-step defense, and a KD-based defense. The first defense scheme is performed at the global server and is based on the similarity of local model parameters. The second defense scheme is mainly performed on the client side as a combination of heterogeneous FL and local data-based validation.

### 5.3.1 Autoencoder-based Two-step Defense

The procedure of an autoencoder-based two-step defense scheme is shown in Fig. 5.3. The core idea is to make the global server find the malicious participants according to the parameters of all the local models and only aggregate parameters of benign local models during the global aggregation.

To identify the malicious models from the benign ones, we perform a two-step malicious user detection on the global server side. In the first step, a coarse detection is performed to make a fast decision on the reliable participant list. After receiving local model parameters, the global server will calculate the distance of parameters of each local model from the previous update, which can be calculated as:

$$G_n^{t+1} = \left\| \theta_n^{t+1} - \theta_{Global}^{t} \right\|_2, \tag{5.14}$$

where $\theta_n^t$ denotes the parameter of the $n^{th}$ local model at the $t^{th}$ iteration. The similarity between the two local models can then be measured as

$$D_{nk}^{t+1} = \left\| G_n^{t+1} - G_k^{t+1} \right\|_2. \tag{5.15}$$

After that, we calculate the average distance of each local model from the other local models:

$$D_n^{t+1} = \frac{1}{N} \sum_{k=1}^{N} D_{nk}^{t+1}. \tag{5.16}$$

Then we sort all the local models according to their average distance from the other local models, and only a few local models with the smallest average distance are selected as a reliable local model set.

In the second step, an autoencoder is used to make a more accurate decision about the benign participant list. This is based on the hypothesis that if a data pattern is used in training, the model would be expected to have a lower loss than a data pattern not present in training datasets [174]. By using an autoencoder, we compare the differences between local model parameter data patterns instead of directly comparing the distances between local model parameters. This can reduce some of the impact of data heterogeneity on the defense.

First, an autoencoder is trained by the parameters of models in the reliable local model set. The autoencoder is then used to reconstruct the parameters of all the local models, and the reconstruction error can be formulated as:

$$E_i = ||\theta_{local}^i - f_2(f_1(\theta_n))||_2, \tag{5.17}$$

where $f_1$ and $f_2$ denote the encoder and the decoder. If the reconstruction error of a local model is larger than the average reconstruction error of all the local models, then the local model will be identified as the malicious local model and will not be included in the global model aggregation. In this way, the global server can prevent the compromised SBSs from poisoning the global model and other benign SBSs.

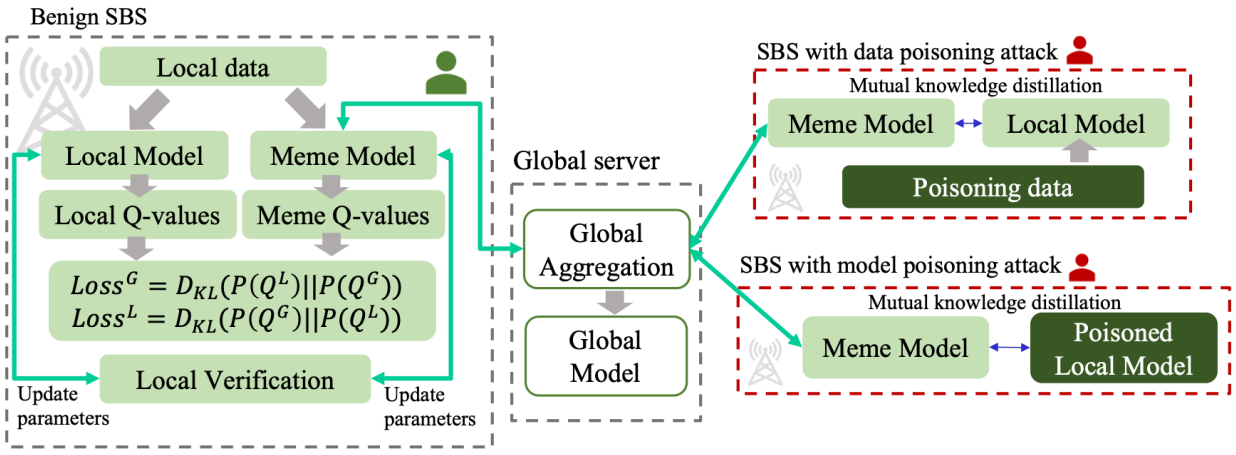



Figure 5.4: The structure of KD-based defense against malicious participants in FL.

### 5.3.2 Knowledge Distillation-based Defense

Although identifying malicious participants according to the similarity of model parameters is a good defense against many attacks, it may lose effectiveness when the attacker purposely designs stealthy attacks [96]. Therefore, we proposed another KD-based defense to defend against attacks on the client sides.

The procedure of the proposed defense scheme is shown in Fig. 5.4. With this defense mechanism, we can provide dual protection for local models. Instead of directly aggregating local models, we set up a meme model for each SBS as a copy of the global model and perform mutual KD between the local model and the meme model. During the mutual KD process, models deliver only vital knowledge, and poisoning factors may be filtered out. This is the first level of protection in this scheme. Other than that, a local verification step is set up for each SBS to determine the direction of knowledge flow between the meme model and the local model. It can be performed by measuring the similarity between the local and meme model outputs based on Kullback–Leibler (KL) divergence. This is the second level of protection.

During each iteration, the updated parameters of the global model are copied to the meme models of all the SBS. The local training data is then fed to the local and meme models to generate local Q and meme-Q values. The KL divergence between the policies held by the local and meme models can be calculated by converting the Q-values into a

probability distribution with a softmax function [175]. It can be formulated as:

$$D_{KL}(p^L||p^M) = \sum_{i \in a} softmax(q_i^L) \ln \frac{softmax(q_i^L)}{softmax(q_i^M)}, \tag{5.18}$$

$$D_{KL}(p^M||p^L) = \sum_{i \in a} softmax(q_i^M) \ln \frac{softmax(q_i^M)}{softmax(q_i^L)}, \tag{5.19}$$

where $\mathbf{a}$ is the action space. $q_i^L$ denotes the output Q values of the local model. $q_i^M$ denotes the output $q$ values of the meme model. It can be given as:

$$\mathbf{q}^L = Q(s, a; \theta_{Local}), \tag{5.20}$$

$$\mathbf{q}^M = Q(s, a; \theta_{Meme}). \tag{5.21}$$

To prevent malicious local models from poisoning benign models through global model aggregation, we perform a local verification by comparing the KL divergence value with a threshold value. If $KL(p^M||p^L) > \Theta$, the global model is supposed to be quite different from the local model. This means that based on the validation results of the local data, the global model may have already been poisoned. Therefore, we only perform a one-way KD from the local model to the global model and stop KD from the global model to the local model. This can be implemented by using the KL divergence as the loss function of the meme model, which can be formulated as:

$$L(\theta_{Meme}) = D_{KL}(p^L||p^M; \theta_{Meme}), \tag{5.22}$$

while the local model will keep using the conventional DQN loss function, which is only related to local data and not to KL divergence.

On the other hand, if $KL(p^M||p^L) < \Theta$, the local model will distill knowledge from the global model by adding the KL divergence to the loss function of the local model, which can be formulated as:

$$L(\theta_{local}^t) = \xi(r^t + \gamma \max_a Q(s^{t+1}, a; \theta_{Local}^t) - Q(s^t, a^t; \theta_{Local}^t))^2 + (1 - \xi) D_{KL}(p^M||p^L; \theta_{Local}^t), \tag{5.23}$$

where $\xi$ is a parameter used to balance the weights between global and local knowledge. At the same time, the KD from the local model to the global model is stopped. In this way, the global model will avoid learning similar knowledge repeatedly and help to learn more knowledge that is different from the current knowledge.

By setting up the local verification scheme, it is guaranteed that there is an upper limit to the effectiveness of the attacks on the benign models involved in FL. This upper limit can be quantified regarding the probability of the model choosing a non-optimal action. The proof is given as follows.

*Preposition* 1: The upper bound of the probability of a benign local model choosing a non-optimal action under any type of attack with the proposed defense scheme is:

$$P_{NO} = \frac{\Theta}{\overline{E_0}}, \tag{5.24}$$

where:

$$E_0 = -(e^{q_1} + e^{q_2}) \ln(e^{q_1} + e^{q_2}) + e^{q_1} \ln(2e^{q_1}) + e^{q_2} \ln(2e^{q_2}). \tag{5.25}$$

*Proof*: To quantify the upper bound, we first assume that in the absence of an attack, the local data can always fully train the local model to output the best action. The sleep control model has three actions, denoted as $i = 1, 2, 3$. According to Eq. (5.19), we have the calculation of the KL divergence on a given batch training data $\mathcal{X}$ as follows:

$$D_{KL}(P^L||p^M) = \sum_{x \in \mathcal{X}} \sum_{i=1,2,3} softmax(q_i^L) \ln \frac{softmax(q_i^L)}{softmax(q_i^M)}). \tag{5.26}$$

By performing the local verification, the KL divergence of the local model and meme model is smaller than the threshold $\Theta$, which can be given as:

$$D_{KL}(P^L||p^M) \leq \Theta|X|. \tag{5.27}$$

Then we suppose there exists a MDP record $x_0 = S_0, a_0, r_0$ in the local training batch $X$, and the outputs of the local DQN model under $s_0$ with the largest, the second largest, and the smallest values are $q_1 > q_2 > q_3$. The output of meme DQN model under $s_0$ is $q_1', q_2', q_3'$. Then we have:

$$softmax(q_i^L) = \frac{e^{q_i}}{\sum_{j=1,2,3} e^{q_j}}, \tag{5.28}$$

$$softmax(q_i^M) = \frac{e^{q_i'}}{\sum_{j=1,2,3} e^{q_j'}}. \tag{5.29}$$

To make an effective attack under state $s_0$, the meme model is supposed to choose different actions from the local model. At the same time, the attacker wants to cause

as many attacks as possible within the limitation of the threshold, so it may expect a minimal KL divergence when performing an attack. We also suppose an ideal situation for the attacker, where the meme model can output any distribution the attacker wants. So we can formulate the problem as:

$$\begin{aligned} &\min_{q'_1,q'_2,q'_3} D^{x_0}_{KL}(P^L||p^M) \\ &s.t.\ \ q'_2 > q'_1 \ \ or \ \ q'_3 > q'_1. \end{aligned} \tag{5.30}$$

To solve this problem, the KL divergence can be firstly expanded as:

$$D^{x_0}_{KL}(P^L||p^M) = \sum_{i=1,2,3} [\frac{e^{q_i}}{\sum_{j=1,2,3} e^{q_j}}(q_i - \ln\left(\sum_{j=1,2,3} e^{q_j}\right) - q'_i + ln(\sum_{j=1,2,3} e^{q'_j}))]. \tag{5.31}$$

Since $q_1, q_2, q_3$ are fixed values, irrelevant items can be removed from the expression of KL divergence and simplify the optimization problem as:

$$\begin{aligned} &\min_{q'_1,q'_2,q'_3} \sum_{i=1,2,3} [e^{q_i} \ln\left(\sum_{j=1,2,3} e^{q'_j}\right) - q'_i e^{q_i}] \\ &s.t.\ \ q'_2 > q'_1 \ \ or \ \ q'_3 > q'_1. \end{aligned} \tag{5.32}$$

According to the definition of KL divergence, the minimum value is achieved when the output of two models have the same distribution, which is $q'_1, q'_2, q'_3 = q_1, q_2, q_3$. Since the optimization function is smoothly continuous, the value of the function will increase in any direction near the minimum value point. Since we also have the prerequisite $q_1 > q_2 > q_3$, the solution of the given optimization problem lies on the tipping point that makes $q'_1 = q'_2$ and $q'_3 = q_3$. Supposing $q'_1 = q'_2 = q^*$,

$$\hat{D^{x_0}}_{KL}(P^L||p^M) = \sum_{i=1,2,3} [e^{q_i} \ln\Big(2e^{q^*} + e^{q^3}\Big)] - q^*(e^{q_1} + e^{q_2}), \tag{5.33}$$

where $\hat{D}$ denotes the expression of KL after simplification. The derivation of $\hat{D^{x_0}}_{KL}(P^L||p^M)$ can then be given as:

$$\frac{\partial \hat{D^{x_0}}_{KL}(P^L||p^M)}{\partial q^*} = 2e^{q^*} \frac{\sum_{i=1,2,3} e^{q_i}}{2e^{q^*} + e^{q_3}} - e^{q_1} - e^{q_2}. \tag{5.34}$$

When the derivation is zero, the minimal value point of $q^*$ can be calculated as:

$$q^* = \frac{e^{q_1} + e^{q_2}}{2}. \tag{5.35}$$

The minimal KL divergence to make an effective attack, which is denoted as $E_0$, can be calculated as:

$$E_0 = -(e^{q_1} + e^{q_2}) \ln(e^{q_1} + e^{q_2}) + e^{q_1} \ln(2e^{q_1}) + e^{q_2} \ln(2e^{q_2}). \tag{5.36}$$

We suppose there is sufficient local training data that is uniformly distributed. The average minimal KL divergence to make an effective attack can be estimated by calculating the average $E_0$ over the batch training data $X$, which can be given as:

$$\overline{E_0} = -\frac{\sum_x \in X(e^{q_1} + e^{q_2}) \ln(e^{q_1} + e^{q_2})}{|X|} + \frac{\sum_x \in X e^{q_1} \ln(2e^{q_1}) + e^{q_2} \ln(2e^{q_2})}{|X|}. \tag{5.37}$$

According to Eq. (5.27), there is an upper bound for the KL divergence. That means the number of action choices that an attacker can change is limited, or the model will choose to learn locally without being affected by the attacks. Therefore, after enough training rounds, the upper bound of the probability of a benign local model choosing a non-optimal action can be estimated as:

$$P_{NO} = \frac{\Theta}{\overline{E_0}}. \tag{5.38}$$

This concludes the proof.

We can give an intuitive interpretation of this result. $E_0$ in the denominator part represents the average energy required for the model to be attacked and indicates the degree of vulnerability of the model to attack. The numerator is the threshold for constraining the dispersion, which regulates how closely the models cooperate in FL. These two factors determine the extent to which the benign models are attacked under the given defense mechanism in FL.

According to this proof, we demonstrate that the performance of our proposed KD-based defense method is not limited by the attack methods and can theoretically be effective against any type of poisoning attack.

### 5.3.3 Baseline Defense: Krum

To make a comparison with two proposed defense schemes, we use a Krum defense proposed in [172] as a baseline. In the Krum defense, the global server will calculate the average distance of each local model from the other local models according to Eq. (5.16) and choose the local model with the smallest average distance. The parameters of the global will be replaced by the parameters of the chosen local model in each iteration.

### 5.3.4 Computational complexity and overhead analysis

In this subsection, we analyze the computational complexity and the overhead of two proposed defense methods. In the autoencoder-based two-step defense method, the main computation comes from calculating the model similarity and training the autoencoder. The model similarity is measured by the Euclidean distance, and the computational complexity is proportional to the dimension of model parameters $len(\theta)$. The complexity of the autoencoder model is proportional to the model layers and the number of neurons. The computational complexity of the autoencoder-based two-step defense method can be formulated as $O(len(\theta) + \sum_{i=1}^{I_A} N_{i-1}^A N_i^A)$, where $N_i^A$ denotes the number of neurons of the autoencoder at the $i^{th}$ layer. $I_A$ denotes the number of layers of the autoencoder.

In the KD-based defense method, the main computation comes from the inference of the meme model and the calculation of the KL divergence. The computational complexity of meme model inference is proportional to the model layers and the number of neurons of the meme model. The computational complexity of KL divergence calculation is proportional to the size of the action space. So the computational complexity of the KD-based defense method is $O(len(a) + \sum_{i=1}^{I_M} N_{i-1}^M N_i^M)$, where $N_i^M$ denotes the number of neurons of the meme model and $len(a)$ denote the size of the action space.

Both of the defense methods do not introduce additional overhead to the system. In the KD-based defense method, instead of submitting the local model parameters to the global server, the meme model parameters are submitted. However, in this paper, since the two models are of the same size, the overhead remains unchanged.

### 5.3.5 Attackers and defenders analysis

In this section, we compare and analyze the knowledge of the attackers and the defenders to carry out different attacks and defenses. The following analysis demonstrates the feasibility of the proposed attack and defense methods.

For GAN-enhanced attacks, the only knowledge attackers have is the structure and parameters of the global model. They do not have any training data. This kind of knowledge can be obtained by intercepting the over-the-air transmission channel between the global server and the clients of FL. Specifically, like Sybil attacks on distributed systems, the attacker can inject fake participants into the FL system instead of controlling existing participants [95].

For regularization-based model poisoning attacks, the attackers are assumed to know the local training data of the malicious participants and the structure and parameters of

the global model. They do not have access to the local training data of other benign participants.

To perform autoencoder-based defense and KD-based defense, the global server has access to the local model parameters updated by each participant. It does not have access to the local datasets or other information about the participants. The participants have access to their local data and the shared global parameters. They do not have access to the data of other participants.

## 5.4 Numeric Results

### 5.4.1 Simulation Settings

To evaluate the effect of attacks and defenses under the wireless communication system, we first simulated a near-realistic 3rd Generation Partnership Project (3GPP) urban macro network environment and built the channel model and communication model with Python 3.8. Next, we modeled the data transmission between the SBS and the UE with the simulation platform, and in this process, MDP data was generated and collected at each SBS. These data are used as the training data for each local DQN model. In each TTI, a set of MDP data is collected by each SBS. During the simulation, we consider a heterogeneous network with 1 MBS and 20 SBSs. To configure a non-IID data distribution among local models, we set various numbers of UEs and different traffic loads for each SBS. The propagation path loss is set as $128.1 + 37.6log(Distance)$. The simulations are repeated in ten runs with a 95% confidence interval. The coefficients of rewards $\omega_1$, $\omega_2$, and $\omega_3$ are 0.25, 1, and 0.05. Other simulation details are given in Table 5.1. In each episode, we simulate a 24-hour typical residential area traffic pattern which is given from [176].

According to the MDP definitions, the dimension of the state and the next state is 16, and the dimension of the action and the reward is 1. Therefore, the dimension of the MDP data is 34. The input dimension of the DQN model is 16 and the output dimension is 3. The DQN has two layers, the first layer has 64 neurons and the second layer has 32 neurons.

We use two communication metrics to evaluate the results, the system throughput and the energy efficiency. The purpose of attacks is to reduce system throughput and energy efficiency, while the purpose of defense methods is to recover system throughput and energy efficiency to the values that it would have without the attacks. To better evaluate our proposed defense methods, we use a well-known Krum defense method proposed in [172] as a baseline for comparison.

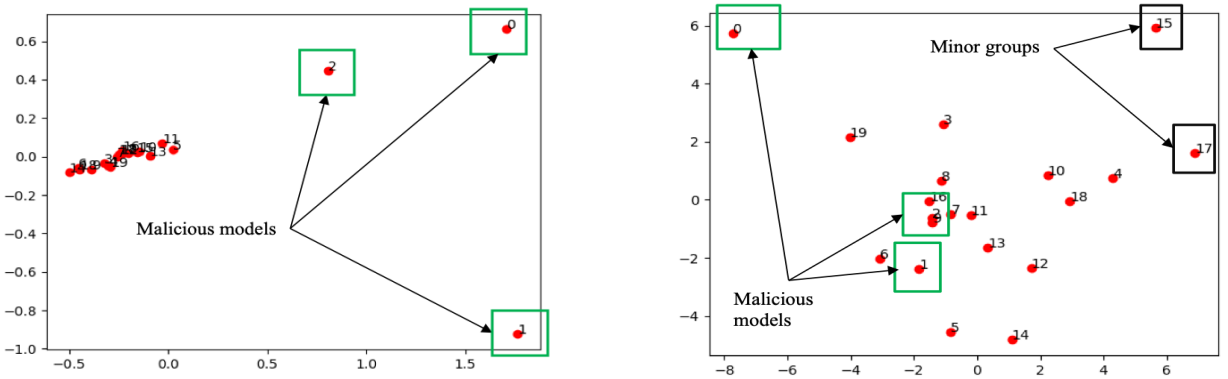
Table 5.1: Simulation settings.

| **Wireless network settings** | **Traffic load settings** |
|---|---|
| 3GPP urban macro network | Peak traffic load of each SBS: |
| Bandwidth: 20MHz | 12/14/16/18 Mbps |
| Number of RBs: 100 | **Sleep mode settings** |
| Subcarriers in each RB: 12 | Energy consumption of active mode: 100% |
| Subcarrier bandwidth: 15kHz | Energy consumption of sleep mode: 50% |
| **BS and UE settings** | Energy consumption of deep sleep mode: 15% |
| Number of SBS: 20 | **DQN settings** |
| Transmission power of MBS: 40W | Initial learning rate: 0.01 |
| Transmission power of SBSs: 4W | Discount factor: 0.8 |
| Carrier frequency of SBS: 3.5 GHz | Epsilon value: 0.05 |
| Carrier frequency of MBS: 2.0 GHz | Hidden layer: 2 |
| Number of UEs: 3-11 each SBS | Batch size: 256 |

### 5.4.2 Model distribution analyses

This section presents the model distribution analyses under FL with independent and identically distributed (IID) data and non-IID data. To generate IID data, we set identical traffic loads, UE numbers, and UE distribution of each SBS. In this way, the distributions of local models will be very close to each other. However, this is an ideal situation that is almost impossible to happen in practice. For non-IID data, we set different traffic loads, UE numbers, and UE distribution of each SBS. So distributions of local models will be far from each other. To visualize the high-dimensional model parameters data, we first perform the principal component analysis (PCA) to extract two principle components and project them into a two-dimensional coordinate system. Suppose we have $m$ participants and each participant has a DQN model with $n$ parameters. We consider it as a set of n-dimensional data with $m$ samples. During the PCA, the samples are first centralized. Then, the covariance matrix for all samples is calculated, and the eigenvalue decomposition of the matrix is performed. The eigenvectors corresponding to the two largest eigenvalues are taken out, and all the eigenvectors are normalized to form an eigenvector matrix. This matrix can transform the high-dimensional model parameter samples into two-dimensional points as the PCA result. The projection result after PCA is shown in Fig. 5.5. In this figure, the $x$ and $y$ axes are the values of two principal components of local model

parameters after PCA.




(a) The distribution of local models with IID data.

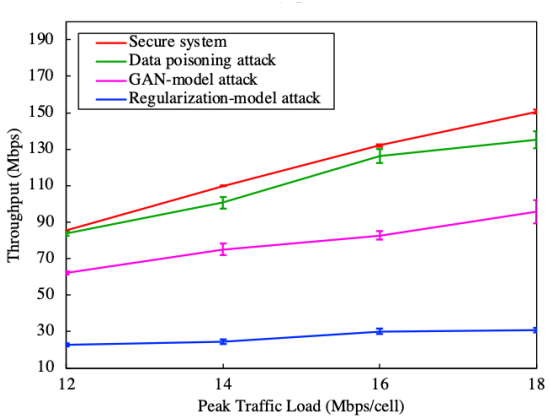
(b) The distribution of local models with non-IID data.

Figure 5.5: A comparison between IID and non-IID data.

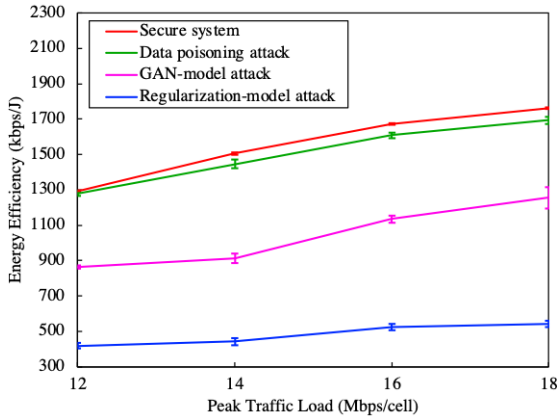
Fig. 5.5 (a) shows the distribution of local models with IID data, and Fig. 5.5 (b) shows the distribution of local models with non-IID data. We also perform a data poisoning attack on the first three local models. It can be observed that in Fig. 5.5 (a), benign models are very close to each other, and malicious models are far away from them. This illustrates that with IID data, malicious models can be easily identified. However, in Fig. 5.5 (b), benign models are not as close as the situation with IID data, and it is tough to intuitively identify the malicious models and benign models from model distribution. And benign models with heterogeneous data distribution from the major data, called minor groups, may be identified as malicious models. This demonstrates our motivation for starting this research that the attack and defense problems of FL with non-IID data are more complex and worth exploring than those of FL with IID data.

### 5.4.3 Performance of Attacks

Fig 5.6 shows the average downlink throughput of all the UEs and energy efficiency under two proposed intelligent attacks, GAN-based model poisoning attack and regularization-based model poisoning attack, and one baseline, data poisoning attack, when there are three attackers out of twenty participants. We also simulated with more attackers, and the results confirmed our findings. However, the number of attackers will be low in practice due to the cost of implementing attacks. The energy efficiency is the ratio of the transmission data of all UEs to the overall energy consumption of the system, which is defined in Eq. (5.4).




(a) The system throughput under different attacks.




(b) The energy efficiency under different attacks.

Figure 5.6: The system performance under three different attacks.

We also add the performance of the secure system for comparison. During the simulations, we change the peak traffic load of the SBS from 12 Mbps to 18 Mbps. From these figures, it can be concluded that among these attacks, model poisoning attacks are more effective than data poisoning attacks. In the proposed data poisoning attack, we changed about 5% of the MDP data. With three attackers, it only decreases the system throughput by 10.2% and reduces the energy efficiency by 3.9%. In contrast, the other two proposed model poisoning attacks have more substantial effects on the system performance. For the GAN-enhanced model poisoning attack, it decreases the system throughput by 36.5% and the energy efficiency by 28.7%. For the regularization-based model poisoning attack, it reduces the system throughput by 69.3% and the energy efficiency by 79.5%.

The reason why model poisoning attacks are more effective is that such attacks can directly change the parameters of the model, and the influence is more definitive. The data poisoning attacks, however, only take effect when poisoning data is selected for training in a local training iteration. Besides, the unpolluted data also places constraints on the model parameters when the model is updated so that the poisoned data do not necessarily influence the model parameters.

The effectiveness of data poisoning will indeed increase as the percentage of poisoned MDP data increases. However, this also means the malicious participants are easier to identify by defense schemes, which will be discussed in the following subsection. Therefore, it does not make sense to enhance the attack simply by increasing the amount of poisoned data. A more practical way to perform intelligent attacks is to design intelligent attacks like

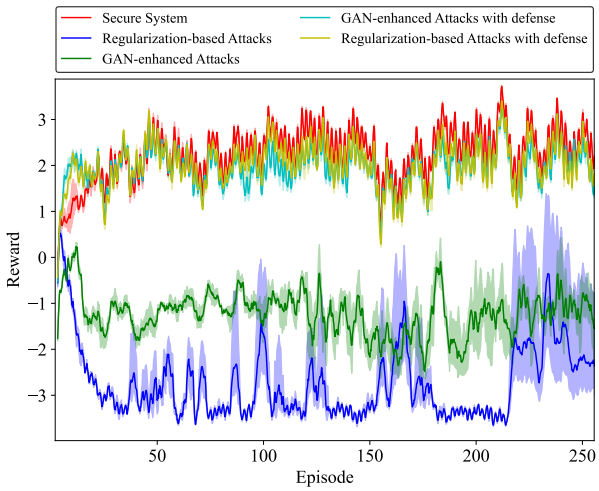



Figure 5.7: The convergence curves of the system reward without attacks, under GAN-based attacks, under regularization-based attacks, and proposed defense.

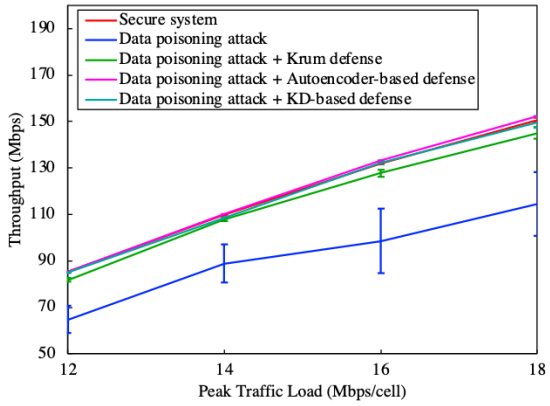
GAN-enhance model poisoning attacks and regularization-based model poisoning attacks.

Fig 5.7 shows the convergence curves of the system reward without attacks, under attacks, and under attacks and defense. We first compare the reward curve of the secure system and the curves of two different attacks. As can be observed in the figure, for the reward of FL without attacks, although there were some fluctuations due to the dynamics of the environment, the reward generally grows and eventually converges. In contrast, in the presence of attacks, the reward value decreases gradually over the process of training.

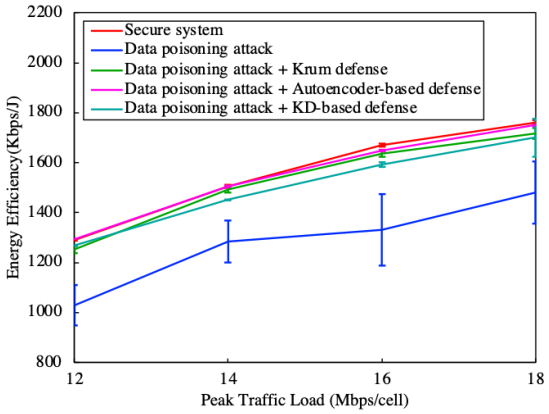
The regularization-based attack leads to a lower reward compared with GAN-enhanced attacks. This is because the regulation-based attack maximizes the loss function during training, which makes it a more proactive and aggressive attack. In contrast, the GAN-enhanced attack is carried out by generating synthetic fake model parameters. These fake model parameters do not contain specific meanings such as increasing the system loss, but only serve as a disturbance to the global model. So the GAN-based attack is not as aggressive as the regularization-based attack. Also, a noticeable oscillation in the reward values of the regularization-based attack is shown. This is because the loss function of the regularization-based attack model consists of two components, an attack component and a regularization component. These two components act jointly and cause the system reward value to be unstable.

Moreover, Fig 5.7 also incorporates our proposed KD-based defense method for each of the two attacks, and the reward convergence curves after defense are shown. As it can be

observed, after the defense, the system reward is close to the reward of the secure system, which illustrates the effectiveness of our proposed defense method. In the next subsection, we provide more simulation results and an analysis of our proposed defense methods.




(a) The system throughput under different defense schemes.



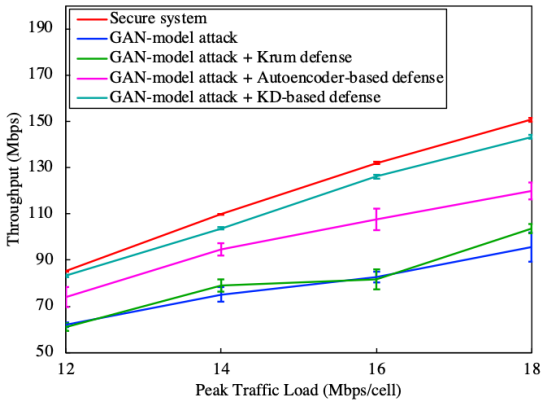

(b) The energy efficiency under different defense schemes.

Figure 5.8: The system performance under data poisoning attacks with different defense schemes.

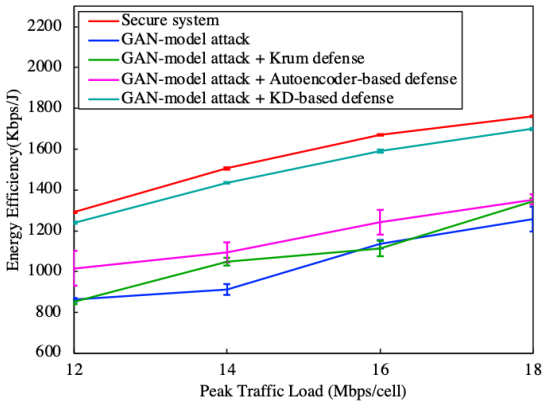
### 5.4.4 Performance of Defense Methods

This section investigates the performance of two proposed defense schemes, autoencoder-based defense and KD-based defense, and one baseline defense scheme, Krum-based defense, under three attacks mentioned in the last subsection. The two model poisoning attacks we proposed can be regarded as upgraded versions of existing poisoning attacks. According to the comparison with the data poisoning attack baseline, they are more intelligent and stealthy and, therefore, more difficult to defend against. So, the three attacks we used for simulation are sufficiently representative to validate the effectiveness of our proposed defense methods. While our defense methods are capable of protecting the models from these intelligent attacks, this implies that they are also effective against other common poisoning attacks.

Fig. 5.8 shows the system throughput and energy efficiency under data poisoning attacks with different defense schemes. We also show the performance of the secure system and the performance under attacks without any defense for comparison. The throughput and energy efficiency of the secure system are used as upper limits to compare with the




(a) The system throughput under different defense schemes.

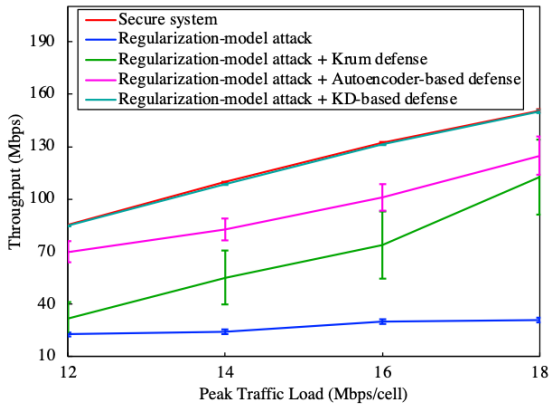



(b) The energy efficiency under different defense schemes.

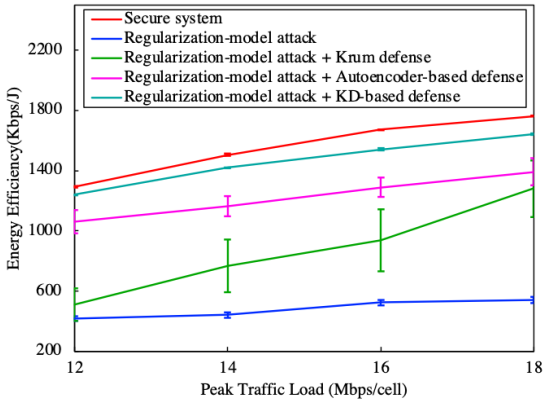
Figure 5.9: The system performance under GAN-enhanced model attacks with different defense schemes.

performance after defense. If the throughput and energy efficiency of the system is close to the upper limit under the secure system after defense, the defense method is considered effective. In this simulation, we increased the number of attackers to five to achieve a more noticeable attack effect and elevate the difficulty of defense. However, each defense scheme can still successfully defend against the data poisoning attack. Among these three defense schemes, the autoencoder-based defense scheme has the best effect and can almost reach the upper limit of the FL system. The other two defense schemes can also improve system performance to at least 95% of the throughput and energy efficiency in a secure system.

Fig. 5.9 shows the system throughput and energy efficiency under GAN-enhanced model poisoning attacks with different defense schemes. According to the simulation results discussed in the last subsection, three attackers are strong enough to bring noticeable attacks to the system performance under GAN-enhanced model poisoning attacks. So in this simulation, we set three attackers out of twenty participants. Among the three defense methods, it can be observed that Krum provides the weakest defense to the attacks. The throughput and energy efficiency after Krum defense are close to the values under attacks but without any defense scheme. In contrast, the effect of autoencoder-based defense is more prominent. It increases the system throughput by 29.9% and the energy efficiency by 9.4%. However, the system performance under this defense scheme is still much worse than a secure system. KD-based defense performs best of the three defense schemes and is closest to the upper limit of the FL system. It can improve the system performance to




(a) The system throughput under different defense schemes.




(b) The energy efficiency under different defense schemes.

Figure 5.10: The system performance under regularization-based model attacks with different defense schemes.

95.3% of the throughput and 95.7% of the energy efficiency in a secure system.

Fig. 5.10 shows the system throughput and energy efficiency under regularization-based model poisoning attacks with different defense schemes. It can be observed that all three defense schemes can provide defensive effect. Among the three defense methods, Krum defense performs worst. In a best-case scenario, it can only improve the system performance to 74.6% of the throughput and 72.3% of the energy efficiency in a secure system. Autoencoder-based defense scheme performs better and it can improve the system performance to 81.17% of the throughput and 82.3% of the energy efficiency in a secure system. Moreover, Krum defense is more favorable when the traffic load is high, while autoencoder-based defense is more advantageous when the traffic load is low. This also illustrates the instability of these two defense schemes, which means the effectiveness of the defense methods can be affected by environmental parameters. In contrast, KD-based defense is the most stable and effective defense scheme. It can increase the energy efficiency to at least 94.2% of the performance in the secure system and achieve almost the same throughput performance as the secure system.

Finally, Fig 5.11 and Fig 5.12 show a dynamic process of how the energy efficiency and the throughput change under intelligent attacks and defense methods. In this simulation, we take the regularization-based model poisoning attack and KD-based defense as examples. In previous simulations, both attack and defense are present throughout the simulation. Differently, in this simulation, we use a special way to conduct the simulation

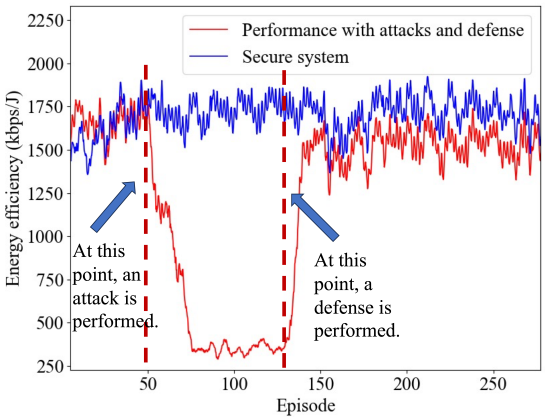



Figure 5.11: The dynamic change of energy efficiency after attacks and defense.

and add the attack and defense in the middle of the simulation to show a more dynamic attack and defense process. At about 50 episodes, we start an intelligent attack, and the attacker learns to degrade the system performance by decreasing the energy efficiency. At about 125 episodes, we start an intelligent defense scheme. The defense scheme can protect the system against attacks and increase energy efficiency.

In summary, the GAN-enhanced model poisoning attack and regularization-based model poisoning attack are more difficult to defend than the data poisoning attack. According to these results, we analyze the advantages and applicable scenarios of two proposed defense methods. First, the autoencoder-based defense method can identify the malicious participants while the KD-based defense method cannot. If the defender wants to know which participants have been attacked, the autoencoder-based defense method is the more appropriate defense method. Second, the autoencoder-based defense method performs better for simpler attack strategies where the attackers do not take any steps to make the attack more stealthy, such as the data poisoning attack [92] [177]. This is because the autoencoder-based defense method detects malicious participants mainly relying on the similarity between local model parameters. Under these attacks, the local model parameters of malicious participants are largely different from benign participants and can be easily identified by autoencoders. But for more advanced attacks such as the two intelligent attacks we propose, extra strategies are added to make the local model parameters of malicious participants close to those of benign participants and to make attacks more stealthy [96]. In this way, these malicious local model parameters sometimes cannot be easily detected by autoencoders, and the performance of the autoencoder-based defense method for these attacks is worse.

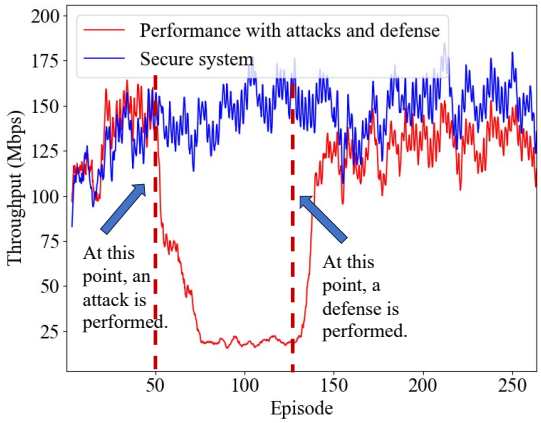



Figure 5.12: The dynamic change of system throughput after attacks and defense.

The advantage of using the autoencoder-based defense method for simpler attack strategies is that it can restore system performance to an unattacked state to a higher extent. Once the autoencoder identifies the malicious participants, the malicious local parameters are excluded from the global aggregation process. In this way, the impact of the attacks is minimized. In contrast, the KD-based defense method defends against attacks by distilling benign information. In the process of distillation, some useful information may also be lost, which makes the KD-based defense method less effective than the autoencoder-based defense method under simpler attacks. However, since the KD-based defense method does not require identifying the malicious participant, the defense effectiveness is less affected by the stealthiness of the attacker. Therefore it demonstrates better results with more advanced attacks. In addition, the KD-based defense method sets up a local validation mechanism to detect the impact of attacks on the model, and it is theoretically effective against any type of poisoning attack. We also proved that there exists an upper bound on the effectiveness of attacks on the benign models involved in FL when KD-based defense is applied. As a result, the KD-based defense method can still provide a relatively strong defense against intelligent attacks and prevent them from significantly degrading the performance of the system.

# Chapter 6

# Dual attention model-weighted robust FL in mobile communication systems

In this chapter, we aim to design a versatile FL framework that simultaneously promotes the performance of the model both in a secure system and under attack for robust radio modulation classification. To this end, we leverage attention mechanisms as a defense against attacks in FL and propose a robust FL algorithm by integrating the attention mechanisms into the global model aggregation step. To be more specific, two attention models are combined to calculate the amount of attention that is cast on each participant and it will then be used to determine the weights of local models during the global aggregation. The proposed algorithm is verified on a real-world dataset and it outperforms existing algorithms, both in secure systems and in systems under data poisoning attacks.

## 6.1 System Model and Problem Formulation

In this work, we consider a wireless radio signal classification system that consists of one transmitter and $K$ receivers. The transmitter transmits signals with $M$ different modulation types, and the receivers will use DNN models to classify the modulation type of the over-the-air received signals.

As shown in Fig. 6.1, we set up a scenario where multiple receivers work collaboratively to train a model for modulation type classification in a distributed manner. We assume that each receiver collects data on the signal it receives and corresponding modulation types as a local dataset, and uses it to train a local radio signal classification model. After

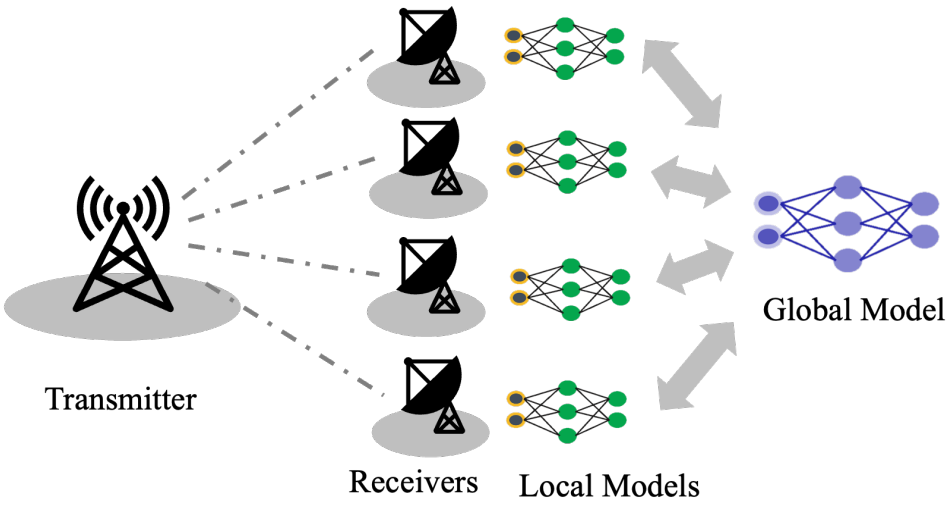



Figure 6.1: System model of FL-enabled radio signal classification.

each local training iteration, FL is performed at a centralized server to aggregate the local models into a global model. The objective function of local training at the $k^{th}$ receiver is defined as:

$$L_k^{Local}(\omega_k) = -\frac{1}{n_k}\sum_{i=1}^{n_k}\sum_{m=1}^{M} y_{i,m} log(p_{i,m}), \tag{6.1}$$

where $n_k$ denotes the number of local training samples at the $k^{th}$ receiver. $\omega_k$ denotes the parameters of the local model at the $k^{th}$ receiver. $p_{i,m}$ denotes the predicted possibility of the DNN model when the modulation type of the $i^{th}$ signal sample is the $m^{th}$ type. $y_{i,m}$ is a binary indicator where $y_{i,m} = 1$ if the modulation type of the $i^{th}$ signal sample is the $m^{th}$ type and 0 otherwise. The local updates can be formulated as follows:

$$\omega_{k,t+1} = \omega_{k,t} - \alpha \frac{\partial L_k^{Local}(\omega_k)}{\partial \omega_k}, \tag{6.2}$$

where $\alpha$ denotes the learning rate of local training.

The optimization goal of FL-enabled radio signal classification can then be defined as:

$$\min_{\omega}\{L^{Global}(\omega) = \sum_{k=1}^{K} L_k^{Local}(\omega_k)\}, \tag{6.3}$$

where $\omega$ denotes the parameters of the local model and $L^{Global}$ denotes the global loss.

The global aggregation can be performed by averaging the model parameters of local models from different receivers, which can be formulated as:

$$\omega_{t+1} = \frac{1}{K} \sum_{k=1}^{K} \omega_{k,t}. \tag{6.4}$$

After each iteration of global aggregation, the global model parameters will then be sent back to local models to accelerate the local training.

In this paper, we consider two scenarios: a secure scenario where all the receivers are benign participants and an attacked scenario where $J$ malicious participants exist. For malicious participants, we perform a label-flipping data poisoning attack and flip the labels of training examples with a source modulation type to a target modulation type while leaving the input signal data unchanged.

Given the common data distribution pattern in practical scenarios, in this work, we consider a non-IID setting. For each receiver, we assume that there is a dominant modulation type accounting for roughly half of the training samples, while all other modulation types account for the other half of the training data combined. For malicious participants with label-flipping data poisoning attacks, all training samples in the local training dataset that have a dominant class label will change their labels to another different class label. Consequently, during local updates, the malicious participants will learn incorrect updates from poisoned data and spread the incorrect updates to the global model through global aggregation. With these settings, the superiority of the proposed algorithm is demonstrated both in a secure environment and under attack.

## 6.2 Dual attention model-weighted FL for radio signal classification

To improve the system performance of FL in both secure and attacked scenarios, we propose a dual attention model-weighted FL based on the dot-product attention scheme. In the following subsections, we first explain the core principle of the attention model and then introduce the dual attention model-weighted FL we design for radio signal classification.

### 6.2.1 Attention model

In this work, scaled dot-product attention is adopted as a fast and space-efficient way to measure the attention weight. The calculation of scaled dot-product attention can be formulated as follows:

$$\eta_k = softmax(\frac{QK^T}{\sqrt{d_k}}), \tag{6.5}$$

where $\eta_k$ denotes the attention weight of the $k^{th}$ participant in the attention assessment. $K$ denotes the key, which is the target value that is expected from the participants. $Q$ denotes the queries, which are the actual values that are observed from the participants. $d_k$ denotes the dimension of the key.

In this way, the closeness between the key and each query is measured by the dot-product. If the direction of the query vector is close to the key vector, the dot-product will be large, and the query will gain more attention. On the contrary, if the query vector has an exact opposite direction from the key vector, the dot-product will be small, and the query will gain less attention. $\sqrt{d_k}$ is a scaling vector that can prevent the dot products from growing large due to the high dimension of the key vectors. The final attention weights are given by a SoftMax function to ensure they sum to one.

However, the dot-product attention model is susceptible to the distributions of the queries and the key, thus leading to unstable performance. Therefore, extra steps are designed to make the output of attention values more stable while applying dot-product attention to FL, and the details are described in the next subsection.

### 6.2.2 Dual attention model-weighted FL.

Fig. 6.2 shows the structure of the proposed dual attention model for FL. In this structure, two attention models are set up to evaluate the importance of each local model during the global aggregation. The first attention model is to perform self-attention among all the local models, and the second attention model is to perform temporal attention evaluation between the local models in the current iteration and the global model in the previous iteration. These attention models help to promote FL training by giving different attention weights to different local models. We further illustrate the scheme with a scenario of a system under attack. As it can be observed from Fig. 6.2, the second participant is assumed to be attacked, and the attack will be reflected in the local model parameters. As a result, the dual-attention scheme will cast less attention on the second local model and

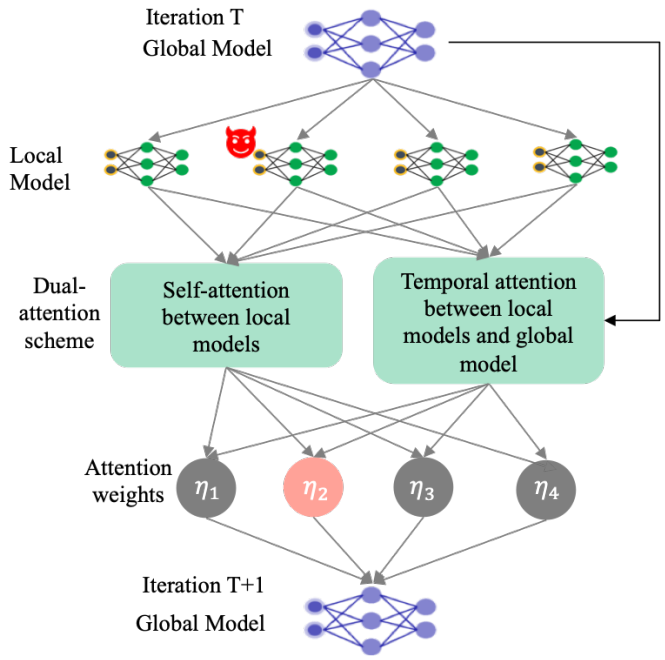



Figure 6.2: The proposed dual attention model for FL.

give it lower weight. In this way, the incorrect updates from the malicious participants are prevented from affecting the global model. The details of the two attention models are explained as follows.

In the first attention model, self-attention calculation among all the local models is performed to decide how much attention each participant will cast on other participants. This helps to evaluate the connections between local models and to cast more attention to local models that have stronger connections with other local models. For each participant, the local model parameters will be used as the key, and the local model parameters of other local model will be used as the queries. The attention value between the $k^{th}$ participant and the $l^{th}$ participant can be first formulated as the dot product of local parameters of the $k^{th}$ local model and the $l^{th}$ local model:

$$\epsilon_{k,l} = \begin{cases} \frac{\omega_k \omega_k^T}{|\omega_k|.|\omega_k^T|}, & if\ k \neq l \\ -inf, & if\ k = l \end{cases} \tag{6.6}$$

where $|\omega_k|$ and $|\omega_k^T|$ denote the modulus of $\omega_k$ and $\omega_k^T$. $inf$ denotes the infinity.

Next, the self-attention value matrix is normalized with mean and variance to ensure the attention model has roughly the same degree of influence on participants in global

aggregation of each iteration. The process can be formulated as:

$$\hat{\epsilon_{k,l}} = \frac{\epsilon_{k,l} - \mu_\epsilon}{\sigma_\epsilon}, \tag{6.7}$$

where $\mu_\epsilon$ denotes the mean value of the self-attention value matrix and $\sigma_\epsilon$ denotes the variance.

Then, the attention weight each participant casts on other participants can be calculated through a SoftMax function, which can be formulated as:

$$\eta_{k,l} = \frac{e^{\hat{\epsilon_{k,l}}}}{\sum_{l=1}^{K} e^{\hat{\epsilon_{k,l}}}}. \tag{6.8}$$

Finally, the attention weights each participant gets from other clients are summed as the attention weights from the first attention model, which can be formulated as follows:

$$\eta_{k1} = \frac{\sum_{K}^{k=1} \eta_{k,l}}{\sum_{K}^{l=1} \sum_{K}^{k=1} \eta_{k,l}}. \tag{6.9}$$

In the second attention model, similar steps are performed to evaluate how much attention the global model in the previous iteration will cast on the local models from all the participants in the current iteration. In this model, the global model parameters are used as the key, and the local model parameters are used as queries. The attention value the $k^{th}$ participant gets from the second attention model can be calculated as:

$$\epsilon_{k2} = \frac{w_t w_{k,t+1}^T}{|w_t|.|w_{k,t+1}^T|}, \tag{6.10}$$

where $w_t$ denotes the global model parameters at the $t^{th}$ iteration. Next, the attention values are normalized as:

$$\hat{\epsilon_{k2}} = \frac{\epsilon_{k1} - \mu_\epsilon}{\sigma_\epsilon}. \tag{6.11}$$

The attention weight of the second attention model can then be calculated as follows:

$$\eta_{k2} = \frac{\eta_{k2}}{\sum_{K}^{k2=1} \eta_{k2}}. \tag{6.12}$$

After obtaining attention weights from two attention models, the final attention weights for each client can be calculated by adding these two attention weights, which can be formulated as:

$$\eta_k = \beta\eta_{k1} + (1-\beta)\eta_{k2}, \tag{6.13}$$

where $\beta$ is a coefficient used to balance the ratio of two forms of attention weights. Tthe global aggregation can be then re-formulated as the attention-weighted sum of the local model parameters, which can be given as:

$$\omega_{t+1} = \sum_{k=1}^{K} \eta_k \omega_{k,t}. \tag{6.14}$$

## 6.3 Numeric Results

### 6.3.1 Simulation settings

In this work, we use a real radio signal classification data set generated in [178]. There are $M = 11$ modulation types, and we assume there are $K = 11$ receivers. The local data set of each receiver is dominated by one receiver. The DNN model used to train the dataset is composed of two convolutional layers, with the number of convolutional kernels being 256 and 80 and the dimensions of the convolutional kernels being $(1, 3)$ and $(2, 3)$, respectively. The convolutional layers are connected with two dense layers, with the number of neurons being 256 and 11, and ended by a SoftMax layer. The batch size during training is set as 32 and the learning rate is set as 0.0005. We selected data with a SNR between 2 dB and 18 dB for model training and prediction, which contains a total of 99,000 samples. In this dataset, 59,400 samples are used as the training and validation set, and 39,600 samples are used as the test set to evaluate the performance of algorithms. We set $\beta = 0.75$ during the simulation.

In the secure system, we compare the proposed dual-attention model-weighted FL algorithm with the classic FedAvg algorithm proposed in [179]. In the system under attack, we use multi-Krum proposed in [172] as the baseline to evaluate the defective effect of our proposed algorithm. We assume the best situation for the multi-Krum baseline, which is that the defender knows the exact number of attackers, which is hard to achieve in real situations. Even under such settings, we still demonstrate that our algorithm is superior to the baseline, while it does not require prior knowledge of the attack.

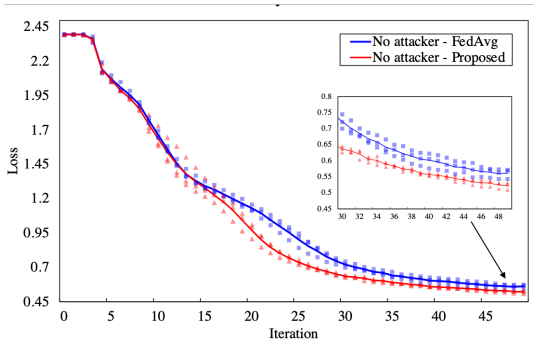



Figure 6.3: The convergence curves of FedAvg and proposed dual attention model-weighted FL in the secure system.

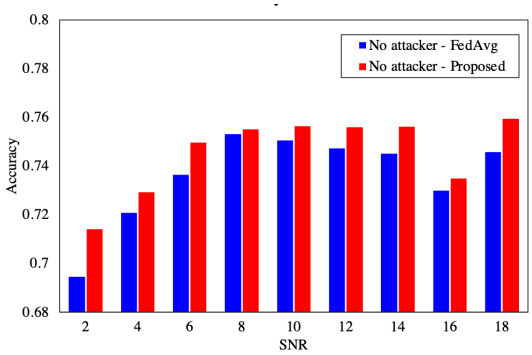



Figure 6.4: The accuracy of modulation type classification under different SNRs in the secure system.

### 6.3.2 Simulation results

In this subsection, we first evaluate the performance of the proposed dual attention model-weighted FL algorithm in a secure system environment and then evaluate it in the presence of attacks on the system.

Fig. 6.3 shows the convergence curves of FedAvg and the proposed algorithm in the secure system with no attacker. It can be observed that during the training process, the loss of both algorithms gradually decreases and eventually reaches a state of convergence. The loss of our proposed algorithm is always lower than the FedAvg algorithm during the training process, which demonstrates our proposed algorithm converges in a faster and more stable way. This is because the integration of the attention mechanism in the FL algorithm can help to identify the connections between participants and enhance cooperation. At the completion of the training, the classification loss of our proposed algorithm is 7.4% lower than the FedAvg algorithm.

Fig. 6.4 shows the accuracy of modulation type classification under different SNRs in the secure system. It can be observed that the proposed algorithm can have higher classification accuracy than FedAvg under all the SNR settings. This demonstrates that the integration of the attention mechanism in the FL algorithm can help to improve the performance of FL.

Then, we perform label-flipping data poisoning attacks on the system and evaluate the robustness of the proposed dual attention model-weighted FL algorithm under attacks. We first set one malicious participant out of 11 participants and flip the labels of a receiver from

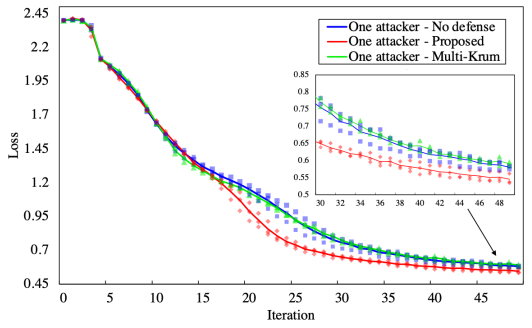



Figure 6.5: The convergence curves of Multi-Krum and the proposed dual attention model-weighted FL in the presence of one attacker.

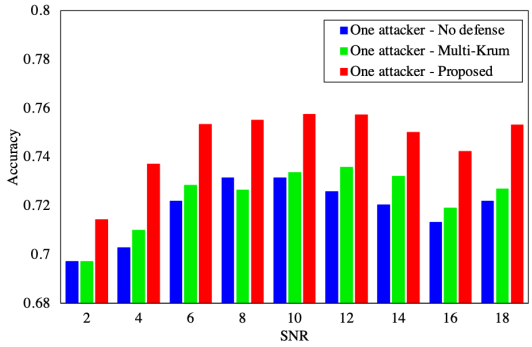



Figure 6.6: The accuracy of modulation type classification under different SNRs in the presence of one attacker.

the 'GFSK' type into the 'CPFSK' type. Fig. 6.5 shows the convergence curves of Multi-Krum and proposed dual attention model-weighted FL, and Fig. 6.6 shows the accuracy of modulation type classification under different SNRs in the presence of one attacker. We also add the convergence curve and the accuracy when there is no defense mechanism as a comparison. It can be observed that in the case of few participants and non-IID training data, multi-Krum has a limited defensive effect. In contrast, our proposed algorithm can still demonstrate a fast and stable convergence in the presence of one attacker. At the completion of the training, the classification loss of our proposed algorithm is 10.2% lower than the multi-Krum algorithm. It also achieves much higher accuracy under different SNRs compared with the multi-Krum defense. With the attention schemes, there is no decrease in the accuracy of classification compared to the secure system.

Next, We increase the number of malicious participants to 2 and flip the labels of another receiver from the 'QPSK' type into the 'CPFSK' type. It is worth noting that our proposed algorithm is still applicable in the case of more attackers. However, considering the limitations of attackers' capabilities in real application scenarios, we only consider the case of up to two attackers in our experiments. Fig. 6.7 shows the convergence curves of Multi-Krum and proposed dual attention model-weighted FL, and Fig. 6.8 shows the accuracy of modulation type classification under different SNRs in the presence of two attackers. It can be observed that the Multi-Krum defense has an even worse convergence compared with the system with no defense. This is because the Multi-Krum defense cannot accurately identify malicious participants in every iteration, and the misjudgment in each iteration will cause the model to lose part of the correct training data, which in turn will

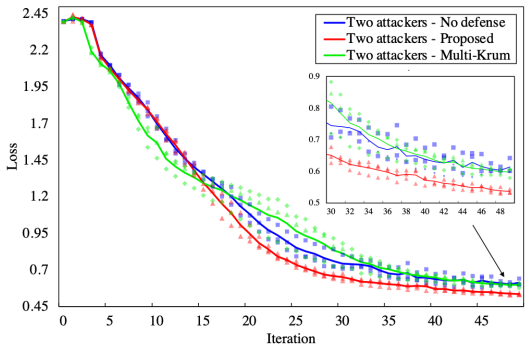



Figure 6.7: The convergence curves of Multi-Krum and the proposed dual attention model-weighted FL in the presence of two attackers.

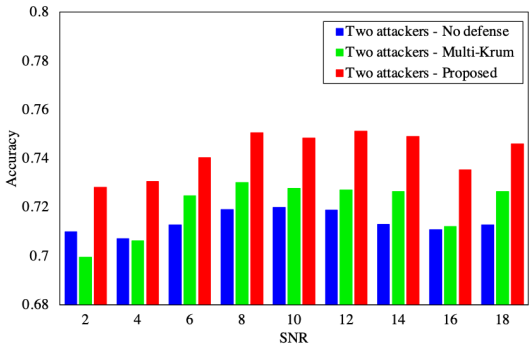



Figure 6.8: The accuracy of modulation type classification under different SNRs in the presence of two attackers.

result in worse performance. In contrast, our proposed algorithm is still able to converge stably and fast and demonstrates almost the same level of accuracy as under a secure system. At the completion of the training, the classification loss of our proposed algorithm is 11.5% lower than the multi-Krum algorithm. These results further validate the defensive effect of the proposed algorithm.

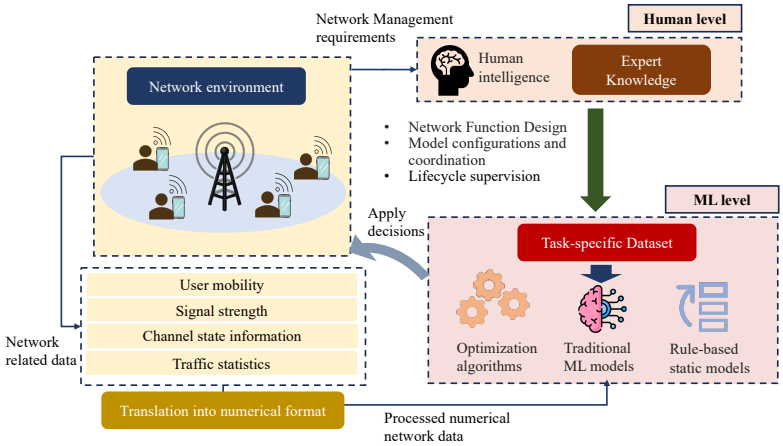
# Chapter 7

# GAI-in-the-loop in mobile communication systems

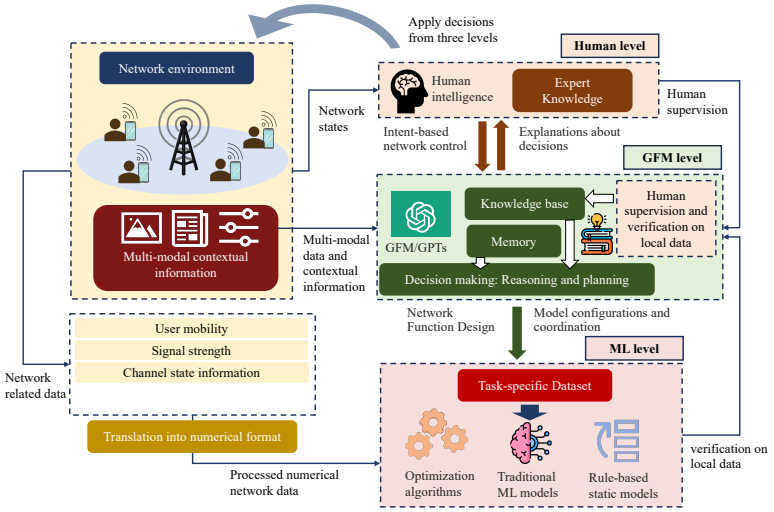
ML algorithms have emerged as effective and promising solutions in intelligent mobile networks. The widespread use of ML techniques has accelerated the automation of selected tasks and laid the foundation for AI-native mobile communication networks. However, the traditional ML techniques also have limitations that hinder the full automation of next-generation networks.

GFMs have recently attracted significant attention in the industry and academia. Benefiting from the remarkably large model size and enriched training data, some pre-trained GFMs, especially GPTs, possess capabilities that have not appeared in traditional small ML algorithms, such as the reasoning capability, the powerful knowledge base, and the interactive capability. Therefore, they have the potential to assist humans with mobile network management tasks that require human intelligence and reduce the engineering cost of human involvement in model design and training [180].

Inspired by these thoughts, in this chapter, we propose the concept of GFM-in-the-loop and explores how GFMs can enhance ML-driven mobile communication. In addition, we also provide a case study and design a pre-trained LLM-empowered framework to perform wireless communication-related tasks.




(a) Human-in-the-loop framework.




(b) GFM-in-the-loop framework.

Figure 7.1: Illustration of the evolution from human-in-the-loop to GFM-in-the-loop in next-generation networks. In the GFM-in-the-loop framework, GFMs act as intermediaries between human-level management and traditional machine learning and optimization algorithms by automating network control based on semantic intentions, generating semantic-based explanations, and performing model and network management.

## 7.1 GFM-in-the-loop

Fig. 7.1 illustrates the evolution from Human-in-the-loop to GFM-in-the-loop frameworks in next-generation networks. In the Human-in-the-loop paradigm, the human level directly interacts with the ML level and is responsible for tasks such as function design, model configuration, and lifecycle supervision. As a result, the system heavily relies on expert knowledge and manual intervention. In contrast, the GFM-in-the-loop framework includes three optional levels of network configurations: human level, GFM level, and ML level. All three levels can observe desired information from the network environment, and they can interact with each other to facilitate collaborative network management. With the introduction of the GFM level, the reliance on human experts is reduced, and additional non-numerical contextual signals, such as multi-modal information, can be incorporated into the decision-making process. The components and interactions within the GFM-in-the-loop framework are explained below.

The ML level contains a diverse set of models, including traditional data-driven ML models such as supervised and reinforcement learning, as well as optimization-based algorithms, rule-based control strategies, and other lightweight mathematical models. It typically processes real-time or historical data in numerical formats, such as user mobility, signal strength, channel state information, and traffic statistics. At this level, models are generally designed for tasks that require high update rates and deterministic outputs, which ML or rule-based models better support than GFMs with a lower inference cost. This process can be enhanced or guided by the GFM level with high-level contextual information, domain-level knowledge and long-term trends. Additionally, model training may occur either offline using ready-made datasets or online through continual learning, depending on the application scenario. Overall, the ML level forms the execution core of the system to give fast, deterministic results while remaining adaptable through its interactions with the higher-level GFM reasoning layer.

On top of the ML level, the GFM level is set. The GFM level with powerful models like GPTs can acquire and comprehend multi-modal input from the network environment. It also provides high-level guidance for the ML level for contextual information analysis, network task-related planning, and decision-making. For example, it can analyze received physical signals, guide the design of ML models, or perform coordination between different models. In addition, considering the uneven development and instability of most GFMs, supervisory mechanisms need to be established to verify the effectiveness of GFM outputs and avoid hallucinations. Possible supervisory methods include human supervision or verification of local data.

The human level is set at the very top of the framework. It is equipped with expert

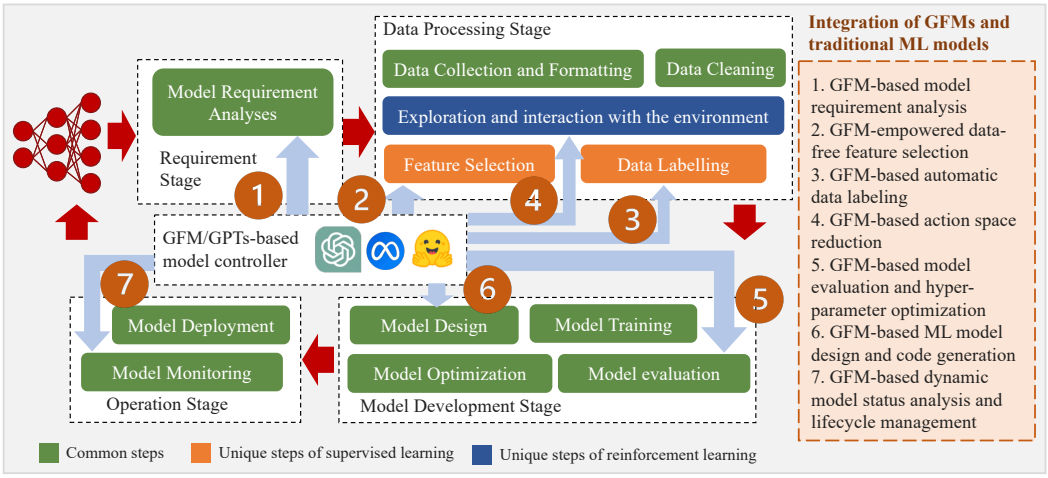



Figure 7.2: Different ways to enhance ML models with GFMs. The life cycle of ML models includes four stages: requirement stage, data processing stage, operation stage, and model development stage. GFMs can be integrated into each stage.

knowledge of mobile networks and can involve considerations related to human factors such as customer demands and commercial costs. The GFM level acts as an interface between the human level and lower levels. It can translate intent-based network control into manageable tasks for ML models and generate explanations of ML layer decisions for humans.

Within the GFM-in-the-loop framework, the combination of GFMs and traditional ML models can capitalize on the benefits of both GFMs and traditional ML models and compensate for the weaknesses of the other.

## 7.2 Enhance traditional ML models with GFM/GPTs

In this section, we start by comparing the life cycle of traditional ML techniques and GFMs. Next, based on the life cycle of traditional ML models, we provide general approaches on leveraging GFMs to improve the performance of existing ML-driven network applications.

GFMs and traditional ML models typically exhibit different lifecycles. GFMs require substantial computational resources and large-scale datasets for pretraining, and they are expensive and difficult to retrain after deployment. In practice, adapting GFMs to specific applications typically relies on lightweight fine-tuning rather than full retraining. Once pre-trained and fine-tuned, GFMs possess generalized, emergent capabilities that allow them to

adapt to a variety of tasks in dynamic environments without frequent model updates. As a result, GFMs generally have a much longer lifecycle. In contrast, traditional ML models are lightweight, designed for specific tasks, and can be retrained or updated frequently as needed, resulting in a much shorter lifecycle than GFMs. Based on these differences, we discuss how GFMs can enhance ML models at different stages of their lifecycle, while assuming that GFMs remain unchanged during these stages.

Fig. 7.2 shows seven different ways to enhance ML models with GFMs. GFMs can use one of these ways or a combination to play a supporting role. In this figure, the life cycle of traditional ML models includes four stages: the requirement stage, the data processing stage, the operation stage, and the model development stage. For different machine learning methods, such as supervised learning, unsupervised learning, and RL, the steps at the data processing stage can be different, which are labeled with different colors. The different roles GFMs play at each stage are explained as follows.

### 7.2.1 Requirement stage

In the requirement stage, the model requirement analysis is performed to clarify the task requirements and decide the number and type of ML models needed for the given task. These designs are traditionally manually done with the help of human experts. The task-planning capabilities of GFMs have been demonstrated in robotics-related studies [181]. Therefore, GPT-based model requirement analysis can be performed to automate the analysis process and reduce the manual cost.

GPT-based model requirement analysis includes several steps. GPTs can first perform task planning and decompose the given task into several small, manageable tasks. Then suitable models are selected for each task according to the type of task. For instance, the traffic steering task can be decomposed into three small tasks, traffic prediction, resource allocation, and handover control. These tasks can then be separately performed with different ML models, such as long short-term memory models and RL models. Similarly, such a method can be migrated to mobile communication applications. The input to the GFMs should include the task description, the mobile communication environment description, available data, and the objective of the task.

### 7.2.2 Data processing stage

In the data processing stage, two common steps are performed. In the first step, data is collected from mobile networks and standardized. Next, data cleaning is conducted to

remove incomplete or anomalous samples from the data set. GPTs can help with data preparation by converting semantic-based non-numeric features. For instance, in the network intrusion detection task, the IP address is a non-numeric feature that needs to be translated into numeric values. The traditional approach to translation is to use the one-hot encoding method, which is difficult to scale and lacks semantic information. Using the knowledge base of GPTs, semantic-based numerical information can be extracted such as whether the IP address is private or public, and whether it is associated with known malicious activities, leading to more efficient data translation.

In addition, there are two unique steps of supervised learning models, feature selection and data labeling. Traditional data-based feature selection methods may lead to overfitting problems and it is challenging to decide the statistical measures [182]. Instead, GPTs perform data-free feature selection based on the semantic understanding of given features. Data labeling is to add informative labels to the raw data so that ML models can learn from it. The traditional data labeling approach is to ask humans to recognize unlabeled data, which results in low scalability and high expenses. In comparison, GFM-based automatic data annotation, or hybrid labeling with both human experts and GFMs can automate the labeling process and lower the cost [183].

Apart from this, RL models usually collect data through exploration and interaction with the environment. When there is a large action space, GPTs can help with action space reduction by removing common sense non-optimal action options. With GPTs, the efficiency of the exploration can be improved. For example, the RL-based slicing resource allocation problem sometimes suffers from the large action space since it is proportional to the number of network slices. GPTs can exclude undesirable actions from the large action space following their knowledge base, e.g., resources should not be biased towards slices with lower traffic loads or lower priority. Furthermore, GFMs can also leverage the generation capability and help synthesize training data to mitigate data shortage.

### 7.2.3 Model development stage

The third stage is the model development stage. It includes model design, training, optimization, and evaluation. In the first two steps, the architecture of the ML model is decided and the model is trained with the data prepared in the previous stage. After that, the ML model is evaluated and optimized.

In this stage, GPTs can first accelerate the model design through code generation according to the functionality and requirements. In addition, GPTs can help understand the

model evaluation results and automatically adjust model architecture and the hyperparameters of model training accordingly to improve the training quality. For example, under a supervised learning task, when GPTs find that the model evaluation results are lower than expected, they may modify the generated code to increase the number of epochs, increase the size of the neural network, and add dropout layers to improve the model performance.

### 7.2.4 Operation stage

The last stage is the operation stage. In this stage, the ML model is deployed at network nodes for model inference. It also needs to be monitored in case of model decay, unexpected data, and attacks. In this stage, GPTs can be used for dynamic model status analysis and life-cycle management based on the knowledge base and the reasoning capability. For example, in a QoS management task, GPTs can be utilized to determine which control functions should be performed under different QoS scenarios, and then the corresponding ML model can be initiated.

Moreover, GPTs can help balance the trade-off between the model performance and the computation cost and enable higher efficiency. For example, GPTs can analyze the relationship between the model complexity and accuracy, and available computation resources, to select the most appropriate model complexity. GPTs can also be used to monitor the ML model performance and evaluate if the model continues to function as expected in realistic mobile network scenarios. If necessary, it can terminate the current ML models and initiate the life cycle of new ML models.

Following the introduction of integration methods, the co-deployment of GFMs and ML models within mobile networks presents significant challenges, which will be addressed in the next section.

## 7.3 Deployments of ML models and GFM in mobile networks

In this section, we discuss simultaneously deploying traditional ML models and GFMs in mobile networks. As shown in Fig. 7.3, there are three different deployment options: fully centralized deployment, mixed centralized and distributed deployment, and fully distributed deployment.

Fully centralized deployment means deploying both GFMs and traditional ML models on the cloud side and combining them for network management. The benefit is that the

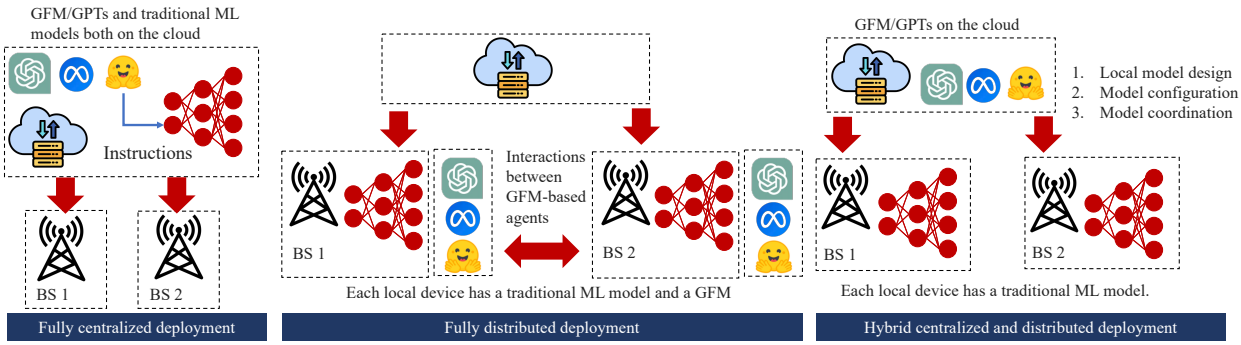



Figure 7.3: Three ways to deploy GFMs and ML models. (a) Both GFMs and ML models are deployed at the center and combined for network management. (b) Both GFMs and ML models are deployed in a distributed manner for Multi-Agent interaction. (c) GFMs are deployed at the center for management, and ML models are deployed in a distributed manner for local training and inference.

powerful computing and storage resources of the cloud server can support the training and deployment of both GFMs and ML models. Moreover, centralized models have access to global information and can make more intelligent decisions.

However, the limitation is that the centralized structure lacks flexibility and scalability and is not suitable to handle a large number of local servers, for example, ultra-dense networks. Also, in this deployment, the cloud server will access sensitive data generated by base stations (BSs) and bring privacy concerns.

This deployment method is applicable when there is a small number of BSs, and the controller wants to have centralized control of the whole network. It also fits the cases when the distributed BSs only have limited computation and storage resources. Some possible applications include interference-aware radio resource management, power control, and network slice provision.

Fully distributed deployment means deploying both GFMs and traditional ML models on the local or edge site and each local BS owns a GFM. The benefit is that the GFMs and ML models are deployed closer to the data source. The sensitive data can be kept locally to enhance data privacy. In addition, fully distributed deployment has better scalability since the workload is jointly undertaken by distributed BSs. The interactions between BSs are also promoted through multi-agent conversations [184].

However, the fully distributed deployment requires a much higher requirement for the resources held by the local servers. The servers should be equipped with powerful com-

puting and storage resources to support both GFMs and ML models. Additionally, the distributed structure will introduce parallelization and synchronization steps and expose models to more vulnerabilities.

In conclusion, this deployment method applies to scenarios with powerful local edge servers and where interactions are needed between BSs. Some suitable applications include traffic prediction and user association.

Hybrid centralized and distributed deployment combines the previous two deployment methods. In this case, GFMs are deployed on the cloud while traditional ML models are deployed on local servers. This method incorporates the benefits of both fully centralized deployment and fully distributed deployment. The cloud server is usually equipped with powerful resources to support the needs of GFMs. The distributed ML models are supported by local devices and are customized for task-oriented applications. As a result, resources can be rationally exploited and data privacy can be protected. The only concern is that the communication between local ML models and the cloud-based GFMs will result in extra communication overhead.

With this deployment, GFMs can be used for distributed ML model management. They can analyze user requirements, plan tasks, and leverage ML models for execution. For example, GPTs can select the appropriate scheduling algorithm models and perform conflict management between different network applications.

## 7.4 Pre-trained LLM-empowered framework for automatic network intrusion detection: A case study

To demonstrate the potential of pre-trained LLMs in performing wireless communication-related tasks, we propose a pre-trained LLM-empowered framework to perform fully automatic network intrusion detection. Three in-context learning methods are designed and compared to enhance the performance of LLMs. With experiments on a real network intrusion detection dataset, in-context learning proves to be highly beneficial in improving the task processing performance in a way that no further training or fine-tuning of LLMs is required. We show that for GPT-4, in-context learning helps improve the accuracy of network intrusion detection.

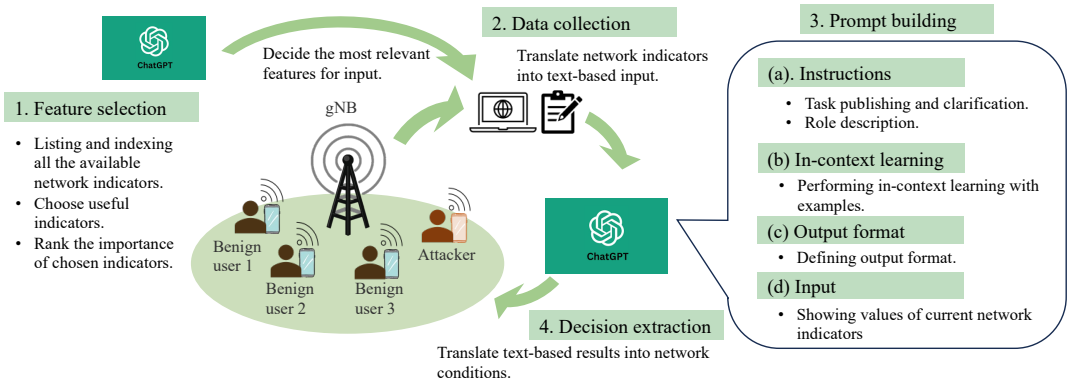



Figure 7.4: System model of pre-trained LLM-empowered network intrusion detection.

### 7.4.1 LLM-empowered network intrusion detection

The system model of the proposed pre-trained LLM-empowered network intrusion detection framework is shown in Fig 7.4. In this work, we consider a cloud-based wireless communication system where malicious attackers can perform distributed denial of service (DDoS) attacks and generate malicious traffic to the networks. At the network controller side, a cloud-based pre-trained LLM is deployed for network security monitoring and intrusion detection.

As shown in Fig 7.4, four main steps are designed in the framework to enable fully automatic intrusion detection for zero-touch networks. In the first step, the most relevant network features for intrusion detection are selected by the LLM. In the second step, data is collected from the networks and processed for LLM input. The third step is to build the prompt for LLM, and the last step is to extract the desired decision from LLM output. These steps automate the interactions between the LLM and the 5G communication system, allowing the LLM to choose desired network information based on its knowledge base, capture the required values from the system, and feedback decisions. Detailed implementation of each step is explained below.

#### (1) Feature selection

To start LLM-based intrusion detection, feature selection is first performed to select the most important features from a large number of network features according to their relevance to the intrusion detection task. This step can be implemented with the knowledge base of pre-trained LLMs.

First, all the accessible network features are indexed and given to the LLM. Next, the LLM is instructed to select the indexing numbers corresponding to the ten most relevant features of the network intrusion detection task. The LLM is then requested to rank the importance of all selected features on three levels: 'very important', 'kind of important', and 'not very important'. Only 'very important' features and 'kind of important' features will be kept as desired features for the following detection tasks. The goal of this step is to concise the length of the LLM input token and to avoid the impact of irrelevant features on the detection results.

**(2) Data collection and processing**

After feature selection, the LLM is employed to monitor the network status for suspected intrusions. First, the values corresponding to the selected features are collected and converted to a text-based format. Next, a text-based template is created to give definitions and semantic explanations about the collected values. Finally, the text-based values are concatenated with pre-defined templates. Based on the above steps, selected input features can be translated into comprehensible human-like descriptions of current network conditions.

**(3) Prompt building**

In this step, prompts are designed to provide the LLM with formalized guidance for effective intrusion detection. The prompts are composed of four parts, instructions, in-context learning examples, output formatting, and input information.

The instructions usually include the task publishing and clarification and role description. In particular, for network intrusion detection tasks, the prompts should first define what a network intrusion is, describe the role of the LLM as a 5G network safety monitor and clarify the task is to determine whether the traffic is from a malicious user based on given information. Next, examples of both benign cases and malicious cases are included in the prompts to provide the LLM with task-specific knowledge through in-context learning. The detailed implementation of in-context learning is given in the next section.

Output formatting is also included as a part of prompt building to give clearer detection results and prevent misunderstandings. In particular, for network intrusion detection tasks, the output formatting can be implemented by using prompts like "If you think the traffic is malicious, answer yes. Otherwise if you think it is benign, answer no". The output formatting part also simplifies the translation from the LLM output to the detection

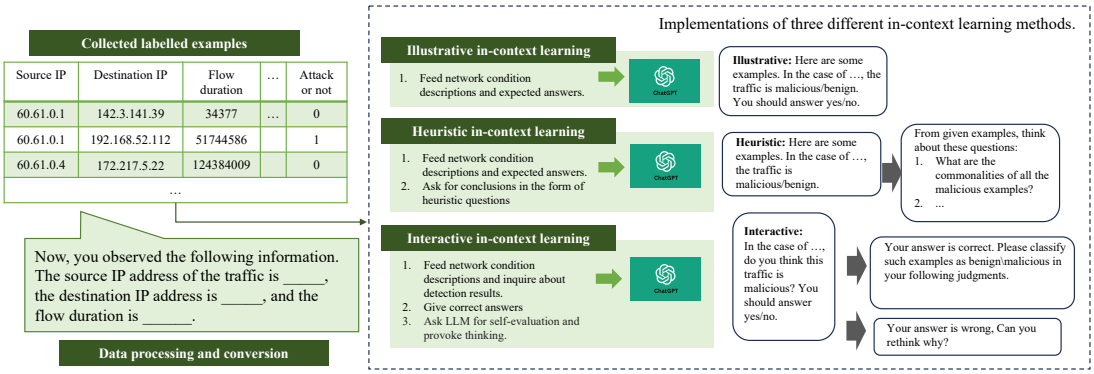



Figure 7.5: Three different methods to implement in-context learning with labelled examples.

results of network conditions in the following step. Input information refers to human-like descriptions of network conditions acquired in the last step.

### (4) Decision extraction

The last step is to understand the output of the LLM and extract the desired results. With output formatting in the last step, the decision extraction can be achieved with keyword searching. In detail, if "yes" or "Yes" appears in the LLM output while "no" or "No" does not appear, then the decision is that the traffic is malicious and vice versa. If neither of the above cases is valid, output formatting will continue to be performed until the desired clear output is obtained.

## 7.4.2 In-context learning-enhanced detection

Although LLMs may have some basic intelligence and knowledge about wireless networks, it is still difficult to directly perform intrusion detection tasks through pre-trained LLMs without fine-tuning. In-context learning is an effective method that can improve the accuracy of LLMs on specific tasks with only a small amount of labelled data.

In this section, we illustrate how we design in-context learning schemes with labelled examples to improve the performance of pre-trained LLMs for domain-specific tasks. In first subsection, general principles of in-context learning in LLMs are introduced. In second subsection, detailed implementation of in-context learning in our framework is given.

### (1) In-context learning for LLM

In-context learning is an effective technique that enables pre-trained LLMs to address specific tasks without updating the parameters of LLM. This is achieved by integrating text-based examples into the prompts and by changing the distribution of the input text to obtain the desired output. The process can be formulated as:

$$y_t^* = \underset{y_t}{argmax}\ P(y_t|(x_1, y_1), ..., (x_n, y_n), x_t; \Theta_{origin}) \tag{7.1}$$

where $y_t^*$ denotes the LLM output of the network intrusion detection task. $P(y_t|...)$ denotes the possibility of getting $y_t^*$ as the output. $x_t$ denotes the network condition description input. $(x_n, y_n)$ denotes the $n^{th}$ in-context learning examples. $\Theta_{origin}$ denotes the pre-trained parameters of the LLM. Following this formula, the output of the LLM is influenced by examples included in the context information and the LLM is more likely to output decisions that are consistent with the given example [185].

### (2) In-context learning for network intrusion detection

In this work, we design and compare three different in-context learning methods, the illustrative in-context learning, the heuristic in-context learning, and the interactive in-context learning. The implementations of three methods are presented in Fig. 7.5.

In the illustrative in-context learning method, examples with labels are converted into human (and LLM)-interpretable descriptions of cases and expected answers. Then, they are laid out and fed directly into the LLM as part of the prompt. The prompts usually start with identifying statements like "Here are some examples" and each example ends with conclusive instructions like "You should answer yes/no."

The illustrative in-context learning method is the simplest among the three in-context learning methods. On this basis, the heuristic in-context learning method is designed by adding heuristic questions to the prompts. Based on the given network intrusion detection task scenario, some critical questions that may shape the outcome are extracted, like "What are the commonalities of all the malicious examples?" or "What is the rational range of flow duration?". These questions can give insights to the LLM on what should be concluded from the given examples. The LLM outputs of these questions will also be included in the prompts as context information for the following intrusion detection tasks.

The last proposed in-context learning method is the interactive in-context learning. In this method, in-context examples are given and analyzed through a question-and-answer format. First, the examples are fed into the LLM without giving the expected answers. Then the LLM is asked to provide detection results and perform self-evaluation. If the result matches the label, the LLM is encouraged to continue to make judgments in this manner. Otherwise, the LLM is asked to perform self-corrections by providing explanations of expected results and reflecting on the reasons for the incorrect answer. Similar to the heuristic in-context learning method, the self-assessments and reflections will also be included in the prompts as a part of the context information.

### 7.4.3 Experimental settings and Results

#### (1) Experimental settings

In this work, we use a real network intrusion detection dataset proposed in [186] to generate testing examples and in-context learning examples. The dataset includes 9 types of DDoS attacks and 84 network traffic features. Three LLMs are used during the experiments, LLAMA-2-7b [187], GPT-3.5, and GPT-4 [188]. CNN-based network intrusion detection is also implemented as a baseline and is used for the comparison between traditional ML models and LLMs. The performance is evaluated according to the accuracy and F1-Score of network intrusion detection. All the models are implemented using Python with the support of PyTorch, OpenAI-API, and HuggingFace.

#### (2) Experimental results

Fig 7.6 shows the feature selection results of the network intrusion detection task from three LLMs, GPT-3.5, GPT-4, and LLAMA. The manually selected features for a CNN-based network intrusion detection model proposed in [185] are also added for comparison. During the experiments, each LLM is asked to choose ten features from the given 84 features, and the importance of these features are ranked according to three levels, 'very important', 'kind of important', and 'not very important'. Only 16 features are mentioned by three LLMs, and among these 16 features, 5 features mentioned by only one LLM are ranked as 'not very important'. This suggests that different LLMs exhibit a high degree of similarity in feature selection for the given task. It can also be observed that the selected features overlap significantly with the manually selected features. This validates the knowledge base of pre-trained LLM in the wireless communication domain and its ability to handle related tasks. In the following experiments, features ranked as 'not very important' are

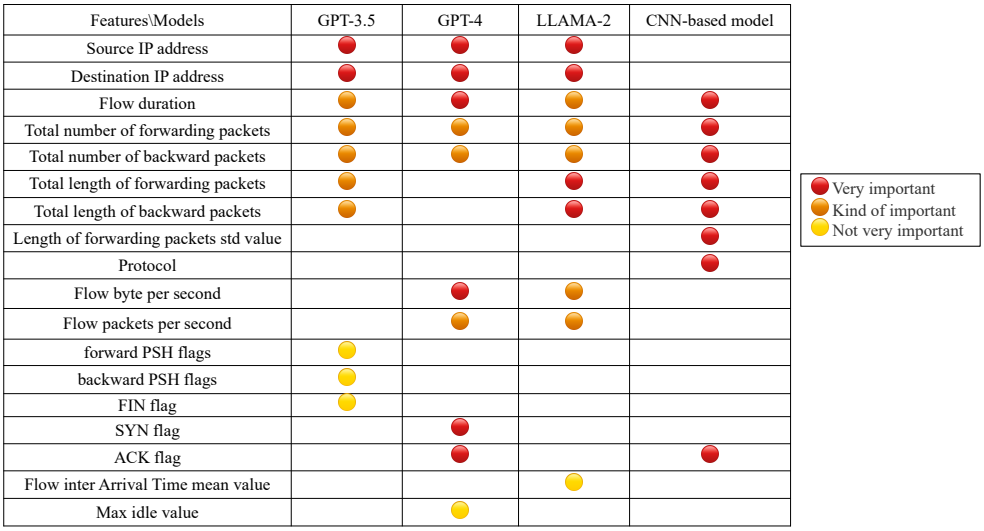



Figure 7.6: The feature selection and importance ranking results from three different LLMs, GPT-3.5, GPT-4, and LLAMA.

removed and only 'very important' features and 'kind of important' features are kept as the input for intrusion detection.

For the in-context learning experiments, we only test with GPT-3.5 and GPT-4. This is because LLAMA2 is an open-source model with a limited token size and is more suitable for fine-tuning rather than in-context learning. Fig. 7.7 and Fig. 7.8 show the comparison of three different in-context learning methods on a TCP ACK attack detection task. It can be observed that the testing accuracy and F1-Score grow with the number of examples. For GPT-3.5, the heuristic in-context learning method shows a poor performance when there are only 2 examples. This is because GPT-3.5 may summarize incorrect conclusions from only 2 examples. As a result, it may give wrong answers to heuristic questions and generate inaccurate detection results. When there are more in-context examples, the interactive in-context learning method and heuristic in-context learning method show better results than the illustrative in-context learning method. Furthermore, increasing the number of in-context learning examples does not necessarily lead to an improvement in performance. This is because of GPT-3.5 only has limited reasoning ability. When there are too many in-context examples, the model may fail to draw valid conclusions and show worse performance. It can also be observed that the accuracy and F1-Score show a slightly different trend. This is because with interactive in-context learning, GPT-3.5 will reflect on its mistakes and be more careful about possible intrusions. But this also makes it take

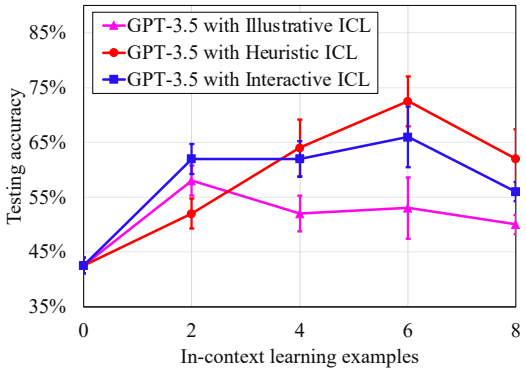



Figure 7.7: Comparison of intrusion detection testing accuracy under three different in-context learning methods with GPT-3.5 while increasing the number of in-context learning examples.

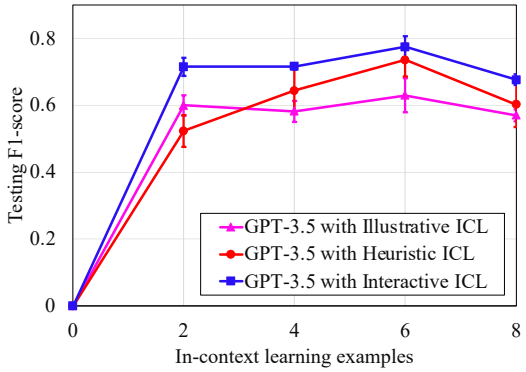



Figure 7.8: Comparison of intrusion detection testing F1-Score under three different in-context learning methods with GPT-3.5 while increasing the number of in-context learning examples.

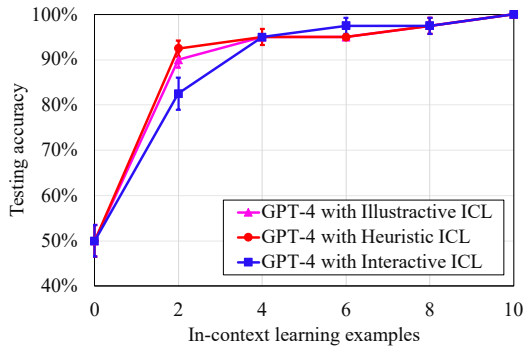



Figure 7.9: Comparison of intrusion detection testing accuracy under three different in-context learning methods with GPT-4 while increasing the number of in-context learning examples.

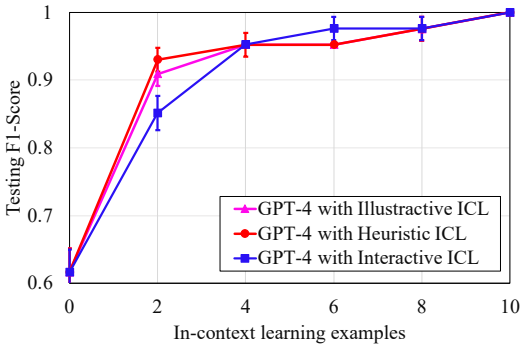



Figure 7.10: Comparison of intrusion detection testing F1-Score under three different in-context learning methods with GPT-4 while increasing the number of in-context learning examples.

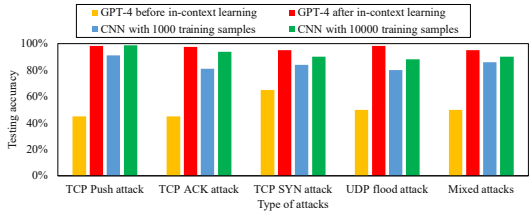



Figure 7.11: Comparison of intrusion detection testing accuracy with GPT-4 and CNN-based method.

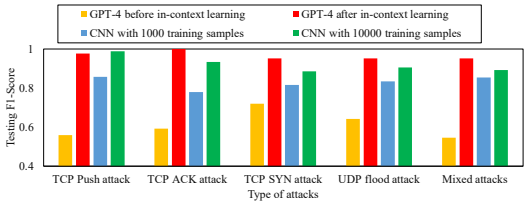



Figure 7.12: Comparison of intrusion detection testing F1-Score with GPT-4 and CNN-based method.

the benign traffic as intrusions sometimes. So interactive in-context learning will lead to a relatively high F1-Score, but slightly lower accuracy.

Such observations do not appear in the experiments using GPT-4, which is equipped with a stronger reasoning ability. As shown in Fig. 7.9 and Fig. 7.10, for GPT-4, the three different in-context learning methods lead to very close performance. The illustrative and the heuristic in-context learning methods perform a little bit better than the interactive in-context learning method when there are fewer examples. This is because GPT-4 has a greater contextual comprehension and reasoning ability [189]. As a result, it requires less guidance to complete the task.

Compared to illustrative in-context learning, heuristic in-context learning and interactive in-context learning will add extra information to the prompts and increase the token length. This will lead to higher inference costs. It is also worth noting that the performance of heuristic in-context learning depends partly on the design of the heuristic questions. Therefore, the design of heuristic in-context learning still requires the intervention of domain experts. This paper mainly focuses on the framework design and thus does not discuss the design of the heuristic questions in detail. In conclusion, illustrative in-context learning is a more cost-effective choice for LLMs with stronger capabilities. Heuristic in-context learning and interactive in-context learning will potentially offer performance boosts for LLMs with less capabilities.

Fig. 7.11 and Fig. 7.12 compare the intrusion detection testing accuracy before and after in-context learning under different types of attacks. The experiments are performed by GPT-4 with 10 in-context learning examples. The performance of CNN-based intrusion detection with different numbers of training samples is also added for comparison. It can be observed that in-context learning can significantly improve the LLM performance on domain-specific tasks. The testing accuracy of LLM after in-context learning is about 90% higher than before. On the other hand, the performance of the CNN model is very much

related to the number of training samples. The accuracy and F1-Score will improve if the number of training samples increases. It can be concluded that the performance of traditional ML models relies much on massive amounts of data. In contrast, LLMs achieve comparable or superior performance to traditional ML models with only a small amount of task-specific data.

It is also worth noting that, while the proposed LLM-based framework demonstrates promising results for detecting the aforementioned network intrusions, it may have limitations in identifying more sophisticated threats such as advanced persistent threats (APTs). APT attacks often employ stealthy, multi-stage strategies over extended periods, making them difficult to detect from limited or isolated data patterns typically available to the LLMs.

# Chapter 8

# Pre-trained LLM-empowered mobile traffic prediction

Mobile traffic prediction is an important enabler for optimizing resource allocation and improving energy efficiency in mobile wireless networks. In this chapter, we introduce a context-aware wireless traffic prediction framework powered by LLMs. To further enhance prediction accuracy, we also leverage ICL and develop a novel two-step demonstration selection strategy, optimizing the performance of LLM-based predictions. The proposed framework shows lower mean squared error and higher $R^2$-Scores compared to the zero-shot prediction method and other demonstration selection methods, such as constant ICL demonstration selection and distance-only-based ICL demonstration selection.

## 8.1 System Design

Fig. 8.1 shows the system design of our proposed LLM-based mobile traffic prediction framework. In this framework, data are first collected from a 5G mobile communication scenario and screened with a two-step ICL demonstration selection scheme. The selected ICL examples are then combined with the test data as input to the LLM for future traffic prediction. In this section, we first define the system model and formulate the traffic prediction problem. Next, we explain how to pre-process data for the ICL demonstration setup and how to generate prompts.

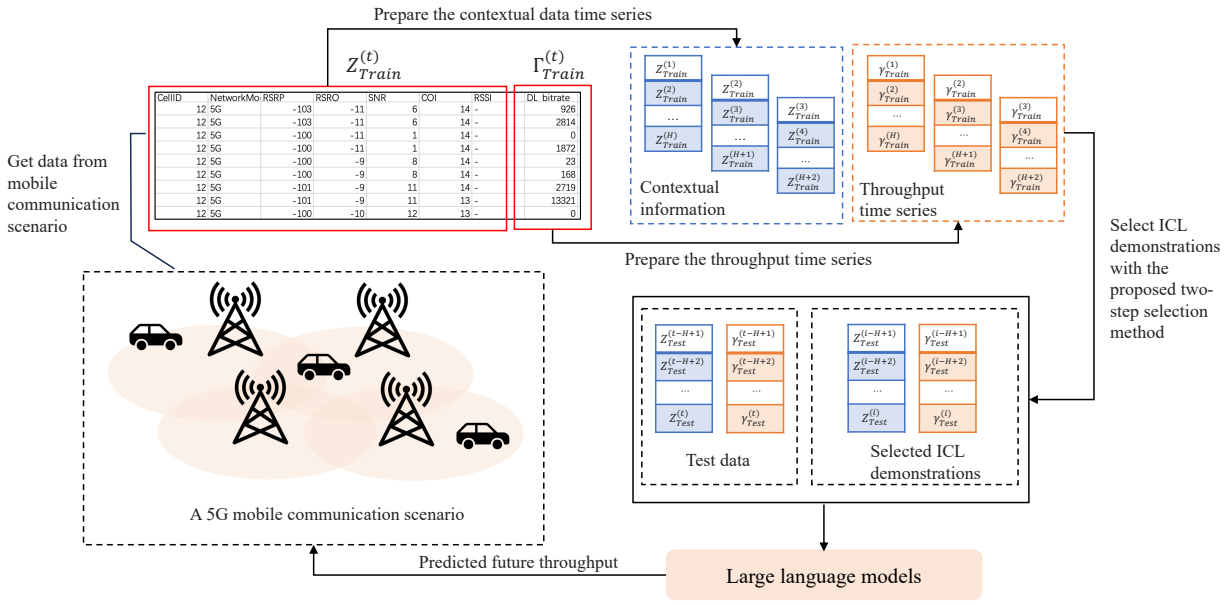



Figure 8.1: System design of LLM-based mobile traffic prediction.

### 8.1.1 Problem Formulation

This study aims to predict the future downlink traffic bitrate of every single mobile device using current and historical traffic data along with contextual information. The BS can periodically measure the bitrate over the communication channel and create discrete time series representing the device's traffic patterns. Additionally, the BS also has access to some contextual data, including the channel metrics and the neighboring cell metrics, to enhance the accuracy of the prediction [190].

Due to the limited size of the prompts, only part of the historical information is used as the input for mobile traffic prediction. The historical time window $H$ is defined to control the input length. In this case, at time $t$, the input downlink throughput time series $\Gamma^{(t)}$ can be formulated as:

$$\Gamma^{(t)} = \{\gamma^{(t)}, \gamma^{(t-1)}, ..., \gamma^{(t-H+1)}\}. \tag{8.1}$$

The input contextual information $Z^{(t)}$ can be formulated as:

$$Z^{(t)} = \begin{pmatrix} z_1^{(t)} & z_2^{(t)} & \cdots & z_N^{(t)} \\ z_1^{(t-1)} & z_2^{(t-1)} & \cdots & z_N^{(t-1)} \\ \vdots & \vdots & \vdots & \vdots \\ z_1^{(t-H+1)} & z_2^{(t-H+1)} & \cdots & z_N^{(t-H+1)} \end{pmatrix}, \tag{8.2}$$

where $\{z_1, z_2, ..., z_N\}$ denotes the $N$ input features selected from all the available contextual information. In this scenario, we focus only on the LLM-enabled one-step ahead traffic prediction problem and the prediction process can be formulated as:

$$\hat{y}^{(t)} = f(\Gamma^{(t)}, Z^{(t)}|\theta^{pre}), \tag{8.3}$$

where $\hat{y}^{(t)}$ denotes the predicted future downlink throughput. $f(.)$ denotes the prediction model and $\theta^{pre}$ denotes the model parameters of pre-trained LLMs.

On this basis, the ICL method is applied to improve the accuracy of prediction. ICL is a prompt engineering method for LLMs to learn new tasks by including known examples in the prompt. It presents the advantage of using off-the-shelf LLMs to solve new tasks without explicit re-training and fine-tuning. However, the performance of ICL can be unstable and highly affected by the demonstration selection [191]. As a result, the selection method of ICL demonstrations is important to improve the accuracy of traffic prediction. To distinguish between the data used to develop the ICL demonstrations and the data used for testing, the ICL demonstrations are assumed to be selected from a training dataset where the true future throughput values are known. The test data are from another testing dataset with no overlap between the two datasets. To perform a $M$-shot prediction, the index set of selected ICL demonstrations $I^{(t)}$ is defined as:

$$I^{(t)} = \{i_1, i_2, ...i_M\}, \tag{8.4}$$

where $M$ denotes the total number of selected ICL demonstrations. The selected ICL demonstrations set are then defined as:

$$S^{(t)} = \{[(\Gamma_{tr}^{(i)}, Z_{tr}^{(i)}), y_{tr}^{(i)}]|\forall i \in I^{(t)}\}, \tag{8.5}$$

where $(\Gamma_{tr}^{(i)}, Z_{tr}^{(i)})$ denotes the input downlink throughput time series and the input contextual information of the $i^{th}$ training data. $y_{tr}^{(i)}$ denotes the corresponding true future throughput value. Eq. (8.3) can be seen as the prediction process without ICL. By integrating the selected demonstrations into Eq. (8.3), the ICL-based traffic prediction can be then rewritten as:

$$\hat{y}_{te}^{(t)} = f([\Gamma_{te}^{(t)}, Z_{te}^{(t)}], S^{(t)}|\theta^{pre}). \tag{8.6}$$

The optimization goal is to minimize the gap between the predicted downlink throughput and the true downlink throughput, which is given as:

$$\underset{I^{(t)}}{\operatorname{argmin}} \frac{1}{T} \sum_{t} L(\hat{y}_{te}^{(t)}, y_{te}^{(t)}), \tag{8.7}$$

where $y_{te}^{(t)}$ denotes the true future throughput value of the given test data. $T$ denotes the length of the test set. $L(\hat{y}_{te}^{(t)}, y_{te}^{(t)})$ denotes the loss function and it can be defined as the squared Euclidean distance, which is given as:

$$L(\hat{y}_{te}^{(t)}, y_{te}^{(t)}) = (\hat{y}_{te}^{(t)} - y_{te}^{(t)})^2 \tag{8.8}$$

### 8.1.2 Data Preprocessing and Prompt design

To create ICL demonstrations and set up tests, the whole dataset is first divided into two parts. The first part is considered the training dataset, while the second part serves as the testing dataset. Next, both datasets are partitioned into multiple partially overlapping short sequences. The length of these sequences is equal to the historical time window $H$.

During the prediction, the ICL sequences are first selected from the training dataset as context series. Then they are combined with the testing time series for prompt generation. In this way, the prediction is made by LLMs based on two parts of memories. The first part of memory is the long-term memory provided by the context series. It shows the historical traffic patterns of the mobile device and indicates some consistent information such as the maximum capacity of the communication channel and the impact of the contextual information on the traffic. The second part of the memory is the short-term memory provided by the testing time series from the most recent previous period. It indicates the current trend of traffic flow.

Based on this design, the prompt of the mobile traffic prediction task includes four parts, task description, demonstrations, input information, and output formatting. The task description part focuses on describing the known information of the scenario and the role LLM plays in the task. For instance, the task description describes the traffic prediction task and indicates the coexistence of 5G and LTE technologies setup.

In the next part, the selected ICL examples are converted into human-like language descriptions by substituting numeric values into a natural language template. In case of

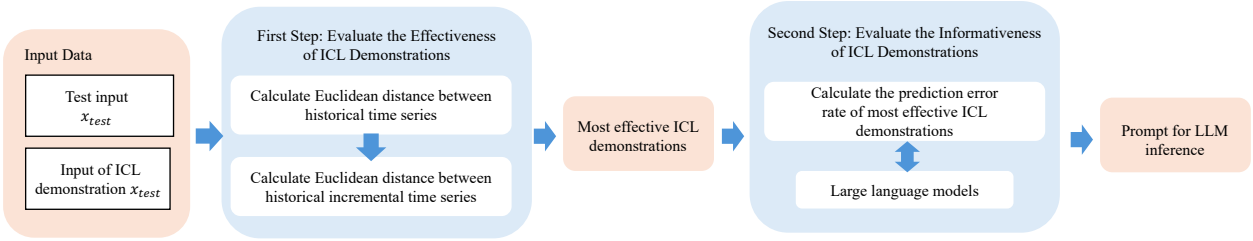



Figure 8.2: The workflow of the two-step ICL demonstration selection method.

missing data, the corresponding contextual descriptions are removed to ensure the coherence of the prompts. After conversion, the ICL examples are aligned and integrated into the prompt. Similarly, in the input information part, the test data are also converted into a natural language format and integrated into the prompt. Specifically, to limit the output length and enhance prediction stability, a mechanism is designed for the automatic conversion of the measurement unit of the downlink traffic.

In our algorithm design, it is observed that the model becomes less accurate when predicting traffic with larger magnitudes. One possible reason is that LLMs generate predictions digit by digit. As the number of digits increases, the correlation between different digits may weaken, leading to reduced prediction accuracy and stability. To mitigate such concerns, when the throughput values in the historical traffic time series are greater than $10^5$, the throughput units are converted from kilobits per second to megabits per second, and the values are scaled down accordingly. With this mechanism, the precision of the predicted values can still reach 1%, and the hallucinations and instability due to long digits are reduced.

Output formatting is also included to define how LLMs should output the results. With clear output formatting, the predicted values can be easily extracted from the natural language output.

### 8.1.3 Deployment Considerations for LLM-based Predictions

In this subsection, the deployment considerations for LLM-based predictions in real-world 5G and 6G networks are discussed. The LLM inference is intended to run on edge servers located at BSs or access points, or cloud servers. These servers are usually equipped with graphics processing units (GPUs) and AI accelerator hardware, which can support the large computation demand of LLM efficiently. Moreover, they also have powerful and fast-access storage resources to meet the storage requirements and low-latency inference requirements. On these infrastructure, a LLM with less than 15 billion parameters is expected to achieve

an inference time of around 100 ms [192], with a network communication latency of around 50 ms [193], depending on hardware settings and network conditions.

These delays remain well within acceptable limits for per-second traffic prediction. For real-world deployment in 5G or 6G networks, hybrid strategies can also be adopted to further ensure the feasibility of the proposed method. Lightweight LLMs can be deployed at the edge to handle predictions involving simpler traffic patterns, while larger LLMs in the cloud can be selectively invoked for more complex scenarios.

Another concern of large-scale deployment in 5G or 6G networks is that LLM inference usually consumes a lot of energy. Deploying LLM to edge or cloud servers helps minimize the energy impact on end-user devices and enables a better resource utilization. Furthermore, in real-world deployment, optimization techniques can be used to further reduce energy consumption. For example, model quantization strategies can compresses model weights to lower bit precision to the memory usage and energy usage of LLMs [194]. Similarly, KV caching reduces redundant computation by storing key and value tensors from previous tokens and this also reduce the energy consumption of LLM inference [195].

In summary, the proposed framework is feasible for real-world deployment in 5G and 6G networks, with acceptable latency and manageable hardware requirements. Moreover, optimization techniques can be used to reduce the energy consumption of LLM in large-scale deployments.

## 8.2 Two-step efficient demonstration selection

In this section, we explain how we perform efficient demonstration selection for ICL-based traffic prediction. Specifically, we first discuss what makes a suitable demonstration for ICL that can improve the prediction accuracy as much as possible. Then, we introduce our proposed two-step selection method.

### 8.2.1 Finding Suitable Demonstrations for ICL

Although the effect of ICL has been validated in previous studies, the performance of ICL still suffers from instability. It is highly task-dependent and easily influenced by ICL demonstrations' selection. To optimize the accuracy gains from integrating ICL while minimizing the computational costs, we first need to decide whether to apply ICL for each test case and then we select the ICL demonstrations that are most likely to improve the prediction accuracy.

An intuitive way to select demonstrations is to include all the available demonstrations in the prompt. Each new demonstration addition is expected to bring more usable information, thus improving the LLM performance. However, in real-world tasks, the length of the prompt is usually restricted and only a minimal number of ICL demonstrations can be integrated into the prompt. In addition, an increased number of ICL examples also results in longer inference time and requires more memory to run. This will constrain the practical use of LLMs in many scenarios. Moreover, undesired ICL examples are of limited help in improving the LLM performance. In some complicated tasks, they may even provide redundant information and add extra difficulties for LLMs to understand the task [196]. Therefore, it is critical to decide whether to apply ICL for each test case and which ICL demonstrations to use.

In general, LLMs expect ICL demonstrations to provide four types of information, namely, the output format, the distribution of the input, the range of the output, and the correlation of inputs and outputs [197]. In our system design, the output format has been indicated by the output formatting part in the prompt. The distribution of the input and the range of the output can also be inferred from the task description and the pre-trained knowledge base of LLMs. Therefore, in the given traffic prediction task, the ICL demonstration selection largely depends on whether the demonstration can provide valid information regarding the correlation of inputs and outputs.

Based on the above thoughts, we propose two rules that should be considered during the ICL demonstration selection: ***effectiveness*** and ***informativeness***. Effectiveness means that the selected ICL demonstrations can provide valid information and are helpful for the test case. Informativeness means that the selected ICL examples can provide information that the original pre-trained LLM does not have. According to these two rules, in this work, we design a two-step ICL demonstration selection method, which is shown in Fig. 8.2. In the first step, the similarity between ICL demonstrations and the test data are calculated by measuring the distance between historical time series and incremental time series. This step is performed to ensure the effectiveness of selected ICL examples. In the second step, the prediction error rate of the selected ICL demonstration is measured to evaluate the informativeness of the example and decide if the ICL should be used for the given test case.

In the next subsection, a theoretical analysis of ICL is provided and detailed explanations about how to develop the two-step ICL demonstration selection method are given.

### 8.2.2 First step: Evaluate the effectiveness of ICL demonstrations

To understand the theory of ICL, the output generation of a pre-trained LLM is first modeled as a hidden Markov model [198]. The general form can be given as:

$$o_j \sim P(o_j = o|s_j = s), \tag{8.9}$$

where $o_j$ denotes the $j^{th}$ output token, and $P(.)$ denotes the probability. $s_j$ denotes the state before generating the $j^{th}$ output token, which is related to the input prompt and the previous state. This formulation means the next output character is sampled from a probability distribution of all the possible tokens according to the current state. The state transition can be formulated as:

$$s_j \sim P(s_{j+1} = s'|s_j = s). \tag{8.10}$$

With this setting, the parameter $\theta$ can be used to illustrate the transition probabilities matrix of the hidden Markov process and it is developed from the prompts.

To evaluate the effectiveness of ICL demonstrations, following [199], we first explain the similarity between the ICL and the kernel regression process. We define the prediction task as $\theta^*$, and the task $\theta^*$ describes the relationship between the input and the output of the prediction. During the prediction, LLMs first calculate the probabilities of all the output tokens. A kernel function can be constructed as:

$$K(x, x') =< \hat{P}(o|x, \theta^{pre})^T, \Sigma_{pre}^{-1} \hat{P}(o|x', \theta^{pre}) >, \tag{8.11}$$

where $x = [\Gamma^{(t)}, Z^{(t)}]$ denotes the input of LLMs in the case of zero-shot prediction. $< ., . >$ denotes the inner product. $\hat{P}(o|x, \theta^{pre})$ is a vector that denotes the probabilities of all the output tokens given the input $x$ under pre-trained LLM parameters $\theta^{pre}$. In the following of this subsection, we denote $\hat{P}(o|x, \theta^{pre})$ as $\hat{P}_x$ for brevity. $\Sigma_{pre}$ is the covariance of the $\hat{P}(o|x, \theta^{pre})$.

By leveraging the property of the kernel function, a series of equations can be constructed, which are then used to establish the relationship between ICL and kernel regression. According to the definition of covariance, the kernel function has the following properties:

$$\mathbb{E}_{x_D}[K(x_{te}, x_D)\hat{P}_{x_D}^T] = \hat{P}_{x_{te}}^T \Sigma_{pre}^{-1} \mathbb{E}[\hat{P}_{x_D} \hat{P}_{x_D}^T] = \hat{P}_{x_{te}}^T, \tag{8.12}$$

where $x_D = [\Gamma_{tr}^{(i)}, Z_{tr}^{(i)}]$ denotes the input of selected ICL demonstrations. $x_{te} = [\Gamma_{te}^{(i)}, Z_{te}^{(i)}]$ denotes the input of test data. $\mathbb{E}_{x_D}[K(x_{te}, x_D)\hat{P}_{x_D}^T]$ denotes the expectations of $K(x_{te}, x_D)\hat{P}_{x_D}^T$ given $x_D$. Considering the linear correspondence between the two ends of the equation, the equation can be transformed as:

$$\mathbb{E}_{x_D}[K(x_{te}, x_D)P(o = y_{te}|x_D, \theta^{pre})] = P(o = y_{te}|x_{te}, \theta^{pre}). \tag{8.13}$$

In kernel regression, if the selected ICL demonstrations are used as observed data points, the predicted values of test data $x_{te}$ can be estimated as:

$$\hat{y}_k = \frac{\sum_{m=1}^{M} e(y_{tr}^{(i_m)})K(x_{te}, x_D)}{\sum_{m=1}^{M} K(x_{te}, x_D)} = \frac{1}{M}\sum_{m=1}^{M} \hat{P}(y|x_{te}^{(i_m)}, \theta^{pre})^T \Sigma_{pre}^{-1} \hat{P}(y|x_D^{(i_m)}, \theta^{pre}) e(y_D^{(i_m)}), \tag{8.14}$$

where $\hat{y}_k$ denotes the predicted values of kernel regression. $e(y_D^{(i_m)})$ denotes the one-hot vector for the output $y_D^{(i_m)}$.

Since the task is described as $\theta^*$, the data pairs $(x_D^{(i_m)}, y_D^{(i_m)})$ can be seen as sampled from the probability distribution $P(y = Y|x_D^{(i_m)}, \theta^*)$. According to Hoeffding's inequality, the upper bound on the probability that the sum of random variables deviates from its expected value can be qualified. If the probability is set as $1 - 2\delta$, the inequality can be converted into:

$$\begin{aligned} &||\hat{y}_k - \mathbb{E}_{x_D}[K(x_{te}, x_D)P(o = y_{te}|x_D, \theta^*)]||_\infty \\ =&||\frac{1}{M}\sum_{m=1}^{M} \hat{P}(y|x_{te}^{(i_m)}, \theta^*)^T \Sigma_{pre}^{-1} \hat{P}(y|x_D^{(i_m)}, \theta^*) e(y_D^{(i_m)}) \\ &- \mathbb{E}_{x_D}[K(x_{te}, x_D)P(o = y_{te}|x_D, \theta^*)]||_\infty \\ <&\sqrt{\frac{1}{2M}\ln\frac{4D}{\delta}}, \end{aligned} \tag{8.15}$$

where $||.||_\infty$ denotes the L-infinity norm. $D$ denotes the dimension of $\hat{P}(y|x_{te}^{(i_m)}, \theta^*)$. We denote the upper bound of Frobenius-norm of $\hat{P}(o|x, \theta)$ as $\eta$. Since the relationship between inputs and outputs of the traffic prediction task is partially consistent with the knowledge gained by the LLMs through pre-training, the difference between the output distribution under $\theta^{pre}$ and $\theta^*$ can be upper-bounded as:

$$|< \hat{P}(y|x_{te}, \theta^*)^T, \Sigma_*^{-1} \hat{P}(y|x_D, \theta^*) > - < \hat{P}(y|x_{te}, \theta^{pre})^T, \Sigma_{pre}^{-1} \hat{P}(y|x_D, \theta^{pre}) > | \leq \eta^2 \epsilon_\theta, \tag{8.16}$$

where $\epsilon_\theta$ is defined to quantize the upper limit. Therefore, the relationship between $\hat{y}$ and $P(o = y_{te}|x_{te}, \theta^*)$ can be derived:

$$||\hat{y} - P(o = y_{te}|x_{te}, \theta^*)||_\infty \leq \sqrt{\frac{1}{2m} \ln \frac{4D}{\delta}} + \eta^2 \epsilon_\theta. \tag{8.17}$$

The above inference process illustrates the high similarity between the estimates of $P(o = y_{te}|x_{te}, \theta^*)$ and kernel regression. According to [200], the behavior of ICL is similar to explicit fine-tuning with ICL demonstrations, which means $P(o = y_{te}|x_{te}, \theta^*)$ also has a high similarity with $P(o = y_{te}|x_{te}, x_D, \theta^{pre})$. Therefore, it can be concluded that the ICL process is comparable to kernel regression based on selected demonstrations.

According to the property of kernel regression, demonstrations that are similar to the test data are supposed to lead to better estimation results. Therefore, the demonstrations can be selected by minimizing the kernel similarity between the test data and the selected ICL demonstration, which is given as:

$$\min_i \ [\hat{P}_{x_{te}}^T \Sigma_{pre}^{-1} \hat{P}_{x_D}]. \tag{8.18}$$

For the traffic prediction problem, it is difficult to compute the above values directly. However, the kernel similarity can be understood as the semantic similarity between the test input $x_{te}$ and input of ICL demonstration $x_D$. In this work, we measure the semantic similarity from two aspects. For the first aspect, the Euclidean distance between the historical downlink throughput time series of test data and the ICL demonstrations is calculated as:

$$e_1 = ||\Gamma_{te} - \Gamma_D||_2. \tag{8.19}$$

For the second aspect, the incremental time series of downlink throughput is first calculated:

$$\Delta\Gamma^{(t)} = \{\gamma^{(t)} - \gamma^{(t-1)}, ..., \gamma^{(t-H+2)} - \gamma^{(t-H+1)}\}. \tag{8.20}$$

Next, the Euclidean distance between the incremental time series of test data and the ICL demonstrations is calculated as:

$$e_2 = ||\Delta\Gamma_{te} - \Delta\Gamma_D||_2. \tag{8.21}$$

The sum of $e_1$ and $e_2$ is used to estimate the effectiveness of ICL demonstrations. A smaller sum indicates a higher similarity between the ICL demonstration and the test sample,

suggesting greater effectiveness of the ICL demonstration. Therefore, we can select the most effective samples by minimizing the sum, which can be written as min $(e_1 + e_2)$.

In the following, a theoretical analysis is provided of how the first-step ICL demonstration selection impacts the LLM performance. Without any demonstrations, the LLM relies solely on its pret-rained knowledge to make predictions. The prediction error can be formulated as:

$$Er(\hat{y_{test}}, y_{test}) = \epsilon_{noise} + \epsilon_{LLM} \tag{8.22}$$

where $Er(\hat{y_{test}}, y_{test})$ denotes the prediction error, $\epsilon_{noise}$ denotes the intrinsic noise error, and $\epsilon_{LLM}$ denotes the error due to LLM's lack of task-specific knowledge. Since ICL is theoretically related to kernel regression, after applying ICL, the error can be reformulated as:

$$Er(\hat{y_{test}}, y_{test}) = \epsilon_{noise} + O(1 - K(x_{test}, x_{ICL})) \tag{8.23}$$

where $O(.)$ is the asymptotic upper bound of the error term. As shown in the formulation, $\epsilon_{LLM}$ was replaced by $O(1 - K(x_{test}, x_{ICL}))$ through ICL. That means ICL reduces prediction error when $K(x_{test}, x_{ICL})$ is maximized.

In the first step of demonstration selection, we maximize $K(x_{test}, x_{ICL})$ by selecting the demonstration most similar to $x_{test}$. This ensures that through ICL, the selected demonstration minimizes the prediction error introduced by the LLM's lack of task-specific knowledge. Moreover, compared with random or constant ICL demonstration selection methods, our proposed method ensures larger $K(x_{test}, x_{ICL})$, thereby leading to greater stability in LLM-based predictions.

### 8.2.3 Second step: Evaluate the informativeness of ICL demonstrations

In this subsection, we explain the second step of ICL demonstration selection and illustrate how the informativeness of ICL demonstrations is evaluated. In this step, ICL will be performed only if the selected ICL demonstrations are informative, i.e., the selected ICL examples can provide information the original pre-trained LLM does not have. In this way, the computational and negative impact of unnecessary ICL can be reduced.

Following the hidden Markov model defined in the last subsection, the loss function of ICL under the set of selected demonstrations $S$ can first be defined as:

$$L(\theta; S, t) = \log P(y_{te}^{(t)} | \theta, S^{(t)}, x_{te}^{(t)}), \tag{8.24}$$

where $P(y_{te}|\theta, S, x_{te})$ denotes the probability of obtaining the desired output of the given test case under the selected demonstrations $S$.

With this loss function, the optimization goal of ICL can be formulated as:

$$\begin{aligned} \theta^{opt} &= \underset{\theta}{\operatorname{argmin}} L(\theta; S, t) \\ &= \underset{\theta}{\operatorname{argmin}}(-\log P(y_{te}^{(t)}|\theta, S^{(t)}, x_{te}^{(t)})), \end{aligned} \tag{8.25}$$

which means the goal is to find an optimal $\theta$ that can maximize the probability of getting correct $y_{te}$ for all of the given test examples.

In this case, it is difficult to directly find the optimal $\theta$ since it is a latent variable that is not explicitly defined. A heuristic method is to first assume a desired $\theta$ which can be denoted as $\theta_0$. Under this setting, to enhance the informativeness of the selected ICL demonstrations, we maximize how much the selected demonstrations can change the latent concept from an initial pattern to a desired pattern. This is captured by the difference in log-probabilities, which can be formulated as:

$$\underset{S^{(t)}}{max}\ [\log P(\theta_0|S^{(t)}) - \log P(\theta^{pre}|S^{(t)})]. \tag{8.26}$$

where $\theta^{pre}$ is the pre-trained latent concept while only the task description is used as the prompt without any ICL demonstrations.

This formulation can be further expanded according to the Bayes rules and simplified as:

$$\begin{aligned} P(\theta_0|S^{(t)}) &= \frac{P(y_D|\theta_0, x_D)P(\theta_0|X = x_D)P(x_D)}{P(y_D|x_D)P(x_D)} \\ &= \frac{P(y_D|\theta_0, x_D)P(\theta_0|x_D)}{P(y_D|x_D)}, \end{aligned} \tag{8.27}$$

$$\begin{aligned} P(\theta^{pre}|S^{(t)}) &= \frac{P(y_D|\theta^{pre}, x_D)P(\theta^{pre}|x_D)P(x_D)}{P(y_D|x_D)P(x_D)} \\ &= \frac{P(y_D|\theta^{pre}, x_D)P(\theta^{pre}|x_D)}{P(y_D|x_D)}. \end{aligned} \tag{8.28}$$

The simplified terms are used to substitute $P(\theta_0|S^{(t)})$ and $P(\theta^{pre}|S^{(t)})$ in Eq. (8.26), which can be written as:

$$
\begin{aligned}
&\underset{S^{(t)}}{max}\ \log \frac{P(\theta_0|S^{(t)})}{P(\theta^{pre}|S^{(t)})}. \\
=&\underset{S^{(t)}}{max}\ \log \frac{P(y_D|\theta^{pre}, x_D)P(\theta^{pre}|x_D)}{P(y_D|\theta_0, x_D)P(\theta_0|x_D)}
\end{aligned}
\tag{8.29}
$$

Since general-purpose LLM without fine-tuning on the target task is used in this work, the model lacks strong prior preferences over latent concepts for any given inputs. The variation of $P(\theta|x_D)$ is typically smaller than that of the likelihood term $P(y_D|\theta, x_D)$. In this case, we can assume that $P(\theta|x_D)$ is approximately uniform across $\theta_0$ and $\theta^{pre}$. Under this assumption, the difference in the priors becomes negligible. In this way, it can be simplified as:

$$
\underset{x_D}{max}\ [\log P(y_D|\theta_0, x_D) - \log P(y_D|\theta^{pre}, x_D)]. \tag{8.30}
$$

An intuitive explanation of this formulation is presented as follows. It measures the difference between the zero-shot predicted results and the expected output of the ICL demonstrations. This indicates how much information about the expected prediction for the given ICL demonstration is not contained in the pre-trained parameters of the LLM. Informative examples are those that cannot be accurately predicted by pre-trained models and, therefore, require ICL. The output of LLM can be viewed as a sequence formed by multiple digits, where each digit follows its own probability distribution. However, these distributions are interdependent, making it difficult to directly compute the overall probability distribution of the final output. Considering the semantic meaning of the output values, the prediction error rate of ICL demonstrations $Er$ is defined to measure the difference:

$$
Er = \frac{|\hat{y}_D - y_D|}{y_D}, \tag{8.31}
$$

where $\hat{y_D}$ is the zero-shot predicted results of ICL demonstration $x_D$:

$$
\hat{y}_D \sim P(y|\theta^{pre}, x_D). \tag{8.32}
$$

In the second step of ICL demonstration selection, the prediction error rate $Er$ is calculated for the demonstration selected in the first step and a threshold is set. If the error rate is above the threshold, the demonstration is considered to be informative and the ICL is performed. If the error value is below the threshold, the sample is considered to be insufficiently informative and ICL is not required for the given demonstration.

To be more straight, in the first step, our proposed method identifies past traffic patterns that are similar to the current input. Then, it evaluates the informativeness by checking whether the LLM can already predict the outcome of each selected demonstration in a zero-shot setting. If it can, the demonstration is considered redundant and excluded. If not, the demonstration is more likely to contain useful information and is kept. These two steps ensures that each selected example is both effective and informative. So our proposed selection method can improve the prediction performance without adding unnecessary length to the prompt.

In the following, we analyze how the second step of ICL demonstration selection influences the prediction error according to Eq. (8.22) and Eq. (8.23). If a selected demonstration has a low zero-shot prediction error, two cases may arise. The first case is that the test sample is similar to the selected demonstration. In this case, a low prediction error of the selected demonstration means that the LLM's prediction error for the test sample, $\epsilon_{LLM}$, is also low. The second case is that although the selected demonstration has a low zero-shot prediction error, the test sample is not similar enough to the selected demonstration. In this case, the kernel product of the test sample input and selected demonstration input is small. As a result, in both cases, using the ICL demonstrations selected in the first step may not significantly reduce prediction error. Therefore, in this step, we filter out ICL demonstrations that do not provide additional performance improvement, ensuring that only ICL demonstrations that reduce prediction error are used.

### 8.2.4 Computational complexity of proposed method

In this subsection, we discuss the computational complexity of the proposed demonstration selection method. For the first step, the calculations are performed during the test time. This step involves traversing all available ICL demonstrations and computing the metrics $e_1$ and $e_2$ for each demonstration. Consequently, the computational complexity of this step is proportional to the number of available ICL demonstrations, denoted as $n_d$, and can be expressed as $O(n_d)$.

For the second step, part of the calculations can be performed offline before the test phase. For instance, the calculation of the predicted error rate $Er$ is independent of the specific test case and can be completed in advance. In this case, during the test time, the only calculation required is to compare the predicted error rates of selected ICL demonstrations with the threshold value, which is computationally negligible. Therefore, the overall computational complexity of the proposed method remains $O(n_d)$ for selecting a single demonstration. For multiple selected demonstrations, the overall computational complexity is $O(n_d n_s)$, where $n_s$ denotes the number of selected demonstrations.

In the following, a comparative analysis of computational complexity with other selection methods is provided. ICL demonstration selection methods generally fall into two categories: gradient-based and retrieval-based selection. Gradient-based methods usually require multiple model evaluations, resulting in a significantly higher complexity, $O(N_d m_{llm})$, where $m_{llm}$ denotes the size of the LLM. So the Gradient-based selection methods have a much higher complexity than our proposed method. Retrieval-based methods select the nearest demonstration based on a similarity metric with a complexity of $O(n_d n_s)$.

The first step of our method can be regarded as a retrieval-based selection step. The difference is that our approach includes the second step with an additional pre-processing step to calculate the predicted error rates of demonstrations. Since this part of the calculation can be performed offline before the test phase, the total computation of the proposed selection approach during the test phase is comparable with the standard retrieval-based methods but increases the prediction performance.

In conclusion, Our method achieves a balance between computational efficiency and performance by maintaining linear complexity while enhancing the prediction performance.

### 8.2.5 Batch calibration

Due to the current limitations of LLMs, the predictive results using a single ICL demonstration may suffer from instability. However, for multiple ICL demonstrations, the performance of ICL is usually sensitive to the permutation of demonstrations [201]. On the other hand, the permutation problem of ICL demonstrations is complex, and finding the optimal permutation directly is very difficult.

In this work, a batch calibration method is adopted to mitigate the influence of permutation with multiple ICL demonstrations. Specifically, other than performing M-shot prediction with all the selected ICL demonstrations at one time, the proposed method performs one-shot prediction separately with different ICL demonstrations. Then the outputs of one-shot predictions are averaged as the final result. This process can be formulated as:

$$\hat{y}_{te} = \frac{1}{M} \sum_{i_1}^{i_m} \hat{y}_{te}^{(i_m)}, \tag{8.33}$$

where:

$$\hat{y}_{te}^{(i_m)} \sim P(y|\theta^{pre}, (\Gamma_{tr}^{(i_m)}, Z_{tr}^{(i_m)}, y_{tr}^{(i_m)}), x_{te}). \tag{8.34}$$

$\hat{y}_{te}^{(i_m)}$ denotes one-shot prediction results with the $i_m^{th}$ ICL demonstration. $i_m$ is the index of the $m^{th}$ selected ICL demonstration. With this method, the robustness and accuracy of LLM-based prediction can be increased.

## 8.3 Numerical Results

### 8.3.1 Experimental Settings and Evaluation Metrics

In this work, the real-world 5G dataset in [202] is used. The dataset includes two kinds of data, static data and driving data, and it also includes three different activities, namely downloading files, watching videos on Netflix, and watching videos on Amazon. The dataset also considers the co-existence of multiple technologies, including 5G, LTE, and HSPA+, which adds extra difficulties to the prediction task and makes it more comparable to real-world situations.

The dataset includes four types of contextual information, the throughput, the channel metrics, the neighboring cell metrics, and other context metrics. There are 25 contextual features in total. In addition to the historical downlink throughput feature, we selected four contextual features for the downlink traffic prediction task: uplink throughput, the reference signal received power of the currently connected cell, the reference signal received power of the neighboring cell, and the network mode. The feature selection was based on the feature ranking results in [54], where the authors calculated the feature importance scores using the same dataset used in this study. Based on these scores, we selected the four most influential contextual features to enhance prediction performance.

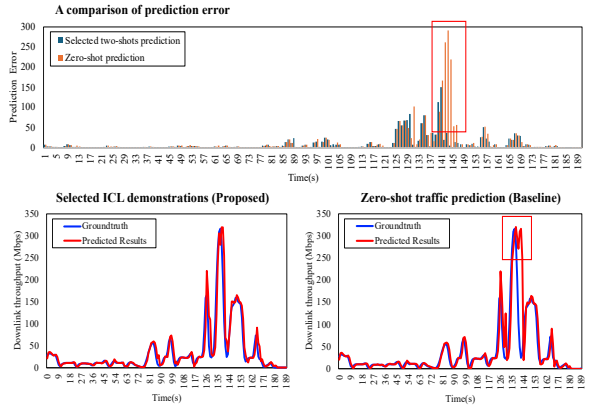
Unlike traditional ML-based traffic prediction methods, this work does not apply other pre-processing steps such as normalization or standardization. The primary reason is to preserve the semantic integrity of numerical values. Since LLMs are pre-trained on diverse textual and numerical data, they can interpret raw numerical values without requiring rescaling. Similarly, pre-processing steps such as one-hot encoding is not used for categorical features because LLMs can directly process natural language-based inputs.

During the experiments, we equally divided the whole dataset into two parts. The first part is used for ICL demonstration selection and the second part is used for testing. The historical time window $H$ is set as 5.

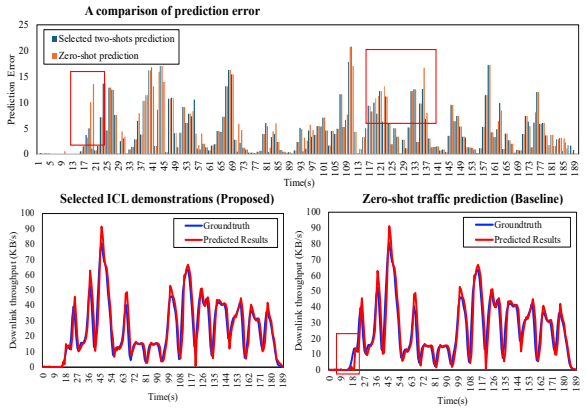
Three indicators are used to evaluate the prediction performance, the mean absolute error (MAE), the root mean squared error (RMSE), and the $R^2$-Score. MAE measures the average of the absolute errors between predicted and actual values. It can be calculated as:

$$MAE = \frac{1}{T}\sum_{t}|y_{te}^{(t)} - \hat{y}_{te}^{(t)}|. \tag{8.35}$$

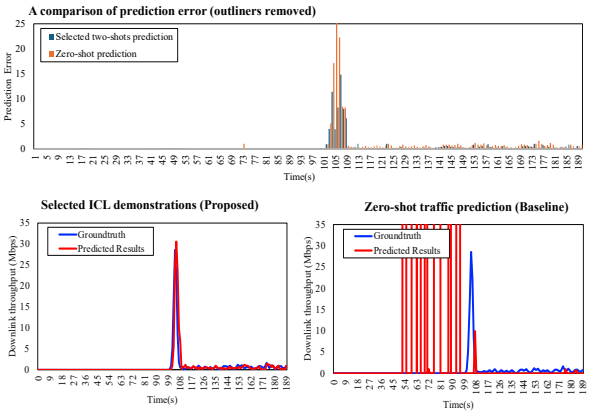
RMSE measures the standard deviation of the predicted errors. It can be calculated



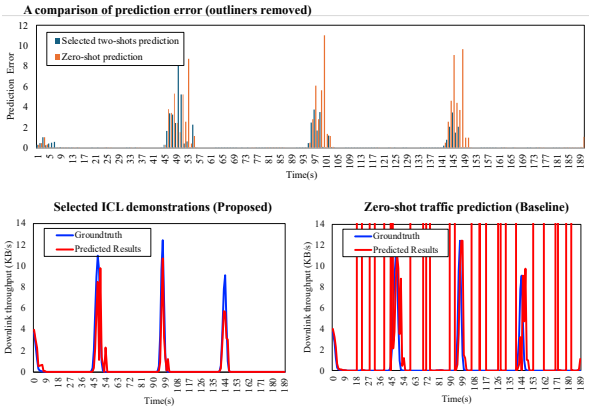

(a) Driving, and downloading files.




(b) Static, and downloading files.




(c) Driving, and watching Amazon videos.




(d) Driving, and watching Netflix videos.

Figure 8.3: A comparison of traffic prediction with the proposed selected ICL demonstrations and zero-shot traffic prediction in four scenarios using PHI-3-medium.

as:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t}(y_{te}^{(t)} - \hat{y}_{te}^{(t)})^2}. \tag{8.36}$$

$R^2$-Score is a commonly used metric to evaluate how well the model predicts the outcome of the dependent variable. It can be calculated as:

$$R^2 - Score = 1 - \frac{\sum_t (y_{te}^{(t)} - \hat{y}_{te}^{(t)})^2/T}{\sum_t (y_{te}^{(t)} - \overline{y_{te}})^2/T}. \tag{8.37}$$

In this work, we use PHI-3 model as the pre-trained LLM for prediction [203]. The PHI-3 model is a fully open-source language model that ranges in size from 3.8 billion to 14 billion parameters depending on the version. If not otherwise specified, we use PHI-3-medium with 14 billion model parameters as the testing model. In the next section, the numerical results are provided to demonstrate the effectiveness of our proposed method.

Table 8.1: MAE, RMSE and $R^2$-Score of mobile traffic prediction under different scenarios.

| | Downloading files - driving | | | Downloading files - static | | | Watching Amazon videos - driving | | | Watching Netflix videos - driving | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Selected two-demo prediction (Proposed) | Zero-shot prediction | Weighted moving average | Selected two-demo prediction (Proposed) | Zero-shot prediction | Weighted moving average | Selected two-demo prediction (Proposed) | Zero-shot prediction | Weighted moving average | Selected two-demo prediction (Proposed) | Zero-shot prediction | Weighted moving average |
| MAE | **9.4913** | 13.816 | 16.052 | **4.4605** | 4.7327 | 8.0972 | **0.42907** | 0.73560 | 0.58080 | **0.29391** | 0.86242 | 0.58674 |
| RMSE | **21.827** | 39.584 | 32.848 | **6.3854** | 6.6346 | 11.125 | **1.7275** | 3.0858 | 2.5201 | **1.0267** | 3.2867 | 1.7135 |
| $R^2$-Score | **0.84743** | 0.49822 | 0.77634 | **0.87349** | 0.87434 | 0.61598 | **0.65831** | -0.0902 | 0.27286 | **0.74821** | -1.5804 | 0.29866 |

### 8.3.2 Performance Analysis of ICL

In this subsection, we analyze how ICL influences the predictive results. We first perform traffic prediction in four different application scenarios, and compare the predictive results obtained using our proposed ICL method with the zero-shot prediction results. During the experiments, we select two ICL demonstrations since more demonstrations will lead to a higher computation cost and according to the numerical result, two demonstrations

already can lead to a better performance and eliminate noise. It should be noted that the choice of the number of ICL demonstrations is also highly dependent on the size of the dataset and cannot be generalized. Using the batch calibration method makes it easier to manage and adjust the number of demonstrations without restructuring the prompt. In practice, we observe that adding more demonstrations does not always lead to better performance. When too many demonstrations are used, additional examples may introduce noise or redundancy. To decide the optimal number of ICL demonstrations, we can first consider the inference time requirements and available computational resources. Then, we can observe how model performance changes as the number of demonstrations increases. A simple approach is to gradually increase the number of demonstrations and monitor the predictions. If the results become more stable and accurate at first but then stop improving or even begin to degrade, we can choose the number just before that turning point.

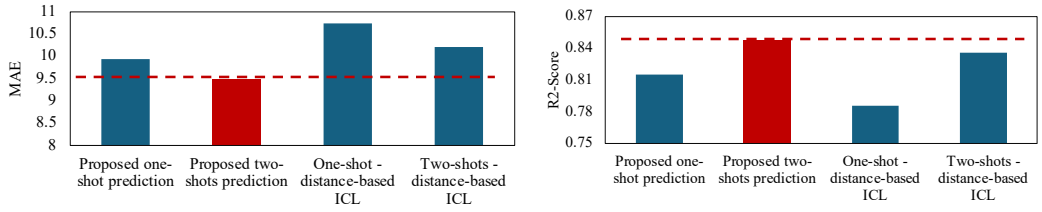
Fig. 8.3 shows the traffic prediction results using PHI-3-medium with selected ICL demonstrations, and zero-shot prediction results. The term zero-shot refers to predictions made without incorporating any ICL demonstrations in the prompt and serves as a baseline. To better highlight the differences between the predicted and ground-truth values over time, we have also included the prediction error bars for both the proposed method and the zero-shot baseline for comparison. The prediction error bars of our proposed method is shown in blue color while the zero-shot baseline is shown in orange color.

As illustrated in Fig. 8.3, our proposed ICL method improves both the accuracy and robustness of predictions across different application scenarios. In the first two scenarios, the performance difference between predictions with and without ICL is relatively small. But it can still be observed that the performance with selected ICL demonstrations is slightly better than zero-shot predictions and some noticeable differences are marked with red boxes.

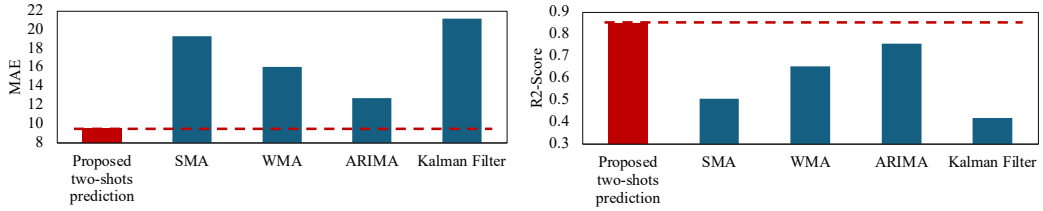
In contrast, the third and fourth scenarios exhibit significant performance degradation under zero-shot prediction. In particular, some predicted values are much higher than the typical traffic bitrate range. These values are generally due to the inability of the LLM to correctly output the end mark, and they are considered hallucinations. The use of selected ICL demonstrations effectively mitigates these hallucination issues, leading to more accurate and stable predictions.

To further analyze prediction performance, in the visualization of the prediction errors, we remove these unstable outlier points. The remaining data shows that predictions made with selected ICL demonstrations consistently still have lower errors compared to zero-shot predictions. This further demonstrate the effectiveness of ICL in enhancing both stability and accuracy.

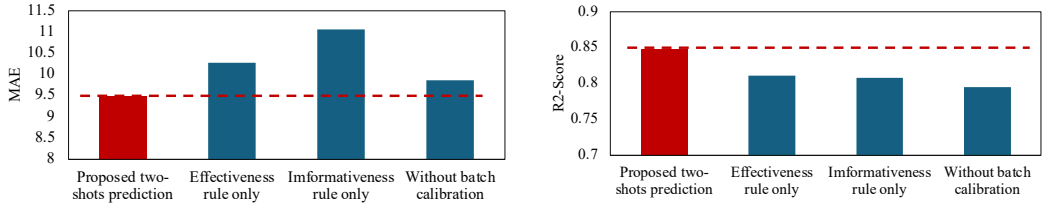
Table 8.1 shows the MAE, RMSE, and $R^2$-Score of the traffic predictions under different application scenarios. The $R^2$-Score ranges from $-\infty$ to 1, where a score closer to 1 reflects better predictive performance. The predictive results with the weighted moving average model [204] are also added for comparison. To enable a better comparison, the hallucinations of zero-shot prediction have been manually removed. As can be observed from the table, our proposed ICL-based traffic prediction method achieves lower MAE, lower RMSE, and higher $R^2$-Scores compared with the other two baselines. The $R^2$-Scores of the ICL-based predictions under different scenarios show that the proposed algorithms perform better in scenarios with gradual changes in traffic flow and perform worse in scenarios with abrupt changes in traffic flow. However, in all the application scenarios, the $R^2$-Scores are above 0.65, indicating that using ICL, the pre-trained LLM can successfully predict the mobile traffic in different scenarios.




(a) A comparison of the proposed similarity measurement and distance-based similarity measurement under varying demonstrations.




(b) A comparison of the proposed method and other traffic prediction methods.




(c) An ablation study on effectiveness and informativeness rules.

Figure 8.4: MAE and $R^2$-Score of mobile traffic prediction under different ICL settings.

Fig. 8.4 further compares the MAE and $R^2$-Scores of mobile traffic prediction under

different ICL settings in the first scenario.

Fig. 8.4(a) assesses the impact of our proposed similarity evaluation method by replacing it with a standard metric based on time series distance measures, keeping the rest of the pipeline identical. According to the simulation results, we show that our similarity evaluation method is more reasonable than the standard distance metric.

Fig. 8.4(b) shows a comparison of the proposed method with three other traffic prediction methods, the simple moving average (SMA) method, the weighted moving average (WMA) method, the ARIMA method and the Kalman Filter-based method. Our proposed method is highlighted in red. According to the results, our method consistently achieves lower MAE and higher $R^2$ than the other four baseline methods.

Fig. 8.4(c) shows a comparison of the MAE and the $R^2$-score for three ablation settings: effectiveness-based selection only, informativeness-based selection only, and the full two-step selection mechanism without batch calibration. As shown in the figure, all three ablated variants result in degraded performance compared to our proposed full two-step selection method. When using only the effectiveness rule, the selected demonstrations may lack sufficient information and lead to redundancy in the prompt. Conversely, when using only the informativeness rule, the selected demonstrations may be less relevant to the test instance and have limited effects on the prediction accuracy. These results highlight the complementary roles of both effectiveness and informativeness in our selection strategy and demonstrate the necessity of each step in achieving optimal performance.

Table 8.2: The ICL ratio, MAE, RMSE and $R^2$-Score under different predicted error rate threshold settings.

| | Zero-shot prediction | Threshold = 0.8 | Threshold = 0.5 | Threshold = 0.35 | Threshold = 0.15 | Threshold = 0.05 | Full ICL |
|---|---|---|---|---|---|---|---|
| ICL ratio | 0.00% | 41.20% | 47.60% | 47.60% | 50.26% | 82.01% | 100.00% |
| $R^2$-Score | 0.49822 | 0.75045 | 0.83880 | 0.84744 | **0.84873** | 0.80990 | 0.81111 |
| MAE | 13.816 | 11.301 | 9.6202 | 9.4913 | **9.4263** | 10.348 | 10.271 |
| RMSE | 39.585 | 27.916 | 22.437 | 21.827 | **21.735** | 24.365 | 24.287 |

### 8.3.3 Performance Analysis under Different Predicted Error Rate Thresholds

In the proposed two-step efficient demonstration selection method, the predicted error rate is used to evaluate the informativeness of selected ICL demonstrations. In this section, we discuss how different predicted error rate thresholds can influence the prediction performance. With these numerical results, we demonstrate the necessity and validity of evaluating the informativeness of ICL demonstrations.

Fig. 8.5 and Table 8.2 show how the setting of the predicted error rate threshold affects the prediction performance. Fig. 8.5 shows the trend in the ICL ratio and the $R^2$-Scores for different error rate threshold settings. The ICL ratio means the ratio of test instances where ICL is used for prediction. In Table 8.2, specific values of ICL ratio, MAE, RMSE, and $R^2$-Scores are given for different prediction error rate threshold settings. As can be observed from the figure and the table, if the threshold is set lower, the ICL ratio will be higher, which means that ICL will be performed more frequently with the test instances. Meanwhile, the $R^2$-Score first increases when the threshold is greater than 0.5, becomes relatively stable when the threshold is greater than 0.15, and finally decreases slightly when the threshold is less than 0.15. This suggests that not all test instances require ICL. In some test instances, the ICL demonstrations will provide useless information and interfere with predictive results. According to the numerical results, setting the threshold between 0.5 and 0.15 is an ideal range to optimize the prediction accuracy and computational cost of ICL.

### 8.3.4 Performance Analysis of LLM-based Prediction and LSTM-based Prediction

In this subsection, we compare the performance of ICL-enabled LLM-based mobile traffic prediction and LSTM-based mobile traffic prediction. Table 8.3 shows the MAE, RMSE, and $R^2$-Scores of LLM-based prediction and LSTM-based prediction in different application scenarios. As can be observed, in most cases, the LSTM-based method achieves better performance than the LLM-based method. This is because the LSTM models are trained based on a huge training dataset, which makes them better capture historical traffic patterns. However, the superiority of the LSTM model comes at the cost of a large number of training computations and a rich training dataset. This will limit the use of LSTM models in some practical application scenarios. Meanwhile, LSTM models trained in one specific scenario cannot be applied to the prediction tasks in other scenarios, whereas LLM-

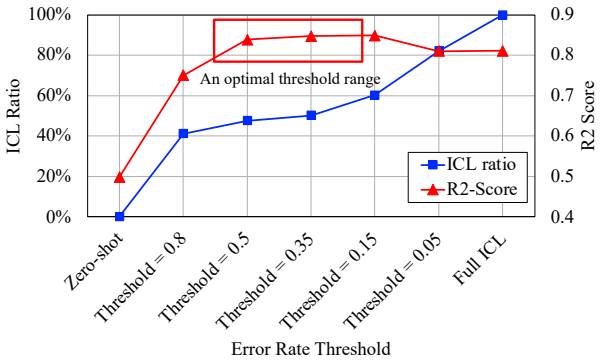



Figure 8.5: The ICL ratio and the $R^2$-Score under different predicted error rate threshold settings.

based methods can work in a one-fits-all manner. In the following, we provide some other numerical results to further illustrate this point.

Table 8.3: MAE, RMSE and $R^2$-Score of mobile traffic prediction of LLM-based method and LSTM-based method.

| | Downloading files - driving | | Downloading files - static | | Watching Amazon videos - driving | | Watching Netflix videos - driving | |
|---|---|---|---|---|---|---|---|---|
| | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction |
| MAE | 9.4913 | **7.3187** | 4.4605 | **2.8560** | 0.42907 | **0.22540** | **0.29391** | 0.30701 |
| RMSE | 21.827 | **17.469** | 6.3854 | **4.6219** | 1.7275 | **0.69268** | 1.0267 | **1.0142** |
| $R^2$-Score | 0.84743 | **0.9022** | 0.87349 | **0.93372** | 0.65830 | **0.9450** | 0.74821 | **0.75427** |

We perform an experiment using the training data and test data from different application scenarios. For the LLM-based method, the test data are from the "Downloading files - Driving" scenario and the ICL demonstrations are selected from other application scenarios. The numerical results are shown in Table 8.4. According to the numerical results, if the ICL demonstrations are also selected from the “downloading files” scenario, the MAE and RMSE of the traffic prediction are relatively low and the $R^2$-Scores are relatively high. If

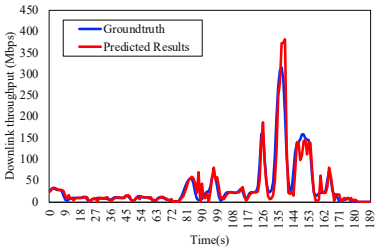
the ICL demonstrations are selected from other scenarios such as watching Amazon videos or watching Netflix videos, the prediction performance is worse but it is still much better than the zero prediction results. This validates that the ICL demonstrations from other scenarios can also provide useful information to the LLMs and are beneficial to the task. For example, the ICL demonstrations from different scenarios may indicate the relationship between the traffic bitrate and the contextual information. With this information and the base knowledge, the pre-trained LLMs are still able to produce relatively satisfactory predictive results and achieve $R^2$-Scores above 0.8. These numerical results show that the ICL demonstrations and test instances are not necessarily from the same application scenario. As a result, the proposed approach naturally adapts to dynamic network environments in a plug-and-play manner.

Table 8.4: MAE, RMSE and $R^2$-Score of LSTM-based mobile traffic prediction and LLM-based prediction while using training data from different scenarios

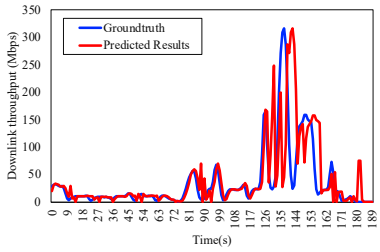
The test data are from the "Downloading files - Driving" scenario.

| Training data is from: | Downloading files - Driving | | Downloading files - Static | | Watching Amazon videos - Driving | | Watching Netflix videos - Driving | | Zero-shot prediction |
|---|---|---|---|---|---|---|---|---|---|
| Methods | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction | LLM-based prediction | LSTM-based prediction | |
| MAE | 9.4913 | **7.3187** | 9.8671 | **6.9010** | **11.027** | 20.380 | **11.246** | 23.974 | 13.816 |
| RMSE | 21.827 | **17.469** | 22.163 | **16.104** | **23.457** | 51.640 | **23.149** | 56.984 | 39.585 |
| $R^2$-Score | 0.84744 | **0.90228** | 0.84271 | **0.91695** | **0.82381** | 0.14606 | **0.82840** | 0.0398 | 0.49822 |

Similarly, for the LSTM-based method, the test data are from the "Downloading files - Driving" scenario, and the training data are from other application scenarios. The numerical results are shown in Table 8.4. According to the numerical results, if both the test data and the training data come from scenarios with the same activities, which is downloading files in our case, then the LSTM model still maintains good performance. However, if the test data and the training data come from scenarios with different activities, then the LSTM model will have a much worse performance compared with the LLM-based method. These simulation results indicate that LSTMs perform well in capturing scenario-specific data patterns, and they can achieve high accuracy when trained on abundant data. However, their generalization ability across scenarios is limited, espe-




(a) Traffic prediction results in the case of driving and downloading files.




(b) Traffic prediction results in the case of static and downloading files.

Figure 8.6: Traffic prediction with selected ICL demonstrations using PHI-3-mini.

cially when scenario-specific data is scarce. These numerical results further corroborate the limitations of the LSTM-based method.

Beyond LSTM-based approaches, transformer-based approaches are also frequently used for traffic prediction. These approaches build upon the foundations of LSTM-based approaches and incorporate more complex architectures and attention mechanisms to further enhance the predictive accuracy. According to the simulation results of LSTM-based approaches, they may also achieve higher prediction accuracy than our method. However, transformer models have similar limitations to LSTM-based methods. They typically require retraining for each new deployment scenario and depend heavily on the availability of scenario-specific labeled data. In contrast, the proposed method offers better generalization across different scenarios without the need for scenario-specific data and retraining, which is a critical advantage in real-world applications.

A potential direction is to integrate LLMs with LSTM models or transformer-based models during online inference. Given the strong generalization capabilities of LLMs, they can be utilized in the initial iterations when the prediction task is changed to a new scenario with limited scenario-specific data. As more scenario-specific data becomes available, the LSTM models or transformer-based models can be trained and employed to enhance the prediction accuracy.

### 8.3.5 Numerical Results with PHI-3-mini model

We also provide some traffic prediction results with the PHI-3-mini model in Fig. 8.6 and Table 8.5. As can be observed from the results, the proposed ICL demonstration selection method is also applicable to the PHI-3-mini model. Specifically, the zero-shot prediction results of the PHI-3-mini model are much worse than those of the PHI-3-medium model.

Table 8.5: MAE, RMSE and $R^2$-Score of mobile traffic prediction using PHI-3-mini.

| | Zero-shot prediction | Proposed one-demo prediction | Proposed two-demo prediction | Full ICL |
|---|---|---|---|---|
| MAE | 23.990 | 11.824 | **9.4348** | 10.112 |
| RMSE | 58.730 | 25.833 | **21.088** | 22.342 |
| $R^2$-Score | -0.10452 | 0.78630 | **0.85759** | 0.84016 |

In the zero-shot prediction case, the PHI-3-mini model yields a negative $R^2$-Score, which indicates that the prediction results are worse than simply using the mean of the target values as a predictor. However, by using ICL with selected demonstrations, PHI-3-mini requires only about one-fourth of the model parameters to achieve predictive performance comparable to that of the PHI-3-medium model.

This study mainly focuses on PHI-3-mini and PHI-3-medium. However, the proposed method is inherently model-agnostic since the architecture of the LLM is treated as a black box. As a result, it is also applicable to other LLMs with different architectures. According to the comparison between PHI-3-mini and PHI-3-medium, we can expect a similar trend if larger LLMs, such as GPT-4 and Llama-3, are used. It is also mentioned in some academic papers that increasing model size does not always improve ICL performance and may even worsen it under certain conditions. So the larger models like GPT-4 and Llama-3 are expected to lead to better performance in traffic prediction tasks without ICL. However, their performance under our proposed ICL framework is not necessarily better than that of smaller models.

On the other hand, using larger LLMs in wireless networks brings several limitations. Large models typically require significantly more memory and computation resources. For mobile traffic prediction, which may need to run in latency-sensitive and resource-constrained environments, using such larger models is kind of impractical.

# Chapter 9

# Supervised Contrastive Learning for Uncertainty-aware Wireless Signal Analysis

AI and deep learning have shown strong capabilities in processing wireless signals and can be used for the AMC task. However, existing AI-based AMC methods usually suffer from unreliable predictions and lack robustness to OOD inputs, which limits their deployment in real-world scenarios.

In this chapter, we propose an uncertainty-aware AMC framework based on SupCon. To be more specific, a ResNet-based SupCon model is first trained to produce class-discriminative representations. These embeddings are then used to train a set of class-wise binary classifiers, which not only perform classification but also provide sample-wise uncertainty estimates. Based on these uncertainty scores, we introduce a rejection mechanism that allows the model to reject uncertain predictions and improve decision reliability.

## 9.1 System Design

Fig. 9.1 shows the system design of our proposed ResNet-based AMC and representation generation framework. As shown in the figure, the raw I/Q signals are used as the input for latent representation generation. In this section, the signal model is introduced and the problem is formulated.

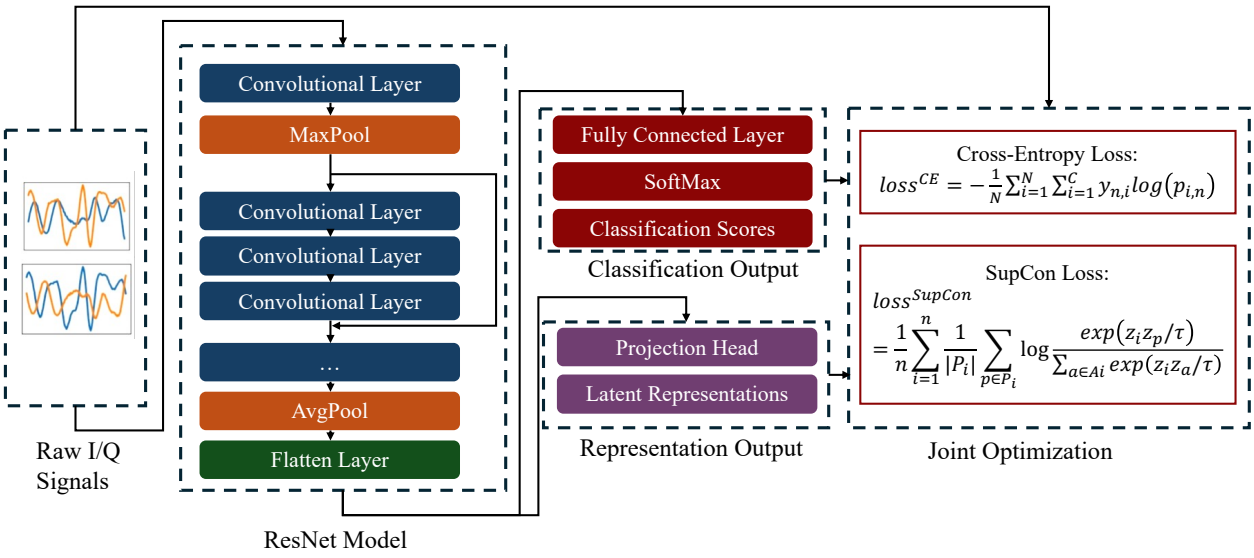



Figure 9.1: System design of ResNet-based automatic modulation classification and representation generation.

### 9.1.1 Signal Model

In this work, we consider a communication system consisting of a transmitter and a receiver. The transmitter selects one of $N$ predefined modulation schemes and encodes the raw bit-stream by modulating it onto a carrier signal. First, the bit-stream is mapped into a sequence of discrete complex symbols $s_k$ using a modulation mapping. Then, these symbols are converted into a continuous base-band analog signal $s_b(t)$ via pulse shaping. This process can be formulated as:

$$s_b(t) = \sum_k s_k g(t - kT), \tag{9.1}$$

where $s_k$ denotes the $k^{th}$ discrete complex symbol. $g(\cdot)$ is the pulse shaping filter, and $T$ is the symbol duration. Alternatively, the base-band signal can be represented in a generalized form as:

$$s_b(t) = a(t)e^{j(2\pi f(t)t + \phi(t) + \phi_0)}, \tag{9.2}$$

where $a(t)$, $f(t)$ and $\phi(t)$ respectively denote the instantaneous amplitude, frequency, and phase of $s_b(t)$ signal and $\phi_0$ is the initial phase offset [205].

Before transmission, the base-band signal is up-converted by mixing it with a carrier signal at frequency $f_c$. The transmitted signal $s(t)$ can be finally formulated as:

$$s(t) = Re[s_b(t)e^{j2\pi f_c t}], \tag{9.3}$$

where $Re[.]$ denotes taking the real part of the signal. Only the real part of the signal will be transmitted.

The goal of the receiver is to capture the noisy signal and identify the modulation type used by the transmitter. The received signal can be modeled as:

$$r(t) = s(t) * h(t) + n(t), \tag{9.4}$$

where $r(t)$ denotes the signal received by the receiver at time $t$ and $y(t)$ denotes the received signal. $h(t)$ denotes the channel impulse response function, which models the attenuation of the signal through propagation. $*$ denotes convolution. $n(t)$ denotes the additive complex Gaussian noise, which can be formulated as:

$$n(t) \sim \mathcal{CN}(0, \sigma^2). \tag{9.5}$$

At the receiver side, the signal is first downconverted to baseband and then digitized using an analog-to-digital converter. The received signal is sampled at a frequency $f_s = \frac{1}{T_s}$ over an obersvation window of length $L$, resulting in a discrete sequence:

$$\boldsymbol{r} = [r[0], r[1], ..., r[L-1]]^T \in \mathbb{C}^L. \tag{9.6}$$

Each sample $r[l]$ is a complex number representing the in-phase component $I$ and quadrature-phase component $Q$. The in-phase component is associated with the cosine carrier $cos(2\pi f_c t)$, and the quadrature-phase component is associated with the sine carrier $sin(2\pi f_c t)$. It can be formulated as:

$$r[l] = r_I[l] + r_Q[l] \quad l \in \{0, 1, ..., L-1\} \tag{9.7}$$

where $r_I[l]$ denotes the in-phase component and $r_Q[l]$ denotes the quadrature-phase component.

Based on this representation, the input IQ signal to our ResNet-based AMC framework is formulated as a $2 \times L$ matrix, which can be given as:

$$x = \begin{bmatrix} \boldsymbol{r_I} \\ \boldsymbol{r_Q} \end{bmatrix} = \begin{bmatrix} r_I[0], r_I[1], ..., r_I[L-1] \\ r_Q[0], r_Q[1], ..., r_Q[L-1] \end{bmatrix}. \tag{9.8}$$

This representation effectively captures the temporal variation of the in-phase and quadrature components of the signal, which allows the ResNet model to learn patterns required for identifying the modulation scheme.

### 9.1.2 Problem Formulation

The primary goal of the AMC task is to learn a model that can correctly predict the modulation type of a given input signal. Formally, we aim to learn a model $f_\theta$, parameterized by $\theta$, that can minimize the expected classification loss over the unknown test data distribution $P_{te}$. The process can be formulated as:

$$f_\theta^* = \arg\min_{f_\theta} \mathbb{E}_{x,y\sim P_{te}}[\ell(f_\theta(x), y)], \tag{9.9}$$

where $\ell(\cdot)$ denotes the loss function. $x$ is the input IQ signal and $y$ is the corresponding modulation type label of the signal.

In this work, we focus on a realistic open-set scenario with OOD test data, which means that the test data may have a different distribution from the training data:

$$P_{tr}(x, y) \neq P_{te}(x, y), \tag{9.10}$$

where $P_{tr}$ denotes the probability distribution of the training data.

Specifically, we assume that only $N_{ID}$ modulation types (indicated as ID classes) are included in the training dataset. The test dataset, however, contains both the ID classes and $N_{OOD}$ OOD modulation types that are not observed during training. The training data set can be formulated as:

$$D^{tr} = (x_i^{tr}, y_i^{tr}) \mid y_i^{tr} \in \mathcal{Y}_{ID} = \{0, 1, ..., N_{ID} - 1\}. \tag{9.11}$$

The test data set can be formulated as:

$$D^{te} = (x_i^{te}, y_i^{te}) \mid y_i^{te} \in \{\mathcal{Y}_{ID} \cup \mathcal{Y}_{OOD}\}, \tag{9.12}$$

where the set of OOD class labels $\mathcal{Y}_{OOD}$ can be denoted as:

$$\mathcal{Y}_{OOD} = \{N_{ID}, N_{ID} + 1, ..., N_{ID} + N_{OOD} - 1\}. \tag{9.13}$$

This setting reflects practical scenarios in AMC. In real-world conditions, the available training data may not include all possible modulation types. Moreover, new modulation schemes may appear at test time due to evolving communication technologies. In such cases, a well-designed model should avoid confidently misclassifying unseen modulation types into known classes.

Therefore, in addition to accurately classifying the ID classes, the second goal of this work is to estimate the predictive uncertainty of the model. Specifically, the model is expected to assign low uncertainty scores to samples that it can correctly classify, and assign high uncertainty scores to ambiguous samples, including examples from OOD classes and noisy signals. This uncertainty estimation is crucial for deploying AMC models in evolving systems and can help to avoid the performance degradation of misclassifications on unknown signals.

In the following sections, our proposed architecture for AMC and uncertainty estimation methods are introduced in detail.

## 9.2 Supervised Contrastive Learning for Latent Representation Generation

To perform AMC, a ResNet-based backbone combined with a SupCon approach is first used to learn meaningful latent representations of the input IQ signals. The ResNet extracts hierarchical features and maps them into a latent space where samples from the same modulation type are closer together. This learned representation is then used for downstream tasks, including classification and uncertainty estimation. In the following, we describe the detailed process for latent representation generation.

### 9.2.1 Supervised Contrastive Learning

SupCon is a representation learning approach extended from the self-supervised batch contrastive approach. Different from self-supervised representation learning methods, it leverages label information to learn semantically meaningful embeddings [206]. The core idea of SupCon is to bring together embeddings of samples from the same class while pushing apart embeddings of samples from different classes in the latent space.

A typical SupCon model contains two components, the encoder part and the projection part. The encoder extracts high-level features from the input data, and the projection head maps these features into a latent embedding space. The output representation for a given input $x_i$ can be formulated as:

$$z_i = Proj(Enc(x_i)) = f^{rep}(x_i), \tag{9.14}$$

where $z_i$ denotes the latent representation produced by the SupCon model. $Enc$ denotes the encoder layers, $Proj$ denotes the projection layers, and $f^{rep}$ denotes the complete representation model.

The loss function of SupCon can be defined as:

$$L_{SupCon} = -\sum_{i=0}^{I} \frac{1}{|P(i)|} \sum_{p \in P(i)} log \frac{\exp(z_i \cdot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a/\tau)}, \tag{9.15}$$

where $P(i)$ denotes the set of positive examples. Positive examples mean the examples that have the same class label as the $i^{th}$ input example $x_i$. $A(i)$ is the set of all examples. $I$ denotes the number of training examples. $\cdot$ is the inner product. $\tau$ denotes the temperature of SupCon. This loss encourages the model to learn embeddings that cluster samples from the same class closely together, while maintaining separation between different classes. By doing so, the model automatically discovers the discriminative features that separate the classes.

However, compared to traditional supervised learning with cross-entropy (CE) loss, SupCon is more difficult to train. Each anchor needs to compute similarities with multiple positive and negative samples, and optimization occurs in a non-linear embedding space. This makes the loss surface more complex than that of CE loss. Additionally, SupCon can suffer from representation collapse, where embeddings of different inputs become indistinguishable [207, 208]. As a result, SupCon training is highly sensitive to initialization, temperature $\tau$, and batch size.

In the given AMC task, training is even more unstable due to variations in SNR. Since signals within the same modulation class may be different under different noise levels, the intra-class variance increases, making it harder for the model to learn consistent embeddings.

To address these challenges, we propose a joint optimization framework that combines SupCon loss with CE loss. As shown in Fig. 7.1, a shared ResNet encoder is followed by two heads, a classification head for predicting class probabilities, and a projection head for generating embeddings. The CE loss is defined as:

$$L_{CE} = -\frac{1}{I} \sum_{i=0}^{I} \sum_{n=1}^{N} y_{i,n} \log(p_{i,n}), \tag{9.16}$$

where $y_{i,n}$ is a one-hot indicator for whether the $i^{th}$ sample belongs to the $n^{th}$ class. $p_{i,n}$ is the output probability of the classification head that the $i^{th}$ sample belongs to the $n^{th}$

class. The CE loss encourages the classification head to assign high probability to the correct class and low probability to all others.

A weight coefficient $\beta \in [0, 1]$ is used to balance the SupCon loss and the CE loss. The total loss can be formulated as:

$$L_{total} = (1 - \beta)L_{SupCon} + \beta L_{CE}. \tag{9.17}$$

During training, $\beta$ is initialized with a large value and is gradually decreased towards zero. This allows the model to first benefit from the strong supervision of CE loss, encouraging discriminative class boundaries, and then refine the learned representations using SupCon to enforce better cluster structures in the embedding space. After training, the classification layer is discarded, and only the projection head is retained. It is combined with the ResNet backbone to form the complete trained SupCon model, which is then used to generate embeddings for input I/Q signals.

### 9.2.2 Data Pre-processing

To improve the generalization of the SupCon model and reduce the risk of overfitting, data augmentation strategies are applied to increase the diversity of the training samples. Based on the characteristics of complex-valued IQ signals, three types of augmentations are used: noise injection, amplitude scaling, and phase rotation.

- Noise injection: In each training iteration, independent and identically distributed Gaussian noise is added to the input samples. The process can be formulated as:

$$x_{noise} = x + \lambda_1 n_0^i, n_0 \sim N(0, 1) \tag{9.18}$$

  where $x$ is the original input, $n_0^i$ is the noise added to the $i^{th}$ sample, and $\lambda_1$ controls the noise level.

- Amplitude scaling: After noise injection, the amplitude of each signal is randomly scaled to simulate channel gain variation. It can be formulated as:

$$x_{amp} = \lambda_2^i x_{noise} \tag{9.19}$$

  where $\lambda_2^i$ is a scaling factor randomly sampled from a pre-defined distribution.

- Phase augmentation: Finally, a small random phase rotation is applied to each signal. The augmented signal can be formulated as:

$$x' = \begin{bmatrix} \boldsymbol{r'_I} \\ \boldsymbol{r'_Q} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{r_{I,amp}} \\ \boldsymbol{r_{Q,amp}} \end{bmatrix} \tag{9.20}$$

  where $x'$ denotes the augmented signal used for training. $\theta_i \sim \mathcal{N}(0, \sigma_\theta^2)$ is a randomly sampled rotation angle for the $i^{th}$ sample.

These augmentation methods introduce small, random, yet realistic variations to each input signal, with the specific augmentations changing randomly across different training iterations. This increases sample diversity without altering the underlying modulation type and enhances the generalization of the SupCon model.

### 9.2.3 Partial Hard Positive Mining

In AMC tasks, the training data often includes signals with a wide range of SNR levels. While high SNR samples are generally well-represented and easily distinguishable, low SNR samples tend to be noisy and ambiguous. As a result, enforcing all samples of the same class to cluster tightly in the embedding space can lead to degraded representations for the clean, high-SNR samples. To address this challenge, we learn from the idea of hard positive/negative mining techniques [209], and selectively optimize a subset of intra-class samples.

The original SupCon framework [206] inherently emphasizes hard positive and hard negative pairs during training. However, this can be problematic in the AMC task. In particular, many of the hard-to-align positive pairs at low SNR are not semantically informative, but rather reflect random noise. Forcing these noisy samples to align may compromise the learned structure for clean samples and increase the risk of overfitting.

To mitigate such concerns, we propose a partial hard negative mining strategy after the initial SupCon training. Specifically, we fine-tune the model with a small learning rate, using only those positive samples within each batch whose cosine similarity to the anchor lies within a defined intermediate range. These pairs are challenging but not dominated by noise, allowing the SupCon model to improve on meaningful distinctions without being misled by unreliable data.

## 9.3 Uncertainty Estimation and Rejection

After the representation model is trained, it is used to transform raw I/Q signal inputs into latent embeddings, which serve as inputs to the downstream classification and uncertainty estimation modules.

To enable both modulation classification and uncertainty estimation, we adopt a one-vs-all scheme by training a set of $N$ binary classifiers, where $N$ is the number of modulation classes. Each binary classifier relates to a single modulation type. The inputs of the classifiers are the embeddings generated by the trained SupCon model, and the outputs are the probability that a given sample belongs to the corresponding modulation type.

These classifiers are trained independently using the binary cross-entropy (BCE) loss, which is defined as follows:

$$\begin{aligned} L_{BCE}^{n} = -\frac{1}{I}\sum_{i=1}^{I} & y_i log(P(y_{i,n})) \\ & +(1-y_{i,n})log(1-P(y_{i,n})), \end{aligned} \tag{9.21}$$

where $P(y_{i,n})$ is the output of the $n^{th}$ classifier, and $L_{BCE}^{n}$ is the loss function of the $n^{th}$ classifier.

During the test stage, each input sample is first encoded into a latent embedding through the trained SupCon model. This embedding is then passed to all $N$ binary classifiers to produce a vector of class-wise probabilities:

$$\boldsymbol{p}_i = \{P(y_{i,n})|n = 0, 1, ..., N-1\}. \tag{9.22}$$

The predicted modulation type is determined by selecting the index corresponding to the maximum probability:

$$\hat{n_i} = \underset{n}{argmax}\ \boldsymbol{p}_i, \tag{9.23}$$

We evaluate the uncertainty with vector $\boldsymbol{p}_i$ from two aspects. The first aspect is the model's confidence in the known class space. This means how much the model believes the sample belongs to any of the known modulation types. The second aspect is the confidence in the predicted class, which means how clearly the model prefers the selected modulation type over the others. For the first aspect, it can be referred to as external uncertainty and can be formulated as $U_{eo} = 1 - P_{max}$. The second aspect can be referred to as internal uncertainty and can be formulated as $U_{io} = 1 - max(softmax(\boldsymbol{p}_i))$. We average the two aspects of uncertainty as the final uncertainty scores.

Table 9.1: Simulation Settings

| **Dataset and Input Data** | **Representation Learning** |
|---|---|
| Dataset Name: RadioML 2018 | Backbone Model: ResNet18 |
| Number of ID Modulation Types: 10 | Projection Head Structure: 2-layer neural network (256, 128) |
| Number of OOD Modulation Types: 10 | Embedding Dimension: 128 |
| Input Length: 1024 sampling points | SupCon Temperature: 0.7 |
| Training data size: 100,000 (ID data only) | Initial Weight Coefficient: 1.0 |
| Test data size: 5,000 ID data + 5,000 OOD data | Weight Decay Schedule: $\times 0.9$ per epoch |
| **Downstream Classifiers** | Optimizer: Adam |
| Model Structure: 2-layer neural network (64, 16) | Batch Size: 1000 (100 per class) |
| Learning Rate: 0.001 | Activation Function: ReLU |
| Optimizer: Adam | Learning Rate: 0.0005 |
| Batch Size: 512 | Partial Hard Positive Mining Range: (0.2, 0.8) |

Based on uncertainty scores, we design a rejection scheme to improve the reliability of the classification system. Specifically, we set an uncertainty threshold, and if the uncertainty score of the prediction is above the threshold, the prediction will be rejected, and the input will be marked as difficult cases. These rejected samples can be further handled by alternative strategies, such as human-in-the-loop analysis.

This uncertainty estimation and rejection mechanism ensures that the system only produces predictions when it is sufficiently confident, thereby enhancing the overall trustworthiness of the AI-empowered wireless communication system.

## 9.4 Numerical Results

### 9.4.1 Experimental Settings

The simulation settings of this work are summarized in Table 9.1, including details on the dataset and input data, the representation learning process, and the downstream classifier design. In this work, we use the RadioML 2018 dataset [210], which contains a total of 24 modulation types under varying SNR conditions ranging from –20 dB to 30 dB. Each sample consists of 1024 I/Q sampling points. For simulation, we select $N_{ID} = 10$ modulation types as ID classes and another $N_{OOD} = 10$ modulation types as OOD classes.

We randomly sample 100,000 training samples from the ID classes, with 10,000 samples per class covering the full SNR range. From the remaining data, we randomly select 5,000 ID samples and 5,000 OOD samples as the test set. We adopt a ResNet-18 backbone for representation learning. The projection head consists of two feedforward layers with 256 and 128 neurons, respectively. The final embedding dimension can be denoted as $z_i \in \mathbb{R}^{128}$.

The model is trained with a combination of SupCon loss and CE loss. The initial weighting coefficient between the two losses is set to 1.0 and decays by a factor of 0.9 per epoch. The batch size is set as 1000, and we employ a class-balanced sampling strategy. Each batch includes 100 samples per class, leading to 99 positive pairs and 900 negative pairs per anchor. The temperature parameter for SupCon is set to 0.07. For optimization, we use the Adam optimizer with an initial learning rate of 0.0005. The hyperparameters selection were determined through an empirical search and evaluation process. In determining values such as the batch size, we also considered computational constraints. For example, larger batch sizes may improve gradient stability but can exceed memory limits, while smaller batches may slow convergence.

For the downstream classification and uncertainty estimation task, we train a set of binary classifiers, each consisting of two feedforward layers with 64 and 16 neurons, respectively. The learning rate is set to 0.001. A standard supervised learning model is used as the baseline for comparison. Details are provided in the following subsection.

### 9.4.2 Evaluation of ResNet Backbone

In this subsection, we evaluate the performance of different backbone architectures to support the use of ResNet in our proposed framework. We compare the modulation classification accuracy of three models: a ResNet model, a CNN model, and a LSTM network

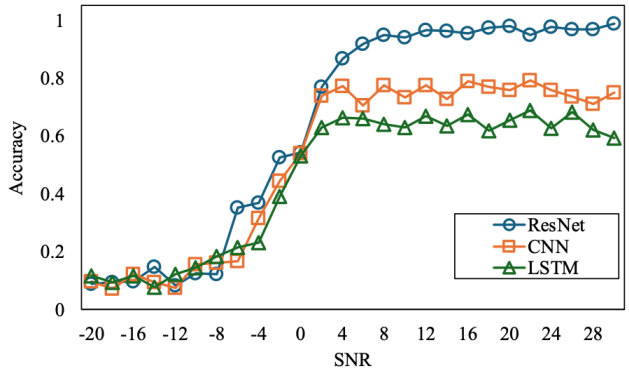



Figure 9.2: Modulation classification accuracy of ResNet, CNN, and LSTM models across different SNR levels.

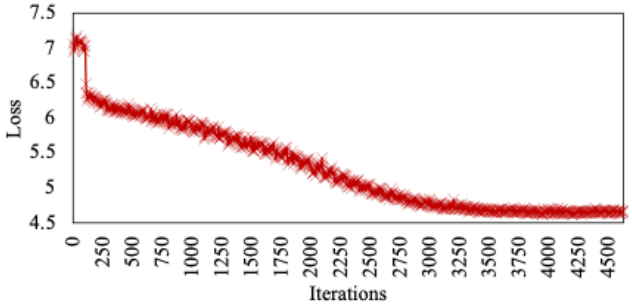
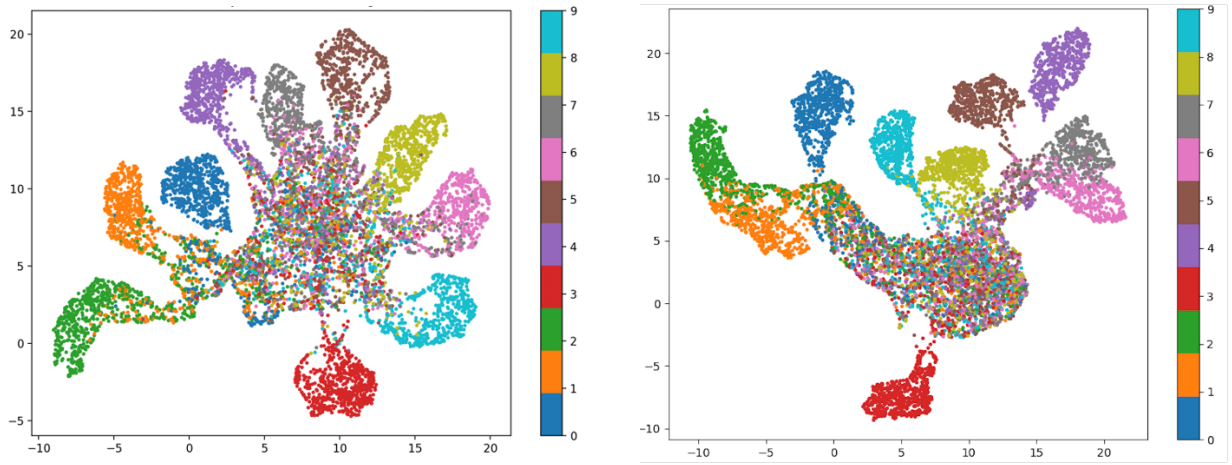



Figure 9.3: Convergence curve of the SupCon model.

model, across a range of SNR levels. As shown in Fig. 9.2, the ResNet model outperforms the other two baselines, particularly under high SNR conditions. For high-SNR signals with clear features, it achieves an accuracy close to 100%, demonstrating its strong feature extraction capability over other model structures. These results provide a foundation for using ResNet as the backbone for SupCon in this work.

### 9.4.3 Performance of Representation Learning

Fig. 9.3 shows the convergence curve of the SupCon loss during training. Although a joint loss that combines SupCon loss and CE loss is used for optimization, we only show the SupCon loss here, as our primary focus of model training at this stage is to generate high-quality representations. As shown in the figure, the training loss decreases steadily



(a) Proposed SupCon model-based representations.

(b) Supervised learning baseline-based representations.

Figure 9.4: Visualization of the learned embeddings from the SupCon model and the supervised learning-based baseline model.

and eventually converges. The smooth decline without noticeable oscillations indicates a stable and effective training process.

Fig. 9.4 presents a visualization of the learned feature embeddings from both the SupCon-based model and the supervised learning-based model. Each color represents a different modulation scheme. For the supervised baseline, the ResNet model is trained using only the CE loss, and the embeddings are extracted from the layer immediately before the final classification layer.

The figures are generated using the uniform manifold approximation and projection (UMAP) method, which is a visualization technique that can perform dimensionality reduction while preserving both local and global structures in high-dimensional data.

As shown in the figures, under the supervised learning setting, the embeddings from different classes exhibit partial overlap, indicating limited inter-class separation. In contrast, the embeddings learned through SupCon form more compact and well-separated clusters. This suggests that SupCon effectively enhances both inter-class separability and intra-class compactness in the latent space, which benefits downstream classification and uncertainty-aware decision-making tasks.

In Fig. 9.5, we provide a quantitative analysis of representation quality by comparing the silhouette scores of the embeddings for each modulation type. The silhouette score is a

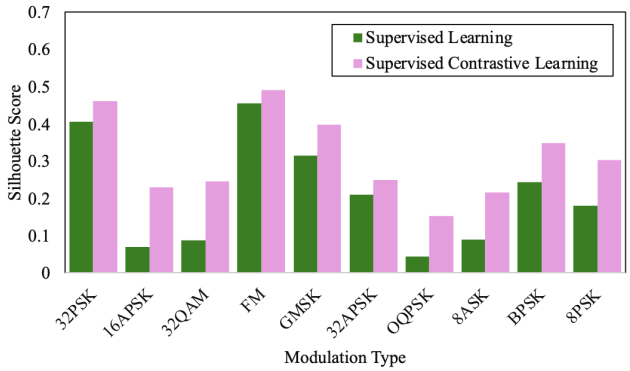



Figure 9.5: A comparison of the silhouette score of the representations.

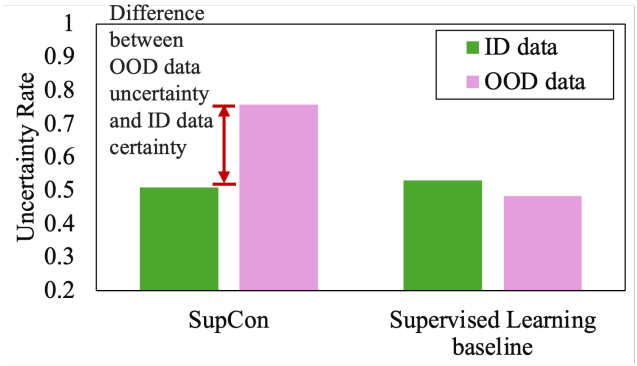


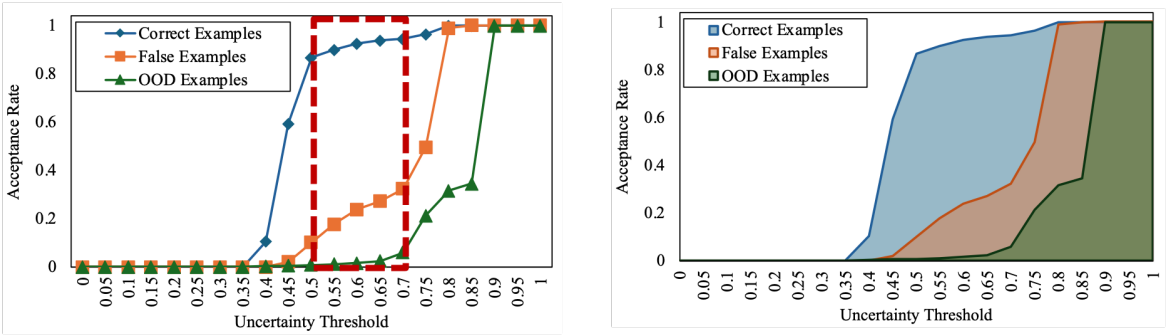

Figure 9.6: A comparison of uncertainty scores for ID and OOD predictions.

widely-used metric to evaluate the quality of class-wise separation in an embedding space. It quantifies how well each data point lies within its class compared to other classes. It can be defined as:

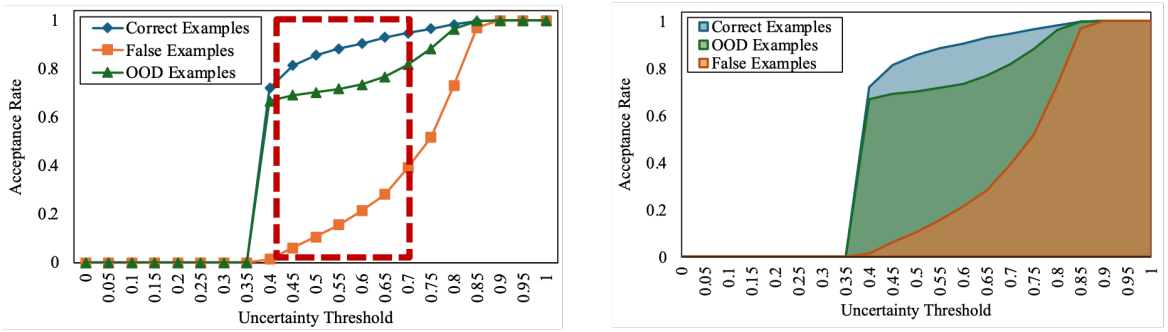
$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{9.24}$$

where $a(i)$ denotes the average distance from $i^{th}$ input sample to all other samples from the same class. $b(i)$ denotes the smallest average distance between that sample and all points in a different class.

As can be observed from the figure, the SupCon-based latent space consistently achieves higher silhouette scores across all classes compared to the baseline method. Moreover, the scores are more balanced among classes. These results confirm that the SupCon-based embeddings exhibit better intra-class compactness and inter-class separability.




(a) SupCon model-based representations.




(b) Supervised learning baseline-based representations.

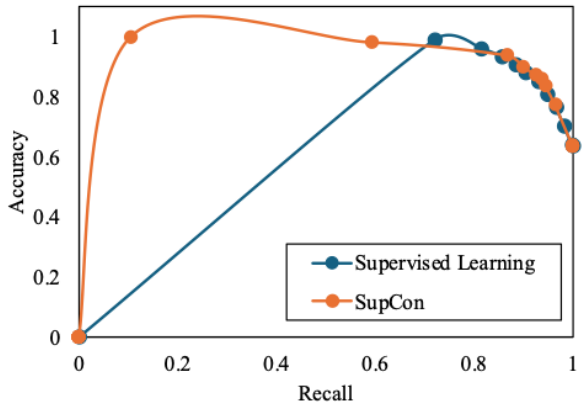
Figure 9.7: Acceptance rate of correct examples, false examples, and OOD examples under different uncertainty thresholds.

### 9.4.4 Uncertainty Estimation and Rejection

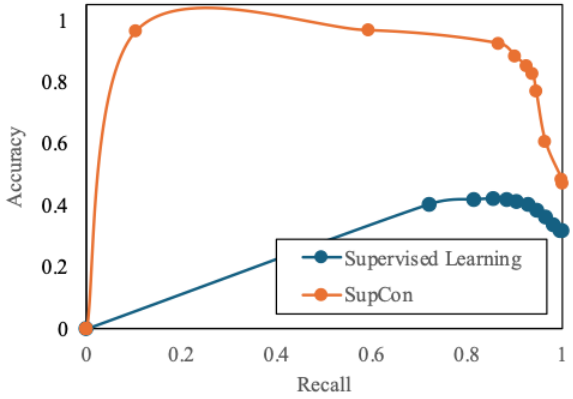
In this section, we present experimental results related to uncertainty estimation and the rejection mechanism. We first evaluate how well the proposed model distinguishes ID and OOD samples based on uncertainty scores.

Fig 9.6 compares the average uncertainty scores of ID and OOD samples predictions under two different training strategies, our proposed SupCon and class-wise classifier-based method, and the supervised learning-based baseline. For our proposed method, uncertainty is estimated based on the binary classifier outputs. For the supervised baseline, we directly use the softmax output from the classification head as the probability for each modulation scheme, and the uncertainty score is defined as one minus the maximum predicted probability.

As shown in the figure, our proposed SupCon-based method produces a significantly




(a) ID data only.



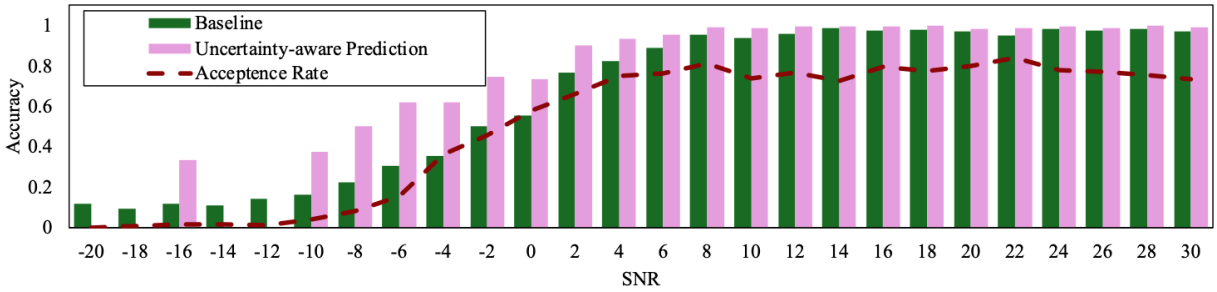

(b) 50% ID data and 50% OOD data.

Figure 9.8: Uncertainty and recall of correct predictions under different uncertainty rejection thresholds.

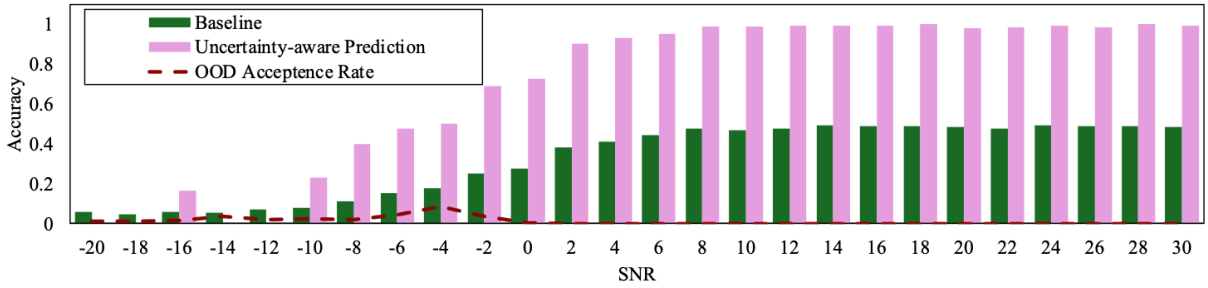
larger gap between the uncertainty scores of OOD and ID samples. In contrast, the supervised learning baseline fails to clearly separate ID and OOD samples in terms of uncertainty. These results demonstrate that our proposed method provides more effective uncertainty estimation, especially for identifying OOD data.

Next, we evaluate the effectiveness of the uncertainty-based rejection strategy in reducing misclassifications and improving decision confidence. In Fig. 9.7, we vary the uncertainty threshold and observe how it affects the acceptance rates of correct predictions, incorrect predictions, and OOD samples. To enhance visual clarity, we also include a version of the figure with shaded areas under each curve to highlight the differences. As shown in the figure, most correct predictions have uncertainty scores concentrated in the range of 0.4–0.5, while incorrect predictions cluster around 0.7–0.8, and OOD inputs are mostly located in the range of 0.8–0.9. This distribution indicates that setting the rejection threshold between 0.5 and 0.7 (highlighted with red boxes in the figure) allows for accepting a high percentage of correct predictions while effectively filtering out incorrect and OOD samples.

In contrast, for the supervised learning baseline, the acceptance rate of OOD samples is very close to that of correct predictions, making it difficult to distinguish between them. This is further illustrated in the shaded visualization, where the blue region—representing accepted correct predictions—is notably larger for the proposed method than for the baseline. These results further demonstrate the effectiveness of our proposed method in supporting reliable, uncertainty-aware modulation classification.




(a) ID data only.




(b) 50% ID data and 50% OOD data.

Figure 9.9: Modulation classification accuracy with uncertainty-aware rejection strategy.

By changing the uncertainty rejection thresholds, we can also get different accuracy and recall values. Fig. 9.8 shows the accuracy–recall curves for SupCon and supervised learning models under two different evaluation settings. In the first figure, we only use ID data for testing, and in the second figure, we use 50% ID data and 50% OOD data for testing. These curves illustrate the trade-off between prediction coverage and correctness when applying uncertainty-based rejection.

As can be observed in the figure, in the ID-only setting, both models achieve comparable performance at high recall levels. SupCon shows a slightly smoother trend, but the overall accuracy is similar to that of the supervised baseline when no OOD data is involved. In the OOD setting, the advantage of SupCon is more obvious. The accuracy remains high at even high recall levels, suggesting that most accepted predictions are correct. Meanwhile, the supervised model shows a great performance degradation due to its inability to distinguish between ID and OOD inputs. These results demonstrate that SupCon not only provides better uncertainty estimates but also supports more reliable rejection decisions especially with OOD data.

Finally, Fig. 9.9 shows how the proposed uncertainty-aware rejection strategy enhances modulation classification performance. We set the uncertainty threshold to 0.5 during the simulation. In the ID-only setting, the model tends to accept more high-SNR samples and reject low-SNR inputs that are inherently noisy and difficult to classify. This selective rejection scheme greatly improves accuracy, especially in the mid-SNR range.

In the OOD setting, the benefit is more obvious. The rejection mechanism effectively filters out most OOD inputs, resulting in a great increase in prediction accuracy. Specifically, accuracy improves from 63.7% to 93.6% without OOD inputs, and from 30.8% to 92.5% when 50% of the inputs are OOD, while maintaining over 85% recall on correctly classified samples. These results further confirm that our proposed method can lead to more robust and trustworthy modulation classification, especially in open-set scenarios with OOD data.

# Chapter 10

# Conclusion

AI techniques have created numerous opportunities for intelligent mobile networks and have accelerated the automation of network operations. However, these techniques also introduce new threats and security concerns. In this thesis, we investigate how to advance next-generation mobile communication systems through the integration of AI techniques. We begin by designing AI-based xAPPs to enhance the performance of key mobile network functions and how to deal with the collaboration and conflicts of distributed agents. We also examine how distributed AI frameworks, such as FL, expose the system to adversarial attacks, and we propose defense mechanisms to ensure the robustness of AI models. Finally, we extend this research by exploring how emerging paradigms such as GAI and LLMs can be adapted to the wireless domain. These models offer new capabilities in generalization and few-shot learning, which can enhance the adaptability and data limitations existing AI frameworks. Together, these directions form a unified effort toward building intelligent, secure, and adaptable next-generation AI-native wireless networks.

## 10.1 Mitigate conflicts between distributed AI-based applications in mobile networks

xAPPs in O-RAN provide higher flexibility for wireless network management. However, it may also lead to conflicts in network function implementations, especially when these functions are implemented by different vendors. To solve this problem, in Chapter 3, we propose two algorithms to mitigate conflicts between xAPPs in the O-RAN architecture, a team learning algorithm and a VFRL algorithm. The core idea of team learning is to

make xAPPs share action information they intend to take. Then the intended actions of other xAPPs will be used as a part of the state for DQN training and the selection of actions. The core idea of VFRL is to make distributed agents submit their local Q-tables to a coordination model for federating, and generate a global Q-table for action selection. These schemes are also applicable to other multi-vendor RANs of the future to enhance the system throughput, reduce delay and PDR.

## 10.2 Knowledge Transfer and Federated Learning Enabled UE-Centric Traffic Steering

UE-centric solutions with on-device intelligence have become a promising technique in wireless networks due to the privacy and scalability concerns. However, there are also challenges in implementing UE-centric ML algorithms. To solve these challenges, in Chapter 4, we continue to discuss the multi-agent control problems in mobile communication systems and propose a federated transfer RL-enabled UE-centric traffic steering framework with knowledge transfer and model compression. The core idea is to use federated learning to aggregate decentralized local models, generate a robust global model, and use knowledge transfer to initialize newcomers. The simulation results show that the proposed framework achieves better convergence, stability, higher system throughput and lower delay compared with independent UE-centric traffic steering and cell-centric traffic steering. Considering these benefits, our proposed framework is not limited to traffic steering. It can also be applied to other distributed wireless applications, such as edge caching and sleep control.

## 10.3 Attacks and defense over federated DRL-enabled cell sleep control scenario

FL is a promising technique for learning-based functions in wireless networks due to its distributed implementation capability. On the other hand, distributed learning may increase the risk of exposure to malicious attacks where attacks on a local model may spread to other models by parameter exchange. In Chapter 5, we study the security problem of FL for cell sleep control application in a heterogeneous network. We consider two intelligent attacks, a GAN-enhanced model poisoning attack, and a regularization-based model poisoning attack. We also propose two defense schemes, a similarity- enabled autoencoder-based defense scheme, and a KD-based defense scheme. According to the simulation results,

intel- ligent attacks are more effective than regular data poisoning attacks and can degrade system performance by causing lower energy efficiency and throughput. Moreover, our proposed defense schemes can effectively defend against these attacks. Especially, the KD-based defense scheme can recover the system performance to approximately 95% of the performance under a secure system regardless of the type of attack.

## 10.4 Dual attention model-weighted robust FL in mobile communication systems

Considering the threats and attacks we discussed in Chapter 5, we expect to design a versatile framework that can be applicable to both the secure system and the attacked system. In Chapter 6, we propose an efficient and robust attention-weighted FL algorithm and applied it to a radio signal classification scenario. We decide on the attention weight for each participant in FL through two attention models. The first attention model performs a self-attention evaluation between all the participants to find the connections between local models. The second attention model decides the temporal attention between the global model in the last iteration and the local models in the current iteration, and it helps to align the training level of participants. The effectiveness of the proposed algorithm is verified on a real-world dataset. It shows great versatility and outperforms existing algorithms, both in secure systems and in systems under attack.

## 10.5 GAI-in-the-loop in mobile communication systems

As new techniques such as GAI and LLM emerged, they can also be applied to wireless communication and bring several benefits. In Chapter 7, we propose that combining the strengths of both GFMs and ML models can yield a synergistic effect, making the whole greater than the sum of its parts. Based on this thought, we explore the effective combination of GFMs with ML models to address mobile network-related challenges. Moreover, we investigated the potential of utilizing pre-trained LLMs for tasks within the realm of wireless communications. Specifically, we proposed a fully automatic network intrusion detection framework in which the LLM carries out the entire process, from feature selection to detection result output. We also proposed three ways to perform in-context learning with task-specific labeled data to enhance the performance. This framework is an initial attempt to apply LLMs to the field of wireless communication, and there are many research interests to delve into in the future.

## 10.6 Pre-trained LLM-empowered mobile traffic prediction

Mobile traffic prediction provides support for wireless communication network optimization and decision-making. However, real-world wireless network traffic data are highly nonlinear and complex, which makes mobile traffic prediction a challenging task. In Chapter 8, we designed an LLM-enabled mobile traffic prediction framework and proposed a two-step ICL demonstration selection scheme. This framework can accurately and stably predict the traffic in various scenarios. Specifically, we proposed two rules—effectiveness and informativeness—to evaluate the ICL demonstrations, based on a thorough theoretical analysis. The superiority of the proposed framework is demonstrated on a real-world 5G dataset. We found that ICL performs better than zero-shot prediction and with the proposed demonstration selection method, two demonstrations are sufficient for accurate prediction.

## 10.7 Supervised Contrastive Learning for Uncertainty-aware Wireless Signal Analysis

Existing AI-based methods often produce unreliable predictions and lack robustness to OOD inputs, which limits their deployment in real-world scenarios. In Chapter 9, we proposed an uncertainty-aware framework for wireless signal analysis, with a focus on the AMC task. Specifically, we design a ResNet-based encoder and connect it with a projection head to enable SupCon training. This design allows the model to learn class-discriminative representations with strong inter-class separability and intra-class compactness. On top of the learned embeddings, we introduce a set of class-wise binary classifiers to estimate prediction uncertainty and identify OOD inputs. We also implement an uncertainty-based rejection mechanism that can avoid unreliable predictions. The effectiveness of the proposed method is validated on the RadioML 2018 dataset. According to the simulation results, the proposed method demonstrates improved representation quality, higher sensitivity to OOD inputs, and enhanced classification robustness compared to baseline methods. In future work, we aim to extend this framework by integrating OOD detection with OOD generalization and developing self-evolving wireless foundation models that are capable of dealing with continuously changing signal environments.

## 10.8 Challenges and Future Directions

AI has achieved great success in mobile communication systems and is expected to play an increasingly important role in shaping the evolution of next-generation networks. However, there are still some challenges and open issues for research efforts.

1) Security challenges: AI techniques can lead to numerous potential threats to network security and user privacy [211]. In our research, we have proposed defense methods and robust frameworks to mitigate the malicious attacks in FL. However, privacy-related attacks such as membership inference attacks should be further investigated, as they are not fully addressed by the current framework. In addition, other AI techniques, such as DRL [212] and TL [213], are also facing similar security threats. Moreover, the prevalent LLMs have brought new types of threats such as prompt injection attacks, excessive agency and ethical considerations. Therefore, addressing the emerging security threats introduced by AI is of paramount importance for future network deployments.

2) Computation challenges: While AI technology has shown promising performance, it often comes at the cost of high computation. Computational efficiency has always been an important consideration for AI model design. As the model size and complexity continue to increase, the amount of training computation and inference computation of the AI model also increases, thus preventing the application of AI technology in some scenarios of wireless communication. In our research, we have proposed the concept of "on-device intelligence" to mitigate the computation concerns. Other possible solutions include edge offloading or model compression.

3) Reliability challenges: AI techniques may come up with inaccurate and unfair results due to the biased or incomplete training data. This can become a critical concern, especially when it comes to some vital decisions and may lead to economic loss in mobile communication systems. Therefore, enhancing the reliability of AI techniques is an important topic. Human-in-the-loop is a common solution to the reliability concerns. In addition, the explainable AI technique can also enhance the reliability. It helps users understand the outputs of AI models and increases the confidence of the decisions.

Furthermore, there are some specific challenges regarding the applications of GAI and GFMs in mobile communication systems:

4) Limited domain knowledge of pre-trained GFMs

The first challenge is that most pre-trained GFMs have limited knowledge of the mobile communication domain, which makes it difficult for them to serve numerous diversified wireless tasks and scenarios directly. This is because the datasets used during the pre-training process rarely contain data specific to the mobile communication domain. It

is a challenging aspect to relate the general knowledge of GFMs to real-world mobile communication scenarios.

There are several potential solutions to this challenge. The first solution is to add additional information related to the specific tasks through the input using techniques such as ICL and RAG. ICL means integrating task-specific demonstrations into the prompt [9]. For instance, in a computational task offloading scenario, the reward values of specific state-action pairs can be used as in-context examples for GPT to develop optimized decision-making strategies [214]. RAG is a technique that can incorporate knowledge from external databases to mobile communication tasks, such as telecommunication documents and standards. How to design ICL and RAG for mobile communication tasks is a potential future research direction.

The second solution is to fine-tune the parameters of GFMs and make models more applicable to the mobile communication domain and given tasks. However, the fine-tuning process may cause other issues, including the large computational resource requirement, over-fitting risks, and model stability concerns. These concerns and corresponding solutions will be further discussed in the following subsections.

Model merging is another effective future solution to improve the capabilities of GFMs and can help with domain adaptation. Integrating general-purpose GFMs with wireless domain-specific GFMs can improve the model's adaptability and performance in mobile communication without forgetting the general knowledge.

5) High computation cost and inference delay

The large size and high computational cost of GFMs also pose a great challenge to the widespread deployment of GFMs in mobile communication networks. The computation of GFMs mainly comes from two aspects, fine-tuning and model inference. The former aspect implies that significant computational resources need to be devoted to GFMs, thus increasing the economic cost of communication networks. The latter aspect leads to high model inference delay, thus increasing the overall system delay and decreasing the QoS.

Efficient fine-tuning methods can be employed to reduce the computation of fine-tuning. The first potential solution is to only perform prompt tuning instead of model parameter fine-tuning. With this method, only a few prompt parameters are trained, resulting in low-cost adaptation to new tasks. Other than that, adapter-based fine-tuning methods can also reduce the computational cost by limiting the number of GFM parameters that are updated. An example is the low-rank adaptation method, which is implemented by freezing the pre-trained model weights and injecting trainable rank decomposition matrices into each layer [215].

On the other hand, to reduce the GFM inference computation, a potential solution is to adopt our proposed GFM-in-the-loop framework and only use the GFMs for high-level tasks. Light-weight ML models can be co-deployed in the network for delay-sensitive applications to reduce the inference time. Future research is needed to arrange different network management tasks between GFMs and ML models. Meanwhile, model compression can also be performed to reduce the size of GFMs and keep only the necessary parts of the parameters that are related to the given tasks. This enables the model to retain task-specific performance while significantly reducing the number of parameters.

6) Robustness concerns and hallucinations

Another significant challenge of applying GFMs to the next-generation network is the robustness and security concerns. Such concerns will have a direct impact on the stability of communication, which is an important indicator of QoS assessment for mobile communication.

Robustness concerns arise from the stochastic nature of sampling mechanisms employed by many GFMs, which can lead to output variability and hallucinations. Hallucination means in some tasks, GFM may generate outputs that are nonsensical, inaccurate, or disconnected from inputs. To mitigate such concerns, on the one hand, our proposed GFM-in-the-loop framework provides the feasibility of supervising model outputs by adding a validation mechanism. For instance, the decision to use the GFM output can be made by examining whether the GFM output is within the normal range.

Additionally, while there have been some attacks targeting GFMs and LLM, there remains a lack of sufficient research into effective defense mechanisms against these threats. A potential solution is to include some reinforcement mechanisms in the fine-tuning process to enhance resistance to attacks. In the future, more research is needed to improve the robustness and security level of GFM-enhanced next-generation networks.

In summary, this thesis focused on the design of intelligent, secure, and adaptable wireless communication systems by integrating advanced AI techniques. We addressed key challenges in applying AI to next-generation wireless networks, including enhancing network performance through AI-based xAPPs, ensuring robustness in collaborative distributed AI frameworks, and adapting emerging techniques such as GAIs and LLMs to wireless-specific tasks. Through a series of proposed frameworks and case studies, we demonstrated how different AI techniques can complement each other to tackle critical issues in decentralized coordination, adversarial vulnerability and data scarcity.

While many open issues remain, such as the hallucinations of large models, real-time inference under hardware constraints, and AI reliability and explainability challenges, this thesis lays a foundation for future research on building artificial general intelligence-native

wireless networks. This design is still in its early stages, but holds strong potential to reshape how future wireless systems work.

# Bibliography


[1] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, "6G wireless systems: Vision, requirements, challenges, insights, and opportunities," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, 2021.

[2] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (O-RAN)," in *ICC 2022 - IEEE International Conference on Communications*, pp. 4938–4943, 2022.

[3] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in O-RAN slicing," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 958–963, 2022.

[4] H. Zhang, H. Zhou, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, "On-device intelligence for 5G RAN: Knowledge transfer and federated learning enabled ue-centric traffic steering," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 2, pp. 689–705, 2024.

[5] H. Zhang, H. Zhou, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, "Intelligent attacks and defense methods in federated learning-enabled energy-efficient wireless networks," *IEEE Transactions on Wireless Communications*, 2025.

[6] H. Zhang, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, "Federated learning with dual attention for robust modulation classification under attacks," in *ICC 2024 - IEEE International Conference on Communications*, 2024.

[7] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Generative ai-in-the-loop: Integrating llms and gpts into the next generation networks," *arXiv preprint arXiv:2406.04276*, 2024.

[8] H. Zhang, H. Zhou, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, "Distributed attacks over federated reinforcement learning-enabled cell sleep control," in *2023 IEEE Globecom Workshops (GC Wkshps)*, pp. 485–490, 2023.

[9] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection," in *GLOBECOM 2024 IEEE Global Communications Conference*, IEEE, 2024.

[10] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Mobile traffic prediction using llms with efficient in-context demonstration selection," *IEEE Transactions on Communications*, 2025.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[13] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[14] H. H. Zhuo, W. Feng, Y. Lin, Q. Xu, and Q. Yang, "Federated deep reinforcement learning," *arXiv preprint arXiv:1901.08277*, 2019.

[15] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.

[16] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[17] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[18] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey.," *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[19] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural networks*, vol. 113, pp. 54–71, 2019.

[20] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[21] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[22] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[23] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[25] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[26] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[27] D. H. Hagos, R. Battle, and D. B. Rawat, "Recent advances in generative ai and large language models: Current status, challenges, and perspectives," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 12, pp. 5873–5893, 2024.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, *et al.*, "Generative adversarial networks," *Commun. ACM*, vol. 63, p. 139–144, Oct. 2020.

[29] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (Republic and Canton of Geneva, CHE), p. 689–698, International World Wide Web Conferences Steering Committee, 2018.

[30] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[32] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*, pp. 8821–8831, Pmlr, 2021.

[33] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of AI-generated content (aigc): A history of generative AI from GAN to chatGPT," *arXiv preprint arXiv:2303.04226*, 2023.

[34] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, *et al.*, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," *International Journal of Machine Learning and Cybernetics*, pp. 1–65, 2024.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[36] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[37] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International conference on machine learning*, pp. 1243–1252, PMLR, 2017.

[38] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024.

[39] K. VM, H. Warrier, Y. Gupta, *et al.*, "Fine tuning LLM for enterprise: Practical guidelines and recommendations," *arXiv preprint arXiv:2404.10779*, 2024.

[40] A. Archi, H. A. Saadi, and S. Mekaoui, "Applications of deep reinforcement learning in wireless networks-a recent review," in *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, vol. 1, pp. 1–8, IEEE, 2023.

[41] B. Ma, W. Guo, and J. Zhang, "A survey of online data-driven proactive 5G network optimisation using machine learning," *IEEE access*, vol. 8, pp. 35606–35637, 2020.

[42] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *ICC 2019-2019 IEEE International Conference On Communications (ICC)*, pp. 1–6, IEEE, 2019.

[43] H. Zhou, M. Elsayed, and M. Erol-Kantarci, "Ran resource slicing in 5G using multi-agent correlated q-learning," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1179–1184, IEEE, 2021.

[44] F. B. Mismar, B. L. Evans, and A. Alkhateeb, "Deep reinforcement learning for 5G networks: Joint beamforming, power control, and interference coordination," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1581–1592, 2019.

[45] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for noma with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2200–2210, 2019.

[46] H. Zhou, M. Erol-Kantarci, and H. V. Poor, "Knowledge transfer and reuse: A case study of ai-enabled resource management in ran slicing," *IEEE Wireless Communications*, vol. 30, no. 5, pp. 160–169, 2022.

[47] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load balancing for ultradense networks: A deep reinforcement learning-based approach," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9399–9412, 2019.

[48] S. Singh and J. G. Andrews, "Joint resource partitioning and offloading in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 2, pp. 888–901, 2013.

[49] X. Wang, J. Li, L. Wang, C. Yang, and Z. Han, "Intelligent user-centric network selection: A model-driven reinforcement learning framework," *IEEE Access*, vol. 7, pp. 21645–21661, 2019.

[50] M. Jiang, "Device-controlled traffic steering in mobile networks," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 7–12, 2015.

[51] M. Zhang, X. Yang, T. Xu, and M.-T. Zhou, "Congestion-aware user-centric cooperative base station selection in ultra-dense networks," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, IEEE, 2017.

[52] K. Adachi, M. Li, P. H. Tan, Y. Zhou, and S. Sun, "Q-learning based intelligent traffic steering in heterogeneous network," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5, IEEE, 2016.

[53] M. Masoudi, M. G. Khafagy, E. Soroush, D. Giacomelli, S. Morosi, and C. Cavdar, "Reinforcement learning for traffic-adaptive sleep mode management in 5g networks," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6, 2020.

[54] L. Mei, J. Gou, Y. Cai, H. Cao, and Y. Liu, "Realtime mobile bandwidth and handoff predictions in 4G/5G networks," *Computer Networks*, vol. 204, p. 108736, 2022.

[55] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, and P. Tengkvist, "A meta-learning scheme for adaptive short-term network traffic prediction," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2271–2283, 2020.

[56] F. Sun, P. Wang, J. Zhao, N. Xu, J. Zeng, J. Tao, K. Song, C. Deng, J. C. Lui, and X. Guan, "Mobile data traffic prediction by exploiting time-evolving user mobility patterns," *IEEE Transactions on mobile computing*, vol. 21, no. 12, pp. 4456–4470, 2021.

[57] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *2018 IEEE 29th annual international symposium on personal, indoor and mobile radio communications (PIMRC)*, pp. 1827–1832, IEEE, 2018.

[58] M. A. Habib, P. E. I. Rivera, Y. Ozcan, M. Elsayed, M. Bavand, R. Gaigalas, and M. Erol-Kantarci, "Transformer-based wireless traffic prediction and network optimization in O-RAN," in *2024 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2024.

[59] Q. Liu, J. Li, and Z. Lu, "ST-Tran: Spatial-temporal transformer for cellular traffic prediction," *IEEE Communications Letters*, vol. 25, no. 10, pp. 3325–3329, 2021.

[60] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5g wireless communications: A deep learning approach," *IEEE transactions on network science and engineering*, vol. 7, no. 1, pp. 227–236, 2018.

[61] T. Peng, R. Zhang, X. Cheng, and L. Yang, "Lstm-based channel prediction for secure massive mimo communications under imperfect csi," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[62] S. Zhang, S. Zhang, Y. Mao, L. K. Yeung, B. Clerckx, and T. Q. Quek, "Transformer-based channel prediction for rate-splitting multiple access-enabled vehicle-to-everything communication," *IEEE Transactions on Wireless Communications*, 2024.

[63] K. Hasan, T. Trappenberg, and I. Haque, "A generalized transformer-based radio link failure prediction framework in 5g rans," *arXiv preprint arXiv:2407.05197*, 2024.

[64] K. Boutiba, M. Bagaa, and A. Ksentini, "Radio link failure prediction in 5g networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[65] A. Almutairi, A. Keshavarz-Haddad, and E. Aryafar, "A deep learning framework for blockage mitigation and duration prediction in mmwave wireless networks," *Ad Hoc Networks*, vol. 162, p. 103562, 2024.

[66] M. Ghassemi, H. Zhang, A. Afana, A. B. Sediq, and M. Erol-Kantarci, "Generative ai-enabled blockage prediction for robust dual-band mmwave communication," *arXiv preprint arXiv:2501.11763*, 2025.

[67] K. Dmytro, T. Sergii, and P. Andiy, "Arima forecast models for scheduling usage of resources in it-infrastructure," in *2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, vol. 1, pp. 356–360, IEEE, 2017.

[68] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "Ran resource usage prediction for a 5g slice broker," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 231–240, 2019.

[69] S. Niknam, A. Roy, H. S. Dhillon, S. Singh, R. Banerji, J. H. Reed, N. Saxena, and S. Yoon, "Intelligent o-ran for beyond 5g and 6g wireless networks," in *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 215–220, IEEE, 2022.

[70] C. Adamczyk and A. Kliks, "Conflict mitigation framework and conflict detection in o-ran near-rt ric," *IEEE Communications Magazine*, vol. 61, no. 12, pp. 199–205, 2023.

[71] H. Xu, X. Liu, Q. Zeng, Q. Li, S. Ge, G. Zhou, and R. Forbes, "Decentran: Decentralized radio access network for 5.5 G and beyond," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 556–561, IEEE, 2023.

[72] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2019.

[73] M. Sana, A. De Domenico, W. Yu, Y. Lostanlen, and E. C. Strinati, "Multi-agent reinforcement learning for adaptive user association in dynamic mmwave networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6520–6534, 2020.

[74] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent reinforcement learning based cognitive anti-jamming," in *2017 IEEE wireless communications and networking conference (WCNC)*, pp. 1–6, IEEE, 2017.

[75] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multi-access edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2021.

[76] Y.-J. Liu, G. Feng, Y. Sun, S. Qin, and Y.-C. Liang, "Device association for ran slicing based on hybrid federated deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15731–15745, 2020.

[77] Y. Cao, S.-Y. Lien, Y.-C. Liang, and K.-C. Chen, "Federated deep reinforcement learning for user access control in open radio access networks," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.

[78] J. Hong, H. Wang, Z. Wang, and J. Zhou, "Federated robustness propagation: sharing adversarial robustness in heterogeneous federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7893–7901, 6 2023.

[79] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open

problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, pp. 1–210, 6 2021.

[80] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 5 2022.

[81] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19586–19597, 12 2020.

[82] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 12 2019.

[83] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *arXiv preprint arXiv:2002.05516*, 2 2020.

[84] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 12 2017.

[85] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, pp. 83–93, 7 2020.

[86] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International conference on machine learning*, pp. 12878–12889, PMLR, 7 2021.

[87] Y. Hu, Y. Hua, W. Liu, and J. Zhu, "Reward shaping based federated reinforcement learning," *IEEE Access*, vol. 9, pp. 67259–67267, 4 2021.

[88] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[89] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[90] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18, IEEE, 2017.

[91] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *31st USENIX security symposium (USENIX Security 22)*, pp. 1397–1414, 2022.

[92] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proceedings of Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Part I 25*, pp. 480–501, Springer, 9 2020.

[93] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, pp. 11365–11375, 11 2021.

[94] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *Proceedings of 29th USENIX security symposium (USENIX Security 20)*, pp. 1605–1622, 8 2020.

[95] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3396–3404, 2022.

[96] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*, pp. 634–643, PMLR, 2019.

[97] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, pp. 551–559, IEEE, 8 2021.

[98] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*, pp. 455–475, Springer, 2021.

[99] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2545–2555, 2022.

[100] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, vol. 22, pp. 191–205, 3 2021.

[101] C.-L. Chen, S. Babakniya, M. Paolieri, and L. Golubchik, "Defending against poisoning backdoor attacks on federated meta-learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 5, pp. 1–25, 2022.

[102] C. Chaccour, A. Karapantelakis, T. Murphy, and M. Dohler, "Telecom's artificial general intelligence (AGI) vision: Beyond the GenAI frontier," *IEEE Network*, vol. 38, no. 5, pp. 21–28, 2024.

[103] W. Saad, O. Hashash, C. K. Thomas, C. Chaccour, M. Debbah, N. Mandayam, and Z. Han, "Artificial general intelligence (agi)-native wireless systems: A journey beyond 6g," *Proceedings of the IEEE*, 2025.

[104] Y. Ouyang, Y. Zhang, P. Wang, Y. Liu, W. Qiao, J. Zhu, Y. Liu, F. Zhang, S. Wang, and X. Wang, "6G network business support system," *arXiv preprint arXiv:2307.10004*, 2023.

[105] Z. Chen, Z. Zhang, and Z. Yang, "Big ai models for 6G wireless networks: Opportunities, challenges, and research directions," *IEEE Wireless Communications*, 2024.

[106] X. Cheng, H. Zhang, J. Zhang, S. Gao, S. Li, Z. Huang, L. Bai, Z. Yang, X. Zheng, and L. Yang, "Intelligent multi-modal sensing-communication integration: Synesthesia of machines," *IEEE Communications Surveys & Tutorials*, 2023.

[107] A. Alkhateeb, G. Charan, T. Osman, A. Hredzak, J. Morais, U. Demirhan, and N. Srinivas, "Deepsense 6G: A large-scale real-world multi-modal sensing and communication dataset," *IEEE Communications Magazine*, vol. 61, no. 9, pp. 122–128, 2023.

[108] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[109] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[110] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[111] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[112] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[113] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[114] Y. Zhu, L. J. Fan, and N. G. Team, "NVIDIA Isaac GR00T N1: An Open Foundation Model for Humanoid Robots," March 2025. Accessed: 2025-03-19.

[115] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[116] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, *et al.*, "BLOOM: A 176B-parameter open-access multilingual language model." working paper or preprint, Nov. 2023.

[117] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, *et al.*, "A generalist agent," *Transactions on Machine Learning Research*, 2022. Featured Certification, Outstanding Certification.

[118] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15180–15190, 2023.

[119] R. Islam and O. M. Moushi, "Gpt-4o: The cutting-edge advancement in multimodal llm," *Authorea Preprints*, 2024.

[120] R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, *et al.*, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, vol. 1, 2023.

[121] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, and R. Xu, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025.

[122] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[123] M. Moradi, K. Yan, D. Colwell, M. Samwald, and R. Asgari, "Exploring the landscape of large language models: Foundations, techniques, and challenges," *arXiv preprint arXiv:2404.11973*, 2024.

[124] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao, "Rethinking interpretability in the era of large language models," *arXiv preprint arXiv:2402.01761*, 2024.

[125] Y. Li, Z. Li, Z. Han, Q. Zhang, and X. Ma, "Automating cloud deployment for real-time online foundation model inference," *IEEE/ACM Transactions on Networking*, 2023.

[126] X. Chen, Z. Guo, X. Wang, H. H. Yang, C. Feng, J. Su, S. Zheng, and T. Q. Quek, "Foundation model based native ai framework in 6g with cloud-edge-end collaboration," *arXiv preprint arXiv:2310.17471*, 2023.

[127] Q. Zheng, J. Wang, and Y. Shen, "Overview and trend of large-scale model deployment mode," in *2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, IEEE, 2024.

[128] J. Xu, Z. Li, W. Chen, Q. Wang, X. Gao, Q. Cai, and Z. Ling, "On-device language models: A comprehensive review," *arXiv preprint arXiv:2409.00088*, 2024.

[129] H. Wu, X. Chen, and K. Huang, "Device-edge cooperative fine-tuning of foundation models as a 6g service," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 60–67, 2024.

[130] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu, *et al.*, "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *IEEE Communications Surveys & Tutorials*, 2024.

[131] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali, "Gemel: Model merging for memory-efficient, real-time video analytics at the edge," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 973–994, 2023.

[132] Y. Park, K. Budhathoki, L. Chen, J. M. Kübler, J. Huang, M. Kleindessner, J. Huan, V. Cevher, Y. Wang, and G. Karypis, "Inference optimization of foundation models on AI accelerators," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6605–6615, 2024.

[133] Y.-H. Chen, R. Sarokin, J. Lee, J. Tang, C.-L. Chang, A. Kulik, and M. Grundmann, "Speed is all you need: On-device acceleration of large diffusion models via GPU-aware optimizations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4651–4655, 2023.

[134] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, and K. Huang, "Pushing large language models to the 6g edge: Vision, challenges, and opportunities," *CoRR*, vol. abs/2309.16739, 2023.

[135] J. Chen, Z. Safar, and J. A. Sorensen, "Multimodal wireless networks: Communication and surveillance on the same infrastructure," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 468–484, 2007.

[136] M. Alrabeiah, A. Hredzak, and A. Alkhateeb, "Millimeter wave base stations with cameras: Vision-aided beam and blockage prediction," in *2020 IEEE 91st vehicular technology conference (VTC2020-Spring)*, pp. 1–5, IEEE, 2020.

[137] S. Jiang, A. Hindy, and A. Alkhateeb, "Camera aided reconfigurable intelligent surfaces: Computer vision based fast beam selection," in *ICC 2023-IEEE International Conference on Communications*, pp. 2921–2926, IEEE, 2023.

[138] D. Wang, G. Hosangadi, P. Monogioudis, and A. Rao, "Mobile device localization in 5G wireless networks," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 185–190, IEEE, 2019.

[139] S. Bartoletti, L. Chiaraviglio, S. Fortes, T. E. Kennouche, G. Solmaz, G. Bernini, D. Giustiniano, J. Widmer, R. Barco, G. Siracusano, *et al.*, "Location-based analytics in 5G and beyond," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 38–43, 2021.

[140] A. Conti, F. Morselli, Z. Liu, S. Bartoletti, S. Mazuelas, W. C. Lindsey, and M. Z. Win, "Location awareness in beyond 5G networks," *IEEE Communications Magazine*, vol. 59, no. 11, pp. 22–27, 2021.

[141] G. Charan, M. Alrabeiah, and A. Alkhateeb, "Vision-aided 6g wireless communications: Blockage prediction and proactive handoff," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10193–10208, 2021.

[142] Y. Ahn, J. Kim, S. Kim, S. Kim, and B. Shim, "Sensing and computer vision-aided mobility management for 6g millimeter and terahertz communication systems," *IEEE Transactions on Communications*, 2024.

[143] Y. Ahn, J. Kim, S. Kim, K. Shim, J. Kim, S. Kim, and B. Shim, "Toward intelligent millimeter and terahertz communication for 6g: Computer vision-aided beamforming," *IEEE Wireless Communications*, vol. 30, no. 5, pp. 179–186, 2022.

[144] R. Kudo, K. Takahashi, T. Inoue, and K. Mizuno, "Using vision-based object detection for link quality prediction in 5.6-ghz channel," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, p. 207, 2020.

[145] Y. Shen, J. Zhang, S. Song, and K. B. Letaief, "Graph neural networks for wireless communications: From theory to practice," *IEEE Transactions on Wireless Communications*, vol. 22, no. 5, pp. 3554–3569, 2022.

[146] Y. Lu, Y. Li, R. Zhang, W. Chen, B. Ai, and D. Niyato, "Graph neural networks for wireless networks: Graph representation, architecture and evaluation," *IEEE Wireless Communications*, 2024.

[147] J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu, *et al.*, "Towards graph foundation models: A survey and beyond," *arXiv preprint arXiv:2310.11829*, 2023.

[148] H. Mao, Z. Chen, W. Tang, J. Zhao, Y. Ma, T. Zhao, N. Shah, M. Galkin, and J. Tang, "Position: Graph foundation models are already here," in *Forty-first International Conference on Machine Learning*, 2024.

[149] Z. Zhang, H. Li, Z. Zhang, Y. Qin, X. Wang, and W. Zhu, "Graph meets llms: Towards large graph models," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

[150] H. A. Tahir, W. Alayed, W. U. Hassan, and A. Haider, "Proposed explainable interference control technique in 6g networks using large language models (llms)," *Electronics*, vol. 13, no. 22, p. 4375, 2024.

[151] X. Li and H. Fu, "Securebert and llama 2 empowered control area network intrusion detection and classification," *arXiv preprint arXiv:2311.12074*, 2023.

[152] S. Yang, J. Liu, R. Zhang, M. Pan, Z. Guo, X. Li, Z. Chen, P. Gao, Y. Guo, and S. Zhang, "Lidar-llm: Exploring the potential of large language models for 3d lidar understanding," *arXiv preprint arXiv:2312.14074*, 2023.

[153] A. Abdallah, A. Albaseer, A. Celik, M. Abdallah, and A. M. Eltawil, "Netorchllm: Mastering wireless network orchestration with large language models," *arXiv preprint arXiv:2412.10107*, 2024.

[154] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in o-ran for data-driven nextg cellular networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[155] M. Elsayed and M. Erol-Kantarci, "Reinforcement learning-based joint power and resource allocation for urllc in 5g," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[156] H. Zhou, M. Erol-Kantarci, and V. Poor, "Learning from peers: Deep transfer reinforcement learning for joint radio and cache resource allocation in 5g ran slicing," *IEEE Transactions on Cognitive Communications and Networking ( Early Access )*, pp. 1–17, Sep. 2022.

[157] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep q learning approach," in *ICC 2019-2019 IEEE International Conference On Communications (ICC)*, pp. 1–6, IEEE, 2019.

[158] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in han, nan and wan," *Computer Networks*, vol. 67, p. 74–88, Jul. 2014.

[159] A. B. Mosbah, D. Griffith, and R. Rouil, "Enhanced transmission algorithm for dynamic device-to-device direct discovery," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–8, IEEE, 2018.

[160] Y. Türk, E. Zeydan, and C. A. Akbulut, "An experimental analysis of differentiated quality of service support for lte users," in *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, pp. 408–412, IEEE, 2018.

[161] D. Panno and S. Riolo, "A new joint scheduling scheme for gbr and non-gbr services in 5g ran," in *2018 IEEE 23rd international workshop on computer aided modeling and design of communication links and networks (CAMAD)*, pp. 1–6, IEEE, 2018.

[162] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[163] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 154–169, 2020.

[164] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: clients clustering for better personalization," *World Wide Web*, vol. 26, no. 1, pp. 481–500, 2023.

[165] H.-G. Han, S. Zhang, and J.-F. Qiao, "An adaptive growing and pruning algorithm for designing recurrent neural network," *Neurocomputing*, vol. 242, pp. 51–62, 2017.

[166] K. Akyol, "Comparing of deep neural networks and extreme learning machines based on growing and pruning approach," *Expert Systems with Applications*, vol. 140, p. 112875, 2020.

[167] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[168] M. A. Habib, H. Zhou, P. E. Iturria-Rivera, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, "Traffic steering for 5G multi-rat deployments using deep reinforcement learning," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pp. 164–169, IEEE, 2023.

[169] A. Lacava, M. Polese, R. Sivaraj, R. Soundrarajan, B. S. Bhati, T. Singh, T. Zugno, F. Cuomo, and T. Melodia, "Programmable and customized intelligence for traffic steering in 5g networks using open ran architectures," *IEEE Transactions on Mobile Computing*, 2023.

[170] T. Zheng and B. Li, "Poisoning attacks on deep learning based wireless traffic prediction," in *Proceedings of IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pp. 660–669, IEEE, 6 2022.

[171] C. Liu, B. Natarajan, and H. Xia, "Small cell base station sleep strategies for energy efficiency," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 1652–1661, 3 2015.

[172] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 12 2017.

[173] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "Bagan: Data augmentation with balancing GAN," *arXiv preprint arXiv:1803.09655*, 6 2018.

[174] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proceedings of 2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282, IEEE, 8 2018.

[175] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 11 2015.

[176] H. Zhou, L. Kong, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, "Hierarchical reinforcement learning for ris-assisted energy-efficient ran," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 3326–3331, IEEE, 2022.

[177] P. Gupta, K. Yadav, B. B. Gupta, M. Alazab, and T. R. Gadekallu, "A novel data poisoning attack in federated learning based on inverted loss function," *Computers & Security*, vol. 130, p. 103270, 2023.

[178] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[179] Y. Wang, G. Gui, H. Gacanin, B. Adebisi, H. Sari, and F. Adachi, "Federated learning for automatic modulation classification under class imbalance and varying noise condition," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, pp. 86–96, 6 2021.

[180] C.-H. Chiang and H.-Y. Lee, "Can large language models be an alternative to human evaluations?," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15607–15631, 2023.

[181] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2998–3009, 2023.

[182] J. Brownlee, "How to choose a feature selection method for machine learning," *Machine Learning Mastery*, vol. 10, 2019.

[183] S. Wang, Y. Liu, Y. Xu, C. Zhu, and M. Zeng, "Want to reduce labeling cost? gpt-3 can help," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4195–4205, 2021.

[184] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, *et al.*, "Autogen: Enabling next-gen llm applications via multi-agent conversation," in *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2023.

[185] J. Li, R. Zhao, Y. Yang, Y. He, and L. Gui, "Overprompt: Enhancing ChatGPT through efficient in-context learning," in *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

[186] B. Farzaneh, N. Shahriar, A. H. Al Muktadir, and M. S. Towhid, "DTL-IDS: Deep transfer learning-based intrusion detection system in 5G networks," in *2023 19th International Conference on Network and Service Management (CNSM)*, pp. 1–5, IEEE, 2023.

[187] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[188] Open AI, "OpenAI developer platform," 2024.

[189] H. Liu, R. Ning, Z. Teng, J. Liu, Q. Zhou, and Y. Zhang, "Evaluating the logical reasoning ability of chatgpt and gpt-4," *arXiv preprint arXiv:2304.03439*, 2023.

[190] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, "Linkforecast: Cellular link bandwidth prediction in LTE networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2017.

[191] Y. Zhang, S. Feng, and C. Tan, "Active example selection for in-context learning," *arXiv preprint arXiv:2211.04486*, 2022.

[192] K. T. Chitty-Venkata, S. Raskar, B. Kale, F. Ferdaus, A. Tanikanti, K. Raffenetti, V. Taylor, M. Emani, and V. Vishwanath, "Llm-inference-bench: Inference benchmarking of large language models on ai accelerators," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1362–1379, IEEE, 2024.

[193] W. Fan, F. Xiao, Y. Pan, X. Chen, L. Han, and S. Yu, "Latency-aware joint task offloading and energy control for cooperative mobile edge computing," *IEEE Transactions on Services Computing*, 2025.

[194] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*, pp. 38087–38099, PMLR, 2023.

[195] X. Li, H. Tang, S. Chen, Z. Wang, R. Chen, and M. Abram, "Why does in-context learning fail sometimes? evaluating in-context learning on open and closed questions," *arXiv preprint arXiv:2407.02028*, 2024.

[196] Q. Guo, L. Wang, Y. Wang, W. Ye, and S. Zhang, "What makes a good order of examples in in-context learning," in *Findings of the Association for Computational Linguistics ACL 2024*, pp. 14892–14904, 2024.

[197] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?," *arXiv preprint arXiv:2202.12837*, 2022.

[198] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An explanation of in-context learning as implicit bayesian inference," *arXiv preprint arXiv:2111.02080*, 2021.

[199] C. Han, Z. Wang, H. Zhao, and H. Ji, "In-context learning of large language models explained as kernel regression," *arXiv preprint arXiv:2305.12766*, p. 3, 2023.

[200] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, and F. Wei, "Why can GPT learn in-context? language models implicitly perform gradient descent as meta-optimizers," in *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

[201] Z. Wu, Y. Wang, J. Ye, and L. Kong, "Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1423–1436, 2023.

[202] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5G dataset with channel and context metrics," in *Proceedings of the 11th ACM multimedia systems conference*, pp. 303–308, 2020.

[203] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[204] D. Kapgate, "Weighted moving average forecast model based prediction service broker algorithm for cloud computing," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 2, pp. 71–79, 2014.

[205] C. Xiao, S. Yang, Z. Feng, and L. Jiao, "Mclhn: Towards automatic modulation classification via masked contrastive learning with hard negatives," *IEEE Transactions on Wireless Communications*, 2024.

[206] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18661–18673, 2020.

[207] S. Seo, J. Kim, G. Kim, and B. Han, "Relaxed contrastive learning for federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12279–12288, 2024.

[208] F. Graf, C. Hofer, M. Niethammer, and R. Kwitt, "Dissecting supervised contrastive learning," in *International Conference on Machine Learning*, pp. 3821–3830, PMLR, 2021.

[209] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[210] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[211] S. Zhang and D. Zhu, "Towards artificial intelligence enabled 6g: State of the art, challenges, and opportunities," *Computer Networks*, vol. 183, p. 107556, 2020.

[212] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 90–109, 2021.

[213] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1526–1539, 2020.

[214] H. Zhou, C. Hu, D. Yuan, Y. Yuan, D. Wu, X. Liu, Z. Han, and C. Zhang, "Generative AI as a service in 6G edge-cloud: Generation task offloading by in-context learning," *arXiv preprint arXiv:2408.02549*, 2024.

[215] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, *et al.*, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, vol. 5, no. 3, pp. 220–235, 2023.