
TRAINING NATIVE INTELLIGENT COMMUNICATION SYSTEMS

KAIDI XU

Supervised by

Prof. Geoffrey Ye Li

July 2025

Department of Electrical and Electronic Engineering

Imperial College London

South Kensington

London SW7 2AZ

A thesis submitted in part fulfilment of the requirements for the Degree of
Doctor of Philosophy in Electrical and Electronic Engineering and the
Diploma of Imperial College London

Copyright declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution Non Commercial 4.0 International License (CC BY-NC).

Under this license, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the license terms clear to others by naming the license and linking to the license text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this license or permitted under UK Copyright Law.

Declaration of originality

I hereby declare that the intellectual content of this thesis is the product of my own work under the guidance of my supervisor Prof. Geoffrey Ye Li. All the assistance received and sources of information derived from others have been properly acknowledged and included as references in the bibliography. This thesis has not been and will not be submitted for any degree at any university or institution.

Abstract

Native intelligence, the paradigm from embedding to bonding artificial intelligence with wireless communication systems together, is crucial for rendering scalable, low-latency, and high-efficiency services in next-generation communication systems. However, the complex nature of communication systems, such as time-varying wireless channel statistics, user mobility, and massive connections, and the distributed data across the communication networks, renders the conventional multi-agent training methods inefficient. This thesis proposes a series of advanced distributed training frameworks from four perspectives.

First, a penalty-based Q-learning (PQL) scheme encourages agents in a multi-agent reinforcement-learning (MARL) system to learn high-value actions first. PQL stabilizes learning and outperforms learning-based baselines by more than 3% in HetNet power-control tasks reaching over 90% of the optimization solution performance. Second, a policy gradient (PG) based ADMM with second moment (PASM) casts model sharing as a federated-learning (FL) optimization problem, achieving faster convergence and around 20% higher test performance in vehicle-to-everything (V2X) resource allocation scenarios. Third, a rescale-invariant federated reinforcement learning (RIFRL) technique exploits the node-wise invariance of deep neural networks by normalizing weight columns before aggregation. This alignment enables 3 times faster convergence speed and around 4% testing performance gain over the FedAvg-based FRL algorithm in V2X scenarios. Finally, a neural collapse inspired deep supervised FL algorithm (NCDSFL) aligns hidden features by fixing auxiliary classifier weights at their NC solutions. The NCDSFL algorithm converges about 3 times faster than the FedAvg algorithm, thus reducing communication rounds in the OFDM signal detection scenario.

Acknowledgments

During my journey as a Ph.D. student, there are many wonderful people, who have influenced, inspired, and supported my work in their various capacities. In retrospect, I would like to express my sincerest gratitude to them.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Geoffrey Li for the continuous support of my Ph.D. study and research. Prof. Li has groomed me as a researcher and his guidance helped me in all the time of research. I could not have imagined having a better supervisor for my Ph.D. study.

I must also thank Dr. Shenglong Zhou for his help in various aspects. Besides, the group in our lab has been a source of friendships as well as good advice and collaboration. I wish to express my heartfelt gratitude to all members of our lab, including but are not limited to Dr. Nguyen Van Huynh, Ouya Wang, Bowen Zhang, Yanzhen Liu, Jiabao Gao, Guangyao Ding, Tianxin Wang, Shan Sha, Jiawei Cao, Zijian Cao and Fabio.

Last but by no means least, I would like to thank my girlfriend, Yifan Yu, and my parents, Guifei Ma and Huidong Xu. Thanks for the continuous love and support, which help me stay determined and go through those hard times. This dissertation is dedicated to them.

Publications

Journal papers:

- [1] **Kaidi Xu**, Nguyen Van Huynh, Geoffrey Ye Li, “Distributed-training-and-execution multi-agent reinforcement learning for power control in HetNet,” in *IEEE Transactions on Communications*, 2023, pp. 5893-5903.
- [2] **Kaidi Xu**, Shenglong Zhou and Geoffrey Ye Li, “Federated Reinforcement Learning for Resource Allocation in V2X Networks,” in *IEEE Journal of Selected Topics in Signal Processing*, 2024, pp. 1210 - 1221.
- [3] **Kaidi Xu**, Shenglong Zhou and Geoffrey Ye Li, “Rescale-Invariant Federated Reinforcement Learning for Resource Allocation in V2X Networks,” in *IEEE Communications Letters*, 2024, pp. 2799-2803.
- [4] **Kaidi Xu**, Shenglong Zhou and Geoffrey Ye Li, “Neural Collapse based Deep Supervised Federated Learning for Signal Detection in OFDM Systems,” submitted to *IEEE Transactions on Signal Processing*.

Conference papers:

- [1] **Kaidi Xu**, Shenglong Zhou and Geoffrey Ye Li, “Federated Reinforcement Learning for Resource Allocation in V2X Networks,” in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, Singapore, June 2024.
- [2] Shenglong Zhou, **Kaidi Xu** and Geoffrey Ye Li, “Communication-Efficient Decentralized Federated Learning via One-Bit Compressive Sensing,” in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, Singapore, June 2024.

Contents

1 Introduction	13
1.1 Background and Motivation	14
1.2 Literature Review	16
1.3 Thesis Overview and Contributions	20
1.4 Preliminaries	23
1.4.1 Policy Gradient	23
1.4.2 Federated Learning	24
2 Distributed-Training-and-Execution Multi-Agent Reinforcement Learning for Power Control in HetNet	26
2.1 Introduction	27
2.1.1 Specific Contributions of this Chapter	27
2.2 System Model and Problem Statement	29
2.3 Problem Description	31
2.4 Multi-agent Reinforcement Learning Systems	32
2.4.1 Independent Q Learning	32
2.4.2 Proposed penalty-based Q-learning for MARL Systems	33
2.4.3 Penalty-based Deep Q Learning	35
2.5 Proposed DTE Power Control Scheme	37
2.5.1 State and Observation Space	37
2.5.2 Action Space	38
2.5.3 Reward Design	38
2.5.4 Distributed Implementation Phase	39
2.5.5 Decentralized Training Phase	39
2.5.6 Network Structure	41
2.5.7 Training and Implementation Complexity	43

2.6	Simulation Results	44
2.7	Conclusions and future research directions	53
3	Federated Reinforcement Learning for Resource Allocation in V2X	
	Networks	54
3.1	Introduction	55
3.1.1	Specific Contributions of this Chapter	56
3.2	System Model and Problem Formulation	58
3.3	FRL via Inexact ADMM and Policy Gradient	62
3.3.1	Multi-Agent Policy Gradient	62
3.3.2	Inexact ADMM	64
3.3.3	Second Moment	66
3.3.4	Convergence Analysis	68
3.4	PASM for Resource Allocation in V2X Networks	70
3.5	Simulation Results	73
3.5.1	Effect of Second Moments	74
3.5.2	Scenario I	76
3.5.3	Scenario II	77
3.6	Conclusion	80
3.7	Proof of Lemma 3.3.1	81
3.8	Proof of Theorem 3.3.2	83
4	Rescale-Invariant Federated Reinforcement Learning for Resource	
	Allocation in V2X Networks	90
4.1	Introduction	91
4.1.1	Specific Contributions of this Chapter	91
4.2	System Model and Problem Formulation	92
4.3	Framework of RIFRL	94
4.3.1	Multi-agent POMDP and policy gradient for the V2X system	94
4.3.2	RIFRL algorithm	97
4.3.3	Rescale for ReLU activated linear layers	97
4.4	Simulation Results	101
4.5	Conclusion	105

5	Neural Collapse based Deep Supervised Federated Learning for Signal Detection in OFDM Systems	106
5.1	Introduction	107
5.1.1	Specific Contributions of this Chapter	107
5.1.2	Related Works	109
5.2	Multi-Binary Classification	112
5.2.1	Problematic Description	112
5.2.2	NC Solutions	113
5.2.3	Deep supervised DNNs with NC Weights	115
5.3	Neural Collapse inspired Deep Supervised Federated learning	118
5.3.1	Federated Learning	118
5.3.2	DS DNNs with NC Weights in FL	119
5.3.3	The NCDSFL Algorithm	120
5.3.4	Application to Signal Detection	121
5.4	Simulation Results	124
5.4.1	Validation of Theorem 5.2.1	125
5.4.2	Performance Comparison with Baselines	126
5.5	Conclusion	132
5.6	Proof of Theorem 5.2.1	133
5.6.1	Useful Lemmas	133
5.6.2	Proof of Theorem 5.2.1	136
6	Summary and Future Directions	138
6.1	Summary	139
6.2	Future Directions	141
	References	143

List of Figures

2.1	HetNet with two different tiers, which includes a macro-cell AP and multiple small-cell APs	29
2.2	Network structure of dueling DQN.	42
2.3	HetNet with three tiers.	44
2.4	Average sum-throughput performance of different algorithms in the training phase.	47
2.5	Relative performance of different algorithms in the training phase.	47
2.6	Relative performance of different algorithms in the training phase when $\rho = 0$.	49
2.7	Average sum-throughput performance of different algorithms in the training phase in the HetNet with 4 additional third-tier cells.	50
2.8	Relative performance of different algorithms in the training phase in the HetNet with 4 additional third-tier cells.	50
2.9	Relative performance of different algorithms in the training phase with different-structure DQNs deployed at different-tier APs.	52
3.1	The V2X network diagram	58
3.2	Descending objective function	66
3.3	PASM workflow in a V2X network with 3 vehicles.	70
3.4	Moving average reward during the training phase in the two scenarios.	75
3.5	V2V link packet delivery rate versus V2V link payload size	76
3.6	Moving average reward of two algorithms during the training phase.	78
4.1	Node-wise invariance.	98
4.2	Moving average reward of 3 algorithms.	102

4.3	V2V delivery probability of 5 algorithms.	103
4.4	V2V delivery probability of 5 algorithms.	104
5.1	System diagram	111
5.2	A deep supervised DNN	116
5.3	$\theta(\mathbf{W}_0, \mathbf{W}_1)$ v.s. epochs	125
5.4	$\vartheta(\mathbf{W})$ v.s. epochs	126
5.5	Testing BER v.s. epochs (SNR=10dB)	127
5.6	Testing BER v.s. epochs	128
5.7	Testing BER v.s. SNR	129
5.8	Testing BER v.s. epochs	130
5.9	Testing BER v.s. SNR	130
5.10	Testing BER v.s. epochs (SNR=10dB)	131

List of Tables

3.1	Simulation Parameters [1]	73
3.2	Testing performance (V2V link packet delivery rate) of Scenario I with different (N, K) pairs	77
3.3	Testing performance (weighted average rate of all links) of Scenario II with different (N, K) pairs	79

Abbreviations

AC	Actor-Critic
ADMM	Alternating Direction Method of Multipliers
APs	Access Points
AR	Augmented Reality
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BRIO	Backward Rescale-Invariant Operation
BS	Base Station
CSI	Channel State Information
CTDE	Centralized-Training-and-Decentralized-Execution
DDPG	Deep Deterministic Policy Gradient
DFT	Discrete Fourier Transform
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DS	Deep Supervision
DTE	Distributed-Training-and-Execution
ETF	Equiangular Tight Frame
FL	Federated Learning
FRL	Federated Reinforcement Learning
HetNets	Heterogeneous Networks
HQL	Hysteretic Q Learning
iid	independent and identically distributed
IQL	Independent Q Learning
KPI	Key Performance Indicator
LLMs	Large-Language Models
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MADRL	Multi-Agent Deep Reinforcement Learning
MAPPO	Multi-Agent Proximal Policy Optimization
MASC	Multiple-Actor-Shared-Critic

MIMO	Multiple-Input Multiple-Output
MMSE	Minimum Mean Squared Error
MPG	Markov Potential Game
NC	Neural Collapse
NCDSFL	Neural Collapse inspired Deep Supervised Federated Learning
NFSP	Neural Fictitious Self-Play
NGMA	Next Generation Multiple Access
NLoS	non-line-of-sight
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	orthogonal frequency-division multiple access
PASM	PG-based ADMM with a Second Moment
PDPs	Power Delay Profiles
POMDP	Partially Observable Markov Decision Process
PQL	penalty-based Q-learning
ReLU	Rectified Linear Unit
RIFRL	Rescale-Invariant Federated Reinforcement Learning
RIS	Reconfigurable Intelligent Surface
RL	Reinforcement Learning
SINR	Signal-to-Interference-plus-Noise Ratio
TD	Temporal Difference
UDs	User Devices
VR	Virtual Reality
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
WCDMA	Wide-band Code-Division Multiple Access
WINNER II	Wireless World Initiative New Radio (Phase II)
WMMSE	Weighted Minimum Mean-Squared Error

Chapter 1

Introduction

This chapter introduces the thesis along with its backgrounds, motivations and the overview, as well as some preliminaries of this thesis. Further details on the four subtopics will be described in later chapters as required.

1.1 Background and Motivation

Wireless communication systems have undergone rapid evolution over the past several decades, marked by a series of milestone technologies that define each new generation, such as Wide-band Code-Division Multiple Access (WCDMA), Orthogonal Frequency Division Multiplexing (OFDM), Multiple-Input Multiple-Output (MIMO) and millimeter-wave communication [2-5]. The upcoming 6G systems are expected to offer significantly enhanced features compared with 5G, including much higher speeds, lower latency, and increased capacity, with the potential to support applications like holographic communication and AI-driven services [6]. This raises the need for high-efficiency real-time resource allocation and signal processing algorithms.

Conventional communication systems have several limitations that prevent them from satisfying 6G requirements. Conventional communication systems are usually divided into different modules and then optimized separately as optimization problems based on specific requirements and aims. However, these mathematical models can be inaccurate in real communication systems due to non-ideal effects, such as nonlinearity of power amplifiers, time/frequency offsets and phase noise, which are difficult to formulate by the mathematical models. In addition, even with ideal models, the formulated optimization problems are usually non-convex and NP-hard [7]. To solve these optimization problems, global channel state information (CSI) and centralized iterative algorithms, which can find sub-optimal solutions with good performance, are required [8,9]. Nevertheless, they are impractical in real systems because of the lack of global CSI and high complexity.

Deep Learning (DL) is a promising technique to address the aforementioned challenges [10]. Compared with conventional model based methods, DL-based methods can directly learn implicit information from the training data and jointly optimize different communication modules wherever the gradient can be propagated through, which cannot be expressed by mathematical models and result in a performance gain. However, the performance of DL-based methods is highly dependent on the quality of the training dataset. The dataset

of wireless systems is usually different from that investigated in computer vision field. Wireless datasets are usually more complicated and distributed across all wireless network entities. For example, the achievable rate has a more complex expression than classification labels and this information with its related CSI are distributed in each pair of transmitter and receiver. Collecting all data over the network can be expensive and a violation of privacy. This results in inefficiency of the training algorithms from the DL community. Therefore, we use DL to develop low-complexity real-time distributed and efficient resource allocation and signal processing schemes for wireless communication systems and propose novel distributed training algorithms to train the distributed intelligence for wireless systems from different perspectives.

1.2 Literature Review

DL has been recognized as a powerful tool to enhance wireless communication systems from various perspectives, such as distributed resource allocation, networking and mobility management and signal detection [11–17]. Typically, DL can be classified into three major categories, supervised learning, unsupervised learning and reinforcement learning (RL), based on their learning paradigms [18].

Supervised learning learns the map from the input data to the ground-truth label data. In wireless systems, the datasets are usually generated by simulations. The deep neural network (DNN) in [10] learns the map from available information to desired sub-optimal solutions, which are generated by weighted minimum mean-squared error (WMMSE) algorithm [8], in a NP-hard power control problem. Similarly, the authors of [19] proposed to use DNN to learn the behavior of Hungarian algorithm [20] to reduce the complexity. In [13], the authors propose to use a DNN as the neural receiver for joint channel estimation and signal detection in an OFDM system. In [14], the authors proposed using a channel estimation DNN to replace the interpolation procedure to exploit implicit subcarrier correlations and obtained a performance gain over conventional methods. End-to-end designs are later proposed in [15, 16], where DNNs are employed as both transmitter and receiver. All of these neural receivers used transmit symbols as ground-truth labels to supervise the DNNs, which are not always available in real wireless systems.

Unsupervised learning usually involves discovering hidden patterns or structures from unlabeled data, while in the context of wireless system applications, unsupervised learning usually refers to directly optimizing the objective function that is related to system performance and thus does not need ground-truth labels. The authors of [21] proposed to use DNN to directly maximize the system sum-throughput in a data-driven unsupervised learning manner. By leveraging the model knowledge, an iterative algorithm induced DNN was proposed in [22] to solve the precoding problem in a MISO system, which unfolds the WMMSE algorithm and introduces some learnable parameters. Simi-

larly, the model-driven DNN in [23] efficiently allocates transmit power in cellular networks addressing the interference management challenge. These algorithms no longer need labeled data and can outperform the optimization based iterative algorithms with lower complexity under some simulation conditions. However, they still require a differentiable loss function, which is not always available in real systems, such as system sum-throughput and its corresponding gradient information.

In addition to supervised learning and unsupervised learning, deep RL (DRL), especially multi-agent DRL (MADRL), as an effective tool in machine learning, which does not require a differentiable reward function or labels for training, has gained popularity in recent years and has been extensively employed to solve various problems in wireless networks, such as network slicing and distributed resource allocation for vehicular networks [24]. There are two main categories of DRL used in wireless communication systems, value-based DRL and policy-optimization DRL.

Value-based DRL, such as those represented by Deep Q-Network (DQN), uses a DNN to estimate the expected accumulated reward for each action at a given state [25]. The authors in [26] have developed a MADRL based power control algorithm for homogeneous cellular networks, where a training center has been employed to train the common DNN parameters for all agents, by exploiting information exchange among neighboring APs and UDs. DQN is also extensively applied in resource allocation for vehicle-to-everything (V2X) networks. In [27], each vehicle is treated as an agent and makes decisions on sub-spectrum and transmitted power selection with limited transmission overhead. To address the scalability issue, the MADRL-based resource allocation scheme in [28] allocates transmit power and sub-channels in a distributed manner in a vehicle-to-vehicle (V2V) network, where fingerprint [29] is employed to stabilize the multi-agent system. The MADRL algorithm is further enhanced in [30] by graph neural networks.

Policy-optimization DRL directly learns a policy DNN to predict the action for a given state [31]. The deep deterministic policy gradient (DDPG) algorithm [32, 33] jointly allocates power and sub-spectrum for cellular networks

in [34]. In [35], the actor-critic (AC) architecture with a continuous action space is employed to jointly optimize transmit power, resource allocation, and UAV trajectory in a joint radar and communication multi-UAV cooperative detection scenario. The multiple-actor-shared-critic (MASC) method in [36] trains the power control policy NNs deployed at each AP in the HetNet. In [37], DDPG is employed to solve the power allocation in D2D-based V2V communications. In [38], a proximal policy optimization [39] DRL algorithm was proposed to optimize the phase-shift matrix of the reconfigurable intelligent surface (RIS) in RIS-assisted full duplex 6G-V2X communications.

The performance of DL-based methods is highly dependent on the quality of the training dataset. However, as mentioned in Section 1.1, in wireless communication networks, data is typically distributed in a decentralized fashion, as different entities generate and store different parts of the data. Considering the data privacy, this results in the local observability and data silos. To address the challenges, federated learning (FL) is a promising technique and has been applied to wireless communication scenarios [40, 41]. In a typical FL system, there is a central server aggregating/broadcasting model parameters from/to the clients without raw data exchange. The aggregated model integrates knowledge from the clients with partial observation on the wireless environment and thus has good generalizability on the global wireless environment. For instance, an online personalized FL algorithm is designed to collaboratively train the neural receiver among multiple cells [42]. However, the local datasets of clients are usually heterogeneous due to users' local observability, which can lead to slow convergence speed of the FL algorithm and thus increase communication overhead [43].

In addition, FL can be used to train agents in a MADRL system, allowing each agent to learn the knowledge beyond its observability without sharing raw data while preserving privacy, which is termed as federated reinforcement learning (FRL) [44–47]. For example, in [46], FRL trains agents for dynamic channel access and power control in a distributed manner while preserving user privacy and reducing communication overhead. A federated MARL scheme in [47] optimizes the cellular sum rate and the reliability and delay

requirements of V2V links, where the FL addresses the limitation of partial observability and accelerate the training process. A meta FRL framework was proposed in [48] to distributively maximize energy efficiency while ensuring quality of service for users in cellular networks. However, all the aforementioned works only exploit the most basic FL framework, the FedAvg algorithm [40], which suffers from its independent and identically distributed (iid) assumption on clients' datasets and low aggregation efficiency. This can lead to additional communication rounds and performance degradation when implementing in real wireless systems.

1.3 Thesis Overview and Contributions

Sixth generation (6G) communication systems are expected to be AI-empowered, or native intelligent networks. Therefore, to address the aforementioned challenges in training native intelligent wireless networks, in this thesis, we investigated advanced distributed training schemes for native intelligent wireless networks from different perspectives, i.e., better cooperation among agents, more advanced optimization framework for FL, better model alignment and better feature alignment.

In Chapter 2, we proposed a MADRL based power control scheme for the heterogeneous networks (HetNet), where each access point makes power control decisions independently based on local information. To promote cooperation among agents, we developed a penalty-based Q-learning (PQL) algorithm for MADRL systems. By introducing regularization terms in the loss function, each agent tends to choose an experienced action with high reward when revisiting a state, and thus the policy updating speed slows down. In this way, an agent's policy can be learned by other agents more easily, resulting in a more efficient collaboration process. We then implemented the proposed PQL in the considered HetNet and compared it with other distributed-training-and-execution (DTE) algorithms. The simulation results show that our proposed PQL can learn the desired power control policy from a dynamic environment where the user locations change episodically and outperform existing DTE MADRL algorithms.

In Chapter 3, we explored resource allocation in a V2X network under the framework of FRL. Specifically, on one hand, the usage of RL overcomes many challenges from the model-based optimization schemes. On the other hand, FL enables agents to deal with a number of practical issues, such as privacy, communication overhead, distributed learning, and exploration efficiency. The framework of FRL is then implemented by the inexact alternating direction method of multipliers (ADMM), where subproblems are solved approximately using policy gradients and accelerated by an adaptive step size calculated from their second moments. The developed algorithm, policy gradient based ADMM

with a second moment (PASM), is proven to be convergent under mild conditions and has a nice numerical performance compared with some baseline methods for solving the resource allocation problems in a V2X network.

In Chapter 4, we further exploited FRL in V2X networks from a different perspective. The data discrepancy among individual agents can significantly degrade the performance of FRL-based algorithms. To address this limitation, we exploited the node-wise invariance property of ReLU-activated neural networks, with the aim of reducing data discrepancy to improve learning performance. Based on this property, we introduced a backward rescale-invariant operation (BRIO) to develop a rescale-invariant FRL (RIFRL) algorithm, which rescales the DNNs at all clients to reduce the model variance while keeping their input-output maps unchanged. Simulation results demonstrate that the proposed algorithm notably enhances both convergence speed and convergent performance.

In Chapter 5, we investigated the signal detection scenario, where user devices have local datasets characterized by different power delay profiles (PDPs). FL is used to deal with this local observability issue while preserving data privacy. In addition, to mitigate the impact of data heterogeneity, we proposed a Neural Collapse inspired Deep Supervised Federated Learning (NCDSFL) algorithm to reduce the communication rounds with little additional costs by adding a regularization term to local loss function of each client. The NCDSFL exploits the NC phenomenon that appears in the terminal phase of training and describes the optimality conditions for the penultimate hidden features and the classifiers. The NC weights are used as fixed auxiliary classifiers and output classifiers to guide the hidden features of different clients to converge to the same subspaces characterized by the same set of NC weights. Moreover, we defined the NC form for the multi-binary classification problem and proved its optimality theoretically. Simulation results validate the NC phenomenon in the considered multi-binary classification problem and show that our proposed NCDSFL algorithm has more efficient aggregation step and thus converges significantly faster than FedAvg algorithm.

In Chapter 6, we summarized conclusions of this thesis and discussed future

research directions.

The key contributions of this thesis are a series of improved distributed training algorithms for intelligent networks from four different perspectives, which can be summarized as follows.

- We proposed the DTE PQL algorithm to solve the power control problem in HetNets, which promotes the cooperation among agents by introducing a regularization term to the local Q loss function.
- We proposed the PASM algorithm, which incorporates the second moment, policy gradient and inexact ADMM, to train the MADRL system in a FL manner for V2X systems.
- We proposed a RIFRL algorithm, which exploits the rescale invariant property of ReLU activated DNNs to align client models, to train the MADRL system for V2X systems.
- We proposed a NCDSFL algorithm, which exploits the NC phenomenon and DS technique to align the hidden features with little cost, to train the signal detection DNNs in a FL manner addressing the data discrepancy challenge.

The following chapters 2-5 are based on three published papers [49-51] and one manuscript [52] under review.

1.4 Preliminaries

1.4.1 Policy Gradient

PG is an important policy optimization algorithm. It directly optimizes the policy to maximize the expected cumulative reward. Specifically, an RL problem can be defined by a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \rho_0 \rangle$, where $\mathcal{S}$ denotes the state space of the environment, $\mathcal{A}$ is the action space, $P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ is the transition probability function, $R(a_t, \mathbf{s}_t)$ is the reward function, and $\rho_0(\mathbf{s}_1)$ denotes the initial state distribution.

At time t , the agent takes its action $a_t \in \mathcal{A}$ based on its policy $\pi_{\theta}(a_t|\mathbf{s}_t)$ in state $\mathbf{s}_t \in \mathcal{S}$, where θ is a collection of the trainable parameters of the parameterized policy. The environment then returns a reward based on the reward function, $r_t = R(a_t, \mathbf{s}_t)$, and transits to the next state $\mathbf{s}_{t+1} \in \mathcal{S}$ based on the transition probability function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. Assume that each episode consists of T time slots. Then we can write the trajectory of one episode as $\tau = \{\mathbf{s}_1, a_1, r_1, \mathbf{s}_2, a_2, r_2, \dots, \mathbf{s}_T, a_T, r_T\}$. The objective of the agent is to maximize the expected cumulative reward over all possible trajectories by optimizing the policy, i.e., to maximize,

$$J(\theta) = \mathbb{E}_{\tau}(G(\tau)), \quad (1.1)$$

where $G(\tau) = \sum_t (r_t(\tau))$ denotes the reward sum of trajectory τ . The probability of trajectory τ , $P(\tau)$, is determined by the state transition probability, the initial state and the policy. More specifically, we can write $P_{\theta}(\tau) = C \prod_t \pi_{\theta}(a_t|\mathbf{s}_t)$, where $C = \rho_0(\mathbf{s}_1) \prod_t P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ is a positive value determined by the state transition probability and the initial state. Then we have,

$$J(\theta) = \sum_{\tau} P_{\theta}(\tau) G(\tau). \quad (1.2)$$

By taking derivative, we have the following PG,

$$\begin{aligned}
\nabla_{\pi_{\theta}} J &= \nabla_{\pi_{\theta}} \sum_{\tau} P_{\theta}(\tau) G(\tau) \\
&= \sum_{\tau} G(\tau) \nabla_{\pi_{\theta}} P_{\theta}(\tau) \\
&= \sum_{\tau} P_{\theta}(\tau) G(\tau) \nabla_{\pi_{\theta}} \log P_{\theta}(\tau) \\
&\approx \mathbb{E}_{\tau} (G(\tau) \sum_t \nabla_{\pi_{\theta}} \log \pi_{\theta}(a_t | \mathbf{s}_t)).
\end{aligned} \tag{1.3}$$

With the estimate of PG (1.3), we are then able to directly optimize the policy via stochastic gradient ascent methods.

1.4.2 Federated Learning

FL is a distributed training paradigm, which allows multiple parties, or clients, to cooperatively train a machine learning model while ensuring data privacy and security [40]. Therefore, in a typical FL architecture, data and the learning process are distributed across the clients. Raw data are not exchanged. Instead, clients transmit only model updates to a coordinating server. The shared model aggregated from different clients learns from diverse datasets and thus usually generalizes better than the model trained on a single client dataset.

Formally, a canonical centralized FL system has one server and K clients aiming to learn a global model for all clients to minimize the total loss function without raw data exchange. The learning process can be formulated as an optimization problem,

$$\begin{aligned}
&\min_{\mathbf{x}, \mathbf{x}^k \in \mathcal{F}, k \in [K]} \sum_{k \in [K]} L(\mathbf{x}^k; \mathcal{D}^k), \\
&\text{s.t.} \quad \mathbf{x}^k = \mathbf{x}, \quad k \in [K],
\end{aligned} \tag{1.4}$$

where $\mathbf{x}^k$, $\mathbf{x}$, $\mathcal{F}$ and $\mathcal{D}^k$ denote the trainable local parameters at client k , the shared global parameters, the feasible region, and the private dataset of client k , respectively, $L(\cdot; \cdot)$ denotes the local loss function used to train the model

locally and depends on the local parameters and the local dataset.

FL solves problem (1.4) in a distributed manner. Specifically, the learning process repeats the following three key steps until the some terminating conditions are met:

- **Local update:** Each client updates its local model based on its local parameters and dataset by gradient descent, that is, $\mathbf{x}^{k,t+1} = \mathbf{x}^{k,t} - \alpha^t \nabla_{\mathbf{x}^{k,t}} L(\mathbf{x}^k; \mathcal{D}^k)$, or other optimizers such as Adam [53]. α^t is a descending learning rate.
- **Aggregation:** After several local updates, all clients send their trained local model parameters to the server. The server aggregates the models by averaging the received parameters, i.e., $\mathbf{x} = \sum_k \frac{|\mathcal{D}^k|}{\sum_i |\mathcal{D}^i|} \mathbf{x}^{k,t+1}$.
- **Broadcast:** The server then broadcasts the aggregated global model to all clients, and each client replaces its local model with the received global model.

This learning process is known as the FedAvg algorithm and has been theoretically proven to converge under the standard assumptions in non-convex settings [54].

Chapter 2

Distributed-Training-and-Execution Multi-Agent Reinforcement Learning for Power Control in HetNet

In this chapter, we focused on promoting cooperation among agents in a multi-agent deep reinforcement learning (MADRL) empowered Heterogeneous Network (HetNet), where MADRL was employed to allocate power for each Access Point (AP) in a distributed manner based on local observations about the wireless environment. Specifically, we introduced a regularization term into the local Q loss function, which penalizes the low-reward state-action pairs during the training phase so that the agents tend to learn the Q value of high-reward state-action pairs first. In this way, the agents' policy updating speed slows down, and the learning process is thus stabilized. The resulting penalty-based Q-learning (PQL) algorithm promotes cooperation among agents with the cost of additional regularization terms in the training. Simulation results validate the effectiveness of our proposed PQL algorithm.

2.1 Introduction

Modern wireless networks handle an enormous amount of data traffic to support a huge number of wireless devices with data-hungry applications like augmented reality (AR), virtual reality (VR), machine learning services, and ultra-high definition video streaming [55–57]. In conventional cellular networks, a macro Base Station (BS) needs to provide access to the core network for all User Devices (UDs) in the cell. To offload the heavy wireless traffic from those macro BSs, Heterogeneous Network (HetNet) is a promising technique [58]. In HetNets, different-tier Access Points (APs) coexist in one macro cell, where a macro AP is deployed in the center of the cell. Small-cell APs, including femto-cell and pico-cell APs, are deployed in the inner regions to offload data traffic or at the macro cell edges to enhance the coverage. These densely deployed low-power and low-cost small-cell APs can increase the spatial and the spectrum efficiency but suffer from severe cross-tier and co-tier interference. As a result, interference management has attracted great research interest.

To suppress interference, various approaches including both conventional model based methods and Deep Learning (DL) based methods have been developed as we have reviewed in Section 1.2. However, conventional methods require global Channel State Information (CSI) and high computational complexity, and Deep Reinforcement Learning (DRL) based resource allocation algorithms in the literature are Centralized-Training-and-Decentralized-Execution (CTDE) schemes and require a training center to train the Deep Neural Networks (DNNs). To address the above challenges, this chapter aims to develop a Distributed-Training-and-Execution (DTE) power control algorithm based on Multi-Agent Deep Reinforcement Learning (MADRL) for HetNets.

2.1.1 Specific Contributions of this Chapter

We study the power control problem in a single-antenna multi-tier HetNet, where a macro AP is deployed in the center of the cell and several small-cell APs are deployed within the macro cell to enhance the performance of cell-edge UD. A novel multi-agent penalty-based deep Q learning scheme is proposed

to carry out the DTE power control scheme for HetNet. Our main contributions can be summarized as follows:

- We propose a MADRL-based DTE power control scheme for HetNets, where each AP acts as an agent. Specifically, at each AP, a DQN is deployed to determine the transmit power independently and simultaneously with some exchanged information from neighboring APs. The proposed learning framework for HetNets can be trained and implemented in a distributed manner and thus has good scalability.
- We propose a novel PQL for multi-agent reinforcement learning (MARL) systems to promote cooperation among agents. In PQL, each agent tends to choose an experienced action with high reward when revisiting a state, and thus the policy updating speed slows down. Compared with the MARL learning algorithms in [59, 60], which slow down the policy updating speed by maintaining an averaged policy, PQL focuses on learning the high-reward experiences first and thus slows down the updating. Therefore, it is easier for other agents to learn the policy of this agent and thus results in a more efficient collaboration process. Compared with the most basic MARL algorithm, i.e., independent Q learning (IQL) [61], PQL only adds a few regularization terms in the loss function. Hence, there is almost no extra cost to implement PQL than the basic IQL in a MARL system. We develop both Q table based and DQN-based PQL for general MARL problems.
- We implement our proposed PQL algorithm in the considered HetNet and compare it with other DTE MARL algorithms. Simulation results show that our proposed PQL can learn the desired power control policy from a dynamic environment where the locations of UDs change episodically, resulting in a better performance compared with IQL and hysteretic Q learning (HQL) [62].

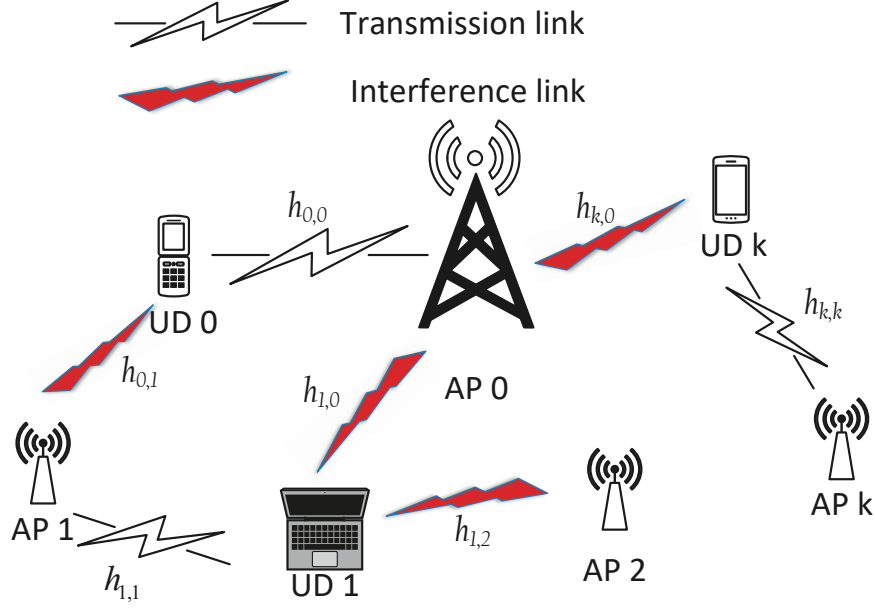


Figure 2.1: HetNet with two different tiers, which includes a macro-cell AP and multiple small-cell APs

2.2 System Model and Problem Statement

We consider a distributed power control problem in a single-antenna HetNet as illustrated in Fig. 2.1. Within a macro cell, several small cells are deployed to offload transmission traffic or to enhance coverage. There are K APs and K UDs denoted as $k \in \mathcal{K} = [0, \dots, K-1]$, where AP 0 is the macro AP, and AP k serves UD k . All APs within the macro cell share the same spectrum to serve the K UDs and may cause interference to others. We consider a synchronized system to simplify the interference model and all channels are modeled as block fading as in [63]. The channel gain, including transmission-link channel gain and interference channel gain, between AP k and UD m at time t is modeled as,

$$h_{m,k}^t = \sqrt{\beta_{m,k}} g_{m,k}^t, \forall m, k \in \mathcal{K}, \quad (2.1)$$

where $g_{m,k}^t \in \mathbb{C}$ is the corresponding small-scale Rayleigh fading and $\beta_{m,k}$ is the large-scale fading, which is determined by the path-loss and log-norm shadowing and remains unchanged over a long time period. We use Jakes fading

model [64] for the small-scale Rayleigh fading update, which can be expressed as follows,

$$g_{m,k}^t = \rho g_{m,k}^{t-1} + \sqrt{1 - \rho^2} i_{m,k}^t, \forall m, k, \in \mathcal{K} \quad (2.2)$$

where $i_{m,k}^t, \forall m, k \in \mathcal{K}$ are independent and identically distributed (i.i.d.) circularly symmetric complex Gaussian random variables with unit variance. The correlation coefficient is $\rho = J_0(2\pi f_D T)$, where $J_0(\cdot)$ denotes the 0-th order Bessel function of first kind and f_D and T are the maximum Doppler frequency shift and the time slot duration, respectively.

At time slot t , AP k transmits signal to user k with transmit power P_k^t . Assume the signal transmitted to user k is s_k^t , then the received signal at user k can be expressed as,

$$y_k^t = \underbrace{h_{k,k}^t \sqrt{P_k^t} s_k^t}_{\text{Desired signal}} + \underbrace{\sum_{k' \neq k} h_{k,k'}^t \sqrt{P_{k'}^t} s_{k'}^t}_{\text{Interference}} + n_k, \quad (2.3)$$

where n_k denotes the additive white Gaussian noise (AWGN) with a variance of σ_k^2 . In the right side of (2.3), the first term is the desired signal, while the second term is the interference term, which includes both cross-tier and co-tier interference. Therefore, the corresponding signal-to-interference-plus-noise ratio (SINR) at user k can be expressed as,

$$\gamma_k^t(\mathbf{P}^t) = \frac{|h_{k,k}^t|^2 P_k^t}{\sum_{k' \neq k} |h_{k,k'}^t|^2 P_{k'}^t + \sigma_k^2}, \quad (2.4)$$

where $\mathbf{P}^t = \{P_k^t \mid \forall k \in \mathcal{K}\}$ denotes a collection of transmit powers from all APs. The achievable rate of UD k can thus be expressed as $r_k^t(\mathbf{P}^t) = \log(1 + \gamma_k^t(\mathbf{P}^t))$.

2.3 Problem Description

We aim to maximize the system sum-throughput via optimizing transmit power of all APs in each time slot t . This power control problem can be formulated as follows,

$$\begin{aligned} \max_{\mathbf{P}} \quad & \sum_{k \in \mathcal{K}} r_k^t(\mathbf{P}^t), \\ \text{s.t.} \quad & P_k \leq P_k^{\max}, \forall k \in \mathcal{K}, \end{aligned} \tag{2.5}$$

where $P_k^{\max}$ is the maximum transmit power budget of AP k .

In HetNet, different-tier APs have different transmit power budget, i.e., different $P_k^{\max}$ for each AP k , and observe the environments with different statistics, which makes it more difficult for agents to learn the desired power control policy. In addition, the dynamic wireless environment with rapidly varying channels requires each AP to make decisions based on partial CSI only and to adjust transmit power before the data transmission stage starts so that the transmit power can maximize the system sum-throughput in every time slot. However, the interference CSI $h_{m,k}^t, \forall m, k \in \mathcal{K}, m \neq k$ is usually difficult to estimate in real systems, which is crucial when solving problem (2.5) by conventional algorithms. Therefore, traditional optimization based power control algorithms [8, 9] fail to be implemented in real-time systems due to the lack of global CSI and their high computational complexity. Although DL-based methods can be trained to predict transmit power based on only partial CSI in a low-complexity manner, the lack of interference CSI leads to the lack of ground truth and gradients of the objective function, and thus limiting the effectiveness of both supervised learning [10] and unsupervised learning [21] methods, if considering the training phase of the DL-based methods. Finally, the scalability problem should be taken into account due to the huge number of APs in real systems. It is thus difficult for centralized DL-based methods to obtain the data and to learn the optimal policy in large-scale wireless systems. Therefore, it is necessary to develop DTE MADRL-based power control algorithms, which will be present in detail in the following sections, to effectively solve the aforementioned challenges.

2.4 Multi-agent Reinforcement Learning Systems

A partially observable MARL system can be described as a partially observable Markov decision process (POMDP), which is defined by a tuple $\langle N, \mathbf{s}_t, \mathbf{a}_t^{(k)}, R_t^{(k)}, \mathbf{z}_t^{(k)}, P, O \rangle$.¹ Specifically, for a MARL system with N agents, in each time step t , each agent k observes the environment in state $\mathbf{s}_t$ and obtains a local observation of the environment $\mathbf{z}_t^{(k)} = O(\mathbf{s}_t, k)$, where $O(\cdot)$ is the observation function mapping the environment state $\mathbf{s}_t$ to the specific observation $\mathbf{z}_t^{(k)}$ of agent k . Then, based on the local observation $\mathbf{z}_t^{(k)}$, agent k chooses its action $\mathbf{a}_t^{(k)}$, which is a part of the system joint action $\mathbf{a}_t$. All agents implement the joint action $\mathbf{a}_t$ and interact with the environment. Each agent k receives a local reward $R_t^{(k)}$ from the environment and the environment state evolves to next state $\mathbf{s}_{t+1}$ with transition probability $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. Thereafter, each agent k receives a new observation of the environment, $\mathbf{z}_{t+1}^{(k)} = O(\mathbf{s}_{t+1}, k)$.

In general, the MARL-powered wireless systems can be formulated as cooperative and endless games.² In this chapter, we focus on cooperative games, where all agents share the same reward or they have a common potential function [65]. We promote cooperation by sacrificing the accuracy of Q value of some sub-optimal actions. To illustrate, we start with the basic learning scheme in MARL, IQL [61].

2.4.1 Independent Q Learning

IQL is a direct extension from single-agent reinforcement learning to MARL, where each agent treats other agents as a part of environment. Therefore, each agent needs to learn both the environment transition and the policy of other agents. Each agent k aims to learn an optimal policy $\pi^{(k)}$ that maximizes the

¹When involving POMDP, researchers usually use a concept, “belief”, to describe the agent’s belief about the invisible environment state. In this chapter, we assume each agent’s observation can well reflect its belief. We thus only use observation to stand for each agent’s local state.

²Although the interference among different cells makes APs perform like a competitive manner, in most cases, we have a network-wise key performance indicator (KPI) to capture the system performance. Therefore, all APs, in fact, cooperate together to maximize the system KPI.

cumulative discounted reward,

$$V_{t,\pi^{(k)}}^{(k)}(\mathbf{s}_t) = R_t^{(k)} + \sum_{j=1}^{\infty} \gamma^j R_{t+j}^{(k)}, \quad (2.6)$$

where γ is the discount factor.

In the traditional IQL, each agent maintains a Q table to record the Q values of all observation-action pairs of the agent. The Q values are learned from the experiences of the agents interacting with the environment. When receiving an experience tuple $\mathbf{e}_t^{(k)} = \langle \mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)}, R_t^{(k)}, \mathbf{z}_{t+1}^{(k)} \rangle$, agent k updates its Q table as follows,

$$Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)}) \leftarrow Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)}) + \alpha \delta_t^{(k)}, \quad (2.7)$$

where $\delta_t^{(k)} = R_t^{(k)} + \gamma \max_{\mathbf{a}^{(k)}} Q_k(\mathbf{z}_{t+1}^{(k)}, \mathbf{a}^{(k)}) - Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)})$ is the temporal difference (TD) error. α is the learning rate.

In IQL, each agent learns its Q table based on its own experience in a distributed manner with other agents' policies treated as a part of the environment. However, since all agents keep updating their policies simultaneously, the environment observed by one agent is no longer static, which violates the assumption of Q learning that the environment state transition probability matrix is fixed. The varying policies of other agents make it difficult for the agent to learn the environmental dynamics and leads to instability and performance degradation of IQL.

2.4.2 Proposed penalty-based Q-learning for MARL Systems

One major challenge of MARL in wireless communication systems is that the reward structure is very complex. Many different resource allocation solutions have similar performance, which results in the fact that at a specific observation, the Q values of different actions are close to each other. On the other hand, for each agent, the covariance of Q value for a specific observation-action pair increases because of the uncertainty of other agents' policies. Therefore, during the training phase, the update of Q table can easily lead to the change of the action with the maximum Q value given an observation. This can result in

$$Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) \leftarrow \begin{cases} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) + \alpha \delta_t^{(k)}, & \text{if } \mathbf{a}^{(k)} = \mathbf{a}_t^{(k)}, \\ Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) - \beta, & \text{if } \mathbf{a}^{(k)} \neq \mathbf{a}_t^{(k)} \text{ and } C_1^k(\mathbf{e}_t^{(k)}) > t_1 \text{ and } C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) < t_2, \end{cases} \quad (2.8)$$

the rapid change of the policy and thus make it harder for agents to learn to cooperate. We promote the cooperation by slowing down the policy updating speed and focusing on the experiences with high reward. That is to say, during the training phase, agents will tend to choose an experienced action with high reward when revisiting an observation and thus learn the Q values of these high-reward observation-action pairs first.

In RL systems, the policy of an agent is given by $\pi^{(k)} = \arg \max_{\mathbf{a}^{(k)}} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)})$, which means the selected action is determined by the maximum Q value given an observation $\mathbf{z}_t^{(k)}$. Therefore, in our proposed PQL, we sacrifice the accuracy of non-maximum observation-action Q values to let the agent stay on observation-action pairs with high reward first. Specifically, we modify the learning steps of the IQL. When receiving an experience tuple $\mathbf{e}_t^{(k)}$, instead of updating the Q value of one observation-action pair, PQL updates the Q values of all actions given the observation $\mathbf{z}_t^{(k)}$. The PQL updating rules are expressed as in (2.8), where $\delta_t^{(k)}$ is the corresponding TD error. β is the penalty factor. $C_1^k(\mathbf{e}_t^{(k)})$ and $C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)})$ are conditional variables, whose expressions are as follows,

$$C_1^k(\mathbf{e}_t^{(k)}) = R_t^{(k)} + \gamma \max_{\mathbf{a}^{*(k)}} Q_k(\mathbf{z}_{t+1}^{(k)}, \mathbf{a}^{*(k)}) - \max_{\mathbf{a}^{*(k)}} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{*(k)}), \quad (2.9)$$

$$C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) = \max_{\mathbf{a}^{*(k)}} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{*(k)}) - Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}). \quad (2.10)$$

Specifically, $C_1^k(\mathbf{e}_t^{(k)})$ is used to determine whether the received experience tuple is "significantly good", and $C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)})$ is used to select those actions with estimated Q values close to the maximum Q value at the current observation $\mathbf{z}_t^{(k)}$. $t_1 > 0$ and $t_2 > 0$ are two thresholds.

With the PQL updating rules in (2.8), when an agent k receives an experience tuple $\mathbf{e}_t^{(k)}$ with significantly high reward, which is triggered by the con-

dition $C_1^k(\mathbf{e}_t^{(k)}) > t_1$, it degrades the Q values of other unselected actions with Q value close to the maximum one, which are filtered out by the condition $\mathbf{a}^{(k)} \neq \mathbf{a}_t^{(k)}$ and $C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) < t_2$.³ Therefore, when agent k revisits this observation, the previously-visited high-reward observation-action is more likely to remain the highest Q value among all actions in this observation due to the Q value degradation to other actions in previous visits although the agent has updated its Q table for many steps. As a result, the agent tends to choose the experienced action with high reward and thus the policy updating speed slows down. After the agent has well learned the high-reward observation-action pairs, the condition $C_1^k(\mathbf{e}_t^{(k)}) > t_1$ is no longer satisfied and then the PQL turns back into IQL, which results in the exploration of other observation-action pairs.⁴

2.4.3 Penalty-based Deep Q Learning

In wireless multi-agent systems, especially in the resource allocation scenarios, the state is usually determined by the system global CSI, which is continuous and not suitable for traditional Q table based RL algorithms. As a consequence, to solve the power control problem in HetNet, we need to extend our PQL to deep Q learning scenarios.

In the context of deep Q learning, the Q table is replaced by the DQN, which estimates the Q values of all actions with a given observation. Therefore, we can realize the PQL in deep Q learning by penalizing the outputs of the DQN corresponding to the unselected actions. Specifically, we add a few regularization terms in the Q loss function. The resulted loss function for agent k of our multi-agent deep PQL algorithm is given by (2.11), where β is the penalty factor, $Q_k(\cdot; \boldsymbol{\theta}_t^{(k)})$ denotes the DQN of agent k at time slot t with respect to the network parameters $\boldsymbol{\theta}_t^{(k)}$. $\delta_t^{(k)} = R_t^{(k)} + \gamma \max_{\mathbf{a}^{*(k)}} Q_k^{\text{tar}}(\mathbf{z}_{t+1}^{(k)}, \mathbf{a}^{*(k)}; \boldsymbol{\theta}^{(k)-}) - Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)}; \boldsymbol{\theta}_t^{(k)})$

³There is no theoretically optimal selection of t_1 and t_2 . We selected their value based on numerical results. Typically, t_1 and t_2 can be set as relative values, i.e., $t_1 = 0.1R_t^{(k)}$ and $t_2 = 0.1Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{*(k)})$.

⁴We employed this Q table based PQL to play the two-agent predator-prey game as described in [62]. The simulation results show the superiority of PQL to the hysteretic Q learning from [62]. For the consistence of this chapter, we omit this experiment in this chapter.

$$L(\mathbf{e}_t^{(k)}; \boldsymbol{\theta}_t^{(k)}) = (\delta_t^{(k)})^2 + \beta \sum_{\mathbf{a}^{(k)} \neq \mathbf{a}_t^{(k)}} C_1^k(\mathbf{e}_t^{(k)}) C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}; \boldsymbol{\theta}_t^{(k)}), \quad (2.11)$$

$$C_1^k(\mathbf{e}_t^{(k)}) = \begin{cases} 1, & \text{if } R_t^{(k)} + \gamma \max_{\mathbf{a}^{*(k)}} Q_k^{\text{tar}}(\mathbf{z}_{t+1}^{(k)}, \mathbf{a}^{*(k)}; \boldsymbol{\theta}^{(k)-}) \\ & - \max_{\mathbf{a}^{*(k)}} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{*(k)}; \boldsymbol{\theta}_t^{(k)}) > t_1 \\ 0, & \text{otherwise,} \end{cases} \quad (2.12)$$

$$C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) = \begin{cases} 1, & \text{if } \max_{\mathbf{a}^{*(k)}} Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{*(k)}; \boldsymbol{\theta}_t^{(k)}) - Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}; \boldsymbol{\theta}_t^{(k)}) < t_2 \\ 0, & \text{otherwise,} \end{cases} \quad (2.13)$$

denotes the TD error of agent k at time slot t , where $Q_k^{\text{tar}}(\cdot; \boldsymbol{\theta}^{(k)-})$ is the target DQN and $\boldsymbol{\theta}^{(k)-}$ is the corresponding target network parameters. As presented in Section 2.4.2, $C_1^k(\mathbf{e}_t^{(k)})$ and $C_2^k(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)})$ are the conditional variables given in (2.12) and (2.13), respectively, where $t_1 > 0$ and $t_2 > 0$ denote the corresponding thresholds.

2.5 Proposed DTE Power Control Scheme

In this section, we implement our proposed PQL to allocate transmit power for the HetNet, in which each AP acts as an agent, which interacts with the environment and gains the experience for training. Each agent needs to gradually adjust its own power control strategy based on its local experience. We design a local reward for each agent, which considers the achievable rates of nearby cells, so that the whole multi-agent system works in a cooperative manner in the interest of global network performance.

Our proposed MARL scheme has two phases, including training phase and implementation phase. In this chapter, we focus on DTE power control schemes. In such DTE schemes, each agent maintains a local DQN and updates the DQN based on its local experience to maximize its local reward. In addition, each agent decides its transmit power based on the collected local observation independently and simultaneously on a time scale on par with the small-scale channel fading.

2.5.1 State and Observation Space

APs are allowed to exchange some information before each transmission stage. At the beginning of each time slot, each AP collects local measurements from its neighbors via wired or wireless backhaul links [66]. Specifically, the local measurement of AP k , $\mathcal{O}_k^t$, includes the achievable rate, $r_k^{t-1}(\mathbf{P}^t)$, and the received interference plus noise power $I_k^{t-1} = \sum_{k' \neq k} h_{k,k'}^{t-1} P_{k'}^{t-1} + \sigma_k^2$ measured in the last time slot $t - 1$, i.e., $\mathcal{O}_k^t = \{r_k^{t-1}, I_k^{t-1}\}$. The local measurement, $\mathcal{O}_k^t$, is then sent to the neighboring APs via backhaul links. Each AP k collects the set of local measurements from the neighboring APs, which is defined by $\{\mathcal{O}_{k'}^t | k' \in \mathcal{N}_k\}$. The neighboring set of AP k , $\mathcal{N}_k$, is the index set of M UDs served by other APs with the smallest distance to AP k . Therefore, together with the estimated transmission link channel gain, $h_{k,k}^t$, the local observation of AP k can be expressed as $\mathbf{z}_t^{(k)} = \{h_{k,k}^t, \mathcal{O}_{k'}^t | k' \in \mathcal{N}_k \cup \{k\}\}$, where $\mathcal{N}_k$ denotes the index set of the nearby cells. The neighboring set of AP k , $\mathcal{N}_k$, is the index set of M UDs served by other APs with the smallest distance to AP k . As a result, each agent

has a local observation of $2M + 3$ dimensions.

Note that the local observation $\mathbf{z}_t^{(k)}$ only includes the transmission link channel gain and the local measurements of itself and its neighbors. Therefore, each AP only needs to send two scalars to its neighbors in each time slot. We thus assume that the local observation can be obtained without delay at the beginning of each time slot. In addition, the local reward we design in Section 2.5.3 does not require any additional information beside the local observation we discuss above.

2.5.2 Action Space

In this chapter, we limit the transmit power to discrete levels for ease of learning and practical circuit restriction. Specifically, each AP k with maximum transmit power budget $P_k^{\max}$ has a discrete power level set of $[0, \frac{P_k^{\max}}{A_C-1}, \frac{2P_k^{\max}}{A_C-1}, \dots, P_k^{\max}]$, where A_C is the cardinality of the action set. Therefore, each agent has an action space of A_C , and each action refers to a specific transmit power level in the action space.

2.5.3 Reward Design

One advantage of RL is the flexibility of reward design. When the reward is correlated to the desired objective, the system performance can be increased. Our goal is to improve the system sum-throughput and train the MARL system locally. Therefore, we need to design a reward obtained from local information and related to the system sum-throughput for each agent. To do so, we use the achievable rates obtained from neighbors and define the reward for time slot t at agent k as follows,

$$R_k^t = \frac{1}{M+1} \sum_{k' \in \mathcal{N}_k \cup \{k\}} r_{k'}^t. \quad (2.14)$$

Since each AP's transmit power mainly affects the achievable rates of its neighboring UDs, the local reward defined in (2.14) can successfully characterize the system sum-throughput.

2.5.4 Distributed Implementation Phase

The distributed power control algorithm is summarized in Algorithm [1](#), where we set $is_train = False$. Specifically, in the implementation phase, at time slot t , each agent k first estimates and collects its local measurement, i.e., $h_{k,k}^t$, r_k^{t-1} and I_k^{t-1} as in Step 3.1. Then it exchanges the local measurement with its neighbors and forms the local observation $\mathbf{z}_t^{(k)}$ as in Step 3.2. Based on local observation $\mathbf{z}_t^{(k)}$, agent k selects the action a_k^t corresponding to the maximum Q value as in Step 3.4. All APs then start transmission with the transmit power corresponding to the selected actions as in Step 3.5.

Note that the information exchange can be done before next time slot starts once the SINR and interference power are estimated with no delay. In addition, each AP only needs to implement a local low-complexity forward propagation to adjust their transmit power. As a result, our proposed MADRL-based power control algorithm can make decision on time and thus is more practical compared with optimization-based algorithms.

2.5.5 Decentralized Training Phase

We focus on the training in a dynamic episodic setting where users' locations are fixed during each episode. When an episode begins, the locations of all users are regenerated uniformly in the corresponding cells and the initial transmit powers of all cells are set to $P_k^{\max}, \forall k \in \mathcal{K}$. The large-scale fading is then generated based on the users' locations. The small-scale fading is updated when a time slot begins. The change of small-scale channel fading triggers a transition of the environment state and causes each agent to adjust its transmit power.

We leverage deep Q learning with experience replay [\[25\]](#) to train the HetNet agents for effective learning of local power control policies. The training phase is summarized in Algorithm [1](#), where $is_train = True$. Compared with the implementation phase, the training phase has two different steps, Step 3.3 and Step 3.4. Step 3.3.1 is the experience collecting step, where all APs store their local experience tuples of the last time slot in their local replay buffers. Steps

Algorithm 1 Proposed power control scheme

1. Episode m starts. Users are randomly located.
 2. Time slot t starts. Small scale channel gain is updated according to (2.2).
 3. For each AP k ,
 - 3.1 Before updating the power, user k estimates its direct link channel gain $h_{k,k}^t$ and the interference plus noise power strength I_i^{t-1} , as well as the achievable rate of the last time slot r_k^{t-1} , and then feeds them back to AP k .
 - 3.2 Each AP k obtains the local measurements $\mathcal{O}_j^t$ from its neighboring APs $j \in \mathcal{N}_k$ through backhaul links and then forms the local observation $\mathbf{z}_t^{(k)}$.
 - 3.3 If $is_train == True$:
 - 3.3.1 Each AP k forms the experience tuple $(\mathbf{z}_{t-1}^{(k)}, a_k^{t-1}, R_k^{t-1})$ and stores it in its local replay buffer.
 - 3.3.2 Each AP k uniformly samples a mini-batch $\mathcal{D}_k$ from the replay buffer.
 - 3.3.3 Each AP k updates its local DQN by minimizing the loss function (2.11) on the mini-batch $\mathcal{D}_k$ using variant of stochastic gradient descent.
 - 3.4 If $is_train == False$:

AP k predicts its transmit power, $\hat{P}_k^t$, via ϵ -greedy, i.e., to select the action corresponding to the highest Q value with a probability of $1 - \epsilon$ or to randomly choose its transmit power with a probability ϵ .

else:

AP k predicts its transmit power, $\hat{P}_k^t$, based on the local observation $\mathbf{z}_t^{(k)}$ by selecting the action corresponding to the highest Q value.
 - 3.5 AP k adjusts the transmit power as $P_k^t = \hat{P}_k^t$.
 4. Time slot t ends. Time slot index $t = t + 1$. Go back to Step 2 if episode m does not end.
 5. Episode m ends. Episode index $m = m + 1$. Go back to Step 1.
-

3.3.2 and 3.3.3 are the network training steps, where each AP k uniformly samples a mini-batch data set $\mathcal{D}_k$ and then updates the trainable parameters of its DQN with the stochastic gradient descent method to minimize the PQL loss function as defined in (2.11). Step 3.4 is the decision making step. In the training mode, each agent chooses its action according to the widely-used epsilon-greedy strategy, i.e., to randomly choose an action with probability of ϵ , or to choose the action corresponding to the highest Q value given the current observation with probability of $1 - \epsilon$.

Note that we can easily change the simplest MADRL training algorithm, IQL, into our PQL easily. The only extra cost comes from the calculation of the added regularization terms. However, in each backward propagation, these regularization terms only involve simple linear and comparison operations and there is no extra computational complexity in back propagating from the output layer to each trainable parameter of the NN. Therefore, PQL only introduces very limited computational complexity of calculating the gradients of the regularization terms during the training phase. As a result, our proposed PQL algorithm has practical distributed training phase due to its IQL-like training manner.

2.5.6 Network Structure

To further improve the performance without imposing any change to the underlying reinforcement learning algorithm, we employ the deep dueling DQN [67] structure for our agents. Specifically, the network structure of dueling DQN is illustrated in Fig. 2.2. In dueling DQN, the Q value of an observation-action⁵ pair is decomposed as the sum of the value function and the advantage function, which is expressed as follows,

$$Q_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) = V_k(\mathbf{z}_t^{(k)}) + A_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}), \quad (2.15)$$

⁵As mentioned before, we use observation to indicate the local state for each agent. Therefore, we use the term "observation-action" instead of "state-action" in this chapter.

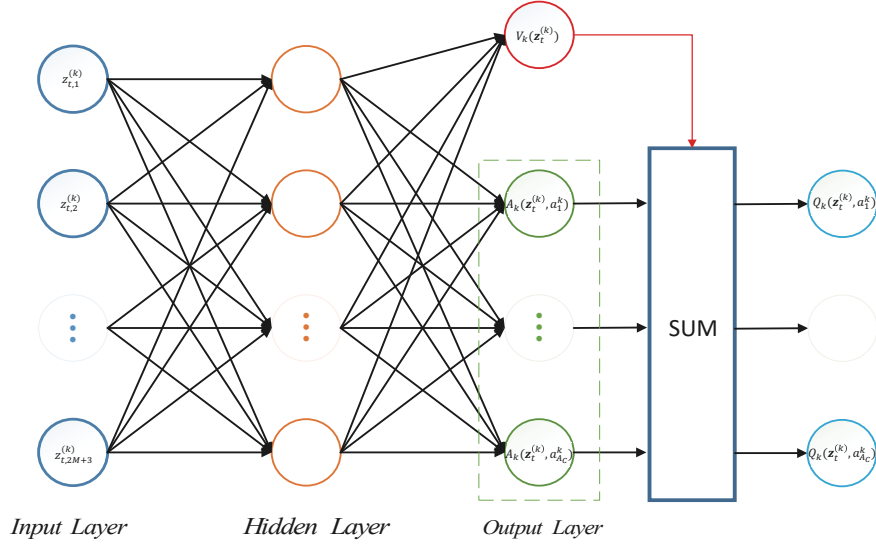


Figure 2.2: Network structure of dueling DQN.

where $V_k(\mathbf{z}_t^{(k)})$ is the value function, which reflects the average of Q values given the observation $\mathbf{z}_t^{(k)}$, and $A_k(\mathbf{z}_t^{(k)})$ is the advantage function, which measures the relative importance of a certain action compared with other actions. There is a zero-sum constraint on the advantage function $A_k(\cdot)$, which is given by,

$$\sum_{\mathbf{a}^{(k)}} A_k(\mathbf{z}_t^{(k)}, \mathbf{a}^{(k)}) = 0, \forall \mathbf{z}_t^{(k)}. \quad (2.16)$$

The intuition behind Dueling DQN is that it is not always necessary to estimate the value of taking each available action. For some states, the choice of action makes limited influence on what happens. On the other hand, with the decomposition (2.15), during the backward propagation, the NN tends to update value function $V_k(\mathbf{z}_t^{(k)})$ instead of the Q value of one observation-action pair, which results in the learning of all Q values of the same observation. For the input and hidden layers in Fig. 2.2, we use the basic fully-connected structure for all agents. Specific hyper-parameters regarding number of layers and neurons and types of activation function will be illustrated in the next section.

2.5.7 Training and Implementation Complexity

The complexity of our proposed method includes the training complexity and the implementation complexity. In the implementation phase, for each action prediction, the computational complexity of our PQL algorithm is the same as that of the IQL algorithm, as well as that of the other training based algorithms with the same DQN structure, i.e., the complexity of a forward neural network propagation. In our considered scenario, the DQN has two hidden layers, which have N_{h1} and N_{h2} neurons respectively. As a result, we can calculate the number of FLOPs [68] in each DQN forward propagation as $2 \left((2(1+M)+1)N_{h1} + N_{h1}N_{h2} + N_{h2}A_C + N_{h2} \right)$. As the number of hidden layer neurons in our considered scenario is relatively small, i.e., we set $N_{h1} = 128$ and $N_{h2} = 64$ in our simulation, the forward propagation has a low complexity. The training computational complexity is highly related to the DQN structure. Compared with the basic IQL algorithm, PQL only needs to calculate the gradients of some linear regularization terms. As a result, the additional computational complexity in the training phase can be neglected.

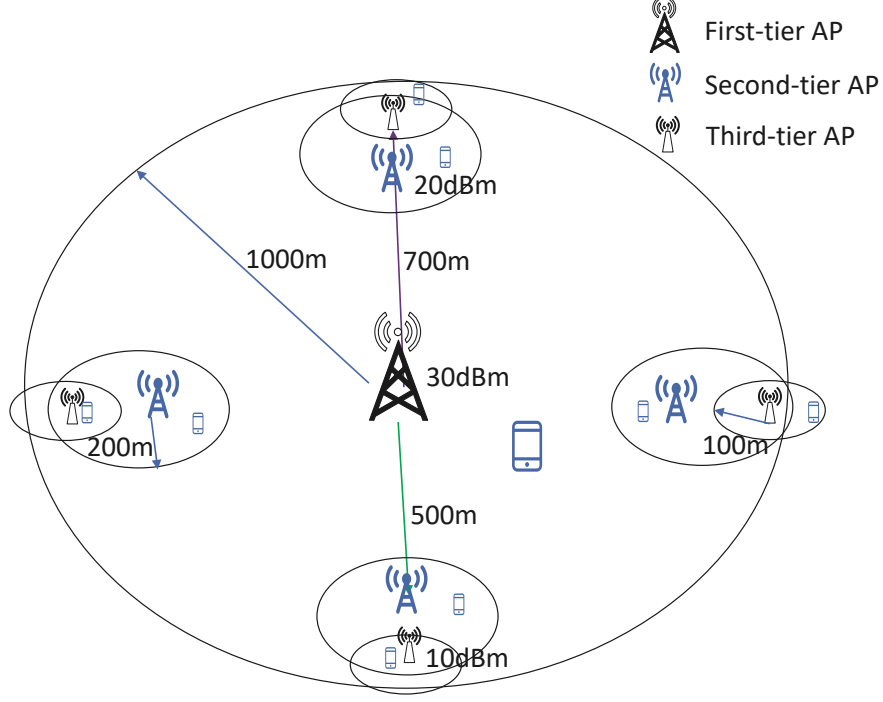


Figure 2.3: HetNet with three tiers.

2.6 Simulation Results

In this section, we present simulation results to verify the effectiveness of our proposed distributed power control scheme based on MADRL in HetNets. Unless otherwise stated, our HetNet system parameters are as the following. We consider a three-tier HetNet as illustrated in Fig. 2.3. In this HetNet, there are nine APs in total, which are located at $(0, 0)\text{m}$, $(500, 0)\text{m}$, $(0, 500)\text{m}$, $(-500, 0)\text{m}$, $(0, -500)\text{m}$, $(700, 0)\text{m}$, $(0, 700)\text{m}$, $(-700, 0)\text{m}$ and $(0, -700)\text{m}$, respectively, indexed by $[0, \dots, 8]$ [36]. AP 0 is the macro-cell or first-tier AP with a coverage defined by a minimum distance, $d_{min} = 10\text{m}$ and a maximum distance, $d_{max} = 1,000\text{m}$, from the AP to UDs. APs $\{1, 2, 3, 4\}$ are small-cell APs in the second tier whose coverage is defined by $d_{min} = 10\text{m}$ and $d_{max} = 200\text{m}$. APs $\{5, 6, 7, 8\}$ are small-cell APs in the third tier whose coverage is defined by $d_{min} = 10\text{m}$ and $d_{max} = 100\text{m}$. UDs are uniformly located in their corresponding cells. The maximum transmit power budget is set to 30 dBm, 20 dBm and 10 dBm for the first-tier, second-tier and third-tier APs, respectively, over a fully

reusable frequency band of 10 MHz. Following the LTE standard [69], the path loss is given by $120.9 + 37.6 \log_{10}(d/1,000)$ (dB), where d (m) is the distance between the transmitter and the receiver. The standard deviation of the log-norm shadow fading is 8 dB. The AWGN power is -114 dBm. The time slot duration $T = 20$ ms and we assume the same maximum Doppler frequency shift $f_D = 10$ Hz for all UDs [26]. The number of considered neighbors, M , is set as 4. The number of discrete transmit power levels is set as $A_C = 11$.

We employ the fully connected dueling DQN structure for all agents. The dueling DQN has two fully connected hidden layers with 128, 64 neurons, respectively. The activation functions of the input layer and first two hidden layers are the Leaky ReLU function with the negative slope being 0.1, while the output layer does not have activation functions. We set the memory size of the replay buffer as $D_{max} = 3,600$ and the number of time slots in each episode as $N_{slots} = 20$ so that the agent can learn the environmental dynamics caused by the change of large scale fading. The mini-batch size is set to 128. We use a decaying ϵ -greedy exploration parameter, which can be expressed as a function of training steps $\epsilon = \max(1 - 0.7 * 1.0005^t, 0.01)$. Adam optimizer with learning rate of 0.0001 is used to train the dueling DQN. Since we only aim to maximize the system sum-throughput in each time slot, the future reward is unimportant. We thus set the discount factor $\gamma = 0$. The penalty factor of PQL, β , is set as 0.05, while the two thresholds are set as $t_1 = 0.1R_t^{(k)}$ and $t_2 = 1$, respectively.

We compare our proposed scheme with some other DTE learning based scheme, including WMMSE, IQL, HQL, and full power.

- 1) WMMSE: The weighted minimum mean-squared error algorithm [8] is centralized with full-CSI and should have the best performance.
- 2) PQL: PQL is our proposed DTE power control scheme.
- 3) IQL: Its settings, including its DQN hyper-parameters, are the same to our proposed PQL scheme, except for the loss function, which still follows the original IQL loss.
- 4) HQL: Hysteretic Q-learning [62] uses two different learning rates to train the DQN. When receiving an experience, the agent first compares the es-

estimated Q value by its DQN with the target Q value, which is calculated by the sum of reward and the discounted Q value of the next step. If the target Q value is higher, it uses a normal learning rate to learn this experience, otherwise, it uses a degraded learning rate to learn this experience. We set the degraded learning rate as 0.4 as in [70].

- 5) Full power: Each AP transmits signal using maximum power.

For comprehensive comparisons, we further compare our proposed method with some other methods with more advanced structure, i.e., the multi-agent deep deterministic policy gradient (MADDPG) [71], the multi-agent proximal policy optimization (MAPPO) [72] and the neural fictitious self-play (NFSP) [59]. Specifically, MADDPG and MAPPO both are actor-critic based and CTDE methods, which can exploit the global observation information for better learning performance. NFSP adopts the idea of slowing down the policy updating speed by maintaining an averaged policy. For fair comparison, we set the policy (actor) network structures of these three algorithms the same as our PQL algorithm, so that all the algorithms have the same computational complexity in the implementation phase. The learning rates of these three algorithms are set to 0.0001 so that these baseline algorithms can converge to good policies in our considered scenarios. Other hyper-parameters remain the same as in the original papers. This work focuses on the training method so we mainly compare it with the learning-based methods.

To begin with, we compare our proposed method with other benchmarks in the considered scenario. Specifically, to better evaluate the performance of the MARL system, we use two identical environments in our experiments, i.e., training environment and testing environment. We train the agents in the training environment, which updates the trainable parameters of all agents, while test the performance in the testing environment with all trainable parameters fixed. In the testing phase, the DQNs are used to optimize the transmit power at each AP for 4,000 time slots, i.e., 200 episodes. Therefore, all the curves in the following figures are the average performance over 200 episodes.

Fig. 2.4 shows the average sum-throughput performance of the proposed al-

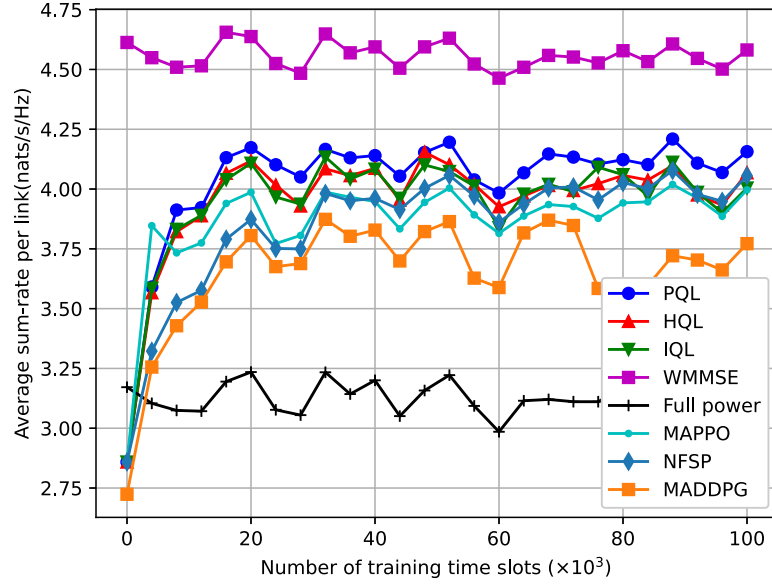


Figure 2.4: Average sum-throughput performance of different algorithms in the training phase.

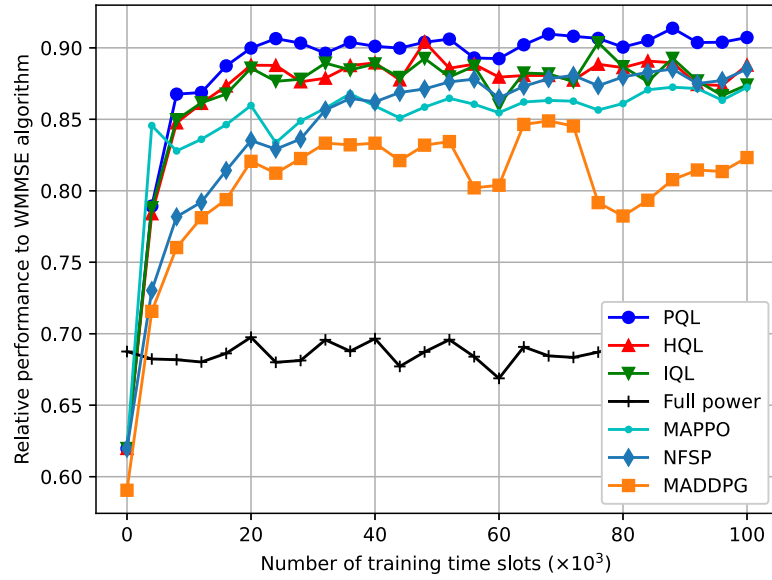


Figure 2.5: Relative performance of different algorithms in the training phase.

algorithm and other benchmark algorithms in the investigated HetNet. From the figure, all the learning based algorithms have worse performance than the full power scheme at the beginning since their policy networks are randomly initialized and thus randomly choose transmit power. After training for 1,000 episodes, i.e. 20,000 time slots, the performance of MADRL-based algorithms comes to a stage. In addition, our proposed PQL outperforms other MADRL-based methods all the time and the performance gain over IQL is around 3%. This is because in PQL, each agent tends to choose an experienced action with high reward when revisiting a state, and thus the policy updating speed slows down. In this way, each agent's policy can be learned by other agents more easily, resulting in a more efficient cooperation-learning process. In addition, the CTDE methods MAPPO and MADDPG fail to outperform our proposed PQL algorithm in this scenario. The superiority of the PQL algorithm over the NFSP algorithm verifies the effectiveness of the learning strategy of the PQL algorithm, i.e., to focus on high-reward experiences first. Note that the UD locations are regenerated randomly in every episode, which causes the performance fluctuation to all algorithms. Therefore, to better illustrate the performance of different learning methods and to show the effectiveness of these learning algorithms, we illustrate the relative performance of the considered methods to the WMMSE algorithm in Fig. 2.5. We can observe that PQL can reach over 90% of the sum-throughput achieved by the WMMSE algorithm. In addition, Fig. 2.5 also indicates that our proposed PQL has less performance variations across training epochs compared with IQL algorithm.

When using MADRL methods, the local observation of each agent includes SINR and interference information regarding the CSI of previous time slot. Therefore, when $\rho = 0$, agents can obtain less information from its local observation about CSI of the current time slot. We thus set the correlation coefficient $\rho = 0$ and keep other system settings unchanged to evaluate the performance of these algorithms as shown in Fig. 2.6. Our proposed PQL still outperforms IQL and HQL with performance gain of around 4%, although it also encounters performance degradation after 60,000 training steps. In this experiment setting, the CTDE MAPPO algorithm successfully outperforms the PQL algorithm, but the MADDPG algorithm still fails to learn a good policy. However,

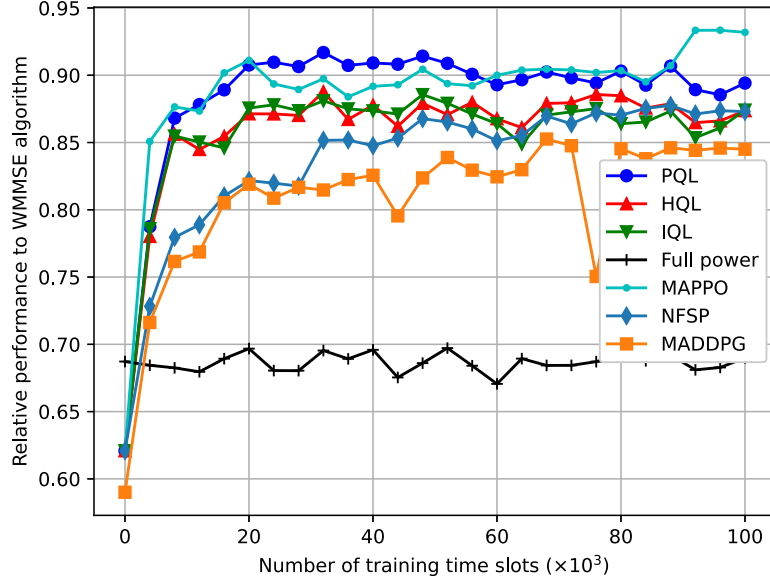


Figure 2.6: Relative performance of different algorithms in the training phase when $\rho = 0$.

our proposed PQL algorithm has the best performance among all algorithms except for the CTDE MAPPO algorithm. This result validates the ability of our PQL to learn to cooperate given less state information.

In the following experiment, we add 4 more third-tier cells at the edge of the macro cell to investigate PQL with more agents. Specifically, the newly added third-tier APs are located at $(250\sqrt{2}, 250\sqrt{2})\text{m}$, $(250\sqrt{2}, -250\sqrt{2})\text{m}$, $(-250\sqrt{2}, -250\sqrt{2})\text{m}$ and $(-250\sqrt{2}, 250\sqrt{2})\text{m}$, respectively, with other system parameters the same as other third-tier APs. Fig. 2.7 and Fig. 2.8 draw the system sum-throughput performance of considered algorithms and the relative performance to the WMMSE algorithm of the considered algorithms during the training phase. From Fig. 2.4 and Fig. 2.7, with more third-tier APs deployed at the macro-cell edge, the system average sum-throughput performance of all algorithms is increased. This result validates the benefit of deploying small cells within macro cells. In addition, from Fig. 2.5 and Fig. 2.8, the performance gain gap between PQL and IQL/HQL is around 5% and increases with the number of agents, although

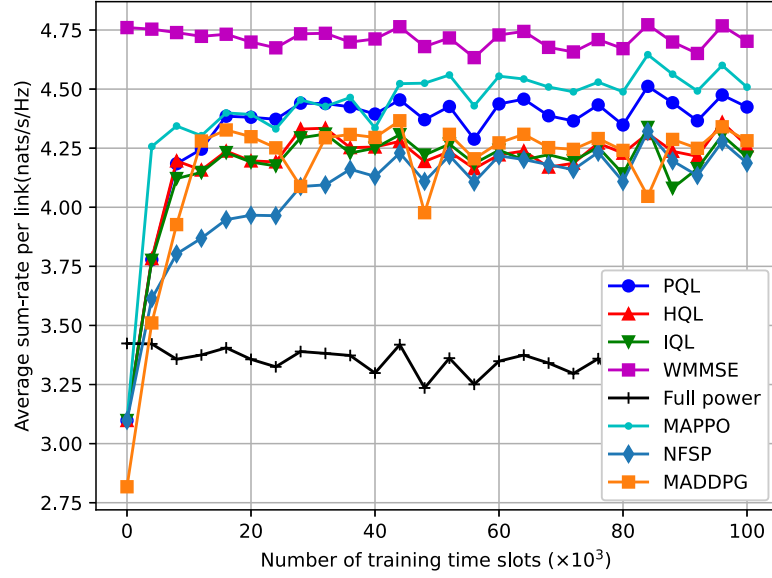


Figure 2.7: Average sum-throughput performance of different algorithms in the training phase in the HetNet with 4 additional third-tier cells.

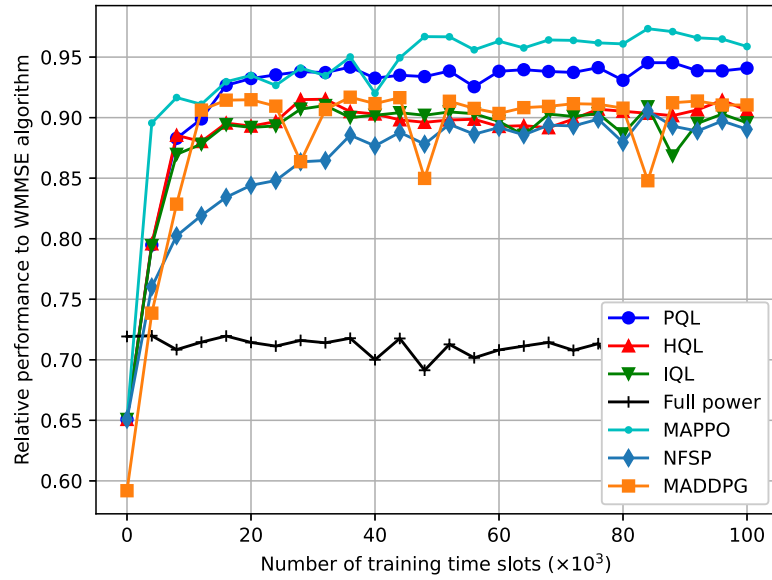


Figure 2.8: Relative performance of different algorithms in the training phase in the HetNet with 4 additional third-tier cells.

the relative performance of all MADRL methods increases. This is because with the number of agents increasing, better cooperation among agents can provide more performance gain. On the other hand, the MAPPO algorithm, which exploits the global information, has the best performance while the MADDPG algorithm only has similar performance with the IQL and the HQL algorithms. The NFSP algorithm has a slower convergence speed and finally has similar performance to the IQL and HQL algorithms. Therefore, the MAPPO algorithm and our proposed PQL algorithm can efficiently promote the cooperation among agents and thus obtains more performance gain over other benchmarks.

Considering the difference of computational capability, battery capacity and environmental statistics of different-tier APs in HetNets, different-tier APs can be deployed with different-structure DQNs. Therefore, in the last experiment, we use different-structure dueling DQNs in different-tier APs to investigate the ability of our PQL to train heterogeneous multi-agent systems. For the first-tier AP, its dueling DQN has three hidden layers with 128, 128, 64 neurons, respectively. For second-tier APs, their dueling DQNs have two hidden layers with 128, 64 neurons, respectively. For third-tier APs, their dueling DQNs have two hidden layers with 64, 64 neurons, respectively. The resulted relative performance during the training phase is shown in Fig. 2.9. Our proposed PQL algorithm and the CTDE MAPPO algorithm have similar performance and reach over 90% sum-rate of the WMMSE algorithm and outperform other MADRL algorithms as expected. The MADDPG algorithm fails to learn a good policy in this experiment. In addition, our proposed PQL has a performance gain of around 4% over IQL. This result validates our PQL on its ability of training a MADRL system with heterogeneous agents.

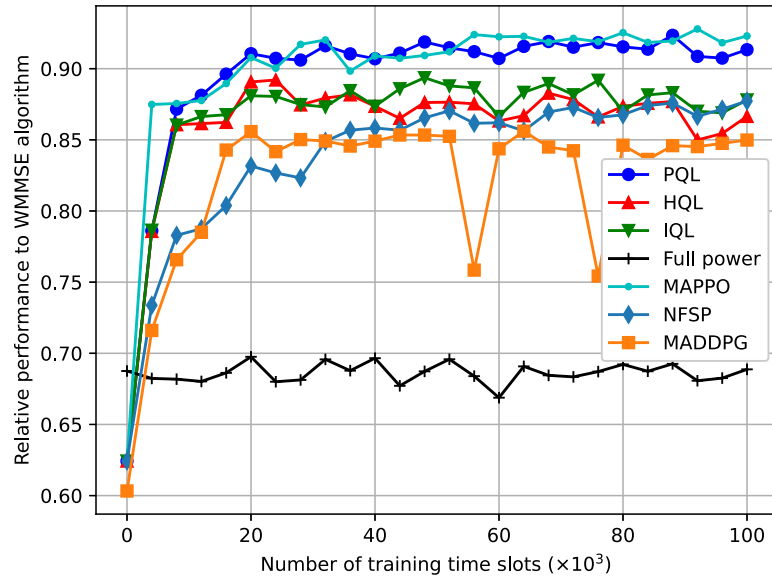


Figure 2.9: Relative performance of different algorithms in the training phase with different-structure DQNs deployed at different-tier APs.

2.7 Conclusions and future research directions

In this chapter, we proposed a DTE MADRL-based power control scheme to allocate transmit power in HetNets. In particular, we introduced a DTE scheme, which decides the transmit power for each AP simultaneously based on local observation. In addition, only the interference power and achievable rates of neighbors are required in both training and testing phases, which is practical in real systems since the unavailable global and interference CSI is not required. To promote cooperation among agents, we proposed PQL algorithm for training MADRL systems, which has almost no extra cost compared with the simple and widely used IQL algorithm. Simulation results show that with limited network layers and local observation, our proposed scheme is able to reach over 90% performance of the conventional WMMSE algorithm and outperforms the existing DTE IQL and HQL algorithms under various system settings. By adjusting the reward design, the proposed DTE scheme can be further extended to optimize other system KPIs, like fairness, coverage and outage probability, so that these KPIs can satisfy more complex requirements in real HetNet systems.

Chapter 3

Federated Reinforcement Learning for Resource Allocation in V2X Networks

In this chapter, we improved Federated Reinforcement Learning (FRL) by advanced optimization framework. We proposed a Policy Gradient based ADMM with a Second Moment (PASM) algorithm to optimize the Policy Gradient (PG) based FRL systems. Specifically, we used inexact ADMM to address the Federated Learning (FL) problem formulated from PG based Multi-Agent Deep Reinforcement learning (MADRL) system with knowledge sharing. The second moment of the Lagrange multipliers is incorporated to accelerate the training. We then applied the PASM algorithm to the distributed resource allocation problems in a vehicle-to-everything (V2X) networks, where vehicle-to-vehicle (V2V) links reuse the spectrum of vehicle-to-infrastructure (V2I) links. Simulation results show that the PASM algorithm can accelerate the convergence speed and improve the convergent performance under different system configurations.

3.1 Introduction

The Next Generation Multiple Access (NGMA) vehicle-to-everything (V2X) networks have attracted considerable research interest since they are capable of delivering many important services, e.g., road safety and traffic efficiency, and enable various applications in smart cities, autonomous driving, and intelligent transport systems [73–75]. Entities, including vehicles and roadside units in V2X networks, communicate and cooperate with each other and thus result in the coexistence of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications on the same spectrum. Therefore, complex mutual interference and severe performance degradation may arise. To overcome such drawbacks, proper resource allocation schemes need to be developed.

It has been noted that resource allocation is usually formulated as an optimization problem, which however is NP-hard in general and lacks universal low-complexity and effective solutions. There is an impressive body of work on developing traditional optimization model-based approaches for resource allocation in V2X networks [76–81]. For example, by considering the density and physical proximity of vehicles, a decentralized algorithm has been proposed in [77] to optimize the transmission delay and successful transmission probability. In [78], a joint optimal centralized spectrum sharing and power control method has been developed to maximize the V2I link sum rate while guaranteeing the reliability of the V2V links with delayed Channel State Information (CSI) feedback. Furthermore, based on the slowly-varying large-scale fading information, the sum ergodic capacity of V2I links with V2V link reliability has been optimized in [79]. Additionally, the graph partitioning tool has been adopted to categorize the highly interfering V2V links into different clusters to reduce computational complexity and signaling overhead [80]. However, due to the fast-varying channel conditions, it is usually hard to obtain global CSI, which limits the practical implementation of the traditional model-based resource allocation schemes in V2X networks, and traditional centralized solutions usually lack scalability in large-scale V2X networks.

DL based methods, especially Federated Reinforcement Learning (FRL) algo-

rithms, which can aggregate the knowledge from different agents observing different parts of the wireless environment, have great potential to address the challenges as we have reviewed in Section 1.2. However, it is known that many Federated Learning (FL) algorithms, such as FedAvg [40] and FedProx [43], have been proposed based on the gradient descent scheme. A separate line of research develops FL algorithms using inexact ADMM [82–85]. The FedGiA algorithm in [85] integrates the gradient descent and inexact ADMM. It has been shown to have high communication efficiency, low computational complexity, and convergence under weaker conditions. In addition, compared with value-based RL systems, we can use continuous optimization techniques to train the policy-optimization RL systems. The policy-optimization multi-agent reinforcement learning (MARL) algorithm is analyzed and connected with optimization problems in [86]. Therefore, we adopted partial ideas from FedGiA to FRL and designed a general PG-based ADMM with a Second Moment (PASM) algorithm to improve the performance of FRL.

Overall, considering the advantageous properties of FRL to overcome the issues of local observability and the efficiency of ADMM for distributed optimization problems, we integrate them to develop a new algorithm for NGMA V2X networks in this chapter.

3.1.1 Specific Contributions of this Chapter

We employ the framework of FRL to train the agents for sub-channel and transmit power level selection in a NGMA V2X network, where each V2V link is deemed as an agent and learns to optimize the V2I link sum rate and the V2V link packet delivery rate based on local observation in a distributed manner. The FRL framework is then implemented by the inexact ADMM where sub-problems are solved approximately using PG. Our main contribution is three-fold.

- We formulate the spectrum-sharing resource allocation problem in NGMA V2X networks as a MARL system to train a distributed resource allocation scheme. Specifically, we consider two different metrics, i.e., the successful

package delivery rate of V2V links and the weighted sum rate of all links, in the V2X networks, where the first metric focuses more on the long-term reward while the second metric focuses more on the instantaneous reward.

- For the training phase, we exploit the FL and PG, and propose a PASM algorithm to train the proposed MARL system in a FL manner. The agent policy-optimization problem is formulated as a FL problem.
- We exploit the inexact ADMM to solve the FL problem, where the second moment is adopted to further improve the algorithmic performance. Such information has been widely used in some popular optimizers in deep learning, e.g., Adam [53] and RMSProp¹. Despite the challenge of establishing the convergence property for an algorithm to solve RL problems, we manage to show that the proposed general method, PASM, which can also be applied to other FL problems, can converge under mild conditions.
- We implement PASM in the considered NGMA V2X network and compare it with a FedAvg-based FRL algorithm and an independent PG algorithm, both of which employ the second-moment based optimizer RMSprop. Simulation results show that PASM can achieve better performance in terms of obtaining moving average rewards.

¹RMSprop is an unpublished adaptive learning rate algorithm proposed by Geoff Hinton in Lecture 6e of his Coursera Class.

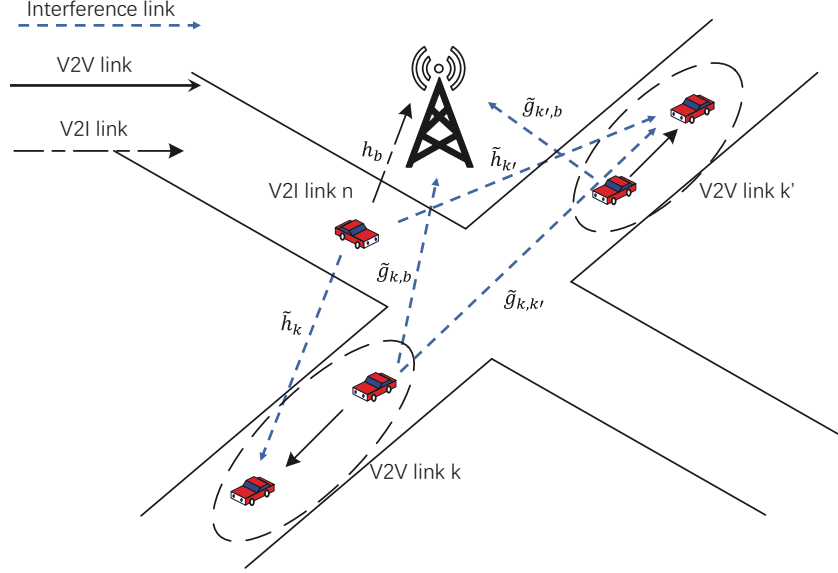


Figure 3.1: The V2X network diagram

3.2 System Model and Problem Formulation

In this chapter, we demonstrate the potential of FRL using resource allocation in V2X networks as an example. As shown in Fig. 3.1, we consider a single-antenna V2X network based on orthogonal frequency-division multiple access (OFDMA), where N V2I links connect the vehicles and the BS, and K V2V links connect the neighboring vehicles. The V2I links support high-data-rate services, and each of them is allocated with an orthogonal sub-channel. As a result, the number of sub-channels matches the number of V2I links in the system under consideration. The V2V links are enabled by device-to-device (D2D) communication and reuse the uplink resource blocks allocated to V2I links to enhance system spectrum efficiency. We denote the set of V2I links as $\mathbb{N} = \{1, 2, \dots, N\}$, the set of V2V links as $\mathbb{K} = \{1, 2, \dots, K\}$, and the set of time slots as $\mathbb{T} = \{1, 2, \dots, T\}$. We make the assumption that the n th sub-channel is allocated to the corresponding n th V2I link. The set of available sub-channels is denoted as $\mathbb{N}$.

In time slot $t \in \mathbb{T}$, the SINR of the n th V2I link can be expressed as follows,

$$\gamma_{n,t}^i[n] = \frac{P_{n,t}^i h_{b,t}[n]}{\sum_{k \in \mathbb{K}} \delta_{k,t}[n] P_{k,t}^v[n] \tilde{g}_{k,b,t}[n] + \sigma^2}, \quad (3.1)$$

where $h_{b,t}[n]$ denotes the channel power gain of the BS on the n th V2I link, $\tilde{g}_{k,b,t}[n]$ is the interference channel power gain from the transmitter of the k th V2V link to the BS on the sub-channel n , σ^2 refers to the received Gaussian noise power, $P_{n,t}^i \leq P_{\max}^i$ and $P_{k,t}^v[n] \leq P_{\max}^v$ denote the transmit power of the n th V2I link and that of the k th V2V link on the n th sub-channel, respectively, and $\delta_{k,t}[n]$ is a binary indicator presenting the sub-channel allocation of V2V link k . If sub-channel n is allocated to V2V link k , $\delta_{k,t}[n] = 1$, otherwise $\delta_{k,t}[n] = 0$. We also limit the number of sub-channels that each V2V link can occupy to 1, namely, $\sum_{n \in \mathbb{N}} \delta_{k,t}[n] \leq 1$ for all $k \in \mathbb{K}$ and $t \in \mathbb{T}$. The resulting achievable rate of V2I link n in time slot t is then given by,

$$C_{n,t}^i = W \log(1 + \gamma_{n,t}^i[n]), \quad (3.2)$$

where W is the sub-channel bandwidth.

For the k th V2V link, in time slot t , the corresponding SINR on sub-channel n is given by,

$$\gamma_{k,t}^v[n] = \frac{\delta_{k,t}[n] P_{k,t}^v[n] g_{k,t}[n]}{I_{k,t}^v[n] + P_{n,t}^i \tilde{h}_{k,t}[n] + \sigma^2}, \quad (3.3)$$

where $I_{k,t}^v[n] = \sum_{k' \in \mathbb{K}, k' \neq k} \delta_{k',t}[n] P_{k',t}^v[n] \tilde{g}_{k',k,t}[n]$ denotes the interference power received by the receiver of V2V link k in time slot t from other V2V link transmitters on sub-channel n , $\tilde{g}_{k',k,t}[n]$ denotes the interference channel power gain from the transmitter of V2V link k' to the receiver of V2V link k on sub-channel n , and $\tilde{h}_{k,t}[n]$ denotes the interference channel power gain from the transmitter of V2I link n to the receiver of V2V link k on sub-channel n . Overall, we can express the corresponding achievable rate of V2V link k in time slot t as,

$$C_{k,t}^v = \sum_{n \in \mathbb{N}} W \log(1 + \gamma_{k,t}^v[n]). \quad (3.4)$$

The V2V links carry the safety-related information generated periodically, which needs to be delivered within a given time duration [28]. This V2V link transmission requirement is mathematically formulated as the delivery rate of packets of size B within T time slots,

$$p(\boldsymbol{\delta}_{k:}, \mathbf{P}_{k:}^v) := \Pr\left(\Delta T \sum_{t \in \mathbb{T}} C_{k,t}^v \geq B\right), \forall k \in \mathbb{K}, \quad (3.5)$$

where ΔT is the channel coherence time, $\boldsymbol{\delta}_{k:} := \{\delta_{k,t}[n] : t \in \mathbb{T}, n \in \mathbb{N}\}$, and $\mathbf{P}_{k:}^v := \{P_{k,t}^v[n] : t \in \mathbb{T}, n \in \mathbb{N}\}$. We consider two scenarios of the resource allocation for the V2X networks.

- Scenario I: one goal is to maximize all V2V packet delivery rates $C_{k,t}^v$ while not affecting the V2I link transmission by properly allocating the sub-channel and transmit power of V2V links with a given power control policy of V2I links, which is formulated as the following problem:

$$\begin{aligned} \max_{\boldsymbol{\delta}_{k:}, \mathbf{P}_{k:}^v, k \in \mathbb{K}} \quad & \omega \sum_{n \in \mathbb{N}} \sum_{t \in \mathbb{T}} C_{n,t}^i + \sum_{k \in \mathbb{K}} p(\boldsymbol{\delta}_{k:}, \mathbf{P}_{k:}^v) \\ \text{s.t.} \quad & \sum_{n \in \mathbb{N}} \delta_{k,t}[n] \leq 1, \forall k, t \\ & \delta_{k,t}[n] \in \{0, 1\}, \forall k, n, t \\ & 0 \leq P_{k,t}^v[n] \leq P_{\max}^v, \forall k, n, t. \end{aligned} \quad (\text{P1})$$

- Scenario II: Another goal is to maximize the weighted sum rate of the V2V links and the V2I links in time slot $t \in \mathbb{T}$, which is a commonly used performance metric in many systems. The problem can be formulated as follows,

$$\begin{aligned} \max_{\boldsymbol{\delta}_{:t}, \mathbf{P}_{:t}^v} \quad & \omega \sum_{n \in \mathbb{N}} C_{n,t}^i + (1 - \omega) \sum_{k \in \mathbb{K}} C_{k,t}^v \\ \text{s.t.} \quad & \sum_{n \in \mathbb{N}} \delta_{k,t}[n] \leq 1, \forall k \\ & \delta_{k,t}[n] \in \{0, 1\}, \forall k, n \\ & 0 \leq P_{k,t}^v[n] \leq P_{\max}^v, \forall k, n. \end{aligned} \quad (\text{P2})$$

where for any $t \in \mathbb{T}$, $\boldsymbol{\delta}_{:t} := \{\delta_{k,t}[n] : k \in \mathbb{K}, n \in \mathbb{N}\}$ and $\mathbf{P}_{:t}^v := \{P_{k,t}^v[n] : k \in \mathbb{K}, n \in \mathbb{N}\}$.

Note that we mainly focus on the resource allocation of the V2V links with a given V2I link power control policy. Therefore we fix the V2I link transmit power to its maximum level, i.e., $\{P_{n,t}^i = P_{\max}^i, \forall n, t\}$ in both considered scenarios. We aim to develop real-time distributed resource allocation schemes, which only require local observations for V2V links in the two scenarios. The V2V packet delivery rate in (P1) can be obtained after every T time slot. On the other hand, the weighted achievable sum rate in (P2) is a short-term metric influenced by the global CSI and the resource allocation policy for each individual time slot t . Both problems are real-time sequential decision-making problems. We thus adopt the FRL techniques to train distributed resource allocation schemes for problems (P1) and (P2) so that each agent can obtain the knowledge from other agents without raw data exchange.

3.3 FRL via Inexact ADMM and Policy Gradient

In this section, we will focus on a specific class of MARL systems and develop the PASM algorithm to train a powerful global policy for the considered systems, Markov potential game (MPG), based on the inexact ADMM.² This section begins with the formulation of the MARL system, followed by the PASM algorithm and its convergence analysis.

3.3.1 Multi-Agent Policy Gradient

A partially observable MARL system can be modeled as a POMDP with a tuple $\langle K, \mathbf{s}_t, \mathbf{a}_t^{(k)}, R_t^{(k)}, \mathbf{z}_t^{(k)}, P, O \rangle$, where K is the number of agents, $\mathbf{s}_t$ is the environment state at time t , $\mathbf{a}_t^{(k)}$ is the action at time t of agent k , $\mathbf{z}_t^{(k)} = O(\mathbf{s}_t, k)$ is the local observation obtained by agent k , observation function $O(\cdot, \cdot)$ maps environment state $\mathbf{s}_t$ to the specific observation $\mathbf{z}_t^{(k)}$ of agent k , $R_t^{(k)}$ is the local reward received by agent k from the environment, and $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ is a transition probability from state $\mathbf{s}_t$ with action $\mathbf{a}_t$ to next state $\mathbf{s}_{t+1}$.

The general idea of MARL is given as follows. At time step t , based on the local observation $\mathbf{z}_t^{(k)}$, agent k selects an action $\mathbf{a}_t^{(k)}$ from the system's joint action $\mathbf{A}_t$ and receives a local reward $R_t^{(k)}$ from the environment. Then current state $\mathbf{s}_t$ transits to next state $\mathbf{s}_{t+1}$ with a transition probability $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. Subsequently, each agent k obtains a new observation of the environment, $\mathbf{z}_{t+1}^{(k)} = O(\mathbf{s}_{t+1}, k)$. In this chapter, we investigate cooperative games, where all agents cooperate to improve the system performance. In other words, we consider a special case of MARL systems, i.e., the MPG.

Moreover, we take advantage of PG to cast our FRL framework. It is noted that the PG-based method directly optimizes the policy of the agents to maximize the accumulative reward. More precisely, it maximizes the accumulative reward during a time period T obtained by implementing the policy, $\pi_k(\mathbf{a} \mid \mathbf{z}_t^{(k)})$, which denotes the probability of performing action $\mathbf{a}$ when observing $\mathbf{z}_t^{(k)}$ for

²Although all agents share a global policy model, this global policy model is able to distinguish different agents from their different states. For example, if we include the agent index in the state, the policy model can easily distinguish different agents from the input indices. This setting enables FRL to train an MARL system.

agent k . Denote Φ the joint policy of all agents by

$$\Phi(\mathbf{A}|\mathbf{s}_t) = \prod_{k=1}^K \pi_k(\mathbf{a}|\mathbf{z}_t^{(k)}). \quad (3.6)$$

Then, given the static environment transition probability and the joint policy of all agents, in each episode a trajectory $\tau = \{\mathbf{s}_0, \mathbf{A}_0, \mathbf{s}_1, \mathbf{A}_1, \dots, \mathbf{s}_T, \mathbf{A}_T\}$ of $T + 1$ steps is sampled based on the policy and the environment. For agent k , the object is to maximize the expected accumulative reward with given policy Φ over all possible trajectories, i.e., to maximize

$$y_k(\Phi) := \mathbb{E}_\tau(R^{(k)}(\tau)), \quad (3.7)$$

where $R^{(k)}(\tau) := \sum_{t=0}^T R_t^{(k)}$ is the accumulative reward over trajectory τ . According to [86], the PG for agent k can be expressed as

$$\nabla_{\pi_k} y_k(\Phi) \approx \mathbb{E}_\tau \left(R^{(k)}(\tau) \sum_{t=0}^T \nabla \log \pi_k(\mathbf{a}_t | \mathbf{z}_t^{(k)}) \right), \quad (3.8)$$

With the PG given in (3.8), the gradient ascent methods can be used to optimize the policy of all agents.

Furthermore, by taking advantage of MPG, we can leverage the potential function of the MARL system, ϕ , to formulate our FRL problem. According to Lemma 4.2 in [86], the stationary point of the potential function of the MARL system implies Nash policies of this MARL system. We thus aim to find the stationary point of the potential function of the MARL system, i.e., to find $\pi_k, \forall k$ that implies $\nabla_{\pi_k} \phi(\Phi) = \mathbf{0}, \forall k$. On the other hand, gradient ascent methods can be used to find the stationary points without knowing the specific expression and derivatives of ϕ due to the equality of derivatives (cf. Proposition B.1 P2. in [86]), given by,

$$\nabla_{\pi_k} \phi(\Phi) = \nabla_{\pi_k} y_k(\Phi), \forall k. \quad (3.9)$$

For the sake of notation consistency, in the rest of this chapter, we use the gradient of the potential function, $\nabla_{\pi_k} \phi(\Phi)$, to present the PG.

3.3.2 Inexact ADMM

Based on the discussions in Section 3.3.1, we can formulate the PG-based FRL as an optimization problem to maximize the system potential function subject to the constraint that all agents share a common global policy model. The formulated FRL optimization problem is thus given by,

$$\begin{aligned} \max_{\Theta, \theta_c} \quad & \phi(\Phi) \\ \text{s.t.} \quad & \theta_k = \theta_c, \forall k \in \mathbb{K}, \end{aligned} \quad (3.10)$$

where θ_c is the shared global model parameters and $\Theta := (\theta_1, \theta_2, \dots, \theta_K)$ is a collection of all local trainable parameters $\{\theta_k, k \in \mathbb{K}\}$. Here θ_k denotes agent k 's policy. In this context, we use a deep neural network to represent agent k 's policy. Therefore, π_k is a function of θ_k , i.e., $\pi_k(\theta_k)$ and thus the joint policy Φ is a function of the collection of local parameters, i.e., $\Phi := \Phi(\Theta)$. Therefore, hereafter, we denote

$$\phi(\Theta) := \phi(\Phi) = \phi(\Phi(\Theta)). \quad (3.11)$$

Note that [86, 87] have proven the smoothness, i.e., the policy gradient Lipschitz continuity, of the expected value function in single-agent case and multi-agent case, respectively, which allows us to assume a gradient Lipschitz continuity on potential function ϕ , namely,

$$\|\nabla\phi(\Theta_1) - \nabla\phi(\Theta_2)\| \leq l\|\Theta_1 - \Theta_2\|, \quad (3.12)$$

where $\|\cdot\|$ is the Frobenius (or Euclidean) norm. We exploit the inexact ADMM to solve problem (3.10) in an FL manner. The augmented Lagrange function of problem (3.10) is

$$\begin{aligned} L(\Theta, \Lambda, \theta_c) &:= -\phi(\Theta) + \sum_{k \in \mathbb{K}} L_k(\theta_k, \lambda_k, \theta_c), \\ L_k(\theta_k, \lambda_k, \theta_c) &:= \lambda_k^\top (\theta_k - \theta_c) + \frac{\rho}{2} \|\theta_k - \theta_c\|^2, \end{aligned} \quad (3.13)$$

where $\rho > 0$ and $\Lambda := (\lambda_1, \lambda_2, \dots, \lambda_K)$ is the collection of all Lagrange multipliers. Then the inexact ADMM takes the framework as follows: given (Θ^0, Λ^0) ,

perform the following steps iteratively

$$\Theta^{j+1} \approx \operatorname{argmin}_{\Theta} L(\Theta, \Lambda^j, \theta_c^j), \quad (3.14a)$$

$$\lambda_k^{j+1} = \lambda_k^j + \rho(\theta_k^{j+1} - \theta_c^j), \quad k \in \mathbb{K}, \quad (3.14b)$$

$$\theta_c^{j+1} = \operatorname{argmin}_{\theta_c} \sum_{k \in \mathbb{K}} L_k(\theta_k^{j+1}, \lambda_k^{j+1}, \theta_c), \quad (3.14c)$$

for $j = 0, 1, 2, \dots$. To solve subproblem (3.14a) in the above scheme, we approximate $\phi(\Theta)$ using first-order information, i.e., its PG $\nabla\phi(\Theta) = (\nabla_{\theta_1}\phi(\Theta), \dots, \nabla_{\theta_K}\phi(\Theta))$. Denote

$$\begin{aligned} \Theta_c &:= (\theta_c, \theta_c, \dots, \theta_c), \\ g_k &:= -\nabla_{\theta_k}\phi(\Theta_c), \quad g_k^j := -\nabla_{\theta_k}\phi(\Theta_c^j). \end{aligned} \quad (3.15)$$

From the above definition, one can observe that g_k is the PG obtained by agent k from the environment. Now for each $k \in \mathbb{K}$, we can solve subproblem (3.14a) inexactly by

$$\begin{aligned} \theta_k^{j+1} &= \operatorname{argmin}_{\theta_k} \theta_k^\top g_k^j + \frac{r_k}{2} \|\theta_k - \theta_c^j\|^2 + L_k(\theta_k, \lambda_k^j, \theta_c^j) \\ &= \theta_c^j - \frac{1}{\rho + r_k} (\lambda_k^j + g_k^j), \end{aligned} \quad (3.16)$$

where $r_k \in (0, l]$ is a non-negative constant. The last step in (3.14) is the aggregation step calculated by

$$\theta_c^{j+1} = \frac{1}{K} \sum_{k \in \mathbb{K}} \mathbf{u}_k^{j+1}, \quad (3.17)$$

where $\mathbf{u}_k^{j+1}$ is the temporary variables to be aggregated from agent k , which is updated locally at agent k before the aggregation step,

$$\mathbf{u}_k^{j+1} = \theta_k^{j+1} + \frac{1}{\rho} \lambda_k^{j+1}. \quad (3.18)$$

Unlike FL, FRL requires agents to interact with the environment to obtain the training data and estimate the PG. The PG, $\nabla\phi(\Theta_c)$, is obtained from the interactions between the agents and the environment.

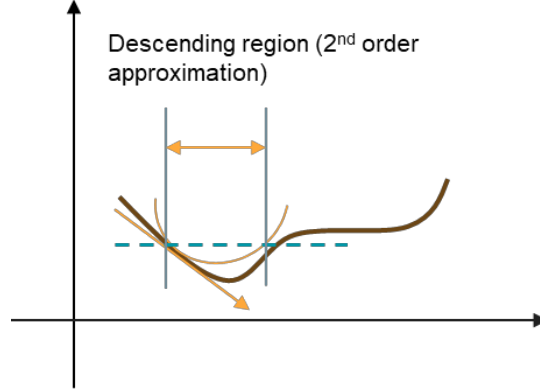


Figure 3.2: Descending objective function

3.3.3 Second Moment

To further improve the learning process, the second moment is adopted to generate an adaptive step size, which has been proven effective in widely used optimizers, e.g., Adam [53] and RMSprop. Typically, the convergence of inexact ADMM in non-convex setting is based on the monotonically decreasing augmented Lagrange function [84, 85]. As shown in Fig. 3.2, in the inexact ADMM paradigm, each subproblem is approximated by its second order approximation. There exists a “descending region” in each updating step and so does the “aggregation” step. We therefore incorporate the second moment into the “aggregation” step of the inexact ADMM algorithm to obtain a decreasing augmented Lagrange function. Specifically, in the above inexact ADMM algorithm, the Lagrange multipliers (3.14b) in fact contain the accumulated gradient information. To some extent, the update of $\mathbf{u}_k^{j+1}$ in (3.18) plays the role which is alike gradient descent. One can treat $1/\rho$ as the stepsize and $\boldsymbol{\lambda}_k^{j+1}$ as the direction. This allows us to integrate the second moment into (3.18). To proceed with that, for an initialized $1 > \beta > 0$ and $\mathbf{v}_c^0 = \mathbf{0}$, the server estimates the second moment by

$$\mathbf{v}_c^{j+1} = \beta \mathbf{v}_c^j + \frac{1}{K} \sum_{k \in \mathbb{K}} (1 - \beta) \boldsymbol{\lambda}_k^{j+1} \odot \boldsymbol{\lambda}_k^{j+1}, \quad (3.19)$$

where β is a moving average constant and $\odot$ is the Hardamard product. Then update (3.18) is modified as

$$\mathbf{u}_k^{j+1} = \boldsymbol{\theta}_k^{j+1} + \frac{1}{\rho(\sqrt{\mathbf{v}_c^{j+1}} + \epsilon)} \odot \boldsymbol{\lambda}_k^{j+1}, \quad (3.20)$$

where ϵ is a small value to prevent zero denominators. Here $1/\sqrt{\mathbf{v}} \in \mathbb{R}^N$ is a vector with the n th entry being $1/\sqrt{v[n]}$.

Algorithm 2 PASM: PG-based inexact ADMM using the second moment for RFL

- 1: **Initialize:** $\boldsymbol{\theta}_c^0 = \mathbf{0}, \mathbf{v}_c^0 = \mathbf{0}$, and $\boldsymbol{\lambda}_k = \mathbf{0}, \forall k \in \mathbb{K}$, and proper hyperparameters $J > 0, \rho > 0, \epsilon \in (0, 1), \beta \in (0, 1)$, and $r_k \in (0, l], \forall k \in \mathbb{K}$.
 - 2: **for** episode index $j = 0, 1, \dots, J$ **do**
 - 3: --Local gradient estimation--
 - 4: **for** each agent $k \in \mathbb{K}$ **do**
 - 5: Updates its local model by $\boldsymbol{\theta}_k^j = \boldsymbol{\theta}_c^j$.
 - 6: Samples a trajectory τ_k^j based on current policy $\pi_k(\cdot; \boldsymbol{\theta}_k^j)$ and calculate the PG of the current episode based on (3.8) to derive $\mathbf{g}_k^j = -\nabla_{\boldsymbol{\theta}_k} \phi(\boldsymbol{\theta}_c^j)$.
 - 7: Update $(\boldsymbol{\theta}_k^{j+1}, \boldsymbol{\lambda}_k^{j+1})$ by (3.16) and (3.14b) and send them to the sever.
 - 8: **end for**
 - 9: --Global aggregation--
 - 10: The server updates the global parameter $\boldsymbol{\theta}_c^{j+1}$ by (3.19), (3.20) and (3.17), and broadcasts it to all agents.
 - 11: **end for**
-

The resulting framework is summarized in Algorithm 2. All agents update their models and aggregate in each episode. In Step 3, the local update of the agents starts to be performed. In Step 6, each agent interacts with the environment, obtains the experience trajectory of the current episode, and calculates the PG. Then in Step 7, each agent updates the local model parameters and the Lagrange multipliers. Step 10 aggregates the knowledge of all agents at the server.

Note that compared with FedAvg-based FRL algorithms, PASM falls into the primal and dual algorithm category, namely, primal variables $\boldsymbol{\theta}$ and Lagrange multipliers (i.e., dual variables $\boldsymbol{\lambda}$) need to be updated simultaneously. This

results in some extra additions and scalar multiplications, which do not incur much computational complexity, as the main complexity of training agents comes from calculating gradients. Therefore, the PASM algorithm has similar computational complexity to the FedAvg-based FRL algorithms.

3.3.4 Convergence Analysis

Before the convergence result of the PASM algorithm, we need the following assumptions.

Assumption 3.3.1. Suppose that 1) ϕ is gradient Lipschitz continuous, i.e. (3.12), 2) $-\phi$ is bounded from below, i.e., $\phi > -\infty$, and 3) $\max_{k \in \mathbb{K}} \|\nabla_{\theta_k} \phi(\Theta_c)\|_\infty \leq 1 - \epsilon$.

We point out that the first two assumptions are commonly used to established the convergence in optimization. The third assumption can be guaranteed if we set up $|R_t^{(k)}|$ to have a small upper bound. Indeed, if we choose tiny reward $|R_t^{(k)}|$, then $R^{(k)}(\tau) = \sum_{t=0}^T R_t^{(k)}$ can be sufficiently small, resulting in small $\|\nabla_{\pi_k} y_k(\Phi)\|_\infty$ by (3.8) and so is $\|\nabla_{\theta_k} \phi(\Theta_c)\|_\infty$ by (3.9). Here $\|\mathbf{x}\|_\infty = \max_{n \in \mathbb{N}} |x[n]|$ is the infinity norm.

Lemma 3.3.1. Suppose $\max_{k \in \mathbb{K}} \|\nabla_{\theta_k} \phi(\Theta_c)\|_\infty \leq 1 - \epsilon$, then $\|\mathbf{v}_c^j\|_\infty < (1 - \epsilon)^2$ for any $j = 0, 1, 2 \dots$.

Based on the above lemma, we establish the following convergence guarantee of the PASM algorithm, where $\mathbf{Z}^{j+1} := (\Theta^{j+1}, \Lambda^{j+1}, \theta_c^j)$.

Theorem 3.3.2. Suppose Assumptions 3.3.1 hold and choose $\rho \geq 10l, \epsilon \in (0.5, 1)$. Then 1) sequence $\{L(\mathbf{Z}^j)\}$ is non-increasing and converges, 2) $\lim_{j \rightarrow \infty} \|\mathbf{Z}^{j+1} - \mathbf{Z}^j\| = 0$, and 3) the policy gradient is vanishing eventually, i.e., $\lim_{j \rightarrow \infty} \|\sum_{k \in \mathbb{K}} \mathbf{g}_k^j\| = 0$.

The conditions given in Theorem 3.3.2 are sufficient but unnecessary, which indicates that there is no need to set up parameters strictly satisfying these conditions for the algorithm to converge in the numerical experiments.

The proofs of Lemma 3.3.1 and Theorem 3.3.2 are given at the end of this chapter. The proof of Theorem 3.3.2 is based on the monotonically descending augmented Lagrange function (3.13), which is proven by showing the difference of the augmented Lagrange function values between two consecutive iterations is non-positive under the given conditions. With this monotonicity, we are then

able to show that the variable updates and the sum of the Lagrange multipliers converge to zero, which indicates that the gradients finally converge to zero.

3.4 PASM for Resource Allocation in V2X Networks

In this section, we apply PASM to the resource allocation problems in the considered NGMA V2X network. Fig. 3.3 depicts an example of PASM in a NGMA V2X network with 3 agents. In the considered FRL-for-V2X-network setting, the base station, which provides V2I link services to the vehicles, is regarded as the central training server in the FRL framework, while the local agent models are trained and deployed at each vehicle. Each vehicle updates the local model parameters and local Lagrange multipliers based on its collected local experience at the end of each episode. During the aggregation phase, each vehicle uploads its local model parameters and Lagrange multipliers to the base station via V2I links. After collecting these messages from the vehicles, the base station aggregates all the local model information to obtain a global model and then broadcasts it to all the vehicles via V2I downlinks.

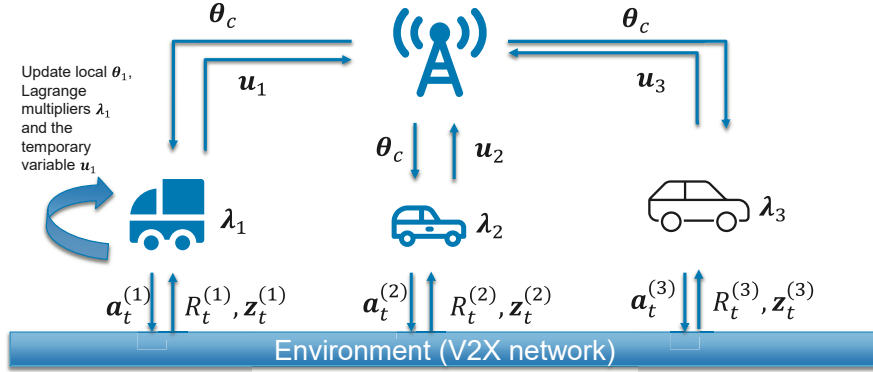


Figure 3.3: PASM workflow in a V2X network with 3 vehicles.

To apply the proposed PASM algorithm, we first need to formulate the resource allocation problem as a MARL system, i.e., to formulate the resource allocation problem in the form of the POMDP tuple, $\langle K, s_t, a_t^{(k)}, R_t^{(k)}, z_t^{(k)}, P, O \rangle$, described in Section 3.3.³ Specifically, each V2V link is treated as an agent in the RL framework. Each agent maintains a policy deep neural network to make decisions. In both considered scenarios described in Section 3.2, the agents have the same

³Here we do not need to know $\langle s_t, P, O \rangle$, as they determine the environment dynamics but are not necessary to the policy optimization process.

observable information. Thus, we use the same observation space but different reward functions for the two scenarios. Since each V2V link determines its sub-channel selection and transmits power level, the action space of each agent, denoted as k , is defined as $\{\mathbf{a}_t^{(k)} = (\ell_k, P_k^v) | \ell_k \in \mathbb{N}, P_k^v \in \mathcal{P}\}$, where ℓ_k denotes the selected sub-channel index, and $\mathcal{P}$ denotes the available discrete transmit power levels defined as $\mathcal{P} = \{23, 10, 5, -100\}$ dBm in the sequel.⁴

For practical implementation, each agent only has local observations and we do not use interference CSI as a part of the local observation. Specifically, the local observation of agent k includes the V2I link channel power gains over all sub-channels, its V2V link channel power gains over all sub-channels, its received interference power over all sub-channels in the last time slot, relative position $\mathbf{q}_k$ between the transmitter and the receiver of V2V link k , velocities vel_k^t, vel_k^r of the transmitter and the receiver of V2V link k , the remaining time budget T_k , remaining payload B_k to be transmitted and the agent index k . Formally, we have

$$\mathbf{z}_k^t = \left\{ h_{b,t}[n], g_{k,t}[n], I_{k,t-1}^v[n] + P_{n,t-1}^i \tilde{h}_{k,t-1}[n], \forall n \in \mathbb{N}, \right. \\ \left. \mathbf{q}_k, vel_k^t, vel_k^r, T_k^t, B_k^t, k \right\}. \quad (3.21)$$

Note that the local observation $\mathbf{z}_k^t$ is easy to obtain at each vehicle. We thus assume that there is no observation collection delay for the agents. Therefore, the agents are able to make real-time decisions for the V2V links.

For Scenario I, we aim to maximize the successful delivery rate of the V2V links and the sum rate of the V2I links. We thus use a common reward for all agents defined as

$$R_t = \omega \sum_{n \in \mathbb{N}} C_{n,t}^i + \sum_{k \in \mathbb{K}} D_{k,t} + \sum_{k \in \mathbb{K}} U_k. \quad (3.22)$$

In (3.22), $D_{k,t}$ is a stimulus used to encourage the agent to transmit packets if the remaining packet data size is positive,

$$D_{k,t} = \begin{cases} C_{k,t}^v & \text{if } B_k > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (3.23)$$

⁴The discrete action space here can be extended to a continuous action space easily as in the continuous PG algorithm [33].

U_k is determined after the episode ends and it indicates whether V2V link k successfully deliver all its packets in the episode,

$$U_k = \begin{cases} \Omega, & \text{if } B_k \leq 0 \text{ at the end of the episode,} \\ 0, & \text{otherwise.} \end{cases} \quad (3.24)$$

Here, Ω is a large positive constant to encourage agents to successfully deliver all their data packets.

For Scenario II, we aim to maximize the weighted achievable sum rate of all V2I links and V2V links. Therefore, we directly use the weighted achievable sum rate as the reward as follows,

$$R_t = \omega \sum_{n \in \mathbb{N}} C_{n,t}^i + (1 - \omega) \sum_{k \in \mathbb{K}} C_{k,t}^v. \quad (3.25)$$

Table 3.1: Simulation Parameters [1]

Parameter	Value
Carrier frequency	2GHz
Bandwidth	4MHz
BS antenna height	25m
BS antenna gain	8dBi
BS receiver noise figure	5dB
Vehicle antenna height	5m
Vehicle antenna gain	3dBi
Vehicle receiver noise figure	9dB
Vehicle speed	10-15m/s
Vehicle drop and mobility model	Urban case of A.1.2 in [1]
V2I transmit power $\{P_{n,t}^i\}$	23dBm
V2V transmit power $\{P_{k,t}^v\}$	[23,10,5,-100]dBm
Noise power σ^2	-114dBm
V2V package delivery time	100ms
V2V link packet size	1060 bytes
Channel fast-fading updating time	1ms

3.5 Simulation Results

In this section, we demonstrate the performance of PASM for resource allocation in a NGMA V2X network through computer simulation. Our simulation environment follows the urban case in Annex A of [1]. The channel models we use in the simulation is the same as in [28]. We consider N V2I links and K V2V links in the V2X network, where the V2V links are formed by each vehicle and its neighbors. We test the performance of the proposed algorithm under different (N, K) pairs. The simulation parameters are summarized in Table 3.1.

The policy deep neural network for each V2V link consists of three fully connected hidden layers with 500, 250, and 120 neurons, respectively. The rectified linear unit (ReLU) is used as the activation function in the input and three hidden layers. The output layer is connected to a softmax function so that the final output is a probability distribution of the action. Each training episode consists of 100 time slots. For Scenario I, we set the V2I link sum rate weight $\omega = 0.01$ and the V2V successful delivery reward $\Omega = 0.5$, as the V2V package deliv-

ery rate is more important. Hyper-parameters of PASM are set as $\rho = 1000$, $\epsilon = 10^{-2}$, $\alpha = 1$, $\beta = 0.999$, and $r_k = 0.1\rho, \forall k$. For Scenario II, we set $\omega = 0.1$, $\rho = 500$, and other parameters the same as those in Scenario I.

We compare our PASM algorithm with the independent PG algorithm and the FedAvg-based FRLPG algorithm [40]. Both algorithms employ Adam [53] optimizer to update the local policy deep neural networks and share the same neural network structure with the PASM algorithm. The learning rate of the PG algorithm and the FRLPG algorithm is set as 10^{-4} and 10^{-3} , respectively⁵. We also use two additional baselines. The first one is the random resource allocation scheme, which randomly chooses the sub-channel and transmits the power level. Typically, we have power control in practical systems, but this random baseline is to show a lower bound of the system performance. The second one is the centralized maxV2V proposed in [28], which provides an upper bound of Scenario I by an exhaustive search scheme.⁶

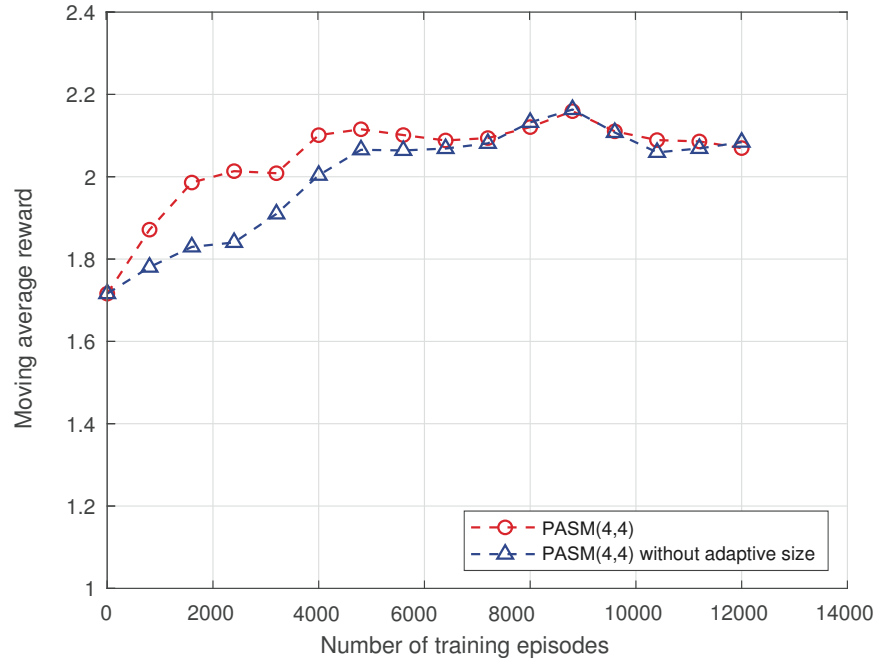
3.5.1 Effect of Second Moments

We first verify the effectiveness of the adaptive stepsize of our algorithm. We compare the moving average reward during the training phase of our proposed algorithms with and without adaptive stepsize in both scenarios with $(N, K) = (4, 4)$.⁷ As shown in Fig. 3.4, the introduction of adaptive stepsize accelerates the convergence speed in Scenario I, while it improves the performance in Scenario II. This is similar to the case of the Adam and RMSprop optimizers, whose effectiveness has been approved in many works in both AI and communication communities. Therefore, we only compare our PASM algorithm with adaptive stepsize with other baselines in the following simulations.

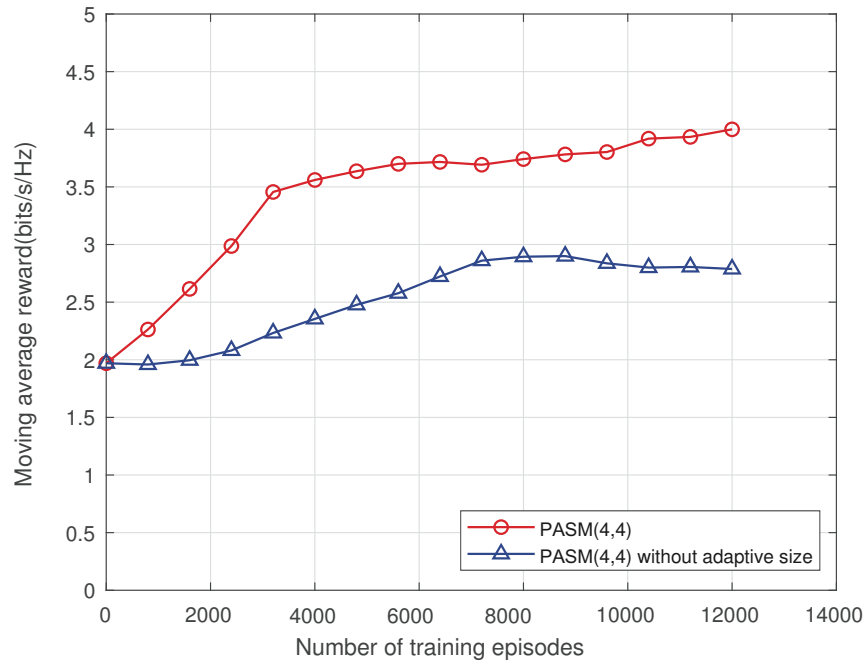
⁵We set a slower learning rate for the PG algorithm because a slightly larger one (e.g., 10^{-3}) makes the PG algorithm fail to learn a good policy. In addition, we find that the Adam optimizer and RMSprop optimizer have very similar performance in the FRLPG and the Independent PG algorithms. Therefore, we only show the results of Adam optimizer in the simulation.

⁶We only apply this baseline to the case of $(N, K) = (4, 4)$, as the brute-force method has an extremely high complexity when the number of agents increases.

⁷The moving average reward is calculated over 2000 episodes to reduce the performance fluctuation caused by the vehicle mobility and the channel fading randomness.



(a) Scenario I



(b) Scenario II

Figure 3.4: Moving average reward during the training phase in the two scenarios.

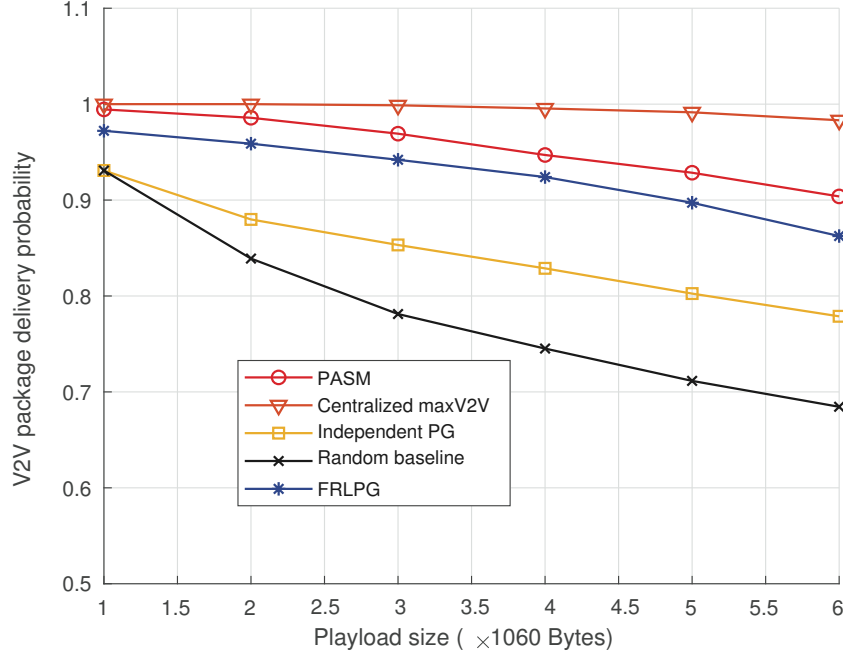


Figure 3.5: V2V link packet delivery rate versus V2V link payload size

3.5.2 Scenario I

Next, we show the experiment results in Scenario I. We train the agents for 12000 episodes with $(N, K) = (4, 4)$ and a payload size of 2120 Bytes and then test them in another testing environment. To test the robustness of the proposed algorithms to the V2V link payload size, we test the trained model in the environment with $(N, K) = (4, 4)$ and increasing V2V payload size. Fig. 3.5 plots the V2V package delivery rate versus the V2V payload size of the considered algorithms. With the payload size increasing, the V2V package delivery rate of all considered algorithms decreases. In addition, with any payload size, the agent models trained by our proposed PASM algorithm always have the best testing performance among all algorithms except for the brute-force method, Centralized maxV2V. The FRLPG algorithm has better performance than that of the independent PG algorithm. When a vehicle moves for a long distance, the environment an agent observes changes significantly. With FL manner, the agent is able to learn the environment from other agents' knowledge, but independent learners cannot. Therefore, the Independent PG algo-

Table 3.2: Testing performance (V2V link packet delivery rate) of Scenario I with different (N, K) pairs

(N, K)	PASM	FRLPG	Independent PG	Random baseline
(4, 4)	0.9858	0.9588	0.8798	0.839
(6, 12)	0.9257	0.9222	0.8865	0.7797
(8, 24)	0.8979	0.8955	0.8295	0.8065
(6, 18)	0.7949	0.7439	0.6893	0.6281

rithm has relatively bad performance among all the considered algorithms.

To further test our proposed algorithm, we use the algorithms to train agents in the environments with different (N, K) pairs, which is determined by the environment, and then test these trained models in the corresponding environments. The (N, K) pair controls the level of training difficulty, as it determines the V2V link density, K/N , the number of agents in the environment, K , and the freedom degree of resource allocation, i.e., the number of available sub-channels, N . The performance of different algorithms under different (N, K) pairs is summarized in Tab. 3.2.⁸ We observe that for all algorithms, the larger link K/N , the worse performance they had, because a higher link density means more crowded subchannels on average, leading to worse transmission. The proposed PASM algorithm always has the best performance. In addition, as we have explained above, the FRLPG algorithm has better performance than the Independent PG algorithm due to the FL manner among the agents. These results validate the efficiency of our proposed PASM algorithm in Scenario I.

3.5.3 Scenario II

In the sequel, we evaluate our proposed algorithm in Scenario II. Fig. 3.6 shows the moving average reward of the PASM algorithm and the FRLPG algorithm versus the number of training episodes in the training phase with different (N, K) pairs. These results are obtained by training the agents using the corresponding algorithms for 15000 episodes and testing in another testing environment. We can see that when $(N, K) = (4, 4)$, the PASM algorithm and

⁸We omit the Centralized maxV2V algorithm in this experiment due to its extremely high computational complexity.

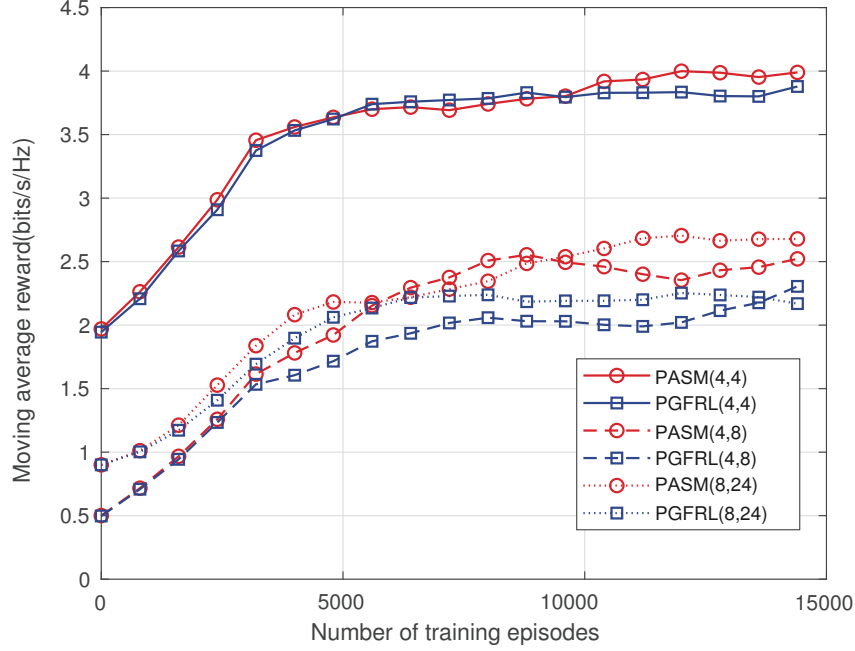


Figure 3.6: Moving average reward of two algorithms during the training phase.

the FRLPG algorithm have similar performance. However, when the number of agents and the V2V link density increase, the PASM algorithm has a significant performance gain over the FRLPG algorithm. This is because when $(N, K) = (4, 4)$, the weighted sum rate problem is relatively simple compared with the case of $(4, 8)$ and $(8, 24)$, where the performance gain is significant. The ADMM-based algorithm has better performance than the Fedavg-based algorithm when the problem is highly non-convex. The relative training performance gain of the PASM algorithm over the FRLPG algorithm even reaches around 20% when $(N, K) = (8, 24)$.

Then we evaluate the corresponding testing performance of the considered algorithms in the testing environment. The testing performance of the considered algorithms under different (N, K) pairs is summarized in Tab. 3.3. We can see that the Independent PG algorithm outperforms the FRLPG algorithm when $(N, K) = (4, 8)$. This is because, in Scenario II, agents only optimize the system weighted sum-throughput in the current time slot, while in Scenario I, agents

Table 3.3: Testing performance (weighted average rate of all links) of Scenario II with different (N, K) pairs

(N,K)	PASM	FRLPG	Independent PG	Random baseline
(4, 4)	4.0 Mbps	3.71 Mbps	3.17 Mbps	1.77 Mbps
(4, 8)	2.59 Mbps	2.15 Mbps	2.41 Mbps	0.91 Mbps
(8, 24)	2.72 Mbps	2.16 Mbps	2.54 Mbps	1.27 Mbps

have to optimize the V2V link package delivery rate, which is obtained after a sequence of decisions. Therefore, it is easier for independent learners to learn a good policy in Scenario II than in Scenario I. In addition, the Independent PG algorithm allows each agent to keep its own policy instead of a shared policy, which results in a larger degree of freedom. The easier problem setting and the larger degree of freedom together contribute to the performance gain of the Independent PG algorithm over the FRLPG algorithm. However, due to the induction of Lagrange multipliers, the PASM algorithm can better tackle the non-convexity of the problem and thus always has the best performance under all environmental conditions, which is resulted from the fact that the “global policy” is able to distinguish different agents from different input states if the learning algorithm is powerful enough to tackle the complexity of the learning problem. These results validate the efficiency of our proposed algorithm in Scenario II.

3.6 Conclusion

We developed a PASM learning algorithm based on the framework of FRL. The algorithm was implemented by inexact ADMM and benefited from two critical techniques: the usage of the PG and the second moment of the Lagrange multipliers. The former enabled the agents to gradually improve their policies while the latter enabled an adaptive learning rate to accelerate and to improve the training. We implemented PASM in a NGMA V2X network to train the agents in an FL manner to optimize the V2V package delivery rate and the system weighted sum-throughput. The numerical experiment has shown that the FL manner does improve the learning process of the agents by sharing the learned knowledge among the agents and that our proposed algorithm can improve the performance of the resource allocation problem in the considered NGMA V2X network.

3.7 Proof of Lemma 3.3.1

Proof. Let $\alpha := r_k/(\rho + r_k)$. It follows from (3.14b) and (3.16) that⁹

$$\begin{aligned}
\lambda_k^{j+1} &= (1 - \alpha)g_k^j + \alpha\lambda_k^j \\
&= (1 - \alpha)g_k^j + (1 - \alpha)\alpha g_k^{j-1} + \alpha^2\lambda_k^{j-1} \\
&= \dots \\
&= (1 - \alpha) \sum_{t=0}^j \alpha^t g_k^{j-t} + \alpha^{j+1}\lambda_k^0 \\
&= (1 - \alpha) \sum_{t=0}^j \alpha^t g_k^{j-t},
\end{aligned}$$

the last equality is from $\lambda_k^0 = 0$, which results in

$$\begin{aligned}
\|\lambda_k^{j+1}\|_\infty &\leq (1 - \alpha) \sum_{t=0}^j \alpha^t \|g_k^{j-t}\|_\infty \\
&\leq (1 - \alpha) \sum_{t=0}^j \alpha^t \|g_k^{j-t}\|_\infty \\
&\leq (1 - \alpha)(1 - \epsilon) \sum_{t=0}^j \alpha^t \\
&= (1 - \alpha^j)(1 - \epsilon)^2 \leq (1 - \epsilon).
\end{aligned}$$

Using the above condition $\mathbf{v}_c^0 = 0$, and (3.19) we have

$$\begin{aligned}
\|\mathbf{v}_c^1\|_\infty &\leq \|\beta \mathbf{v}_c^0\|_\infty + \frac{1}{K} \sum_{k \in \mathbb{K}} (1 - \beta) \|\lambda_k^1 \odot \lambda_k^1\|_\infty \\
&\leq (1 - \beta)(1 - \epsilon)^2 < (1 - \epsilon)^2,
\end{aligned}$$

⁹For better readability, we omit the indices of the unreferenced equations in the appendix.

which further leads to

$$\begin{aligned}\|\mathbf{v}_c^2\|_\infty &\leq \|\beta \mathbf{v}_c^1\|_\infty + \frac{1}{K} \sum_{k \in \mathbb{K}} (1 - \beta) \|\boldsymbol{\lambda}_k^2 \odot \boldsymbol{\lambda}_k^2\|_\infty \\ &\leq \beta(1 - \epsilon)^2 + (1 - \beta)(1 - \epsilon)^2 = (1 - \epsilon)^2.\end{aligned}$$

By deduction, we can show that $\|\mathbf{v}_c^j\|_\infty \leq (1 - \epsilon)^2$ for any $j = 1, 2, 3, \dots$. $\square$

3.8 Proof of Theorem 3.3.2

For convenience, we define some updating gaps as follows,

$$\begin{aligned}\Delta\theta_k^{j+1} &= \theta_k^{j+1} - \theta_k^j, \quad \Delta\lambda_k^{j+1} = \lambda_k^{j+1} - \lambda_k^j, \\ \Delta\theta_c^{j+1} &= \theta_c^{j+1} - \theta_c^j, \quad \Delta\theta_{kc}^{j+1} = \theta_k^{j+1} - \theta_c^j, \\ \Delta\tilde{\theta}_{kc}^{j+1} &= \theta_k^{j+1} - \theta_c^{j+1}\end{aligned}\tag{3.26}$$

For notational convenience, we write

$$\sum := \sum_{k \in \mathbb{K}}.$$

Based on the gradient Lipschitz continuity, we have the following descent inequality for a gradient-Lipschitz-continuous function $f(\cdot)$,

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \nabla f(\mathbf{w})^\top (\mathbf{x} - \mathbf{y}) + \frac{l}{2} \|\mathbf{x} - \mathbf{y}\|^2,\tag{3.27}$$

where $\mathbf{w}$ can be $\mathbf{x}$ or $\mathbf{y}$, and $l > 0$ is the Lipschitz constant. For $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K)$, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$, if $f(\mathbf{X})$ is Lipschitz continuous, we have

$$\begin{aligned}f(\mathbf{X}) - f(\mathbf{Y}) \\ \leq \sum \left(\nabla_{\mathbf{w}_k} f(\mathbf{W})^\top (\mathbf{x}_k - \mathbf{y}_k) + \frac{l}{2} \|\mathbf{x}_k - \mathbf{y}_k\|^2 \right),\end{aligned}\tag{3.28}$$

where $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K)$ can be $\mathbf{X}$ or $\mathbf{Y}$. Similarly, let $\Theta^i := (\theta_1^i, \theta_2^i, \dots, \theta_K^i)$, $i = 1, 2$, by (3.12) we have,

$$\sum \|\nabla_{\theta_k} \phi(\Theta^1) - \nabla_{\theta_k} \phi(\Theta^2)\|^2 \leq l^2 \sum \|\theta_k^1 - \theta_k^2\|^2.\tag{3.29}$$

Proof. We prove the first part of Theorem 3.3.2 by analyzing the gap between

two consecutive updates. We rewrite the gap as a sum of three parts as follows,

$$\begin{aligned}
L(\mathbf{Z}^{j+1}) - L(\mathbf{Z}^j) &= e_1^{j+1} + e_2^{j+1} + e_3^j, \\
e_1^{j+1} &:= L(\Theta^{j+1}, \Lambda^j, \theta_c^j) - L(\Theta^j, \Lambda^j, \theta_c^j), \\
e_2^{j+1} &:= L(\mathbf{Z}^{j+1}) - L(\Theta^{j+1}, \Lambda^j, \theta_c^j), \\
e_3^j &:= L(\Theta^j, \Lambda^j, \theta_c^j) - L(\mathbf{Z}^j).
\end{aligned}$$

1) We first derive the upper bound of e_1^{j+1} , which indicates the updating impact of $\theta_k, \forall k$. Based on the gradient Lipschitz continuity of ϕ , we have,

$$\begin{aligned}
e_1^{j+1} &= -\phi(\Theta^{j+1}) + \phi(\Theta^j) \\
&+ \sum \left((\lambda_k^j)^\top \Delta \theta_k^{j+1} + \frac{\rho}{2} (\|\Delta \theta_{kc}^{j+1}\|^2 - \|\Delta \tilde{\theta}_{kc}^j\|^2) \right) \\
&\leq \sum \left((\tilde{g}_k^{j+1} + \lambda_k^j)^\top \Delta \theta_k^{j+1} + \frac{l}{2} \|\Delta \theta_k^{j+1}\|^2 \right. \\
&\quad \left. + \frac{\rho}{2} (\|\Delta \theta_{kc}^{j+1}\|^2 - \|\Delta \tilde{\theta}_{kc}^j\|^2) \right), \tag{3.30}
\end{aligned}$$

where $\tilde{g}_k^{j+1} := -\nabla_{\theta_k} \phi(\Theta^{j+1})$ and the inequality is from (3.28). Then, we have,

$$\begin{aligned}
p_k^{j+1} &:= (\tilde{g}_k^{j+1} + \lambda_k^j)^\top \Delta \theta_k^{j+1} + \frac{l}{2} \|\Delta \theta_k^{j+1}\|^2 \\
&+ \frac{\rho}{2} (\|\Delta \theta_{kc}^{j+1}\|^2 - \|\theta_k^j - \theta_k^{j+1} + \theta_k^{j+1} - \theta_c^j\|^2) \\
&= \left(\tilde{g}_k^{j+1} + \lambda_k^j + \rho \Delta \theta_{kc}^{j+1} \right)^\top \Delta \theta_k^{j+1} + \frac{l - \rho}{2} \|\Delta \theta_k^{j+1}\|^2 \\
&\stackrel{(3.16)}{=} \left(\tilde{g}_k^{j+1} - g_k^j - r_k \Delta \theta_{kc}^{j+1} \right)^\top \Delta \theta_k^{j+1} + \frac{l - \rho}{2} \|\Delta \theta_k^{j+1}\|^2.
\end{aligned}$$

Then using two facts $2\mathbf{a}^\top \mathbf{b} \leq t\|\mathbf{a}\|^2 + (1/t)\|\mathbf{b}\|^2$ for any $\mathbf{a}, \mathbf{b}$ and $t > 0$ and

$r_k \in (0, \ell]$, we have

$$\begin{aligned}
p_k^{j+1} &\leq \frac{1}{2l} \|\tilde{\mathbf{g}}_k^{j+1} - \mathbf{g}_k^j - r_k \Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \\
&\leq \frac{1}{l} \|\tilde{\mathbf{g}}_k^{j+1} - \mathbf{g}_k^j\|^2 + \frac{r_k^2}{l} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \\
&\leq \frac{1}{l} \|\tilde{\mathbf{g}}_k^{j+1} - \mathbf{g}_k^j\|^2 + \frac{l^2}{l} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \\
&= \frac{1}{l} \|\tilde{\mathbf{g}}_k^{j+1} - \mathbf{g}_k^j\|^2 + l \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2.
\end{aligned}$$

Therefore, from (3.30) and the above condition we derive

$$\begin{aligned}
e_1^{j+1} &\leq \sum p_k^{j+1} \\
&\leq \sum \left(\frac{1}{l} \|\tilde{\mathbf{g}}_k^{j+1} - \mathbf{g}_k^j\|^2 + l \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \right) \\
&\stackrel{(3.29)}{\leq} \sum \left(2l \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \right) \\
&\stackrel{(3.14b)}{=} \sum \left(\frac{2l}{\rho^2} \|\Delta \boldsymbol{\lambda}_k^{j+1}\|^2 + \frac{2l - \rho}{2} \|\Delta \boldsymbol{\theta}_k^{j+1}\|^2 \right). \tag{3.31}
\end{aligned}$$

2) For e_2^{j+1} regarding the impact of updating $\boldsymbol{\lambda}_k, \forall k$, we have

$$e_2^{j+1} = \sum (\Delta \boldsymbol{\theta}_{kc}^{j+1})^\top \Delta \boldsymbol{\lambda}_k^{j+1} \stackrel{(3.14b)}{=} \frac{1}{\rho} \sum \|\Delta \boldsymbol{\lambda}_k^{j+1}\|^2. \tag{3.32}$$

3) We next derive the upper bound for e_3^j about $\boldsymbol{\theta}_c$. For simplicity, let $\mathbf{w}_c^j := \sqrt{\mathbf{v}_c^j} + \epsilon$. By Lemma 3.3.1, we have $\|\mathbf{w}_c^j\|_\infty \in [\epsilon, 1)$. Based on the updating steps (3.17) and (3.20), we have the following equation,

$$\sum \left(\Delta \tilde{\boldsymbol{\theta}}_{kc}^j + \frac{1}{\rho \mathbf{w}_c^j} \odot \boldsymbol{\lambda}_k^j \right) = \mathbf{0},$$

which immediately results in

$$\sum -\boldsymbol{\lambda}_k^j = \sum \rho \mathbf{w}_c^j \odot \Delta \tilde{\boldsymbol{\theta}}_{kc}^j. \tag{3.33}$$

It follows from (3.13) and the above condition that

$$\begin{aligned} e_3^j &:= \sum \left(-(\lambda_k^j)^\top \Delta \theta_c^j + \frac{\rho}{2} (\|\Delta \tilde{\theta}_{kc}^j\|^2 - \|\Delta \theta_{kc}^j\|^2) \right) \\ &= \sum \left(\rho(\mathbf{w}_c^j \odot \Delta \tilde{\theta}_{kc}^j)^\top \Delta \theta_c^j + \frac{\rho}{2} (\|\Delta \tilde{\theta}_{kc}^j\|^2 - \|\Delta \theta_{kc}^j\|^2) \right). \end{aligned} \quad (3.34)$$

The following part aims to estimate the right-hand side of (3.34).

$$\begin{aligned} q_k^j &:= \rho(\mathbf{w}_c^j \odot \Delta \tilde{\theta}_{kc}^j)^\top \Delta \theta_c^j + \frac{\rho}{2} (\|\Delta \tilde{\theta}_{kc}^j\|^2 - \|\Delta \theta_{kc}^j\|^2) \\ &= \rho(\mathbf{w}_c^j \odot \Delta \tilde{\theta}_{kc}^j)^\top \Delta \theta_c^j - \rho(\Delta \tilde{\theta}_{kc}^j)^\top \Delta \theta_c^j - \frac{\rho}{2} \|\Delta \theta_c^j\|^2 \\ &= \frac{\rho}{2} \left\| \sqrt{\mathbf{w}_c^j} \odot (\Delta \tilde{\theta}_{kc}^j + \Delta \theta_c^j) \right\|^2 - \frac{\rho}{2} \|\Delta \theta_c^j\|^2 \\ &\quad - \frac{\rho}{2} \left\| \sqrt{\mathbf{w}_c^j} \odot \Delta \tilde{\theta}_{kc}^j \right\|^2 - \frac{\rho}{2} \left\| \sqrt{\mathbf{w}_c^j} \odot \Delta \theta_c^j \right\|^2 \\ &\quad - \frac{\rho}{2} \|\Delta \tilde{\theta}_{kc}^j + \Delta \theta_c^j\|^2 + \frac{\rho}{2} \|\Delta \tilde{\theta}_{kc}^j\|^2 + \frac{\rho}{2} \|\Delta \theta_c^j\|^2 \\ &\leq \frac{\rho(\|\mathbf{w}_c^j\|_\infty - 1)}{2} \|\Delta \tilde{\theta}_{kc}^j + \Delta \theta_c^j\|^2 - \frac{\rho}{2} \|\Delta \theta_c^j\|^2 \\ &\quad + \frac{\rho(1 - \epsilon)}{2} \|\Delta \tilde{\theta}_{kc}^j\|^2 + \frac{\rho(1 - \epsilon)}{2} \|\Delta \theta_c^j\|^2 \\ &\leq \frac{\rho(1 - \epsilon)}{2} \|\Delta \tilde{\theta}_{kc}^j\|^2 - \frac{\epsilon\rho}{2} \|\Delta \theta_c^j\|^2 \\ &\leq \frac{\rho}{4} \|\Delta \tilde{\theta}_{kc}^j\|^2 - \frac{\rho}{4} \|\Delta \theta_c^j\|^2. \end{aligned}$$

where the last three inequalities used $\|\mathbf{w}_c^j\|_\infty \in [\epsilon, 1)$ and $\epsilon \in [1/2, 1]$. By (3.14b), we have $\Delta \tilde{\theta}_{kc}^j = \frac{1}{\rho} \Delta \lambda_k^{j+1} - \Delta \theta_k^{j+1}$ and hence

$$\begin{aligned} q_k^j &\leq \frac{\rho}{4} \left\| \frac{1}{\rho} \Delta \lambda_k^{j+1} - \Delta \theta_k^{j+1} \right\|^2 - \frac{\rho}{4} \|\Delta \theta_c^j\|^2 \\ &\leq \frac{1}{\rho} \|\Delta \lambda_k^{j+1}\|^2 + \frac{\rho}{3} \|\Delta \theta_k^{j+1}\|^2 - \frac{\rho}{4} \|\Delta \theta_c^j\|^2, \end{aligned}$$

where the second inequality is from $\|\mathbf{a} + \mathbf{b}\|^2 \leq (1 + t)\|\mathbf{a}\|^2 + (1 + 1/t)\|\mathbf{b}\|^2$ for

any a, b and $t > 0$, which together with (3.34) derives

$$\begin{aligned} e_3^j &= \sum q_k^j \\ &\leq \sum \left(\frac{1}{\rho} \|\Delta \lambda_k^{j+1}\|^2 + \frac{\rho}{3} \|\Delta \theta_k^{j+1}\|^2 - \frac{\rho}{4} \|\Delta \theta_c^j\|^2 \right). \end{aligned}$$

4) It follows from (3.16) and (3.14b) that $\lambda_k^{j+1} = r_k(\theta_c^j - \theta_k^{j+1}) - g_k^j$. Then, we have,

$$\Delta \lambda_k^{j+1} = r_k(\Delta \theta_c^j - \Delta \theta_k^{j+1}) + g_k^{j-1} - g_k^j, \quad (3.35)$$

which by $r_k \in (0, l]$ and (3.29) suffices to

$$\begin{aligned} &\sum \|\Delta \lambda_k^{j+1}\|^2 \\ &\leq \sum (3l^2 \|\Delta \theta_k^{j+1}\|^2 + 3l^2 \|\Delta \theta_c^j\|^2 + 3\|g_k^j - g_k^{j-1}\|^2) \end{aligned} \quad (3.36)$$

$$\leq \sum (3l^2 \|\Delta \theta_k^{j+1}\|^2 + 6l^2 \|\Delta \theta_c^j\|^2). \quad (3.37)$$

5) We sum e_1^{j+1} , e_2^{j+1} and e_3^j and use (3.37) to obtain

$$\begin{aligned} L(\mathbf{Z}^{j+1}) - L(\mathbf{Z}^j) &= e_1^{j+1} + e_2^{j+1} + e_3^j \\ &= \sum \left(\frac{6l - \rho}{6} \|\Delta \theta_k^{j+1}\|^2 - \frac{\rho}{4} \|\Delta \theta_c^j\|^2 + \left(\frac{2l}{\rho^2} + \frac{2}{\rho} \right) \|\Delta \lambda_k^{j+1}\|^2 \right) \\ &\leq \sum \left(\frac{6l^3}{\rho^2} + \frac{6l^2}{\rho} + \frac{6l - \rho}{6} \right) \|\Delta \theta_k^{j+1}\|^2 \\ &\quad + \sum \left(\frac{12l^3}{\rho^2} + \frac{12l^2}{\rho} - \frac{\rho}{4} \right) \|\Delta \theta_c^j\|^2 \\ &\leq \sum \left(-\frac{61l}{150} \|\Delta \theta_k^{j+1}\|^2 - \frac{9l}{50} \|\Delta \theta_c^j\|^2 \right). \end{aligned} \quad (3.38)$$

where the last inequality is due to $\rho \geq 10l$. From (3.38), we can conclude that sequence $\{L(\mathbf{Z}^j)\}$ is a non-increasing.

6) Based on the descent inequality (3.28), we have,

$$\begin{aligned}
& \phi(\Theta^{j+1}) - \phi(\Theta_c^j) \\
& \leq \sum \left((\mathbf{g}_k^j)^\top \Delta \boldsymbol{\theta}_{kc}^{j+1} + \frac{l}{2} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 \right) \\
& = \sum \left((\boldsymbol{\lambda}_k^{j+1} + r_k \Delta \boldsymbol{\theta}_{kc}^{j+1})^\top \Delta \boldsymbol{\theta}_{kc}^{j+1} + \frac{l}{2} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 \right) \\
& \leq \sum \left((\boldsymbol{\lambda}_k^{j+1})^\top \Delta \boldsymbol{\theta}_{kc}^{j+1} + \frac{3l}{2} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 \right),
\end{aligned}$$

where the equality is due to (3.14b) and (3.16). This results in

$$\begin{aligned}
& L(\mathbf{Z}^{j+1}) \\
& = -\phi(\Theta^{j+1}) + \sum \left((\boldsymbol{\lambda}_k^{j+1})^\top \Delta \boldsymbol{\theta}_{kc}^{j+1} + \frac{\rho}{2} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2 \right) \\
& \geq -\phi(\Theta_c^j) + \frac{\rho - 3l}{2} \sum \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|^2.
\end{aligned}$$

Therefore, we have $L(\mathbf{Z}^{j+1}) > -\infty$ for due to $\rho \geq 10l$. This together with the non-increasing property of $\{L(\mathbf{Z}^j)\}$ shows that $\{L(\mathbf{Z}^j)\}$ is convergent. Then taking the limit of the both sides of (3.38) immediately leads to $\lim_{j \rightarrow \infty} \|\Delta \boldsymbol{\theta}_k^j\| = 0$ and $\lim_{j \rightarrow \infty} \|\boldsymbol{\theta}_c^j\| = 0$, which by (3.37) contributes to $\lim_{j \rightarrow \infty} \|\Delta \boldsymbol{\lambda}_k^{j+1}\| = 0$ and $\lim_{j \rightarrow \infty} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\| = 0$ by (3.14b).

7) Based on (3.14b) and (3.16), we have,

$$\begin{aligned}
\left\| \sum \mathbf{g}_k^j \right\| &= \left\| \sum (\boldsymbol{\lambda}_k^{j+1} + r_k \Delta \boldsymbol{\theta}_{kc}^{j+1}) \right\| \\
&\leq \left\| \sum \boldsymbol{\lambda}_k^{j+1} \right\| + \sum l \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\|.
\end{aligned} \tag{3.39}$$

For the first term in (3.39), by (3.33), we can conclude that

$$\begin{aligned}
\left\| \sum \lambda_k^{j+1} \right\| &= \left\| \sum \rho \mathbf{w}_c^{j+1} \odot (\boldsymbol{\theta}_k^{j+1} - \boldsymbol{\theta}_c^{j+1}) \right\| \\
&= \left\| \sum \rho \mathbf{w}_c^{j+1} \odot (\Delta \boldsymbol{\theta}_{kc}^{j+1} - \Delta \boldsymbol{\theta}_c^{j+1}) \right\| \\
&\leq \rho \sum \|\mathbf{w}_c^{j+1}\|_\infty \|\Delta \boldsymbol{\theta}_{kc}^{j+1} - \Delta \boldsymbol{\theta}_c^{j+1}\| \\
&\leq \rho \sum \|\mathbf{w}_c^{j+1}\|_\infty \left(\|\Delta \boldsymbol{\theta}_{kc}^{j+1}\| + \|\Delta \boldsymbol{\theta}_c^{j+1}\| \right) \\
&\rightarrow 0.
\end{aligned}$$

The above condition, $\lim_{j \rightarrow \infty} \|\Delta \boldsymbol{\theta}_{kc}^{j+1}\| = 0$, and (3.39) show $\lim_{j \rightarrow \infty} \|\sum_k \mathbf{g}_k^j\| = 0$. $\square$

Chapter 4

Rescale-Invariant Federated Reinforcement Learning for Resource Allocation in V2X Networks

In this chapter, we improved the Federated Reinforcement Learning (FRL) by model alignment. We proposed a Rescale-Invariant FRL (RIFRL) framework to train the Policy Gradient (PG) based Multi-Agent Deep Reinforcement Learning (MADRL) systems, which can be applied to Federated Learning (FL) problems targeting at ReLU-activated Deep Neural Networks (DNNs). Specifically, we exploited the node-wise invariant property of ReLU-activated DNNs, by which we can rescale the weights of the DNN without changing the input-output map of the DNN. We rescaled the DNNs to align local models across all clients to improve the aggregation efficiency without introducing additional communication overhead. The RIFRL algorithm was then applied to the resource allocation problem in a V2X network. Simulation results show that the RIFRL has faster convergence speed and better convergent performance than the baseline algorithms.

4.1 Introduction

As we have already discussed in Chapters 1 and 3, Federated Reinforcement Learning (FRL) is a promising technique for the resource allocation problems in vehicle-to-everything (V2X) networks. Despite the great potential of FRL, many FL-based algorithms, such as FedAvg [40], FedProx [43], and FedGiA [85], face performance degradation when local devices hold significantly different data, including non-independent-and-identically-distributed (non-iid) data or inhomogeneous data. Moreover, these algorithms pay much attention to the optimization model of a learning problem and thus do not make full use of the structures of neural networks. For example, ReLU-activated neural networks exhibit the node-wise invariance property [88]. By harnessing this feature, we can rescale the models for local devices to mitigate the influence of data differences and consequently enhance learning performance.

4.1.1 Specific Contributions of this Chapter

The major contribution of this chapter lies in the development of the Rescale-Invariant FRL (RIFRL) algorithm for resource allocation in V2X networks. The algorithm fully exploits the node-wise invariance property of ReLU-activated neural networks, thereby remarkably improving the learning process of the FRL system under various configurations. Furthermore, the proposed Backward Rescale-Invariant Operation (BRIO) could prove advantageous in enhancing the performance of many distributed algorithms when addressing challenges posed by data discrepancy.

4.2 System Model and Problem Formulation

This chapter considers the scenario I of Chapter 3. For readability, we rephrase the system model in this section. The resource allocation for an orthogonal frequency-division multiple access based single-antenna V2X network is introduced as follows.

In the V2X system, a roadside unit provides vehicle-to-infrastructure (V2I) link services to N vehicles in the system and there are K vehicle-to-vehicle (V2V) links connecting neighbour vehicles. Assume that each V2I link is allocated an orthogonal sub-channel, as it provides high-data-rate services to each vehicle. Therefore, the number of orthogonal sub-channels is the same as the number of the V2I links. The V2V links are enabled by device-to-device communication and reuse the uplink sub-channels of the V2I links to improve the spectral efficiency. By denoting the sets of V2I and V2V links as $\mathbb{N} = \{1, 2, \dots, N\}$ and $\mathbb{K} = \{1, 2, \dots, K\}$, respectively, V2I link n is allocated with sub-channel n .

At time slot t , V2I link n receives the interference signals from V2V links. Therefore, the SINR for V2I link n can be expressed as follows,

$$\gamma_n^{i,t} = \frac{|h_n^t[n]|^2 P_n^{i,t}}{\sigma^2 + \sum_{k \in \mathbb{K}} \rho_k^t[n] |\tilde{h}_k^t[n]|^2 P_k^{v,t}}, \quad (4.1)$$

where $h_k^t[n]$ denotes the channel gain of V2I link n and $\tilde{h}_k^t[n]$ represents the interference channel gain between the transmitter of V2V link k and the road side unit in sub-channel n at time slot t . $P_n^{i,t}$ and $P_k^{v,t}$ are the transmit power of V2I link n and that of V2V link k at time slot t , respectively. σ^2 is the variance of the additive Gaussian white noise. The binary variable, $\rho_k^t[n]$, represent whether sub-channel n is allocated to V2V link k for transmission, i.e., $\rho_k^t[n] = 1$ if allocated, $\rho_k^t[n] = 0$ otherwise. Thus the rate for V2I link n at time slot t can be expressed as,

$$R_n^{i,t} = W \log(1 + \gamma_n^{i,t}), \quad (4.2)$$

where W denotes the bandwidth of each sub-channel.

Similarly, at time slot t , V2V link k receives the interference signals from other

V2V and V2I links. The corresponding SINR for V2V link k in sub-channel n is given by,

$$\gamma_k^{v,t}[n] = \frac{\rho_k^t[n] |g_{k,k}^{v,t}[n]|^2 P_k^{v,t}}{\sigma^2 + I_k^{i,t}[n] + I_k^{v,t}[n]}, \quad (4.3)$$

where $g_{k,k}^{v,t}[n]$ denotes the channel gain of V2V link k at time slot t , $I_k^{i,t}[n]$ and $I_k^{v,t}[n]$ represent the received interference power at the receiver of V2V link k from all V2I links and from other V2V links, respectively. Specifically, $I_k^{i,t}[n] = |\tilde{h}_{k,n}^t[n]|^2 P_n^{i,t}$ and $I_k^{v,t}[n] = \sum_{k' \in \mathbb{K}, k' \neq k} \rho_{k'}^t[n] |\tilde{g}_{k,k'}^{v,t}[n]|^2 P_{k'}^{v,t}$, where $\tilde{h}_{k,n}^t[n]$ is the interference channel gain between the transmitter of V2I link n and the receiver of V2V link k , and $\tilde{g}_{k,k'}^{v,t}[n]$ is that between the transmitter of V2V link k' and the receiver of V2V link k in sub-channel n . The rate of V2V link k is thus given by,

$$R_k^{v,t} = \sum_{n \in \mathbb{N}} W \log(1 + \gamma_k^{v,t}[n]). \quad (4.4)$$

V2V links are responsible for transmitting the periodically generated safety-related information. Therefore, we aim to allocate the transmit power and sub-channels to the V2V links to maximize the V2V package delivery rate within a given time duration, i.e., $Pr(T_s \sum_{t \in \mathbb{T}} R_k^{v,t} > D_s), \forall k$, for the V2X system without causing severe interference to the V2I links, where D_s denotes the size of data package, T_s denotes the duration of each time slot, and $\mathbb{T} = \{1, 2, \dots, T\}$ denotes the set of transmission time indices. Note that the V2V link delivery rate is a long-term value, which can be effectively dealt with by Reinforcement Learning (RL) as it suits sequential decision-making for a long-term goal. In addition, in the system, we also employ Federated Learning (FL) to combat local observability.

4.3 Framework of RIFRL

In this section, we first formulate the resource allocation problem in the V2X system as a multi-agent Partially Observable Markov Decision Process (POMDP) and then design the RIFRL algorithm to solve the problem.

4.3.1 Multi-agent POMDP and policy gradient for the V2X system

The designed V2X system can be formulated as a POMDP with a tuple $\langle K, \mathbf{s}_t, \mathbf{a}_t^{(k)}, R_t^{(k)}, \mathbf{z}_t^{(k)}, P, O \rangle$, where K is the number of agents, $\mathbf{s}_t$ is the environment state at time t , $\mathbf{a}_t^{(k)}$ is the action at time t of agent k , $\mathbf{z}_t^{(k)} = O(\mathbf{s}_t, k)$ is the local observation obtained by agent k , observation function $O(\cdot, \cdot)$ maps state $\mathbf{s}_t$ to specific observation $\mathbf{z}_t^{(k)}$ of agent k , $R_t^{(k)}$ is the local reward received by agent k from the environment, and $P(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ is a transition probability from state $\mathbf{s}_t$ with joint action $\mathbf{a}_t = (\mathbf{a}_t^{(1)}, \mathbf{a}_t^{(2)}, \dots, \mathbf{a}_t^{(K)})$ to next state $\mathbf{s}_{t+1}$. In practice, we only use available $\langle K, \mathbf{a}_t^{(k)}, R_t^{(k)}, \mathbf{z}_t^{(k)} \rangle$ to train the system. Other quantities are automatically and implicitly learned by the agents during training.

In the V2X system, each V2V link k is deemed as an agent k and makes decisions for itself. Therefore, we define

$$\mathbf{a}_t^{(k)} = (P_k^{v,t}, f_t^k), \quad (4.5)$$

where $P_k^{v,t} \in \{23, 15, 5, -100\}$ dBm is limited to discrete values for the sake of both ease of learning and practical circuit restriction and $f_t^k \in \mathbb{N}$ denotes the selected sub-channel index for agent k . (We restrict the number of selected sub-channels for each V2V link to 1.) We use the available information [28] as the observation information for each agent, i.e.,

$$\mathbf{z}_t^{(k)} = \left\{ h_n^t[n], \tilde{h}_{k,n}^t[n], g_{k,k}^{v,t}[n], \tilde{g}_{k,k'}^{v,t}[n], I_k^{v,t-1}[n] + I_k^{i,t-1}[n], \right. \\ \left. \forall n \in \mathbb{N}, k' \in \mathbb{K}, \mathbf{q}_k, T_k^t, B_k^t, k \right\}, \quad (4.6)$$

where $\mathbf{q}_k$ is the relative position of the transmitter to the receiver of link k , T_k^t

and B_k^t are the remaining time budget and the remaining data package size. Moreover, all agents share a common reward so that they can learn to cooperate. We give all the agents a reward of 0.5, once a V2V link successfully delivers its data package within the given time period. Therefore, considering the V2I link transmission quality, the reward for agent k is given by,

$$R_t^{(k)} = 0.5 * Q^t + 0.1 * \sum_n R_n^{i,t}, \quad (4.7)$$

where Q^t denotes the number of V2V links that complete data package transmission in time slot t .

The general idea of POMDP is given as follows. At time t , each agent k simultaneously selects and implements an action $\mathbf{a}_t^{(k)}$ based on its observation of the environment, $\mathbf{z}_t^{(k)}$. Subsequently, the environment returns a local reward, $R_t^{(k)}$, to each agent k and its state $\mathbf{s}_t$ transits to the next state, $\mathbf{s}_{t+1}$, with a transition probability $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. In the next time step, $t + 1$, each agent k obtains a new observation of the environment, $\mathbf{z}_{t+1}^{(k)} = O(\mathbf{s}_{t+1}, k)$. Throughout the chapter, we set a common shared reward for all agents so that the targeted system is a Markov potential game. According to [86], the policy gradient (PG)-based algorithm for the game can finally converge to an approximate Nash equilibrium. Therefore, we build the FRL framework based on a PG-based algorithm.

PG-based methods directly optimize the policy of agents to maximize the long-term reward, i.e., each agent k aims to maximize its accumulative reward $R^{(k)}(\tau) = \sum_{t=1}^T R_t^{(k)}$ over trajectory τ , namely

$$\max y_k(\Phi) = \mathbb{E}_\tau(R^{(k)}(\tau)), \quad (4.8)$$

where Φ is the joint policy of all agents and trajectory $\tau = \{(\mathbf{z}_t^{(k)}, \mathbf{a}_t^{(k)}, R_t^{(k)}) \mid \forall k \in \mathbb{K}, t \in \mathbb{T}\}$ is collected by implementing the joint policy of all agents and interacting with environment for a time period of T .

We use a fully ReLU-based multi-layer neural network as an approximator of the policy. For agent k , we denote its trainable parameters as θ_k (i.e., networks' weights and biases to be trained). The corresponding policy is then expressed

as $\pi_k(\mathbf{a}_t^{(k)}|\mathbf{z}_t^{(k)}; \boldsymbol{\theta}_k)$ (i.e., networks' outputs). After interacting with the environment for T time slots, each agent k can calculate its approximate PG as follows [86],

$$\nabla_{\boldsymbol{\theta}_k} y_k(\Phi) \approx \mathbb{E}_{\tau} \left(R^{(k)}(\tau) \sum_{t=1}^T \nabla \log \pi_k(\mathbf{a}_t^{(k)}|\mathbf{z}_t^{(k)}; \boldsymbol{\theta}_k) \right). \quad (4.9)$$

With the above PG, the gradient ascent-based optimizers (e.g., RMSprop [89]) can be used to train the networks.

Algorithm 3 RIFRL

- 1: **Input:** Initial global model $\tilde{\boldsymbol{\theta}}_g^0$, number of training episodes J , and aggregation interval G .
 - 2: **for** episode index $j = 0, 1, \dots, J$ **do**
 - 3: --Local learning steps--
 - 4: **for** each agent $k \in \mathbb{K}$ in parallel **do**
 - 5: Update its local model by $\boldsymbol{\theta}_k^j = \tilde{\boldsymbol{\theta}}_g^j$, if receive $\tilde{\boldsymbol{\theta}}_g^j$ from the server.
 - 6: Sample a trajectory τ_k^j based on current policy $\pi_k(\cdot; \boldsymbol{\theta}_k^j)$ and calculate the PG based on (4.9).
 - 7: Update $\boldsymbol{\theta}_k^{j+1}$ by applying a gradient ascent-based optimizer to train the neural networks.
 - 8: **if** $j \bmod G == 0$ **then**
 - 9: Rescale $\boldsymbol{\theta}_k^{j+1}$ by **BRIO** to get $\tilde{\boldsymbol{\theta}}_k^{j+1}$.
 - 10: Upload rescaled model $\tilde{\boldsymbol{\theta}}_k^{j+1}$ to the server.
 - 11: **end if**
 - 12: **end for**
 - 13: --Global aggregation--
 - 14: **if** $j \bmod G == 0$ **then**
 - 15: **for** the central server **do**
 - 16: Update global model $\boldsymbol{\theta}_g^{j+1} = \frac{1}{K} \sum_{k=1}^K \tilde{\boldsymbol{\theta}}_k^{j+1}$.
 - 17: Rescale $\boldsymbol{\theta}_g^{j+1}$ by **BRIO** to get $\tilde{\boldsymbol{\theta}}_g^{j+1}$
 - 18: Broadcast rescaled model $\tilde{\boldsymbol{\theta}}_g^{j+1}$ to all agents.
 - 19: **end for**
 - 20: **end if**
 - 21: **end for**
 - 22: **Output:** Trained global model, $\boldsymbol{\theta}_g^{J+1}$.
-

4.3.2 RIFRL algorithm

The RIFRL algorithm is presented in Algorithm 3. In this FRL system, we treat K agents as edge nodes, place a central server to aggregate local models $\{\theta_k : \forall k \in \mathbb{K}\}$ from agents, and broadcast the aggregated global model, θ_g , to all agents. All agents update their local models in each episode and aggregation happens every G episode. At Step 5, local updates for all agents begin with the same models, $\theta_k^j = \tilde{\theta}_g^j$. In Step 6, each agent interacts with the environment based on the policy represented by their local models, obtains the experience trajectory of the current episode, and calculates the corresponding PG using (4.9). In Step 7, each agent updates the local model using the obtained PG and the gradient ascent method. At Step 16, the central server aggregates the knowledge of all agents.

Due to data discrepancy or heterogeneity, local updates can experience client drifts, i.e., the local model updating direction varies among different clients. This client drift degrades the aggregation efficiency and thus slows down the convergence speed. To mitigate this effect, we exploit the node-wise invariance property of ReLU-activated neural networks, i.e., force all local models to stay in the same solution space without changing the neural network input-output mapping. We improve the FRL algorithm by adding two backward rescale steps before and after the aggregation step in Steps 9 and 17. The BRIO is introduced in the next subsection.

Note that although the FRL algorithm targets a global shared model, only model parameters are shared among all agents and the same model can still output a unique policy for each agent based on different observations of different agents.

4.3.3 Rescale for ReLU activated linear layers

As shown in Fig. 4.1, node-wise invariance is based on the fact that for a ReLU-activated neuron network or subnetwork, if we multiply the input weights of this node by $\alpha > 0$ and multiply the output weights of this node by $\frac{1}{\alpha}$, then the

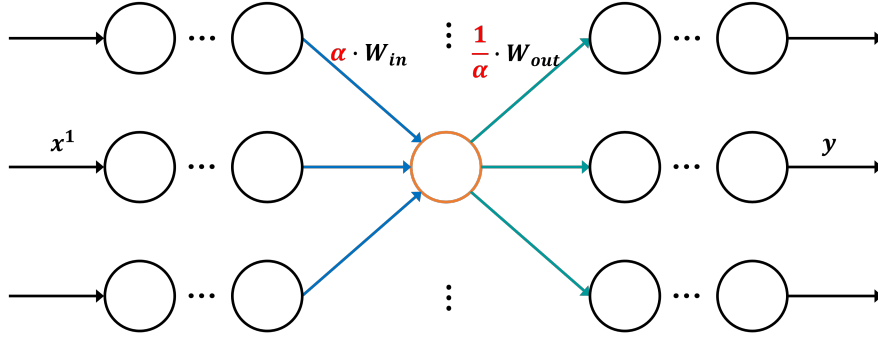


Figure 4.1: Node-wise invariance.

input-output map of this node does not change^[1] Now we integrate this idea in a fully connected network with $L - 1$ ReLU-activated linear layers.

Let the input, output, weight matrices, and bias of the ReLU activated network be $\mathbf{x}^1, \mathbf{y}, (\mathbf{W}^1, \dots, \mathbf{W}^L)$ and $(\mathbf{b}^1, \dots, \mathbf{b}^L)$, respectively. Then we have the input-output relationships as follows,

$$\begin{aligned} \mathbf{x}^{\ell+1} &= \text{ReLU}(\mathbf{W}^\ell \mathbf{x}^\ell + \mathbf{b}^\ell), \ell = 1, \dots, L - 1, \\ \mathbf{y} &= \mathbf{W}^L \mathbf{x}^L + \mathbf{b}^L. \end{aligned} \quad (4.10)$$

Left multiplying a diagonal matrix $\mathbf{D}^{\ell+1}$ with positive diagonal entries to both sides of (4.10) yields that

$$\begin{aligned} \tilde{\mathbf{x}}^{\ell+1} &= \mathbf{D}^{\ell+1} \mathbf{x}^{\ell+1} \\ &= \mathbf{D}^{\ell+1} \text{ReLU}(\mathbf{W}^\ell \mathbf{x}^\ell + \mathbf{b}^\ell) \\ &= \text{ReLU}(\mathbf{D}^{\ell+1} \mathbf{W}^\ell \mathbf{x}^\ell + \mathbf{D}^{\ell+1} \mathbf{b}^\ell) \\ &= \text{ReLU}(\mathbf{D}^{\ell+1} \mathbf{W}^\ell (\mathbf{D}^\ell)^{-1} \tilde{\mathbf{x}}^\ell + \mathbf{D}^{\ell+1} \mathbf{b}^\ell), \\ \mathbf{D}^L \mathbf{y} &= \mathbf{D}^{L+1} \mathbf{W}^L (\mathbf{D}^L)^{-1} \tilde{\mathbf{x}}^L + \mathbf{D}^{L+1} \mathbf{b}^L, \end{aligned} \quad (4.11)$$

where the third equation is due to the homogeneity^[2] of ReLU and the last two equations are from $\tilde{\mathbf{x}}^\ell = \mathbf{D}^\ell \mathbf{x}^\ell$. The above equations indicate that we obtain

¹For expression consistence, we use “network” to refer to network or subnetwork in this section.

²A function f is homogeneous if $f(\alpha x) = \alpha f(x)$ for any $\alpha > 0$.

another network with $\tilde{\mathbf{x}}^\ell$ being the input vector of the ℓ -th linear layer. The input and output of the new network are $\tilde{\mathbf{x}}^1$ and $\mathbf{D}^{L+1}\mathbf{y}$. To make the two networks equivalent, we set $\mathbf{D}^1$ and $\mathbf{D}^{L+1}$ to be identity matrices. Then we get a new equivalent network with the ℓ -th layer's weight matrix and bias vector being

$$\tilde{\mathbf{W}}^\ell = \mathbf{D}^{\ell+1}\mathbf{W}^\ell(\mathbf{D}^\ell)^{-1} \text{ and } \tilde{\mathbf{b}}^\ell = \mathbf{D}^{\ell+1}\mathbf{b}^\ell. \quad (4.12)$$

To choose a proper diagonal matrix $\mathbf{D}^\ell$, we adopt the idea to normalize $\tilde{\mathbf{W}}^\ell$ to have unit columns. To fulfil that, we can calculate the i -th diagonal entry of matrix $\mathbf{D}^\ell$ in a backward rescale order of $\ell = \{L, \dots, 1\}$ as follows,

$$\mathbf{D}_{ii}^\ell = \begin{cases} 1, & \ell \in \{1, L+1\}, \\ \|(\mathbf{D}^{\ell+1}\mathbf{W}^\ell)_{:i}\|, & \ell \in \{2, 3, \dots, L\}, \end{cases} \quad (4.13)$$

where $\|(\mathbf{D}^{\ell+1}\mathbf{W}^\ell)_{:i}\|$ denotes the Euclidean norm of the i -th column of $\mathbf{D}^{\ell+1}\mathbf{W}^\ell$. One can observe that the above choice of $\mathbf{D}^\ell$ normalizes each column of $\mathbf{D}^{\ell+1}\mathbf{W}^\ell$ so that $\tilde{\mathbf{W}}^\ell$ has unit columns. We call such an operation the BRIO and present it in Algorithm 4.

Algorithm 4 BRIO

- 1: **Initialize:** A fully connected neural network with $L - 1$ ReLU activated linear layers and a linear output layer with weight matrices $(\mathbf{W}^1, \dots, \mathbf{W}^L)$ and bias $(\mathbf{b}^1, \dots, \mathbf{b}^L)$.
 - 2: For $\ell = L$ to 1, calculate $(\mathbf{D}^{L+1}, \dots, \mathbf{D}^1)$ by (4.13).
 - 3: Return rescaled weight matrices $(\tilde{\mathbf{W}}^1, \dots, \tilde{\mathbf{W}}^L)$ and bias $(\tilde{\mathbf{b}}^1, \dots, \tilde{\mathbf{b}}^L)$ by (4.12).
-

To integrate BRIO into RIFRL, we let $\boldsymbol{\theta} = (\mathbf{W}^1, \dots, \mathbf{W}^L, \mathbf{b}^1, \dots, \mathbf{b}^L)$ and the rescaled model $\tilde{\boldsymbol{\theta}} = (\tilde{\mathbf{W}}^1, \dots, \tilde{\mathbf{W}}^L, \tilde{\mathbf{b}}^1, \dots, \tilde{\mathbf{b}}^L)$. Thanks to this, for each agent k , the components in $\tilde{\boldsymbol{\theta}}_k$ corresponding to $\tilde{\mathbf{W}}^\ell, \ell \in \{2, 3, \dots, L\}$ have unit columns so that the main components in $\boldsymbol{\theta}_k$ corresponding to $\mathbf{W}^\ell, \ell \in \{2, 3, \dots, L\}$ for different agents become similar, which is beneficial to the model aggregation and thus improves the learning process.

Remark 4.3.1. To end this section, we would like to point out that the BRIO can be extended to neural networks with other activation functions if they are homogeneous.

This is because equations (4.11) still hold if we replace ReLU with these functions. Typical homogeneous activation functions are piecewise linear functions such as leaky ReLU [90] and parametric ReLU [91]. Therefore, the RIFRL algorithm can be flexibly adjusted based on other neural networks with homogeneous activation functions.

4.4 Simulation Results

In this section, we demonstrate the performance of our proposed RIFRL algorithm for the resource allocation problem in a V2X system. We follow the urban case in Annex A of [1], which simulates 9 blocks in total with an area size of $1299\text{m} \times 750\text{m}$ and with both line-of-sight and non-line-of-sight channels. The corresponding channel parameters are set the same as Table I and Table II in [28]. We consider 8 vehicles and thus 8 V2I links as well as 8 sub-channels. The number of V2V links, K , is set to 24 and V2V package data size D_s is set to 2120 Bytes if not specified.

The policy neural network for each V2V link consists of three fully connected hidden layers with 500, 250, and 120 neurons, respectively. ReLU is employed as the activation function in all layers except for the outermost layer where the softmax function is adopted so that the policy network can output a probability vector for the action candidates.

We compare three learning methods and a random baseline that randomly selects actions is used to show the system performance without optimization. The three learning methods are the independent PG algorithm, the FedAvg-based FRL algorithm (short for FRL), and RIFRL. We set the learning rate for FRL and RIFRL as 10^{-3} but for independent PG as 10^{-4} and the optimizer as RMSprop optimizer. This is because a slightly larger learning rate makes independent PG fail to learn a good policy. We set the time duration of each time slot as 1 ms and each training episode consists of 100 time slots. Both FRL and RIFRL aggregate agents' models once every 5 training episodes, i.e., $G = 5$.

We begin by comparing and analyzing the learning processes of three algorithms. Fig. 4.2 illustrates the normalized moving average reward throughout the training phase.³ In comparison with independent PG, two FRL algorithms can enhance the learning process by empowering each agent to acquire knowledge beyond its individual observations. This is because the V2V delivery probability is a long-term optimization target, which is challenging for inde-

³The moving average is calculated over the reward of 500 episodes and each curve is the mean of 35 independent training trajectories.

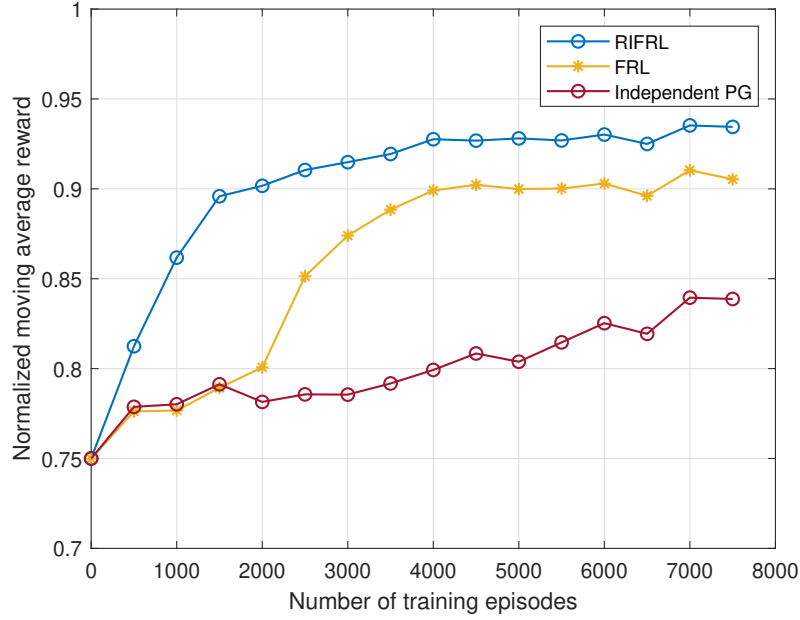


Figure 4.2: Moving average reward of 3 algorithms.

pendent PG using a single agent to explore the relation between the long-term reward and the instantaneous local observations.

Among the three algorithms, RIFRL improves both the learning speed and convergent performance remarkably. These improvements stem from the BRIO, which adjusts the local models of agents to make them more similar to each other, thereby shrinking the solution space of each agent’s optimization problem and enhancing the aggregation performance. We highlight that RIFRL is able to achieve comparable performance to FRL with around 1600 fewer training episodes. Please note that a performance of 1 cannot be reached even if infinite computational resource is available, because severe channel fading can lead to the failure of V2V link transmission.

Next, we evaluate the trained models of all algorithms, as well as a MARL algorithm from [28], by employing them in another testing environment. Fig. 4.3 illustrates the performance of different algorithms in terms of V2V packet delivery probability when varying the size of V2V data package. It is noticeable that the probability decreases as the data size increases for all algorithms. It is

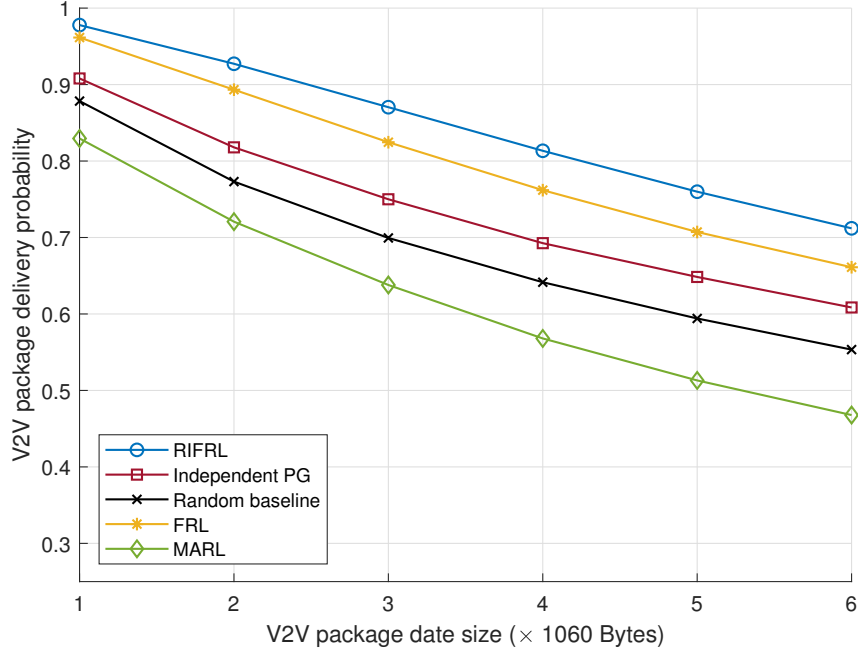


Figure 4.3: V2V delivery probability of 5 algorithms.

clear that RIFRL outperforms the others. The MARL algorithm from [28] does not deliver desirable results. This is because it has been designed to address the static environment, while the environment considered for our simulation is dynamic due to the change of positions of vehicles.

Finally, to assess the adaptation ability of the algorithms under varying environmental configurations, we modify the number of V2V links (i.e., the number of agents), which impacts the V2V link density per resource block and consequently results in diverse environmental statistics. As illustrated in Fig. 4.4, it becomes evident that as the number of V2V links increases, the V2V package delivery probability decreases, indicating poorer performance. This degradation is attributed to the increased competition among V2V links sharing the same sub-channel. Once again, RIFRL exhibits the highest probability, highlighting its desirable effectiveness in adapting to different environmental configurations.

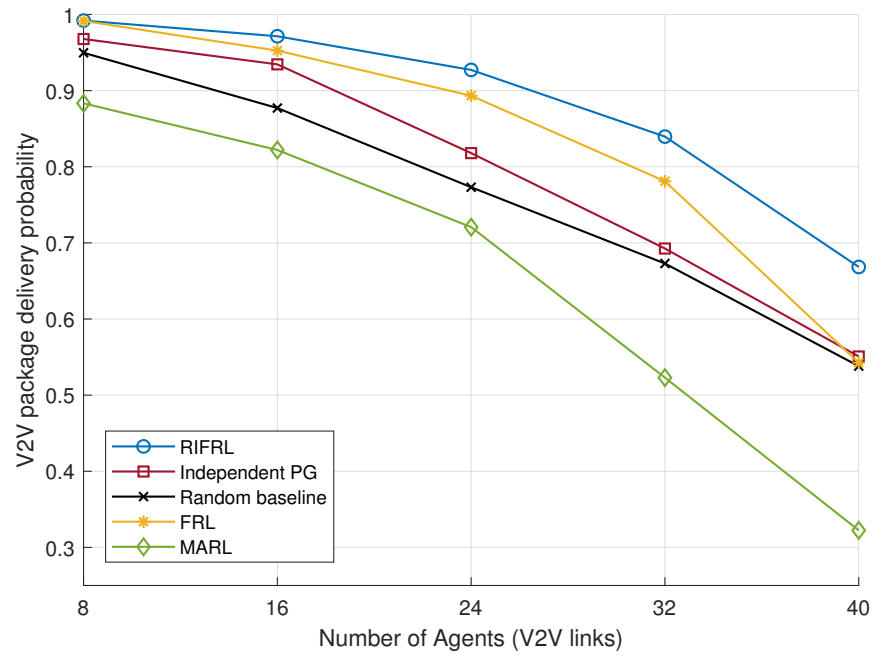


Figure 4.4: V2V delivery probability of 5 algorithms.

4.5 Conclusion

The performance enhancement for resource allocation problems in V2X networks arises from two aspects: the integration of FL and RL and the exploitation of node-wise invariance of ReLU-activated neural networks. This chapter has enabled us to design a useful procedure BRIO. We believe that the developed architecture and the BRIO hold the potential to address many other applications, as long as they can be formulated as a FL problem and the targeted DNN model is ReLU-activated. This deserves further investigation.

Chapter 5

Neural Collapse based Deep Supervised Federated Learning for Signal Detection in OFDM Systems

In this chapter, we improved Federated Learning (FL) by hidden feature alignment. We exploited the Neural Collapse (NC) phenomenon, where the last layer hidden features and the output layer weights finally converge to Equiangular Tight Frames (ETFs) and are linearly aligned under the universal approximation assumption of Deep Neural Networks (DNNs). We introduced an auxiliary classifier to the DNN at each client and used the NC weights to deeply supervise the hidden features across all clients. In this way, the hidden weights were better aligned without additional communicational and computational overheads. To apply this Neural Collapse based Deep Supervised Federated Learning (NCDSFL) framework to signal detection task, we further derived the NC solution for multi-binary classification problems and proved its optimality. Simulation results show that the proposed NCDSFL algorithm has faster convergence speed than the baseline algorithm under various channel conditions.

5.1 Introduction

Deep learning (DL) has been recognized as a powerful tool to enhance wireless communication systems in various perspectives (see Chapter 1 for detailed background and literature review), whose performance is highly dependent on the quality of the training dataset. In many wireless communication systems, as illustrated in Fig. 5.1, user devices can only observe a partial wireless environment, which is characterized by its unique Power Delay Profile (PDP) determined by the positions of User Devices (UDs), the base station and the scatterers. Collecting data from all users is expensive and a privacy violation, which makes DL-based methods less practical for training a generalizable model. Federated Learning (FL) is thus employed to address this challenge. However, the local datasets of clients are usually heterogeneous due to users' local observability, which can lead to slow convergence speed of the FL algorithm and thus increase communication overhead [43].

To mitigate the client drift effect caused by data discrepancy, feature alignment can be exploited [43, 92–94]. FedProx [43] penalizes the difference between local client model and the global model during the local training updates, which implicitly aligns the features of local model and that of the global model. MOON [93] uses contrastive learning to encourage the similarity between the local model feature representation and the global model feature representation. The authors of [94] permute the hidden elements within each layer so that the hidden elements with similar feature extraction signatures can be matched and aggregated. The aforementioned methods validate the benefits of feature alignment in FL but require additional computational overhead or treat model parameters equally ignoring the effect of dataset. To implement the FL algorithm in real wireless systems, it is important to reduce both communication and computation overhead.

5.1.1 Specific Contributions of this Chapter

In this chapter, we focus on training a joint channel estimation and signal detection neural receiver by FL aggregating the knowledge learned by dif-

ferent client user devices with local observability while preserving data privacy. We propose a Neural Collapse based Deep Supervised Federated Learning (NCDSFL) framework to mitigate the client drift effect. Neural collapse (NC) [95] describes the optimal condition for classifiers and hidden features under the universal approximator assumption on DNNs. The NC duality establishes the geometric link between the classifiers and the hidden features, i.e., they converge to the same Equiangular Tight Frame (ETF) and are finally linear aligned. Therefore, it is possible to align the hidden features by using the same fixed auxiliary classifiers and the same loss function to supervise the hidden features at each client user device by Deep Supervision (DS) [96]. This NC inspired DS is carried out by adding a regularization term to the local training loss at each client user device. Therefore, the NCDSFL algorithm reduces the model parameters to be updated and transmitted. In addition, the signal detection task can be formulated as a multi-binary classification problem. We derive the NC form for the multi-binary classification problems, which has not been addressed in the literature yet. The main contributions of this work can be summarized as follows.

- We provide an explicit NC solution for a multi-binary classification problem and rigorously establish its optimality by revealing the relationship between an NC and a global solution to the problem. To the best of our knowledge, this has not been addressed in the existing literature.
- We then introduce the technique of DS in a DNN by adding one auxiliary layer connected to the third-to-last layer. This approach can be interpreted as incorporating a regularization term into the objective function of the target optimization model. During training, we fix the weights at both the output layer and the auxiliary layer using two derived NC solutions. As a result, the number of trainable parameters is reduced.
- We integrate DS-based DNNs, with a portion of their weights fixed using NC solutions, into an FL paradigm. This design encourages different clients to produce similar hidden features at both the penultimate and auxiliary layers. A hidden feature refers to the output of a specific layer of neurons; see (5.11) for a detailed definition. Consequently, the

proposed NCDSFL algorithm can effectively address data heterogeneity and achieve fast convergence, as demonstrated by simulation results on signal detection in Orthogonal Frequency Division Multiplexing (OFDM) systems compared to several baseline methods. Furthermore, we empirically validate the emergence of the NC phenomenon in multi-binary classification tasks.

5.1.2 Related Works

NC [95], first defined in [97], is a terminal state of neural networks at the end of the training phase. NC provides an elegant mathematical analysis on the features and the parameters of the last layer by peeling them off the neural network and treating them as free variables under the universal approximator assumption on neural networks. A specific NC form depends on formulation of the DNN training problem, but usually includes four properties: collapse of variability, preference towards a simplex ETF, self-dual alignment and simplification to nearest class center. A family of researches on NC with different configurations has been conducted. The authors of [97] study the canonical classification problem with CrossEntropy loss and first define NC. The authors of [98] further analyze the landscape of the CrossEntropy loss with weight decay and find that only global minimizers are simplex ETFs while all other critical points are strict saddle points, which explains why CrossEntropy is an effective loss function for classification problem. In [99], the authors investigated how label corruption influences collapse, showing that neural collapse can still manifest but with altered geometric structure due to memorization effects. In [100], the authors analyze the gradients of CrossEntropy loss and propose replacing the final learnable classifier with fixed ETF classifiers and an alternative loss function, which has a theoretical convergence rate not worse than the CrossEntropy loss with learnable classifiers. Beside the multiclass classification problem, NC for multivariate regression is recently proved in [101]. In addition, the authors of [102] investigate the NC for multi-label classification problems using pick-all-label loss, where each data sample can be classified to multiple classes. DL based signal detection can be modeled as a multi-binary classification prob-

lem, which is equivalent to multi-label classification problem. However, the method from [102] requires a threshold to determine how many labels to pick for one data sample in the prediction phase, which does not fit our considered signal detection task. We therefore derive a different NC form for multi-binary classification problems and prove its global optimality.

DS was first proposed in [96], which adds supervision to the intermediate hidden features of DNNs. By doing so, DS can mitigate the gradient vanishing issue, help the DNN learn more discriminative features and thus improve the DNN performance [103]. It is later applied to many different computer vision tasks like image segmentation, object detection, image super-resolution, etc. [104-106]. The aforementioned MOON [93] algorithm can also be classified into the DS category, as it introduces regularization terms of the intermediate hidden features, which is then optimized with the main loss function together. NC defines the optimal solution form for the output layer related variables, naturally, we can use the fixed ETF weights as the auxiliary classifiers as in the output layer due to their optimality. However, to the best of our knowledge, DS related works haven't exploit the NC solutions yet.

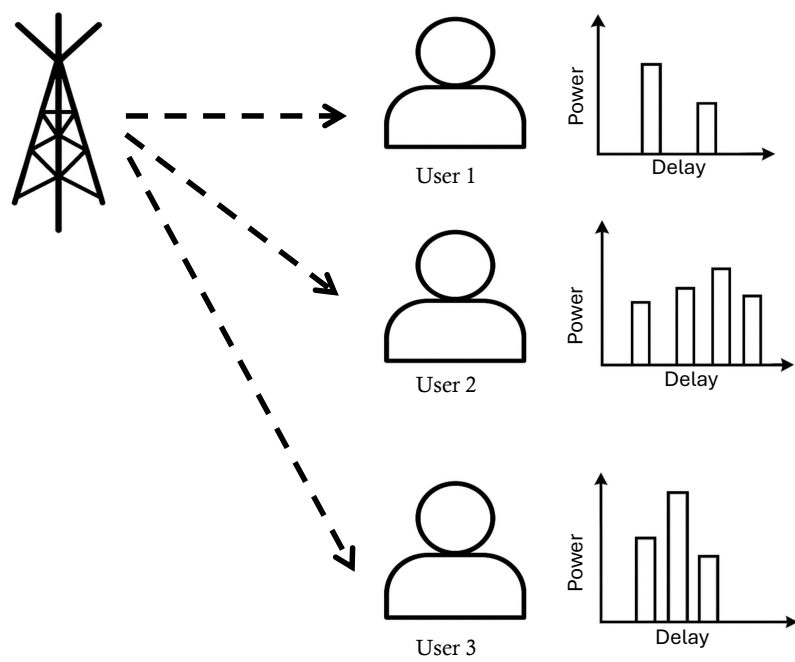


Figure 5.1: System diagram

5.2 Multi-Binary Classification

In this section, we introduce the multi-binary classification problem and propose the NC solution of the corresponding layer peeled model, followed by its application in DNNs.

5.2.1 Problematic Description

Let $s \in \mathcal{S}^I$ denote a binary label sequence of I binary classes, where $\mathcal{S}^I$ denotes the space of binary sequences with length I . The i -th value, denoted by s_i , of s represents the i th class of the data sample labeled by s . Taken $I = 3$ as an example, $\mathcal{S}^I = \{000, 001, 010, 011, 100, 101, 110, 111\}$ and $s = 011$ means that the sample labeled as s belongs to the second or third classes due to $s_2 = s_3 = 1$. In the sequel, for notational simplicity, we denote

$$\mathcal{S} = \mathcal{S}^I = \left\{ s^{(0)}, s^{(1)}, \dots, s^{(2^I-1)} \right\}, \quad (5.1)$$

as $\mathcal{S}^I$ has 2^I elements. The superscript j in $s^{(j)}$ is determined by binary sequence s itself. For example, $j = 1$ if $s = 001$ and $j = 6$ if $s = 110$. We consider the balanced distribution case, where there are K samples for each label $s \in \mathcal{S}^I$.¹ As a result, there are $K2^I$ data samples in total. In addition, we denote the linear classifiers for the i th binary class as $\mathbf{w}_{i,1} \in \mathbb{R}^d$ and $\mathbf{w}_{i,0} \in \mathbb{R}^d$ and the hidden feature for the k th data sample labeled by sequence s as $\mathbf{h}_s^{(k)} \in \mathbb{R}^d$.

Following the canonical classification paradigm, we use cross entropy with regularized weights and features as our loss function. The resulting loss function is given by

$$\begin{aligned} L(\mathbf{W}, \mathbf{H}) = & \lambda \|\mathbf{W}\|^2 + \lambda \|\mathbf{H}\|^2 \\ & + \frac{1}{KI2^I} \sum_{k=1}^K \sum_{i=1}^I \sum_{s \in \mathcal{S}} \ln \left(1 + \exp \left(\phi_{kis}(\mathbf{W}, \mathbf{H}) \right) \right), \end{aligned} \quad (5.2)$$

¹This assumption is reasonable in a wireless communication scenario, because the transmitted binary sequences are usually uniformly distributed binary sequences.

where $\lambda > 0$ and $\phi_{kis}(\mathbf{W}, \mathbf{H})$ is defined by

$$\phi_{kis}(\mathbf{W}, \mathbf{H}) = \langle \mathbf{w}_{i,1-s_i} - \mathbf{w}_{i,s_i}, \mathbf{h}_s^{(k)} \rangle. \quad (5.3)$$

Here, $\langle \cdot, \cdot \rangle$ denotes the vector inner product, $1 - s_i, s_i \in \{0, 1\}$, and $\mathbf{W}$ and $\mathbf{H}$ denote the collection of all classifiers and all features, namely,

$$\begin{aligned} \mathbf{W} &= [\mathbf{w}_{1,0}, \mathbf{w}_{2,0}, \dots, \mathbf{w}_{I,0}, \mathbf{w}_{1,1}, \mathbf{w}_{2,1}, \dots, \mathbf{w}_{I,1}], \\ \mathbf{H} &= [\mathbf{h}_{s(0)}^{(1)}, \dots, \mathbf{h}_{s(2^I-1)}^{(1)}, \dots, \mathbf{h}_{s(0)}^{(K)}, \dots, \mathbf{h}_{s(2^I-1)}^{(K)}]. \end{aligned} \quad (5.4)$$

5.2.2 NC Solutions

It is known that a layer-peeled model [100,107] ignores the impact of the backbone network and treats the features of the penultimate hidden layer $\mathbf{H}$ as free variables, which is true when the DNN is over-parameterized due to the universal approximation property of DNNs. The layer peeled model of the multi-binary classification problem can be expressed as the following optimization problem,

$$\min_{\mathbf{W}, \mathbf{H}} L(\mathbf{W}, \mathbf{H}). \quad (5.5)$$

To analyze the above problem, let $\text{vec}(\mathbf{H})$ denote the column-wise vectorization of $\mathbf{H}$, namely,

$$\text{vec}(\mathbf{H}) = \left[\mathbf{h}_{s(0)}^{(1)T}, \dots, \mathbf{h}_{s(2^I-1)}^{(1)T}, \dots, \mathbf{h}_{s(0)}^{(K)T}, \dots, \mathbf{h}_{s(2^I-1)}^{(K)T} \right]^T, \quad (5.6)$$

where $\mathbf{h}_{s(i)}^{(k)T} = (\mathbf{h}_{s(i)}^{(k)})^T$. Let $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ be an identity matrix, and $\mathbf{A} \in \mathbb{R}^{dI \times dK2^I}$ and $\mathbf{C}^i \in \mathbb{R}^{d \times d2^i}$ be defined by,

$$\mathbf{A} = \underbrace{[\mathbf{B}^I, \dots, \mathbf{B}^I]}_{K \text{ times}}, \quad \mathbf{C}^i = \underbrace{[\mathbf{I}_d, \dots, \mathbf{I}_d]}_{2^i \text{ times}}, \quad (5.7)$$

where $\mathbf{B}^I$ is recurrently generated as follows,

$$\mathbf{B}^1 = \begin{bmatrix} -\mathbf{I}_d & \mathbf{I}_d \end{bmatrix}, \quad \mathbf{B}^{i+1} = \begin{bmatrix} \mathbf{B}^i & \mathbf{B}^i \\ -\mathbf{C}^i & \mathbf{C}^i \end{bmatrix}, \quad (5.8)$$

for $i = 1, 2, \dots, I - 1$. Based on these notations, an NC solution to problem (5.5) is introduced as follows.

Definition 1. A point $(\mathbf{W}, \mathbf{H})$ is called an NC solution to problem (5.5) for the multi-binary classification if it satisfies the following conditions:

- (NC1: hidden feature collapse). All features converge to the mean, i.e., $\mathbf{h}_s^{(k)} = \frac{1}{K} \sum_{j=1}^K \mathbf{h}_s^{(j)}, \forall s, k$.
- (NC2, classifier collapse). Classifiers of all parallel binary classification problems form an orthogonal set, i.e., $\langle \mathbf{w}_{i,1}, \mathbf{w}_{j,1} \rangle = 0, \forall i \neq j$, and $\mathbf{w}_{i,1} + \mathbf{w}_{i,0} = \mathbf{0}, \forall i$. In addition, $\|\mathbf{w}_{i,1}\| = \|\mathbf{w}_{j,1}\|, \forall i, j$.
- (NC3: duality). The classifier and the hidden features are linearly aligned by

$$\text{vec}(\mathbf{H}) = c_0 [\mathbf{A}^T, -\mathbf{A}^T] \text{vec}(\mathbf{W}), \quad (5.9)$$

for a scaling constant $c_0 > 0$.

The following theorem reveals the relationship between the NC solution and the global solution to problem (5.5).

Theorem 5.2.1. If $0 < \lambda < 1/(2I\sqrt{2K2^I})$, then any global minimizer of problem (5.5) is an NC solution.

Proof. The proof of Theorem 5.2.1 is given in Appendix 5.6. $\square$

Remark 5.2.2. The proposed NC for multi-binary classification problem can also be extended to a multi-label classification scenario, where each data sample has an arbitrary number of class labels from I classes. This multi-label classification problem is equivalent to the multi-binary classification problem with I parallel binary classification sub-tasks, where each binary classification represents whether the sample is classified into the corresponding class or not.

5.2.3 Deep supervised DNNs with NC Weights

Now we introduce O -layer neural networks with $(O - 1)$ hidden layers. Specifically, let d_o be the number of hidden units of the o -th hidden layer for $i \in [O - 1]$, where $[O] = \{1, 2, \dots, O\}$. Let d_0 and d_O represent the number of input and output units. Denote $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_O\}$ with $\mathbf{W}_o \in \mathbb{R}^{d_o \times d_{o-1}}$ being the weight matrix of the o th layer. For simplicity, we integrate both the weights and the bias into $\mathbf{W}$. Let $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n) : n \in [N]\}$ be a given dataset, where $\mathbf{x}_n \in \mathbb{R}^{d_0}$ and $\mathbf{y}_n \in \mathbb{R}^{d_O}$ is the n th sample's feature and label, and N is the number of samples. The optimization model of DNNs can be built as follows,

$$\begin{aligned} \min_{\mathcal{W}} \quad & L(\mathbf{Y}, \tilde{\mathbf{Y}}) \\ \text{s.t.} \quad & \tilde{\mathbf{Y}} = \text{softmax}(\mathbf{W}_O \sigma(\mathbf{W}_{O-1} \cdots \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{X})))), \end{aligned} \quad (5.10)$$

where L is a loss function, $\sigma(\cdot)$ is an activation function, such as the ReLU used in the sequel, $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, and $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$. For simplicity, let

$$\mathbf{H}_0 = \mathbf{X}, \quad \mathbf{H}_o = \sigma(\mathbf{W}_o \mathbf{H}_{o-1}), \quad o = 1, 2, \dots, O - 1. \quad (5.11)$$

Then $\mathbf{H}_o$ is the output of the neurons at the o th layer and is referred to as the hidden features at this layer.

The motivation for considering the NC solution stems from observations in many applications, such as the signal detection task illustrated in Figs. 5.3 and 5.4, where the penultimate hidden layer of a DNN exhibits classifier collapse (i.e., NC2), as well as classifier and feature linear alignment (i.e., NC3), and the NC-related studies introduced in the introduction section. To leverage this structural behavior, we embed NC solutions as the weights at the penultimate hidden layer of a DNN.

Direct embedding

The first embedding is employed directly on the original DNN at the O th layer, i.e., the output layer, see the red rectangle in Fig. 5.2, where $\mathbf{W}_O$ in model (5.10) will be fixed by using an NC weight defined in Definition 1. Specifically, we

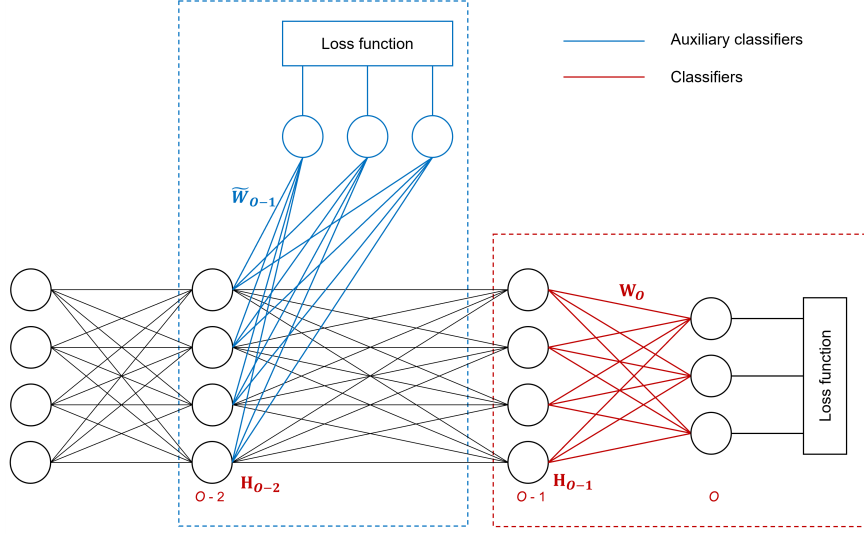


Figure 5.2: A deep supervised DNN

generate a set of orthogonal vectors $\{\mathbf{w}_{i,0}\}$ and let $\mathbf{w}_{i,1} = -\mathbf{w}_{i,0}$ for all $i \in [I]$ to form an NC weight $\mathbf{W}_{NC}$. The weight is then fixed during the whole training phase, as fixed NC classifiers do not harm classification performance [98]. To do so, the constraint in model (5.10) becomes

$$\tilde{\mathbf{Y}} = \text{softmax}(\mathbf{W}_{NC} \sigma(\mathbf{W}_{O-1} \cdots \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{X}))). \quad (5.12)$$

Auxiliary embedding under the DS framework

The second embedding is applied for an auxiliary layer. We introduce one auxiliary layer with weight matrix $\tilde{\mathbf{W}}_{O-1}$ collecting the $(O-2)$ th neuron in the original network. This idea is termed as DS in [103]. As shown in the blue rectangle in Fig. 5.2, the $(O-2)$ th neuron output in the original network is connected to the auxiliary loss function by the weight matrix $\tilde{\mathbf{W}}_{O-1}$. Again, weight $\tilde{\mathbf{W}}_{O-1}$ is fixed by another NC weight $\tilde{\mathbf{W}}_{NC}$ defined similarly as above. This indicates that we can add a regularization term in model (5.10),

$$\text{CE}(\mathbf{Y}, \text{softmax}(\mathbf{H}_{O-2} \tilde{\mathbf{W}}_{NC})), \quad (5.13)$$

where $\text{CE}(\cdot)$ is a cross entropy loss. This together with (5.12), model (5.10) turns to have the following form,

$$\begin{aligned} \min_{\widetilde{\mathcal{W}}} \quad & L(\mathbf{Y}, \widetilde{\mathbf{Y}}) + \mu \text{CE}(\mathbf{Y}, \mathbf{H}_{O-2} \widetilde{\mathbf{W}}_{NC}) \\ \text{s.t.} \quad & \widetilde{\mathbf{Y}} = \text{softmax}(\mathbf{W}_{NC} \sigma(\mathbf{W}_{O-1} \cdots \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{X})))), \end{aligned} \quad (5.14)$$

where $\widetilde{\mathcal{W}} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_{O-1}\}$ and $\mu > 0$. In the next section, we will explore the above model to develop an efficient algorithm for signal detection problems.

5.3 Neural Collapse inspired Deep Supervised Federated learning

In this section, we first introduce the framework of FL and develop the NCDSFL algorithm based on it.

5.3.1 Federated Learning

A FL system consisting of one central server and J clients aims to learn a shared model for all clients to minimize the total loss function without raw data exchange. Specifically, this learning process can be formulated as the following optimization problem,

$$\begin{aligned} \min_{\mathbf{v}, \mathbf{v}^i \in \mathcal{F}, i \in [J]} \quad & \sum_{i \in [J]} \left(L(\mathbf{v}^i; \mathcal{D}^i) + R(\mathbf{v}^i; \mathcal{D}^i) \right), \\ \text{s.t.} \quad & \mathbf{v}^i = \mathbf{v}, \quad i \in [J], \end{aligned} \tag{5.15}$$

where $\mathbf{v}^i$, $\mathbf{v}$, $\mathcal{F}$ and $\mathcal{D}^i$ denote the trainable local parameters at client i , the shared global parameters, the feasible region, and the dataset of client i , respectively, $L(\cdot; \cdot)$ denotes the local loss function used to train the model locally and depends on the local parameters and the local dataset, and $R(\cdot; \cdot)$ is a regularization term added to a local loss to guide training.

In a canonical centralized FL system, the learning process repeats the following three key steps until the model parameters converge or some terminating conditions are met:

- **Local update:** Each client updates its local model based on its local parameters and dataset by an optimization algorithm such as stochastic gradient descent or Adam [53].
- **Aggregation:** After several local updates, all clients or a part of them send their trained local model parameters to the server. The server aggregates the models by averaging the received parameters.
- **Broadcast:** The server then broadcasts the aggregated global model to all

clients.

5.3.2 DS DNNs with NC Weights in FL

Note that in FL, model aggregation occurs after several local updates. This can lead to client drift and slow down the convergence, especially when local datasets are heterogeneous. To facilitate more efficient aggregation, a natural approach is to align hidden features across clients. According to the NC theory [95], the hidden features (e.g., $\mathbf{H}_{O-1}$ in the direct embedding and $\mathbf{H}_{O-2}$ in the auxiliary embedding) and their classifiers (i.e., $\mathbf{W}_{NC}$ and $\tilde{\mathbf{W}}_{NC}$) eventually collapse (i.e., satisfying NC1 and NC2) and become linearly aligned (i.e., satisfying NC3). At the same time, DS [103] provides direct guidance to intermediate hidden features (e.g., $\mathbf{H}_{O-2}$).

By integrating these two insights, we can guide and align hidden feature distributions across clients, thereby promoting consistency among local models. To achieve this, we incorporate model (5.14) into the FL framework. Specifically, for J clients, we equip each client i with E DNNs with $E \geq 1$, which have different learnable parameters. Therefore, there are EJ DNNs in total. The dataset, $\mathcal{D}^i$, for client i is

$$\mathcal{D}^i = \{\mathcal{D}^{i,1}, \mathcal{D}^{i,2}, \dots, \mathcal{D}^{i,E}\}, \quad (5.16)$$

$$\mathcal{D}^{i,e} = \{(\mathbf{x}_1^{i,e}, \mathbf{y}_1^{i,e}), \dots, (\mathbf{x}_N^{i,e}, \mathbf{y}_N^{i,e})\}, \quad e \in [E], \quad (5.17)$$

where $\mathcal{D}^{i,e}$ is the dataset for the e th DNN for client i . Let

$$\mathbf{X}^{i,e} = [\mathbf{x}_1^{i,e}, \dots, \mathbf{x}_N^{i,e}], \quad \mathbf{Y}^{i,e} = [\mathbf{y}_1^{i,e}, \dots, \mathbf{y}_N^{i,e}] \quad (5.18)$$

be the input and the label matrix for the e th DNN of client i . Then variables in (5.15) for each $i \in [J]$ are specified by,

$$\mathbf{v}^i = \{\tilde{\mathcal{W}}^{i,1}, \tilde{\mathcal{W}}^{i,2}, \dots, \tilde{\mathcal{W}}^{i,E}\}, \quad (5.19)$$

$$\tilde{\mathcal{W}}^{i,e} = \{\mathbf{W}_1^{i,e}, \mathbf{W}_2^{i,e}, \dots, \mathbf{W}_{O-1}^{i,e}\}, \quad e \in [E], \quad (5.20)$$

Algorithm 5 NCDSFL: Neural Collapse inspired Deep Supervised Federated Learning

- 1: **Initiation:** Initialize learnable parameters $\{\mathbf{v}^{i,0}, i \in [J]\}$ for all clients and $\mathbf{v}^0 = (1/[J]) \sum_{i=1}^J \mathbf{v}^{i,0}$. Generate two NC weights according to NC2 in Definition 1.
 - 2: **for** epoch index $k = 0, 1, \dots, K$ **do**
 - 3: --Local learning--
 - 4: **for** each client $i \in [J]$ in parallel **do**
 - 5: Updates its local model by $\mathbf{v}^{i,0} = \mathbf{v}^k$.
 - 6: **for** local update iteration $u = 0, 1, \dots, U$ **do**
 - 7: Update $\mathbf{v}^{i,u+1} = \mathcal{G}(\mathbf{v}^{i,u}; \mathcal{D}^i)$ locally.
 - 8: **end for**
 - 9: **end for**
 - 10: --Global aggregation--.
 - 11: The server collects $\mathbf{v}^{i,U}$ from all clients $i \in [J]$, updates the global model by $\mathbf{v}^{k+1} = (1/[J]) \sum_{i=1}^J \mathbf{v}^{i,U}$, and broadcasts it to all clients.
 - 12: **end for**
-

where $\widetilde{\mathcal{W}}^{i,e}$ collects all weights to be trained for the e th DNN of client i . Then functions in (5.15) for each i are specified by,

$$L(\mathbf{v}^i; \mathcal{D}^i) = \sum_{e \in [E]} L(\mathbf{Y}^{i,e}, \widetilde{\mathbf{Y}}^{i,e}), \quad (5.21)$$

$$R(\mathbf{v}^i; \mathcal{D}^i) = \mu \sum_{e \in [E]} \text{CE}(\mathbf{Y}^{i,e}, \mathbf{H}_{O-2}^{i,e} \widetilde{\mathbf{W}}_{NC}), \quad (5.22)$$

where $\widetilde{\mathbf{Y}}^{i,e}$ is given similar to that in (5.14), namely,

$$\widetilde{\mathbf{Y}}^{i,e} = \text{softmax}(\mathbf{W}_{NC} \sigma(\mathbf{W}_{O-1}^{i,e} \cdots \sigma(\mathbf{W}_2^{i,e} \sigma(\mathbf{W}_1^{i,e} \mathbf{X}^{i,e}))))).$$

One can observe that for each client i , we use two fixed NC weights: $\mathbf{W}_{NC}$ fixed at the penultimate hidden layer for all E primal networks and $\widetilde{\mathbf{W}}_{NC}$ fixed at the auxiliary layer for all E auxiliary networks. These two weights are irrelevant to client i and can be calculated in advance according to NC2.

5.3.3 The NCDSFL Algorithm

The overall NCDSFL algorithm is given in **Algorithm 5**, where $\mathcal{G}(\cdot; \cdot)$ in step 7 denotes a local update step based on some optimization criterion, such as gradient descent. One advantageous property of the NCDSFL algorithm is the exploration of two fixed NC weights, which helps guide and align the hidden feature distributions across clients. This promotes consistency among local models and mitigates the effects of data heterogeneity. Another key benefit lies in its reduced computational complexity and communication overhead. By fixing the auxiliary classifiers and output layer weights, the algorithm requires fewer variables to be updated and transmitted, making it more practical for real-world deployment.

5.3.4 Application to Signal Detection

In this subsection, we use the NCDSFL algorithm to train the neural receiver, which jointly estimates the channel and detects the signal based on the received pilot-and-data OFDM symbols.

The signal detection task can be formulated as a multi-binary classification problem by treating the prediction of each bit as a binary classification problem. The time-domain transmission model of the OFDM system can be formed by

$$b(t) = h(t, \tau) * a(t) + u(t), \quad (5.23)$$

where $b(t) \in \mathbb{C}$ and $a(t) \in \mathbb{C}$ denote the received signal and the time-domain transmit signal at time t , respectively, $u(t) \in \mathbb{C}$ is the AWGN at the receiver side, $\{h(t, \tau) : \tau = 0, 1, \dots, Q - 1\}$ is the channel impulse response sequence with a length of Q at time t , which is generated from the PDPs defined in the wireless world initiative for new radio (WINNER II) channel model [108]. After removing the cyclic prefix signals, $b(t)$ is passed through a discrete Fourier transform (DFT) module, and we have the following corresponding frequency-domain transmission expression of (5.23),

$$B(r) = A(r)H(r) + U(r), \quad (5.24)$$

where $B(r)$, $A(r)$, $H(r)$ and $U(r)$ are the DFT of $b(t)$, $a(t)$, $h(t, \tau)$ and $u(t)$, re-

spectively.

We adopt the commonly used multi-path channel model [108] to characterize the OFDM channels, namely,

$$h(t, \tau) = \sum_{m=0}^{M-1} \sqrt{P_m} A_m(t) \exp(-j\phi_m(t)) \delta(\tau - \tau_m), \quad (5.25)$$

where $P_m \in \mathbb{R}$ denotes the channel power gain over path m , $A_m \in \mathbb{C}$ characterizes the attenuation fluctuation and the phase shift of path l , which depends on the initial phase shift, the central frequency, the small scale fading, and the shadowing of path m , τ_m denotes the propagation delay of path m . These parameters highly depend on the PDPs of users. Note that in this work, we focus on the low-mobility case. We therefore ignore the time variance caused by the user's mobility in (5.25).

In wireless communication systems, the PDPs of different users depend on the relative positions among users, base stations, and scatterers. Therefore, in cases of low mobility, as illustrated in Fig. 5.1, each user can only observe the wireless channels characterized by one PDP within a given time period, resulting in the local observability problem. Users need to cooperate to learn a general model that works for newly joined users while preserving data privacy. We thus adopt the idea of FL to learn the general model, i.e., model (5.15), for all users in a distributed manner.

Mathematically, each user (i.e., client) i trains E DNNs to predict bits at different positions of the transmit bit stream and these E DNNs use the same training dataset with the same inputs and different labels $\mathcal{D}^{i,e} = \{(\mathbf{x}_1^i, \mathbf{y}_1^{i,e}), \dots, (\mathbf{x}_N^i, \mathbf{y}_N^{i,e})\}$, where $\mathbf{x}_n^i \in \mathbb{R}^{2d_x^i}$ and $\mathbf{y}_n^{i,e} \in \mathbb{R}^{2d_y^{i,e}}$ are calculated by received signal $b^i(t)$ and

transmit signal $A^i(r)$ as follows,

$$\begin{aligned} \mathbf{x}_n^i = & \left(\text{Re}(b^i(t_n^i)), \text{Im}(b^i(t_n^i)), \right. \\ & \text{Re}(b^i(t_n^i + 1)), \text{Im}(b^i(t_n^i + 1)), \dots, \\ & \left. \text{Re}(b^i(t_n^i + d_x^i - 1)), \text{Im}(b^i(t_n^i + d_x^i - 1)) \right)^T, \end{aligned} \quad (5.26)$$

$$\begin{aligned} \mathbf{y}_n^{i,e} = & \left(\text{Re}(A^i(r_n^{i,e})), \text{Im}(A^i(r_n^{i,e})), \right. \\ & \text{Re}(A^i(r_n^{i,e} + 1)), \text{Im}(A^i(r_n^{i,e} + 1)), \dots, \\ & \left. \text{Re}(A^i(r_n^{i,e} + d_y^{i,e} - 1)), \text{Im}(A^i(r_n^{i,e} + d_y^{i,e} - 1)) \right)^T, \end{aligned} \quad (5.27)$$

where $\text{Re}(b)$ and $\text{Im}(b)$ represent the real and imagery part of $b \in \mathbb{C}$, and $r_n^{i,e+1} = r_n^{i,e} + d_y^{i,e}$. The above formulation means that we use d_x^i received signals $\{b^i(t) : t \in t_n^i - 1 + [d_x^i]\}$ to generate the n th input $\mathbf{x}_n^i$ and $d_y^{i,e}$ transmit signals $\{A^i(r) : r \in r_n^{i,e} - 1 + [d_y^{i,e}]\}$ to generate the n th label $\mathbf{y}_n^{i,e}$ of the e th DNNs. Denote the total length of the label for user i by $d_y^i = \sum_{e \in [E]} d_y^{i,e}$. After specifying local training data $\mathcal{D}^i$ for each user i , we apply NCDSFL to solve model (5.15), yielding a solution $\mathbf{v}_*^i \approx \mathbf{v}_*, i \in [J]$, where $\mathbf{v}_*^i$ represents all the weights of the E DNNs for client i . These trained networks can then be used to detect newly emerged signals based on the learned representations.

5.4 Simulation Results

In this section, we conduct some experiments on joint channel estimation and symbol detection to evaluate the performance of the proposed NCDSFL algorithm. We adopt the B1 non-line-of-sight (NLoS) scenario from the WINNER II model [108] as the channel generator. Specifically, the channel consists of 24 propagation paths. The carrier frequency is set to 2.6 GHz with a bandwidth of 20 MHz. We use typical urban channel conditions with a maximum delay of 16 sampling periods and a shadow fading standard deviation of 4 dB. The delay spread follows a log-normal distribution with a mean of $-7.12(\log_{10}[s])$ and a standard deviation of $0.12(\log_{10}[s])$, where s represents ‘second’.

There are 10 users cooperating to train a general model so that newly joined users can also use this model without retraining. Client users only have access to their own data, and they can only observe a partial wireless environment. Therefore, we use the same PDP to generate 500 channels for the same user with randomly generated small scale channel parameters. These channel realizations are then fixed as a local dataset during the whole training process. The transmit data bit streams are randomly generated for each local training iteration, while the pilot bits are fixed in both training and testing phases. We test the bit error rate (BER) performance of the trained model on a general testing dataset generated from all PDPs with different receiving Signal-to-Noise-Ratios (SNRs).

For each user $i \in [10]$, $E = 4$ fully connected DNNs are trained, and each DNN consists of three hidden layers with 500, 250, and 128 neurons, using the ReLU function as activation function σ . The n th sample $(\mathbf{x}_n^i, \mathbf{y}_n^i)$ is generated as (5.26) and (5.27) with $d_x^i = 128$ (including pilot signal) and $d_y^{i,e} = 16$. In Algorithm 5, Each local training epoch consists of 50 local iterations to update $\mathbf{v}^{i,u+1} = \mathcal{G}(\mathbf{v}^{i,u}; \mathcal{D}^i)$ using the RMSprop optimizer². The regularization coefficient in (5.14) is set as $\mu = 0.5$.

We compare the proposed NCDSFL algorithm with two baselines: (1) the FedAvg-

²RMSprop is an unpublished adaptive learning rate algorithm proposed by Geoff Hinton in Lecture 6e of his Coursera Class.

based signal detector [40], denoted as FL, and (2) the independent learning-based signal detector, denoted as IL. Both baselines use the same learning hyperparameters as NCDSFL. We also compare our method with the conventional minimum mean squared error (MMSE) signal detection algorithm, which uses MMSE-based channel estimates to recover the transmitted symbols.

5.4.1 Validation of Theorem 5.2.1

We first validate the NC phenomenon in the signal detection scenario. In this experiment, we focus on the training process of a single client without the FL manner. To validate Theorem 5.2.1, we track two quantities during the training process,

$$\begin{aligned}\theta(\mathbf{W}_0, \mathbf{W}_1) &= \left\| \frac{I\mathbf{W}_0^T\mathbf{W}_0}{\|\mathbf{W}_0^T\mathbf{W}_0\|} - \mathbf{I} \right\| + \left\| \frac{I\mathbf{W}_1^T\mathbf{W}_1}{\|\mathbf{W}_1^T\mathbf{W}_1\|} - \mathbf{I} \right\|, \\ \vartheta(\mathbf{W}) &= \left\| \frac{\text{vec}(\mathbf{H})}{\|\text{vec}(\mathbf{H})\|} - \frac{[\mathbf{A}^T, -\mathbf{A}^T] \text{vec}(\mathbf{W})}{\|[\mathbf{A}^T, -\mathbf{A}^T] \text{vec}(\mathbf{W})\|} \right\|,\end{aligned}\tag{5.28}$$

where $\mathbf{W}_0 = [\mathbf{w}_{1,0}, \dots, \mathbf{w}_{I,0}]$ and $\mathbf{W}_1 = [\mathbf{w}_{1,1}, \dots, \mathbf{w}_{I,1}]$. The results are shown in Figs. 5.3 and 5.4.

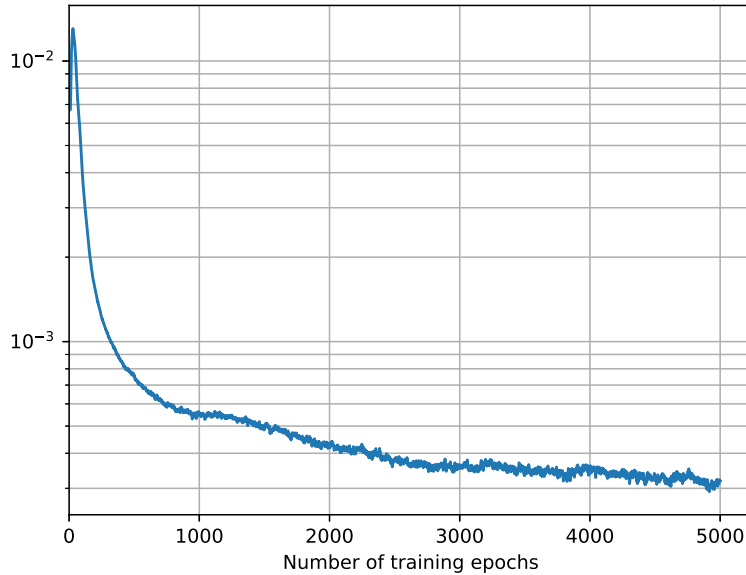


Figure 5.3: $\theta(\mathbf{W}_0, \mathbf{W}_1)$ v.s. epochs

The two figures show that both θ and ϑ decrease monotonically throughout the training process. The decreasing trend of θ suggests that the orthogonality among classifiers in the parallel binary classification tasks strengthens as training progresses. Similarly, the decline in ϑ indicates an increasing duality between the hidden features and their corresponding classifiers. The behavior of θ supports our approach of employing fixed orthogonal classifiers, while the trend in ϑ implies that the hidden features gradually converge toward the subspace spanned by these orthogonal classifiers. The convergence underpins the feature alignment achieved via deep supervision using NC weights. Together, these confirm the presence of the NC phenomenon in the signal detection task.

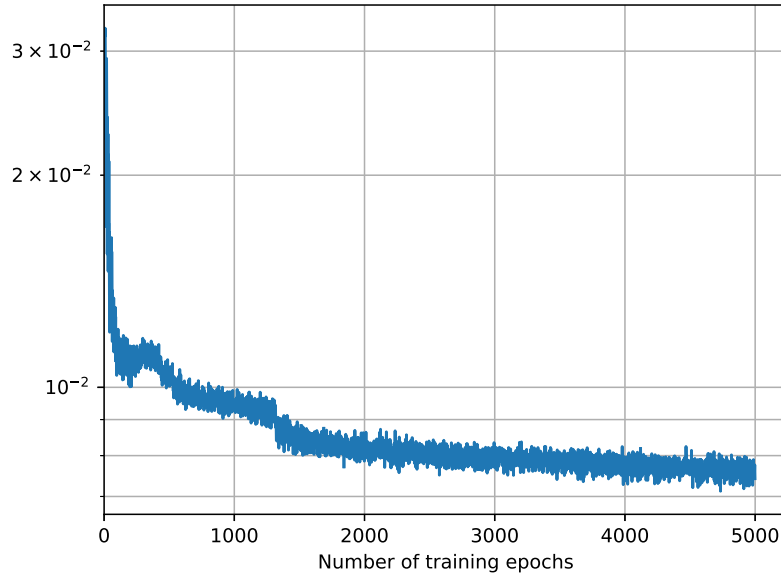


Figure 5.4: $\vartheta(\mathbf{W})$ v.s. epochs

5.4.2 Performance Comparison with Baselines

Next, we compare the considered algorithms in our aforementioned FL learning scenario with a received signal SNR of 10dB. Fig. 5.5 shows the BER performance on the validation dataset versus the number of training epochs. We can observe that the IL algorithm has the slowest convergence speed and worst

performance, because each client only has partial observations of the wireless environment and cannot learn a general model for a general validation dataset. Our proposed NCDSFL algorithm and the FL algorithm have the same convergent BER performance. However, our proposed NCDSFL algorithm converges much faster than an FL algorithm. The NCDSFL algorithm converges at about 60 training epochs, while the FL algorithm converges at about 150 training epochs. This is because the hidden features of different clients are guided to a similar distribution defined by the NC weights as stated in Theorem 5.2.1, which alleviates the impact of client drift effect and data discrepancy. Therefore, the NCDSFL algorithm can save communication rounds.

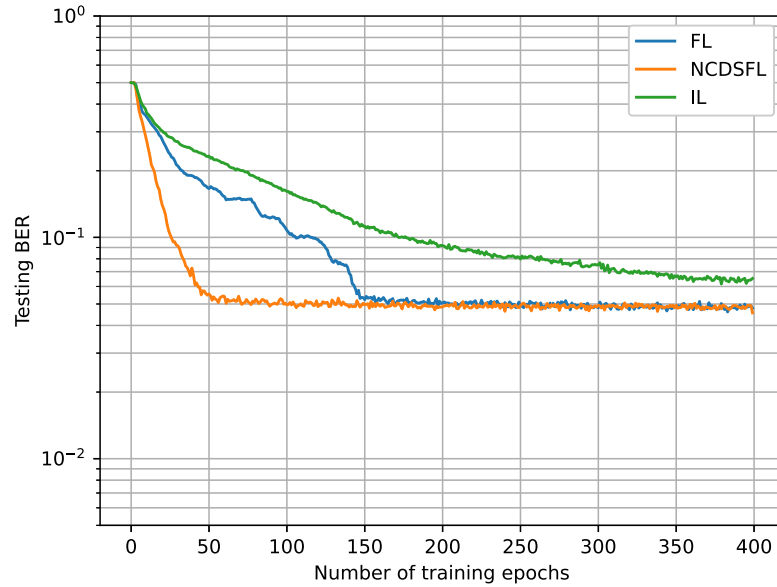


Figure 5.5: Testing BER v.s. epochs (SNR=10dB)

In wireless systems, the large scale fading parameters, which highly depend on the relative positions of transmitters, receivers, and scatters, mainly determine the SNR of the received signal, even if the transmit power is adjustable. Therefore, we simulate a more practical scenario, where different client users receive signals with different SNRs. In this experiment, the SNRs are picked from $\{0, 5, 10, 15, 20\}dB$, with each SNR value corresponding to two client users.

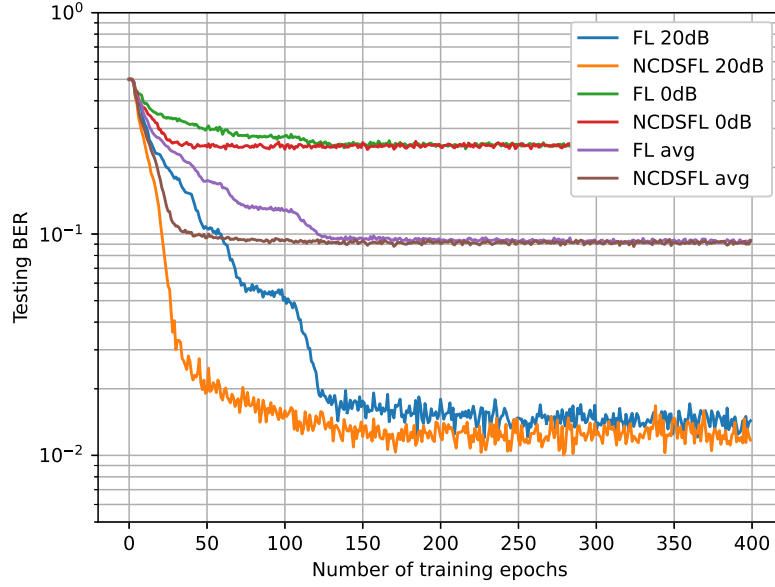


Figure 5.6: Testing BER v.s. epochs

Fig. 5.6 draws the testing BER curves of the FL algorithm and the NCDSFL algorithm under different SNR conditions during the training process. We can observe from the figure that our proposed NCDSFL algorithm converges faster than an FL algorithm. In addition, as the client heterogeneity increases, the convergent BER of our proposed NCDSFL algorithm also outperforms the FL algorithm. This validates the effectiveness of aligning clients by NC based DS in FL. We summarize the testing performance of different algorithms versus SNR in Fig. 5.7. We can observe that our proposed NCDSFL algorithm always has equal or better testing performance than the FL algorithm. Moreover, the testing performance gap between the FL algorithm and the NCDSFL algorithm enlarges with the received signal SNR value. In addition, both the FL algorithm and the NCDSFL algorithm outperform the conventional MMSE baseline. This is because, due to a limited number of pilots, the MMSE algorithm cannot estimate the channels accurately and thus has a worse BER performance, while both the FL algorithm and the NCDSFL algorithm can learn the implicit OFDM channel structure from the received signals.

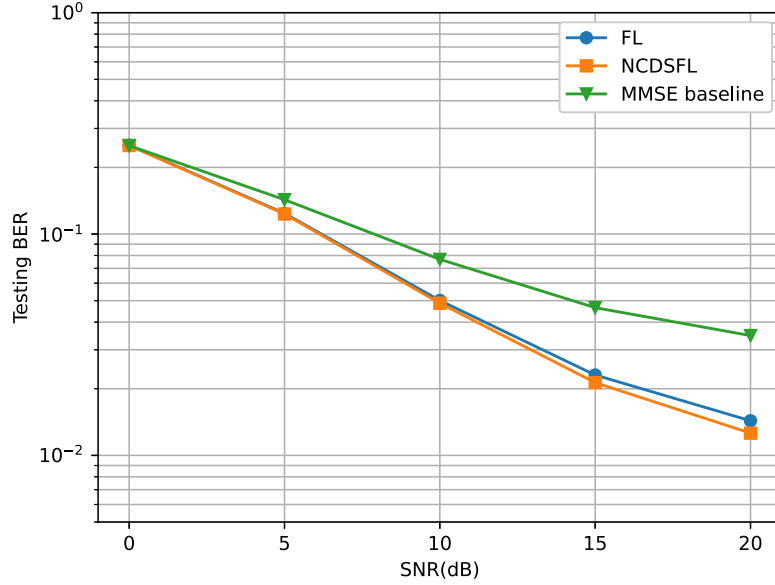


Figure 5.7: Testing BER v.s. SNR

In the next experiment, we further investigate the impact of the LoS path. We add a LoS path to each client's channel and set different Rician factors for different users. The SNRs are set the same as in the last experiment, while the Rician factors are picked from $\{-5, 0, 5, 10, 15\}dB$, with each value corresponding to two client users. Fig. 5.8 draws the validation BER curve of different algorithms with different SNRs. We can observe that our proposed NCDSFL algorithm still converges faster than the FL algorithm. Besides, the BER performance of all algorithms is improved as the presence of LoS paths makes the signal detection problem easier to solve. Fig. 5.9 draws the testing BER performance of different algorithms with different SNRs. We can observe that the testing performance of all algorithms is improved due to the presence of LoS path, and the learning based methods outperform the MMSE baseline due to their ability to learn the implicit information from the training data.

In the final experiment, we introduce 5 additional clients whose channels are Rician channels with a Rician factor of 0dB. The SNR is set to 10dB. Fig. 5.10 draws the BER performance on WINNER II dataset and on Rician dataset of the

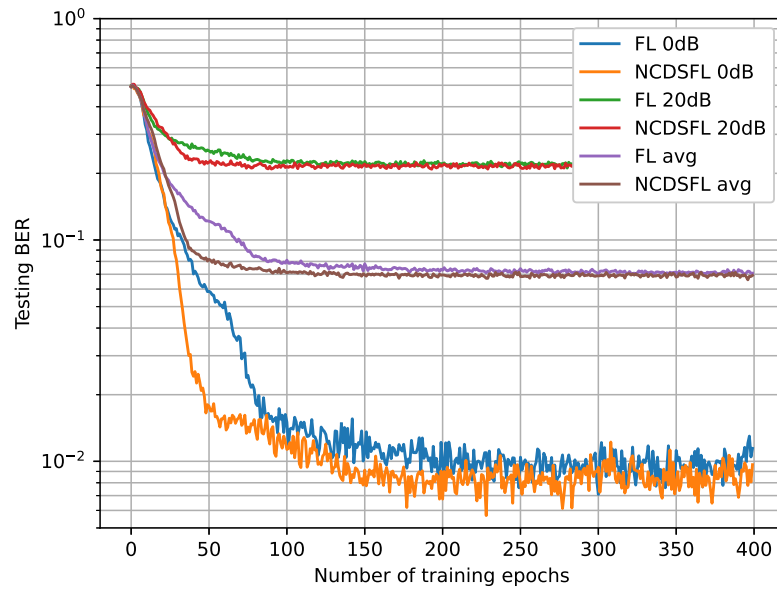


Figure 5.8: Testing BER v.s. epochs

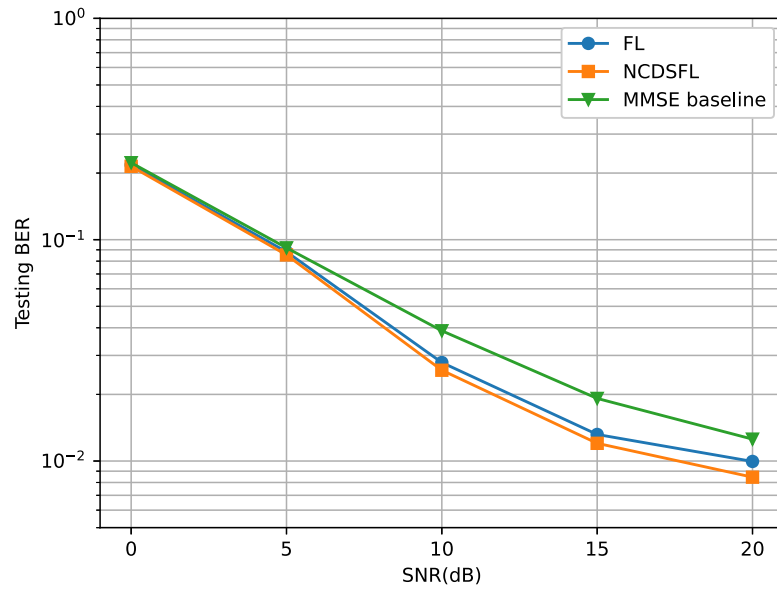


Figure 5.9: Testing BER v.s. SNR

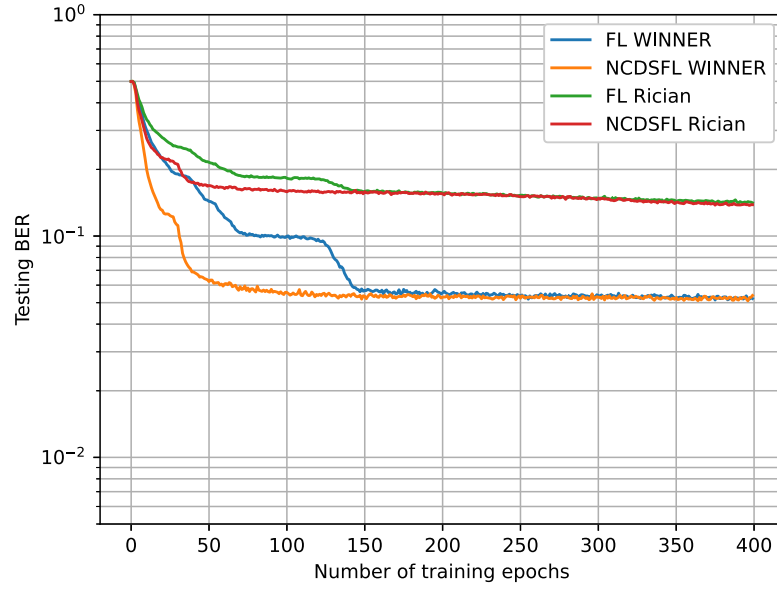


Figure 5.10: Testing BER v.s. epochs (SNR=10dB)

trained model during the training process. We can observe that our proposed NCDSFL algorithm converges faster than the FL algorithm, due to the more efficient aggregation of the NCDSFL algorithm. Moreover, the performance of Rician channels is worse than the WINNER II channel. In Rician channels, except for the LoS path, the NLoS components are irrelevant. Therefore, with a limited number of pilots, it is more difficult to estimate the channels across all subcarriers for Rician channels compared with WINNER II channels.

5.5 Conclusion

Based on the NC phenomenon, A NCDSFL algorithm was proposed to aggregate knowledge from client users while preserving data privacy, which employed the NC weights as auxiliary classifiers to locally supervise the intermediate hidden features of the DNNs to mitigate the data heterogeneity. The NC phenomenon of multi-binary classification problems was given and theoretically proved. The simulation results on signal detection under various channel conditions show that the FL scheme addresses the local observability issue and our proposed NCDSFL improves the aggregation efficiency and thus has a faster convergence speed than the baseline algorithm. Beyond this specific application, the NCDSFL algorithm holds potential for broader use in other domains, such as computer vision, which warrants further investigation in future research.

5.6 Proof of Theorem 5.2.1

5.6.1 Useful Lemmas

Lemma 5.6.1. *Matrix $\mathbf{A}$ is row-orthogonal and has an equal norm of $\sqrt{K2^I}$ for each row.*

Proof. Since $\mathbf{C}^i(\mathbf{C}^i)^T = 2^i\mathbf{I}$, each matrix $\mathbf{C}^i, i = 1, 2, \dots, I$ is row-orthogonal with equal norm row vectors. Now, we prove $\mathbf{B}^i$ for each $i = 1, 2, \dots, I$ is row-orthogonal by mathematical induction. When $i = 1$, $\mathbf{B}^1(\mathbf{B}^1)^T = 2^1\mathbf{I}$. Assume that $\mathbf{B}^i$ satisfies $\mathbf{B}^i(\mathbf{B}^i)^T = 2^i\mathbf{I}$ for any $i = 1, 2, \dots, I-1$. For $i = I$, it follows from (5.8) that

$$\begin{aligned}\mathbf{B}^I(\mathbf{B}^I)^T &= \begin{bmatrix} 2\mathbf{B}^{I-1}(\mathbf{B}^{I-1})^T & \mathbf{0} \\ \mathbf{0} & 2\mathbf{C}^{I-1}(\mathbf{C}^{I-1})^T \end{bmatrix}, \\ &= \begin{bmatrix} 2^I\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 2^I\mathbf{I} \end{bmatrix} = 2^I\mathbf{I}.\end{aligned}\tag{5.29}$$

Using this fact and (5.7), we have

$$\mathbf{A}\mathbf{A}^T = K\mathbf{B}^I(\mathbf{B}^I)^T = K2^I\mathbf{I},\tag{5.30}$$

finishing the proof. $\square$

For simplicity, let $\mathbf{I}_{dI} \in \mathbb{R}^{dI \times dI}$ and

$$\begin{aligned}\mathbf{h} &= \text{vec}(\mathbf{H}), \\ \mathbf{w} &= [\mathbf{I}_{dI}, -\mathbf{I}_{dI}]\text{vec}(\mathbf{W}) \\ &= \left[(\mathbf{w}_{1,0} - \mathbf{w}_{1,1})^T, \dots, (\mathbf{w}_{I,0} - \mathbf{w}_{I,1})^T \right]^T.\end{aligned}\tag{5.31}$$

Based on definitions of $\mathbf{H}$, $\mathbf{W}$, and $\mathbf{w}$ in (5.4) and (5.31), one can represent P as a bilinear function of $\mathbf{w}$ and $\mathbf{h}$ as follows,

$$P = \sum_{k=1}^K \sum_{i=1}^I \sum_{s \in \mathcal{S}} \phi_{kis}(\mathbf{W}, \mathbf{H}) = \mathbf{w}^T \mathbf{A} \mathbf{h},\tag{5.32}$$

where $\mathbf{A}$ is defined in (5.7).

Lemma 5.6.2. For P defined in (5.32), it holds

$$P \geq -\sqrt{K2^{I-1}}(\|\mathbf{W}\|^2 + \|\mathbf{H}\|^2). \quad (5.33)$$

The equality holds iff the following conditions are met:

- 1) $\mathbf{w} = -c\mathbf{A}\mathbf{h}$ for some constant $c > 0$;
- 2) $\mathbf{h}$ is a linear combination of row vectors of $\mathbf{A}$;
- 3) $\mathbf{w}_{i,0} + \mathbf{w}_{i,1} = \mathbf{0}, \forall i$;
- 4) $\|\mathbf{W}\| = \|\mathbf{H}\|$.

Proof. Direct verification leads to

$$\begin{aligned} P = \mathbf{w}^T \mathbf{A} \mathbf{h} &\stackrel{(a)}{\geq} -\|\mathbf{w}\| \|\mathbf{A} \mathbf{h}\| \\ &\stackrel{(b)}{\geq} -\sqrt{K2^I} \|\mathbf{w}\| \|\mathbf{h}\| \stackrel{(c)}{=} -\sqrt{K2^I} \|\mathbf{w}\| \|\mathbf{H}\| \\ &\stackrel{(d)}{\geq} -\sqrt{2K2^I} \|\mathbf{W}\| \|\mathbf{H}\| \\ &\stackrel{(e)}{\geq} -\sqrt{K2^{I-1}}(\|\mathbf{W}\|^2 + \|\mathbf{H}\|^2), \end{aligned} \quad (5.34)$$

where (a) is because $\|\mathbf{a}\mathbf{b}\| \geq -\|\mathbf{a}\| \|\mathbf{b}\|$, and the equality in (a) holds iff $\mathbf{a} = -c\mathbf{b}$ for some constant $c > 0$, namely, condition 1) holds; (b) is due to $\|\mathbf{A}\mathbf{h}\| \leq \|\mathbf{A}\|_2 \|\mathbf{h}\|$ and $\|\mathbf{A}\|_2 = \sqrt{K2^I}$ from Lemma 5.6.1, where $\|\mathbf{A}\|_2$ is the spectral norm of $\mathbf{A}$, and the equality in (b) holds iff condition 2) is met. (c) is from the definition of $\mathbf{H}$ in (5.4); (d) is from definitions of $\mathbf{W}$ and $\mathbf{w}$ in (5.4) and (5.31), and the equality in (d) holds iff condition 3) is met. The equality in (e) holds iff condition 4) is met. $\square$

Lemma 5.6.3. If $\phi_{kis}(\mathbf{W}, \mathbf{H}) = c_1, \forall s \in \mathcal{S}, \forall i$ for a constant $c_1 \leq 0$, and the equality conditions of Lemma 5.6.2 hold, then the NC conditions in Definition 1 hold.

Proof. $\mathbf{h}$ is a linear combination of row vectors of matrix $\mathbf{A}$. Together with definition (5.31) of $\mathbf{h}$ and definition (5.7) of $\mathbf{A}$, we can obtain NC1, the hidden features collapse to their corresponding feature means.

Then we prove NC2. Based on condition 2) from Lemma 5.6.2, there is $\mathbf{z}$ such that $\mathbf{h} = \mathbf{A}^T \mathbf{z}$, which by condition 1) from Lemma 5.6.2 results in

$$\mathbf{w} = -c\mathbf{A}\mathbf{h} = -c\mathbf{A}\mathbf{A}^T \mathbf{z} = -cK2^I \mathbf{z}. \quad (5.35)$$

where the last equation is from (5.30). This suffices to

$$\mathbf{h} = \mathbf{A}^T \mathbf{z} = -\frac{1}{cK2^I} \mathbf{A}^T \mathbf{w}. \quad (5.36)$$

By denoting $\Delta \mathbf{w}_i = \mathbf{w}_{i,1} - \mathbf{w}_{i,0}$, based on the definitions of $\mathbf{A}$ and $\mathbf{w}$ in (5.7) and (5.31), the above condition yields

$$\mathbf{h}_s^{(k)} = \frac{1}{cK2^I} \sum_{i=1}^I (2s_i - 1) \Delta \mathbf{w}_i. \quad (5.37)$$

Then the inner product condition becomes, for $\forall s, i$

$$\begin{aligned} c_1 &= \phi_{kis}(\mathbf{W}, \mathbf{H}) \\ &= \langle -(2s_i - 1) \Delta \mathbf{w}_i, \mathbf{h}_s^{(k)} \rangle \\ &= -\frac{1}{cK2^I} \left\langle (2s_i - 1) \Delta \mathbf{w}_i, \sum_j (2s_j - 1) \Delta \mathbf{w}_j \right\rangle. \end{aligned} \quad (5.38)$$

Consider two binary sequences s and t , which are same except for the k -th position, i.e., $s_i = t_i, \forall i \neq k$ and $s_k + t_k = 1$. We have the following equalities for $\forall i$,

$$-\frac{1}{cK2^I} \left\langle (2s_i - 1) \Delta \mathbf{w}_i, \sum_{j=1}^I (2s_j - 1) \Delta \mathbf{w}_j \right\rangle = c_1, \quad (5.39)$$

$$-\frac{1}{cK2^I} \left\langle (2t_i - 1) \Delta \mathbf{w}_i, \sum_{j=1}^I (2t_j - 1) \Delta \mathbf{w}_j \right\rangle = c_1. \quad (5.40)$$

For $\forall i \neq k$, subtracting the above two equations leads to

$$(2t_i - 1) \langle \Delta \mathbf{w}_i, 2(s_k - t_k) \Delta \mathbf{w}_k \rangle = 0. \quad (5.41)$$

indicating that

$$\langle \Delta \mathbf{w}_i, \Delta \mathbf{w}_k \rangle = 0, \forall i \neq k. \quad (5.42)$$

Together with condition 3) of Lemma 5.6.2, we can conclude that $\{\mathbf{w}_{i,0}\}$ and $\{\mathbf{w}_{i,1}\}$ are two sets of orthogonal vectors. Furthermore, by applying the orthogonality to (5.38), we have

$$-\frac{1}{cK2^I}(2s_i - 1)^2 \langle \Delta \mathbf{w}_i, \Delta \mathbf{w}_i \rangle = c_1, \quad (5.43)$$

which implies that

$$\|\Delta \mathbf{w}_i\|^2 = -cc_1K2^I. \quad (5.44)$$

This completes the proof of NC2. Finally, NC3 is obtained by (5.36). This completes the proof. $\square$

5.6.2 Proof of Theorem 5.2.1

As function $f(x) = \ln(1 + \exp(x))$ is convex, we can apply Jensen's inequality to the following function,

$$\begin{aligned} l(\mathbf{W}, \mathbf{H}) &= \frac{1}{KI2^I} \sum_{k=1}^K \sum_{i=1}^I \sum_{s \in \mathcal{S}} \ln \left(1 + \exp \left(\phi_{kis}(\mathbf{W}, \mathbf{H}) \right) \right) \\ &\geq \ln \left(1 + \exp \left(\frac{1}{KI2^I} \sum_{k=1}^K \sum_{i=1}^I \sum_{s \in \mathcal{S}} \phi_{kis}(\mathbf{W}, \mathbf{H}) \right) \right) \\ &= \ln \left(1 + \exp \left(\frac{P}{KI2^I} \right) \right), \end{aligned} \quad (5.45)$$

where the equality in ' $\geq$ ' holds iff $\phi_{kis}(\mathbf{W}, \mathbf{H}) = c_1, \forall s \in \mathcal{S}, \forall i$. Let $\rho = \|\mathbf{W}\|^2 + \|\mathbf{H}\|^2$. From Lemma 5.6.2, it holds

$$P \geq -\sqrt{K2^{I-1}}\rho. \quad (5.46)$$

Using the above facts, we have

$$\begin{aligned}
L(\mathbf{W}, \mathbf{H}) &= l(\mathbf{W}, \mathbf{H}) + \lambda\rho \\
&\geq \ln \left(1 + \exp \left(\frac{P}{KI2^I} \right) \right) + \lambda\rho \\
&\geq \ln (1 + \exp (-t\rho)) + \lambda\rho = L(\rho),
\end{aligned} \tag{5.47}$$

where $t = 1/(I\sqrt{2K2^I})$. It is clear that $L(\rho)$ is a convex function of ρ and problem $\min_{\rho} L(\rho)$ has zero solution if $\lambda \geq t/2$. In other words, problem $\min_{\rho} L(\rho)$ admits a non-trivial global minimizer if $0 < \lambda < t/2$. Using the first-order optimality condition, $L'(\rho_{opt}) = 0$, one can find optimal solution ρ_{opt} by

$$\rho_{opt} = \frac{1}{t} \ln \frac{t - \lambda}{\lambda}. \tag{5.48}$$

Then we have the following inequality,

$$L(\mathbf{H}, \mathbf{W}) \geq L(\rho_{opt}), \tag{5.49}$$

which indicates that problem (5.5) also admits non-trivial global minimizers. Recall the sufficient and necessary conditions for the global minimizers, i.e., $\phi_{kis}(\mathbf{W}, \mathbf{H}) = c_1, \forall s \in \mathcal{S}, \forall i$ and conditions in Lemma 5.6.2. Based on Lemma 5.6.3, we obtain Theorem 5.2.1. $\square$

Chapter 6

Summary and Future Directions

6.1 Summary

This thesis has explored distributed training algorithms for native intelligent wireless networks from four different perspectives.

In Chapter 2, we have proposed a DTE MADRL-based power control scheme to allocate transmit power in HetNets. To promote cooperation among agents, the PQL algorithm is proposed, which introduces a regularization term in the local Q loss function to penalize the low-reward state-action pairs. In this way, agents tend to learn the high-reward state-action pairs first and thus their policy updating speed slows down, which results in a more stable learning process. This PQL algorithm is then employed to train the MADRL system, which decides the transmit power for each AP simultaneously based on local observation. Simulation results show that with limited network layers and local observation, our proposed scheme is able to reach over 90% performance of the conventional WMMSE algorithm and outperforms a series of existing DTE learning algorithms under various system settings.

To alleviate the local observability issue, we employ FL to train the MARL powered intelligent V2X systems, which leads to FRL frameworks. In Chapter 3, we have proposed a novel policy optimization based FRL framework, namely PASM algorithm, which incorporates the advanced optimization algorithm, inexact ADMM, and the second moment. The former enables agents to gradually improve their policies, while the latter enables an adaptive learning rate to speed up and improve the training process. The convergence of the PASM algorithm is theoretically established. The PASM is then implemented in the considered V2X networks to train agents to distributively optimize the V2V package delivery rate and the system weighted sum-throughput. Numerical experiments have confirmed that the FL approach accelerates agent training by knowledge sharing and that the proposed PASM algorithm has faster convergence and around 20% weight sum rate gain in the intelligent V2X network resource allocation scenario.

Moreover, FRL is improved via model alignment in Chapter 4. We exploit the natural RI property of ReLU activated DNNs to rescale the DNNs at each

clients before and after the aggregation step to align the client models. The rescale operation can reduce the model variance and thus alleviate the client drift effect without additional communication overhead. The proposed RIFRL algorithm is then implemented to train the intelligent V2X networks. Simulation results showed that the proposed RIFRL had 3 times faster convergence speed and improved the testing performance by around 4%, compared with the classical FedAvg based FRL baseline in the V2X resource allocation scenario.

Finally, we improve the FL algorithm by hidden feature alignment in Chapter 4. We developed the NCDSFL algorithm by integrating the NC solutions with the DS scheme within an FL framework. This approach not only enables the effective aggregation of local knowledge from clients but also mitigates the impact of data heterogeneity, thereby accelerating convergence. Simulation results on joint channel estimation and symbol detection in OFDM systems showed that the NCDSFL algorithm had around 3 times faster convergence compared with the FedAvg baseline and thus successfully cut communication rounds.

6.2 Future Directions

Chapters 3 to 5 have introduced three different FL algorithms addressing FL from complementary perspectives, i.e., optimization framework, model alignment, and feature alignment. Therefore, a natural research direction is to combine these three methods together to obtain a cumulative gain. This combined method can be used to address the FL problems targeting at training ReLU activated DNN model with a specific NC form. The formulated training problem has the following typical FL form:

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{x}_i \in \mathcal{F}} \quad & \sum_i (L(g(\mathcal{D}_i; \mathbf{x}_i); \mathcal{D}_i) + R(g(\mathcal{D}_i; \mathbf{x}_i); \mathcal{D}_i)) \\ \text{s.t.} \quad & \mathbf{x} = \mathbf{x}_i, \forall i, \end{aligned} \tag{6.1}$$

where $L(., .)$ is the local loss function which depends on the output of the DNN and the local dataset $\mathcal{D}_i$. $\mathcal{F}$ is the feasible region. $g(\mathcal{D}; \mathbf{x})$ denotes the ReLU-activated DNN function with $\mathbf{x}$ being the trainable parameters and $\mathcal{D}$ being the corresponding input dataset. $R(., .)$ denotes the regularizer used to supervise the intermediate features. The PASM framework will be used as the optimization framework to solve problem (6.1), while the NC regularizer in Chapter 5 will be used as R in problem (6.1). Finally, the backward rescale invariant operation (BRIO) will be applied to the ReLU activated DNN directly before and after the aggregation step as in Chapter 4. This resulted FL algorithm can be applied to many FL based classification problems in wireless networks, such as the signal detection problem in Chapter 5.

Another promising research direction is training large language models (LLMs) for wireless networks. LLMs such as ChatGPT have recently attracted intense research attention because they exhibit near-human reasoning and language abilities. Integrating LLMs into future wireless networks would allow base stations and edge devices to interpret semantic-level information and deliver highly personalized, context-aware services that elevate user quality of service. FL can protect user privacy by keeping raw data local, but efficiently training LLMs under FL constraints remains an open problem. It is prohibitively expensive to transmit full parameters of an LLM due to its huge number of pa-

rameters. To address this issue, split learning (SL) is a promising solution [109]. Specifically, multiple clients split the model with a server segment and periodically federate the client segments. In this SL-based FL system, our previously introduced BRIO (Chapter 4) and PASM (Chapter 3) can still be applied, i.e., BRIO to rescale ReLU-activated modules, and PASM as the optimization backbone. Upcoming work should therefore focus on FL algorithms that are LLM-specific, featuring lower communication overhead and reduced on-device update complexity and making FL training for LLMs practical for wireless edge deployments.

References

- [1] 3GPP Technical Specification Group Radio Access Network. Study on LTE-based V2X services; (Release 14). Technical Report TR 36.885, 3GPP, June 2016.
- [2] Harri Holma and Antti Toskala. *WCDMA for UMTS: Radio access for third generation mobile communications*. John Wiley & Sons, 2005.
- [3] Stephen Weinstein and Paul Ebert. Data transmission by frequency-division multiplexing using the discrete Fourier transform. *IEEE transactions on Communication Technology*, 19(5):628–634, 2003.
- [4] Gerard J Foschini. Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. *Bell labs technical journal*, 1(2):41–59, 1996.
- [5] Theodore S Rappaport, Shu Sun, Rimma Mayzus, Hang Zhao, Yaniv Azar, Kevin Wang, George N Wong, Jocelyn K Schulz, Mathew Samimi, and Felix Gutierrez. Millimeter wave mobile communications for 5g cellular: It will work! *IEEE access*, 1:335–349, 2013.
- [6] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A vision of 6g wireless systems: Applications, trends, technologies, and open research problems. *IEEE network*, 34(3):134–142, 2019.
- [7] Zhi-Quan Luo and Shuzhong Zhang. Dynamic spectrum management: Complexity and duality. *IEEE journal of selected topics in signal processing*, 2(1):57–73, 2008.
- [8] Qingjiang Shi, Meisam Razaviyayn, Zhi-Quan Luo, and Chen He. An iteratively weighted mmse approach to distributed sum-utility maximization for a mimo interfering broadcast channel. *IEEE Transactions on Signal Processing*, 59(9):4331–4340, 2011.
- [9] Kaiming Shen and Wei Yu. Fractional programming for communication systems—part i: Power control and beamforming. *IEEE Transactions on Signal Processing*, 66(10):2616–2630, 2018.

- [10] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.
- [11] Yaohua Sun, Mugen Peng, Yangcheng Zhou, Yuzhe Huang, and Shiwen Mao. Application of machine learning in wireless networks: Key techniques and open issues. *IEEE Communications Surveys & Tutorials*, 21(4):3072–3108, 2019.
- [12] Quoc-Viet Pham, Nhan Thanh Nguyen, Thien Huynh-The, Long Bao Le, Kyungchun Lee, and Won-Joo Hwang. Intelligent radio signal processing: A survey. *IEEE Access*, 9:83818–83850, 2021.
- [13] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. Power of deep learning for channel estimation and signal detection in ofdm systems. *IEEE Wireless Communications Letters*, 7(1):114–117, 2017.
- [14] Xuemei Yi and Caijun Zhong. Deep learning for joint channel estimation and signal detection in ofdm systems. *IEEE Communications Letters*, 24(12):2780–2784, 2020.
- [15] Hao Ye, Le Liang, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning-based end-to-end wireless communication systems with conditional gans as unknown channels. *IEEE Transactions on Wireless Communications*, 19(5):3133–3143, 2020.
- [16] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning based end-to-end wireless communication systems without pilots. *IEEE Transactions on Cognitive Communications and Networking*, 7(3):702–714, 2021.
- [17] Mikko Honkala, Dani Korpi, and Janne MJ Huttunen. Deeprx: Fully convolutional deep learning receiver. *IEEE Transactions on Wireless Communications*, 20(6):3925–3940, 2021.
- [18] Ian Goodfellow. Deep learning, 2016.
- [19] Mengyuan Lee, Yuanhao Xiong, Guanding Yu, and Geoffrey Ye Li. Deep neural networks for linear sum assignment problems. *IEEE Wireless Communications Letters*, 7(6):962–965, 2018.
- [20] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [21] Fei Liang, Cong Shen, Wei Yu, and Feng Wu. Towards optimal power control via ensembling deep neural networks. *IEEE Transactions on Communications*, 68(3):1760–1776, 2019.

- [22] Qiyu Hu, Yunlong Cai, Qingjiang Shi, Kaidi Xu, Guanding Yu, and Zhi Ding. Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser mimo systems. *IEEE Transactions on Wireless Communications*, 20(2):1394–1410, 2020.
- [23] Arindam Chowdhury, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. Unfolding wmmse using graph neural networks for efficient power allocation. *IEEE Transactions on Wireless Communications*, 20(9):6004–6017, 2021.
- [24] Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Qihui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24(2):1240–1279, 2022.
- [25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [26] Yasar Sinan Nasir and Dongning Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on selected areas in communications*, 37(10):2239–2250, 2019.
- [27] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang. Deep reinforcement learning based resource allocation for v2v communications. *IEEE Trans. Veh. Technol.*, 68(4):3163–3173, 2019.
- [28] Le Liang, Hao Ye, and Geoffrey Ye Li. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE J. Sel. Areas Commun.*, 37(10):2282–2292, 2019.
- [29] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 1146–1155. PMLR, 2017.
- [30] Ziyang He, Liang Wang, Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang. Resource allocation based on graph neural networks in vehicular communications. In *2020 IEEE Glob. Commun. Conf.*, pages 1–5, 2020.
- [31] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.*, 12, 1999.
- [32] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

- [33] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Int. Conf. Mach. Learn.*, pages 387–395. Pmlr, 2014.
- [34] Yasar Sinan Nasir and Dongning Guo. Deep reinforcement learning for joint spectrum and power allocation in cellular networks. In *2021 IEEE globecom workshops (GC wkshps)*, pages 1–6. IEEE, 2021.
- [35] Tao Zhang, Kun Zhu, Shaoqiu Zheng, Dusit Niyato, and Nguyen Cong Luong. Trajectory design and power control for joint radar and communication enabled multi-uav cooperative detection systems. *IEEE Transactions on Communications*, 71(1):158–172, 2022.
- [36] Lin Zhang and Ying-Chang Liang. Deep reinforcement learning for multi-agent power control in heterogeneous networks. *IEEE Transactions on Wireless Communications*, 20(4):2551–2564, 2020.
- [37] Khoi Khac Nguyen, Trung Q. Duong, Ngo Anh Vien, Nhien-An Le-Khac, and Long D. Nguyen. Distributed deep deterministic policy gradient for power allocation control in d2d-based v2v communications. *IEEE Access*, 7:164533–164543, 2019.
- [38] Prajwalita Saikia, Sonia Pala, Keshav Singh, Sandeep Kumar Singh, and Wan-Jen Huang. Proximal policy optimization for ris-assisted full duplex 6g-v2x communications. *IEEE Trans. Intell. Veh.*, pages 1–16, 2023.
- [39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artif. Intell. Stat.*, pages 1273–1282. PMLR, 2017.
- [41] Zhijin Qin, Geoffrey Ye Li, and Hao Ye. Federated learning and wireless communications. *IEEE Wireless Communications*, 28(5):134–140, 2021.
- [42] Ouya Wang, Shenglong Zhou, and Geoffrey Ye Li. New environment adaptation with few shots for ofdm receiver and mmwave beamforming. *arXiv preprint arXiv:2310.12343*, 2023.
- [43] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
- [44] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Merouane Debbah. Federated learning for ultra-reliable low-latency v2v communications. In *2018 IEEE Glob. Commun. Conf.*, pages 1–7, 2018.

- [45] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.
- [46] Ziyang Lu, Chen Zhong, and M Cenk Gursoy. Dynamic channel access and power control in wireless interference networks via multi-agent deep reinforcement learning. *IEEE Trans. Veh. Technol.*, 71(2):1588–1601, 2021.
- [47] Xiang Li, Lingyun Lu, Wei Ni, Abbas Jamalipour, Dalin Zhang, and Haifeng Du. Federated multi-agent deep reinforcement learning for resource allocation of vehicle-to-vehicle communications. *IEEE Trans. Veh. Technol.*, 2022.
- [48] Zelin Ji, Zhijin Qin, and Xiaoming Tao. Meta federated reinforcement learning for distributed resource allocation. *IEEE Transactions on Wireless Communications*, 23(7):7865–7876, 2024.
- [49] Kaidi Xu, Nguyen Van Huynh, and Geoffrey Ye Li. Distributed-training-and-execution multi-agent reinforcement learning for power control in hetnet. *IEEE Transactions on Communications*, 71(10):5893–5903, 2023.
- [50] Kaidi Xu, Shenglong Zhou, and Geoffrey Ye Li. Federated reinforcement learning for resource allocation in v2x networks. *IEEE Journal of Selected Topics in Signal Processing*, 18(7):1210–1221, 2024.
- [51] Kaidi Xu, Shenglong Zhou, and Geoffrey Ye Li. Rescale-invariant federated reinforcement learning for resource allocation in v2x networks. *IEEE Communications Letters*, 28(12):2799–2803, 2024.
- [52] Kaidi Xu, Shenglong Zhou, and Geoffrey Ye Li. Neural collapse based deep supervised federated learning for signal detection in ofdm systems. *arXiv preprint arXiv:2506.19476*, 2025.
- [53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [54] Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.
- [55] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE communications surveys & tutorials*, 18(3):1617–1655, 2016.
- [56] Akhil Gupta and Rakesh Kumar Jha. A survey of 5g network: Architecture and emerging technologies. *IEEE access*, 3:1206–1232, 2015.
- [57] Lin Zhang, Ying-Chang Liang, and Dusit Niyato. 6g visions: Mobile ultra-broadband, super internet-of-things, and artificial intelligence. *China Communications*, 16(8):1–14, 2019.

- [58] Yongjun Xu, Guan Gui, Haris Gacanin, and Fumiyuki Adachi. A survey on resource allocation for 5g heterogeneous networks: Current research, future trends, and challenges. *IEEE Communications Surveys & Tutorials*, 23(2):668–695, 2021.
- [59] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.
- [60] Delin Guo, Hui Ding, Lan Tang, Xinggan Zhang, Lvxi Yang, and Ying-Chang Liang. A proactive eavesdropping game in mimo systems based on multi-agent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 21(11):8889–8904, 2022.
- [61] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. 10th Int. Conf. Mach. Learn.*, pages 330–337, 1993.
- [62] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007.
- [63] Jungang Ge, Ying-Chang Liang, Jingon Joung, and Sumei Sun. Deep reinforcement learning for distributed dynamic mimo downlink-beamforming coordination. *IEEE Transactions on Communications*, 68(10):6070–6085, 2020.
- [64] Taejoon Kim, David J Love, and Bruno Clerckx. Does frequent low resolution feedback outperform infrequent high resolution feedback for multiple antenna beamforming systems? *IEEE Transactions on Signal Processing*, 59(4):1654–1669, 2010.
- [65] Yong Huat Chew, Boon-Hee Soong, et al. *Potential game theory*. Springer, 2016.
- [66] Ahmad Abu Al Haija and Chintla Tellambura. Small-macro cell cooperation for hetnet uplink transmission: Spectral efficiency and reliability analyses. *IEEE Journal on Selected Areas in Communications*, 35(1):118–135, 2016.
- [67] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.
- [68] Jack J Dongarra, Piotr Luszczek, and Antoine Petit. The linpack benchmark: past, present and future. *Concurrency and Computation: practice and experience*, 15(9):803–820, 2003.
- [69] Radio Frequency (RF) System Scenarios. TR 25.942, 3GPP, 2017. Available at: https://www.3gpp.org/ftp/Specs/archive/25_series/25.942/25942-1400.zip.

- [70] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*, pages 2681–2690. PMLR, 2017.
- [71] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- [72] Delin Guo, Lan Tang, Xinggan Zhang, and Ying-Chang Liang. Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(11):13124–13138, 2020.
- [73] Syed Adeel Ali Shah, Ejaz Ahmed, Muhammad Imran, and Sherali Zeadally. 5g for vehicular communications. *IEEE Commun. Mag.*, 56(1):111–117, 2018.
- [74] Haixia Peng, Le Liang, Xuemin Shen, and Geoffrey Ye Li. Vehicular communications: A network layer perspective. *IEEE Trans. Veh. Technol.*, 68(2):1064–1078, 2019.
- [75] Md. Noor-A-Rahim, Zilong Liu, Haeyoung Lee, G. G. Md. Nawaz Ali, Dirk Pesch, and Pei Xiao. A survey on resource allocation in vehicular networks. *IEEE trans. Intell. Transp. Syst.*, 23(2):701–721, 2022.
- [76] Tengchan Zeng, Omid Semiari, Walid Saad, and Mehdi Bennis. Joint communication and control for wireless autonomous vehicular platoon systems. *IEEE Trans. Commun.*, 67(11):7907–7922, 2019.
- [77] Muhammad Ikram Ashraf, Mehdi Bennis, Cristina Perfecto, and Walid Saad. Dynamic proximity-aware resource allocation in vehicle-to-vehicle (v2v) communications. In *2016 IEEE Glob. Commun. Conf.*, pages 1–6, 2016.
- [78] Le Liang, Joonbeom Kim, Satish C. Jha, Kathiravetpillai Sivanesan, and Geoffrey Ye Li. Spectrum and power allocation for vehicular communications with delayed csi feedback. *IEEE Wireless Commun. Lett.*, 6(4):458–461, 2017.
- [79] Le Liang, Geoffrey Ye Li, and Wei Xu. Resource allocation for d2d-enabled vehicular communications. *IEEE Trans. Commun.*, 65(7):3186–3197, 2017.
- [80] Le Liang, Shijie Xie, Geoffrey Ye Li, Zhi Ding, and Xingxing Yu. Graph-based resource sharing in vehicular communication. *IEEE Trans. Wirel. Commun.*, 17(7):4579–4592, 2018.
- [81] Jie Mei, Kan Zheng, Long Zhao, Yong Teng, and Xianbin Wang. A latency and reliability guaranteed resource allocation scheme for lte v2v communication systems. *IEEE Trans. Wirel. Commun.*, 17(6):3850–3860, 2018.

- [82] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. FedPD: A federated learning framework with adaptivity to non-iid data. *IEEE Trans. Signal Process.*, 69:6055–6070, 2021.
- [83] Shenglong Zhou and Geoffrey Ye Li. Communication-efficient admm-based federated learning. *arXiv preprint arXiv:2110.15318*, 2021.
- [84] Shenglong Zhou and Geoffrey Ye Li. Federated learning via inexact admm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.
- [85] Shenglong Zhou and Geoffrey Ye Li. FedGiA: An efficient hybrid algorithm for federated learning. *IEEE Trans. Signal Process.*, 71:1493–1508, 2023.
- [86] Stefanos Leonardos, Will Overman, Ioannis Panageas, and Georgios Piliouras. Global convergence of multi-agent policy gradient in markov potential games. In *Int. Conf. Learn. Rep.*, 2022.
- [87] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.*, 22(1):4431–4506, 2021.
- [88] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. *Advances in neural information processing systems*, 28, 2015.
- [89] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.
- [90] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3. Atlanta, GA, 2013.
- [91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1026–1034, 2015.
- [92] Sunny Gupta, Vinay Sutar, Varunav Singh, and Amit Sethi. Fedalign: Federated domain generalization with cross-client feature alignment. *arXiv preprint arXiv:2501.15486*, 2025.
- [93] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.
- [94] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

- [95] Vignesh Kothapalli. Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*, 2022.
- [96] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. Pmlr, 2015.
- [97] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.
- [98] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.
- [99] Duc Anh Nguyen, Ron Levie, Julian Lienen, Gitta Kutyniok, and Eyke Hüllermeier. Memorization-dilation: Modeling neural collapse under label noise. *arXiv preprint arXiv:2206.05530*, 2022.
- [100] Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? *Advances in neural information processing systems*, 35:37991–38002, 2022.
- [101] George Andriopoulos, Zixuan Dong, Li Guo, Zifan Zhao, and Keith Ross. The prevalence of neural collapse in neural multivariate regression. *arXiv preprint arXiv:2409.04180*, 2024.
- [102] Pengyu Li, Xiao Li, Yutong Wang, and Qing Qu. Neural collapse in multi-label learning with pick-all-label loss. *arXiv preprint arXiv:2310.15903*, 2023.
- [103] Renjie Li, Xinyi Wang, Guan Huang, Wenli Yang, Kaining Zhang, Xiaotong Gu, Son N Tran, Saurabh Garg, Jane Alty, and Quan Bai. A comprehensive review on deep supervision: Theories and applications. *arXiv preprint arXiv:2207.02376*, 2022.
- [104] Qi Dou, Hao Chen, Yueming Jin, Lequan Yu, Jing Qin, and Pheng-Ann Heng. 3d deeply supervised network for automatic liver segmentation from ct volumes. In *International conference on medical image computing and computer-assisted intervention*, pages 149–157. Springer, 2016.
- [105] Hengli Wang, Rui Fan, Peide Cai, and Ming Liu. Sne-roadseg+: Rethinking depth-normal translation and deep supervision for freespace detection. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1140–1145. IEEE, 2021.

- [106] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016.
- [107] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proc. Natl. Acad. Sci. U.S.A.*, 118(43):e2103091118, 2021.
- [108] Pekka Kyosti. Winner ii channel models. *IST, Tech. Rep. IST-4-027756 WINNER II D1. 1.2 V1. 2*, 2007.
- [109] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8485–8493, 2022.