# Towards Intelligent Mobile Edging: High Efficiency, Low Cost, and Good Decision

by

Dian Shi

A dissertation submitted to the Department of Electrical and Computer Engineering,

Cullen College of Engineering

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Electrical Engineering

Chair of Committee: Miao Pan

Committee Member: Zhu Han

Committee Member: Tomoaki Ohtsuki

Committee Member: Xin Fu

Committee Member: David Mayerich

University of Houston

May 2022

## ACKNOWLEDGMENTS

From the bottom of my heart, I would like to express my deepest gratitude to my advisor Dr. Miao Pan for his patient guidance, passionate support, and constructive criticisms during my Ph.D. study. His enthusiasm, knowledge, and exacting attention to detail kindled in me a fascination with the research and my determination to take scientific research as a career. I am very honored to be guided by him, which made me spend five unforgettable and meaningful years in my life.

Furthermore, I would like to extend my sincere appreciation to my committee, Dr. Zhu Han, Dr. Tomoaki Ohtsuki, Dr. Xin Fu, and Dr. David Mayerich, who provided thoughtful comments and recommendations on my research. This dissertation would not have been possible without their support. I am grateful to have the pleasure of working with Dr. H. Vincent Poor, Dr. Yuguang Fang, Dr. Jixiang Lu, Dr. Lixin Li, Dr. Jie Wang, Dr. Xuanheng Li, Dr. Xiangwei Zhou, Dr. Hao Wang, Dr. Li Wang, Dr. Hao Yue, Dr. Ming Li, Dr. Pan Li, and Dr. Minglei Shu. I really appreciate their invaluable feedback on my analysis and framing and every discussion that has benefited me a lot.

Special thanks to all my dear colleagues and friends, Dr. Jingyi Wang, Dr. Sai Mounika Errapotu, Dr. Debing Wei, Dr. Xinyue Zhang, Jiahao Ding, Pavana Prakash, Rui Chen, Chenpei Huang, Huai-an Su, Dr. Feng Hu, Dr. Maoqiang Wu, Dr. Shihao Ran, Pengyu Yuan, Lening Wang, and many others. Their company in Houston encouraged me to make continuous progress in my studies and filled my life with sunshine and hope.

Last but not least, I am profoundly grateful to my parents and families for their unwavering support and belief in me, and my most enormous thanks to Dr. Liang Li for constantly accompanying me throughout my life. Without you, the most stunning view turns only to regret.

# ABSTRACT

In recent years, with the continuous prosperity of the mobile Internet industry and the Internet of Things, the advance of widely used mobile terminals promotes the integration of mobile networks and Artificial Intelligence (AI), two of the most disruptive technologies world has seen nowadays. While each technology has spawned a large number of applications to facilitate our lives, the combination of mobile networks and AI, i.e., intelligent mobile edging, is going to be genuinely transformative. One perspective of combining these two is to exploit mobile networks to better support AI (Wireless for AI), where federated learning (FL) over mobile devices can greatly extend the scale of AI functions and preserve data privacy. Another perspective of the combination of AI and mobile networks is to tackle the challenges of wireless communication with AI strengths (AI for Wireless). Specifically, AI techniques can represent hard-to-model wireless problems and find feasible solutions with low computational complexity. The last point is that the combination can enable various smart mobile applications and services (AI & Wireless for applications). Though intelligent mobile edging has infiltrated many areas due to its advantages, several critical challenges still limit the efficient implementation of intelligent mobile edging, among which efficiency, cost, and performance are major considerations of this dissertation.

Huge energy/time consumption is one of the most significant obstacles restricting the development of AI functions on resource-constrained mobile devices. Besides, the delay-sensitive property of intelligent mobile applications also puts forward higher requirements for the efficiency of AI methodologies and communication service decision-making. Therefore, given these challenges, the objectives of this dissertation are to develop high efficiency, low cost, and good decision intelligent mobile edging methodologies from the three perspectives mentioned above through a combination of theoretical, simulation, and experimental studies. Specifically, this dissertation firstly endeavors to develop a series of efficient FL over mobile devices approaches, where computing and communication resources are well balanced to reduce the total cost during training; and then focuses on making good decisions and improving the performance of implementing AI functions on wireless networks and intelligent wireless services.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# 1 Introduction

With the rapid development of digital and information technologies, data produced by users and digital systems showed an explosive growth trend. In this big data era, Artificial intelligence (AI), particularly deep learning technique (DL), fully exploits the potential of the enormous data and has become one of the most disruptive technologies the world has witnessed in the last few years. With the maturity of the DL technique and its wide application in all fields, AI continues to seep into our daily lives. The convolutional neural network (CNN) can efficiently extract the features of pictures to intelligently classify or recognize the content of images, such as object detection or face recognition. Deep reinforcement learning (DRL) enables automatically control strategies selection, reaching or even exceeding the limits of human beings in many fields, like the game AI or solving scientific challenges. The wide application of the DL technique pushes AI to a new stage. However, some issues such as data privacy, scalability, and efficiency of the technology itself still need to be explored.

Another recent booming technology, mobile networks, is greeted with an avalanche of publicity. From the widely used 4G network to the maturing 5G network, and then to the 6G and beyond the network soon, the accelerating data transmission speed and increasingly flexible connection modes make the society move towards the era of the interconnection of all things. With the advanced mobile communications technology, data and information can be shared more efficiently among users and devices, which has spawned many industries with mobile technology as the carrier, such as mobile Internet. At the same time, due to the advance of hardware design, a growing number of mobile devices are armed with ever-increasingly high-performance computation units, such as the central processing units (CPUs) and graphics processing units (GPUs) which enable them to host computation-intensive tasks. The rapid development of mobile devices provides the basis for mobile communication to support more complex application scenarios and puts forward higher requirements for transmission reliability and low latency.

Artificial intelligence and mobile networks are two of the most innovative technologies

in recent years. While each technology has made breakthrough progress and brought a new experience to users, the combination of AI and mobile networks is going to push for a sea-change in the world. The interaction between these two emerging technologies derives a new nexus, i.e., intelligent mobile edging, which coordinates these technologies to further develop both sides and create new application scenarios. With mobile communications, the AI functions can be brought from remote cloud servers to edge devices, enabling on-device processing or providing intelligent mobile services. Meanwhile, the AI functions on the edge devices can also support the wireless transmission process with intelligent management. Specifically, intelligent mobile edging can be illustrated in the following perspectives.

First of all, exploiting mobile networks can better support AI. The recent surge of ML technologies is primarily due to the power of cloud computing and the availability of big data. Unfortunately, cloud-centric ML generates tremendous traffic and causes serious privacy concerns, which is not suitable for many resource-constrained applications. In order to scale and move beyond the cloud-centric ML, Google has introduced federated learning (FL), the currently popular distributed machine learning paradigm, which aims to enable mobile devices to collaboratively learn a joint global ML model without sharing their privacy-sensitive raw data [1]. With the help of advanced mobile networks, distributed data stakeholders (e.g., mobile devices) in FL only need to periodically transmit their updated local models to the aggregation server for global updates wirelessly, instead of uploading their potentially private raw data, thus significantly lowering the risk of privacy leakage. Recent successes in mobile networks, such as 5G and beyond (5G+) technology [2, 3], can significantly facilitate the implementation of FL over mobile devices. For example, 5G+ mobile devices are usually armed with high-performance computation units, which enable them to host computation-intensive learning tasks. Besides, the unique properties of advanced mobile networks, like the multi-access edge computing (MEC) [4] structure in 5G standard and the high data rate and ultra-low latency of the 5G transmission, paving the way for performing computing and communication for edge intelligence.

Such a combination of mobile networks and FL prompts tremendous successful applications over mobile devices, including keyboard prediction [5], cardiac event prediction, financial risk management, etc. While deploying FL over mobile devices is promising to have so many exciting applications, severe challenges are foreseeable, of which energy and time consumption is the dominant concern. On the one hand, executing on-device computing and performing local model updates are both resource-hungry, inducing a significant surge of energy and time consumption on mobile devices, which seriously drains battery power and affects the user experience. On the other hand, there will be a trade-off between computing and communication over resource-constrained mobile devices for both energy and time. Coordinating the relationship between computing and communication is also crucial for total resource consumption. Thus, investigating the energy/time efficient FL with low cost over mobile devices from both the computing and communication perspectives is urgently needed.

Moreover, applying AI can overcome challenges in mobile networks. With the continuous prosperity of the mobile Internet industry and the Internet of Things, more promising communication paradigms are proposed for the next generation of mobile networks to support flexible and efficient transmissions. For example, the MEC architecture pushes computational capabilities closer to end-users, efficiently solving the long latency and backhaul bandwidth limitation problem in cellular networks. Device-to-Device (D2D) communication enabling direct data transmission between two mobile users has emerged as a vital component for 5G cellular networks to improve spectrum utilization and enhance system capacity. However, there are still some critical technical challenges to implementing these new communication modes. The first point is that such wireless problems are hard to model, particularly considering the natural characteristics of dynamic communication conditions. The second point is that the wireless model is always in a non-convex and non-linear formula, which brings great difficulties to solve and find feasible solutions, even if it consumes many computing resources.

For the above-mentioned challenges, AI is a power tool to solve them efficiently due to

its strengths. With neural networks, almost any form of problem can be well represented by DL models. Especially, the deep reinforcement learning (DRL) methods show excellent performance in coping with dynamic communication environments and making sequential decisions under uncertainty. Furthermore, the optimization methods, such as stochastic gradient descent (SGD) or Adam, utilized in DL, enable DL to effectively find convincing results for such difficult problems. Nevertheless, the convergence rates and the stability of the training process, which have a great impact on user experience and final performance, are non-negligible and still needed to be further improved.

Finally, AI and mobile networks can enable a variety of intelligent applications. In addition to promoting each other's development, the combination of their advantages will have broader application prospects. AI applications have developed significantly in the past few years, and have been applied in almost every business sector. Its high efficiency and automation enable AI to provide flexible system solutions for a variety of new and enhanced experiences. With the help of the high-speed and low latency characteristics of advanced mobile networks, AI-processing can be brought to a variety of end devices. Through the continuous and rapid interaction of data and smart decisions, end devices has become an indispensable part of intelligent society. For example, the construction of intelligent transportation system, including transportation network company cars, intelligent public transportation, etc., has brought great convenience to urban daily traveling. With the popularity of internet of everything, a growing portion of internet of things (IoT) devices are created for consumer use, such as smart home, elder care, and security monitoring, etc., and commercial use, like in medical or industry. In addition, the combination of AI and mobile networks also lays the foundation for new applications such as augmented reality (AR), virtual reality (VR) and metaverse, making it possible for the continuous emergence of various new things.

This dissertation focuses on intelligent mobile edging, where we marry AI and mobile networks for fruitful use without frictions to achieve high efficiency, low cost, and good decision. This dissertation will first illustrate works on how mobile networks can better

support AI (Wireless for AI), where the high efficiency and low cost in FL training are our key concerns. Next, we will show how AI can overcome challenges in mobile networks (AI for Wireless). In this section, the DRL techniques are used to model the wireless problems and make good decision strategies with high convergence speed. Finally, we will share our thoughts on how AI and mobile networks can efficiently enable various intelligent applications with good performance (AI & Wireless for applications).

Specifically, we first provide the preliminaries of FL and DRL, the main ML techniques used in this dissertation, in Section 2. From the wireless for AI perspective, we implement the dynamic batch sizes technique during the training process and propose a time-efficient FL algorithm in Section 3. Then, in Section 4, we present an energy-efficient FL algorithm considering over-the-air computing. Basically, we derive the corresponding convergence analysis for both algorithms and make a good trade-off between computing and communication in FL training, thus significantly reducing the overall resources consumption. From the AI for wireless perspective, in Section 5, we investigate resource allocation problems in D2D networks, where we develop a Stackelberg game guided multi-agent DRL approach to make the smart power control and channel allocation decisions. From the perspective of AI and wireless for applications, we show two case studies on intelligent services. In Section 6, we present a DRL-based route scheduling scheme for TNC cars to improve the revenue and reduce the cruising time. In Section 7, we introduce a deep learning-based child detection system to prevent hot car deaths by using the WiFi signal. At last, we present some possible future works related to intelligent mobile edging in Section 8.

## 2 Deep Reinforcement Learning and Federated Learning

Machine learning (ML), particularly deep learning (DL), is one of the most disruptive technologies the world has witnessed in the last few years. Typically, ML technology can be classified into three categories: supervised learning, unsupervised learning, and reinforcement learning (RL). Besides, according to different training modes, ML technology can also be divided into centralized learning and distributed learning, in which federated learning is a typical representative of distributed learning (FL). In this section, two main technologies used in this paper, RL and FL, will be introduced in the following.

### 2.1 Reinforcement Learning

RL [6] has been widely used in finding optimal policies and achieving optimal control. In RL, the agent observes the environment and discovers which action $a \in A$ yields the most numerical reward $r \in R$ by trying it, and then generates the policy of mapping state $s \in S$ to action $a$. Following the policy, it moves to the next state with probability $p \in P$, and then repeats this process step by step. Therefore, a four-tuple $(S, A, R, P)$ can be used to denote the RL model. The agent's goal is to maximize the expected cumulative reward.

The expected cumulative reward starts from state $s_k$ and following action $a_k$, which is called state action value $Q(s_k, a_k)$ or $Q$ value. The state action value $Q(s_k, a_k)$ is given by

$$Q(s_k, a_k) = \mathbb{E}[r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + ...|s_k, a_k] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{k+t}|s_k, a_k]. \tag{1}$$

Let $Q^*(s_k, a_k)$ be the optimal state action value function, which can be represented by the optimal Bellman equation as $Q^*(s_k, a_k) = \mathbb{E}[r_k + \gamma \max_{a'_k} Q^*(s'_k, a'_k)|s_k, a_k]$, where $s'_k$ and $a'_k$ represent the next state and the next action, respectively. This establishes a correlation between one state and the next state, and the optimal state action value can be computed iteratively step by step. In RL, we want to find a learned policy to maximize the expected

cumulative reward, which can be written as

$$\pi^* = \arg\max_{\pi \in A} \ R_k^{\pi}, \tag{2}$$

where $R_k^{\pi} = \sum_{t=k}^{K} \gamma^{t-k} r_k$ and $\gamma$ is the discount factor which can balance the instant reward and the future rewards.

Typically, RL methods can be divided into two classed, the value-based methods and the policy-based methods. For the value-based RL methods, the optimal can be obtained indirectly through the Q-value. In value-based RL methods, the iterative value function is defined as

$$Q(s_k, a_k) = Q(s_k, a_k) + \alpha[r_k + \gamma \max_{a'_k} Q(s'_k, a'_k) - Q(s_k, a_k)], \tag{3}$$

where $\alpha$ is the learning rate, $\gamma$ is the discount rate. Finally, the state action value will converge to the optimal value, i.e. $Q_t \to Q^*$ when $k \to \infty$, and then the optimal policy $\pi^*$ can be obtained from the value function, i.e. $\pi^* = \arg\max_{a_k} \ Q^*(s_k, a_k)$ by checking the *Q-table* in the memory. For the policy-based RL methods, the optimal policy can be directly formulated and optimized through the optimization approaches.

## 2.2 Federated Learning

As an emerging decentralized learning paradigm, FL has taken advantage of the computing resources across massive participants. Specifically, all participants collaboratively contribute to one global learning task in a distributed manner with continuous interactions for model parameter updates. We consider a FL system consisting of $K$ participating users, where each user $k \in \{1, 2, \ldots, K\}$ has its own data set, denoted by $\mathcal{D}_k$. The goal of FL is to collaborate the users to perform a unified optimization task, written as

$$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}) \triangleq \frac{1}{K} \sum_{k=1}^{K} f_k(\mathbf{w}), \tag{4}$$

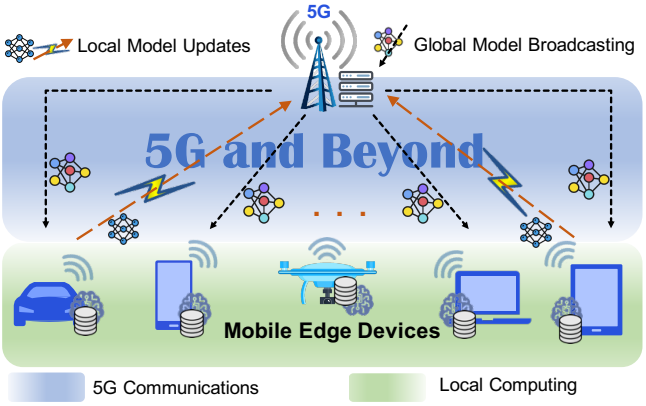



Figure 1: The illustration of the FL over mobile devices.

where $f_k$ is the local loss function corresponding to user $k$, and $d$ is the dimension of the model parameters.

Let $r \in \{1, 2, \ldots, R\}$ denote the FL global round index between the server and local devices, and $H$ be the number of local computing iterations executed between two consecutive global communication rounds. Moreover, We define $\mathbf{w}^r$ as the global model at the $r$-th communication round and $\mathbf{w}_k^{r,h}$ as the local model of user $k$ at the $h$-th local iteration under the $r$-th communication round. Therefore, the local updating process of user $k$ under the $r$-th communication round is denoted as

$$\mathbf{w}_k^{r,h+1} = \mathbf{w}_k^{r,h} - \eta \nabla F_k(\mathbf{w}_k^{r,h}) \text{ for } h = 0, 1, ..., H-1, \tag{5}$$

where $\nabla F_k(\mathbf{w}_k^{r,h})$ is a stochastic gradient of $f$ with a random batch-size data, and $\eta$ is the local learning rate. Here, $\nabla F_k(\mathbf{w}_k^{r,h})$ is the unbiased estimation of $\nabla f_k(\mathbf{w}_k^{r,h})$, i.e., $\mathbb{E}_{\xi \sim \mathcal{D}_k}[\nabla F_k(\mathbf{w}) \mid \boldsymbol{\xi}] = \nabla f_k(\mathbf{w})$, where $\xi$ represents the randomness like the batch-size index. After finishing the local training, every participate upload its local model updates to the server for global aggregation, i.e., $\eta \sum_{h=0}^{H-1} \nabla F_k(\mathbf{w}_k^{r,h})$, and the server then broadcasts the most recent global model to initiate a new round of local training. The above process is repeated until the global model converges. A typical paradigm of the FL over mobile devices, including local computation and wireless communications parts, is shown in Fig. 1.

# 3 To Talk or to Work: Dynamic Batch Sizes Assisted Time Efficient Federated Learning over Future Mobile Edge Devices

## 3.1 Introduction

Embracing recent advances in ultra-high speed wireless transmissions (e.g., WiFi-6, 5G, 6G, etc.) and mobile hardware technologies, multi-access edge computing (MEC) has been recently emerged as a promising paradigm enabling local data analysis and real-time service provisioning. With this trend, federated learning (FL) further pushes artificial intelligence (AI) functions to mobile edge devices, thereby being instrumental in spearheading the vision of intelligent mobile edge networks. As we know, through local training updates, FL enables data stakeholders (e.g., mobile edge devices) to collaboratively learn a joint global machine learning (ML) model without sharing their private raw data. Various features of MEC make it perfectly fit to support FL. First, although current mobile edge devices (e.g., 4G or 5G smartphones, tablets, etc.) can only do learning inferences, future mobile edge devices in MEC will be widely equipped with high-performance integrated processors (e.g., high-performance GPUs) and are expected to have on-device local training capability [7], given the promising research advances in mobile computer architecture designs and computing hardware development. Besides, the distributed architecture of MEC is consistent with that of FL, where future mobile edge devices can serve as local FL clients and the edge server can serve as the FL aggregator. Moreover, the widespread popularity of social networking applications will breed a wealth amount of data continually generated on future mobile edge devices, which provides data basic for on-device training. Therefore, such the coupling of MEC and FL can prompt a broad range of applications, including keyboard prediction [5], cardiac event prediction [8], and financial risk management [9], etc.

However, deploying FL over future mobile edge devices in practice is non-trivial and poses great challenges, of which the time consumption is a critical concern. In particular, the proliferation of real-time and lifelong applications with stringent requirements on low

latency, such as augmented reality and voice assistant, has spurred a growing need for continuous training on mobile edge devices. The mismatch between the delay-sensitive property of these applications and the limited resources of mobile edge networks raises two fundamental issues of FL over future mobile edge devices. One is how to fundamentally accelerate the training process from the algorithmic perspective while guaranteeing the learning convergence and model accuracy. It is also challenging for future mobile edge devices to efficiently execute the time consuming neural network training, which may cause intolerable latency for certain delay-sensitive FL applications. The other is how to determine the key training parameters adapting to the mobile edge environment in practice, so that the system resources, such as GPU resources and wireless bandwidth, can be fully utilized to improve the time efficiency.

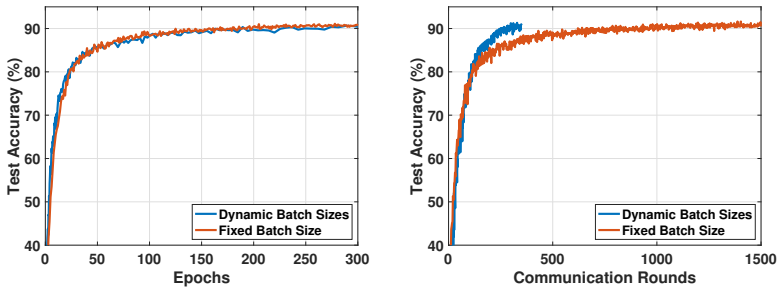
Most existing works in the machine learning community focus on improving the communication efficiency of FL since communication is usually regarded as a bottleneck in the training procedure. In [1], the local SGD (a.k.a. FedAvg) algorithm is proposed to allow every participating device to perform multiple stochastic gradient descent (SGD) iterations locally before synchronizing with others, thereby avoiding communication after every local iteration. Another notable method to reduce the necessary communication rounds is gradually enlarging the batch sizes during training, which properly increases local computations in a single round to reduce the variance [10, 11]. Although these methods exhibit great potential in communication burden alleviation, it is questionable whether the communication is a bottleneck in itself. In fact, the rapidly expanding 5G networks can provide a high transmission rate up to 1 Gbps, and WiFi-6 claims to have the peak throughput of 9.6 Gbps. Furthermore, the forthcoming 6G networks are envisioned to open up a "Tbps" era. These advanced technologies make transmission delay no longer a dominant issue hindering FL's implementation in wireless networks [12, 13]. In this situation, wireless transmissions and local computing are comparable in time consumption. For example, performing a single-step local iteration (including data accessing and computing) of a ResNet-50 model with 100MB parameters on one GPU typically takes hundreds of milliseconds [14], which is similar to

the transmission delay introduced by uploading the model via 1 Gbps wireless links (i.e., 800 ms).

Recognizing the comparability between the communication cost and computing cost, some recent works in [15, 16, 17, 18, 19] study to reduce the FL system cost over mobile edge devices via joint communication and computing resource management. Their methods just allocate system resources under a given budget and strive to acquire a proper resource allocation strategy matching with the network environment. However, essentially saving the total time consumption from the learning's perspective is widely overlooked in these works, such as how to adjust the appropriate batch size. Thus, it is worthwhile to investigate the trade-off between "working" (i.e., local computing) and "talking" (i.e., wireless communication) from both the learning algorithm and resource allocation perspectives for efficient FL over future mobile edge devices.

To bridge this gap, this work [20] targets at accelerating the FL training process by jointly considering the computing and communication conditions over future mobile edge devices. To this end, we first propose a time-efficient FL algorithm, named **d**ynamic **b**atch sizes assisted **f**ederated **l**earning (**DBFL**), with the convergence guarantee. Unlike ordinary local SGD using fixed batch sizes throughout the training, the DBFL allows users to exponentially increase the batch sizes with an incremental factor, leading to a reduction of communication rounds required to complete the training. We will explain the reduction in detail in the following section. We then employ the proposed DBFL algorithm in wireless networks and develop a batch size control scheme adapting to the specific network conditions. In particular, we determine the optimal incremental factor via an elaborate optimization problem, where "working" and "talking" at future mobile edge devices are well balanced to minimize the total training time consumption. In addition, the goal of improving time efficiency is in general consistent with that of improving energy efficiency, as the shortening of overall time consumption leads to the reduction of overall energy consumption. Our salient contributions are summarized as follows:

- Capturing the learning dynamics, we propose the DBFL, a time efficient FL algorithm




(a) Test accuracy vs. epochs (b) Test accuracy vs. communications

Figure 2: FL with dynamic batch sizes (ResNet20 on CIFAR-10).

allowing the batch size to increase exponentially in the training process. We also provide a theoretical analysis of the DBFL algorithm in terms of convergence rate and communication complexity.

- Guided by the theoretical results of DBFL, we further study to minimize the total time consumed for training an FL model to converge over future mobile edge devices. In particular, we develop a batch size control scheme to derive the optimal batch size incremental factor, catering to the GPU computing performances and wireless communication conditions of mobile edge devices.

- We conduct extensive simulations to evaluate the performance of the proposed scheme on various learning models and system settings. Our scheme exhibits great superiority in terms of time consumption reduction for FL over future mobile edge devices compared with the state-of-the-art FL solutions.

### 3.2 Preliminaries of Dynamic Batch Sizes

In order to better explain the motivation of this paper, the learning schemes with the fixed batch size and dynamic batch sizes are compared through experiments in this part. Taking the ResNet20 [21] model on the CIFAR-10 dataset as an example, we consider an FL scenario with 10 participating users, and each user sequentially takes 10 local SGD steps. Based on this, we deploy this scenario with the fixed scheme (FedAvg [1] with batch

Table 1: GPU Performance.

| Size | 50 | 100 | 200 | 250 | 500 | 1000 | 2500 |
|---|---|---|---|---|---|---|---|
| Time(s) | 0.021 | 0.022 | 0.023 | 0.024 | 0.040 | 0.072 | 0.171 |
| Ratio | 6.140 | 3.216 | 1.681 | 1.404 | 1.170 | 1.052 | 1 |

size 200) and the dynamic scheme (gradually increasing the batch sizes from 50 to $2,500$), respectively, and the experiment results are shown in Fig. 2. Note that the fixed scheme needs the relatively small batch sizes (e.g., smaller than 200) in the FL to guarantee the convergence [1]. Fig. 2a shows that two schemes need almost the same data epochs to achieve the target accuracy, and one epoch refers to one cycle through the full training dataset. This means that the gradient calculation operations of two schemes, i.e., the computation loads during the training, are similar. The training curves with data epochs as x-axis are similar in 2a. But for the global updates, i.e., communication cost, shown in Fig. 2b, the dynamic scheme needs far fewer communication rounds than the fixed scheme. The reason is that the large batch size implemented in the latter training stage leads to the reduction of update frequency.

In this paper, we focus on the time consumption for the overall FL procedure, including the communication time and the computing time. From Fig. 2b, it can be easily found that the communication cost of the dynamic scheme is much less than that of the fixed scheme due to the fewer required communication rounds. In terms of the computing cost, when using a similar number of data epochs to achieve FL convergence in the two schemes (as shown in Fig. 2a), the total computing time of the dynamic scheme is lower than that of the fixed scheme due to the efficient large-batch training in the later stage of training. Specifically, the local gradient calculation delay does not increase linearly with the batch size increase. For example, when the batch is doubled, the increase in time consumption will be less than twice. Such a relationship between batch size and time consumption indicates the efficiency of the large batch training. This is because the GPU pipeline is a kind of parallel processing, where it can directly process far more data simultaneously [22]. As

shown in Table 1, we conduct the experiments on a single "RTX 8000" GPU and record the time consumption for different batch sizes. The "Ratio" indicates the ratio of "Time for computing 2,500 samples with specified batch size" and "Time for computing 2,500 samples with batch size 2,500". This "Ratio" implies that a larger batch size can save more time when accessing the same number of stochastic gradients, and similar results can be found in [23]. Thus, compared with the fixed scheme that always uses relatively small batch sizes, the dynamic batch size scheme consumes less computing time for convergence, which is consistent with the trend of communication consumption. Moreover, to further reduce the system time consumption and balance the communication and computing in the dynamic scheme, we are going to adjust the batch size increasing rate to achieve the minimum time consumption for the training.

## 3.3 Federated Learning with Dynamic Batch Sizes

In this section, we propose a general time efficient FL framework with dynamic batch sizes in Sec. 3.3.1, named **DBFL**, and derive the convergence analysis for it in Sec. 3.3.2.

### 3.3.1 Dynamic Batch Sizes Assisted FL Algorithm Design

We consider a multi-access edge computing system for the distributed machine learning task with one edge server and a set $\mathcal{K} := \{1, ..., K\}$ of $K$ participating future mobile edge devices, as shown in Fig. 3. Specifically, each device $i$ has its own dataset $\mathcal{D}_i$ and cannot access other devices' datasets. Moreover, all devices maintain their local machine learning models with the same model structure and attempt to achieve a global goal, i.e., obtain a common model to minimize the training loss, under the coordination of the edge server. Such a scenario can be considered as a FL task, where the future mobile edge server can serve as the FL aggregator and future mobile edge devices can serve as local FL clients. In addition, FL can be also formulated as the following distributed non-convex optimization

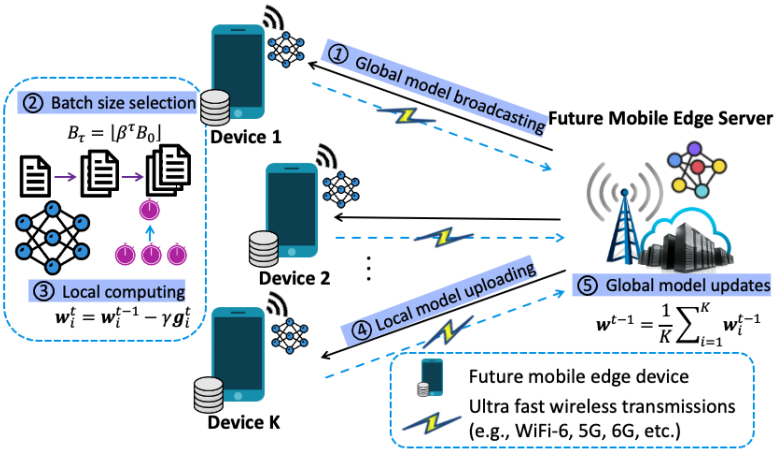



Figure 3: The illustration of the dynamic batch sizes assisted federated learning.

problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) \triangleq \frac{1}{K}\sum_{i=1}^{K} f_i(\mathbf{w}), \tag{6}$$

where $f_i(\mathbf{w})$ is the training loss for device $i$ over the dataset $\mathcal{D}_i$, i.e., $f_i(\mathbf{w}) \triangleq \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[F_i\left(\mathbf{w};\xi_i\right)\right]$. Here, we assume all users have the same size of dataset $\mathcal{D}_i$ for concise expression, and the size of $\mathcal{D}_i$ can be possibly different for different $i$. $\xi$ denotes the randomness in the training process, e.g., different data points and different batch sizes selection. In every training iteration $t$, each device will take one SGD step, which means each device observes the unbiased stochastic gradients with one batch data based on the model $\mathbf{w}_i^{t-1}$ obtained at the last iteration step as $\mathbf{g}_i^t = \nabla F_i\left(\mathbf{w}_i^{t-1};\xi_i\right)$. Here, $\mathbf{g}_i^t$ is the unbiased estimation of $\nabla f_i(\mathbf{w}_i^{t-1})$, i.e., $\mathbb{E}_{\xi_i^t\sim\mathcal{D}_i}\left[\mathbf{g}_i^t \mid \boldsymbol{\xi}\right] = \nabla f_i(\mathbf{w}_i^{t-1})$.

One classical method to coordinate all devices' local model is to collect and take the average of the observed gradients among all devices in each iteration, then adopting the averaged gradients to update the global model. Furthermore, each device downloads the updated global model and continues to take the SGD step mentioned above for training. Such an optimization method is called mini-batch SGD. However, updating the global model in each iteration with only one SGD step is extremely communication inefficient, which consumes a tremendous amount of the limited communication resource, especially for the

FL over mobile edge devices. Hence, we employ a communication efficient optimization method, called local SGD, where each device performs $H$ sequential SGD steps and then updates the global model. Note that each data sample can be repetitively used during local updates. But it will have different effects on the model since the local model accumulates the previous gradient calculation results. Therefore, we will update the global model every $H$ iterations. For explanation convenience, we define a virtual global model at each iteration step $t$ as

$$\overline{\mathbf{w}}^t \triangleq \frac{1}{K}\sum_{i=1}^{K}\mathbf{w}_i^t, \tag{7}$$

and the virtual global model can be iteratively calculated as

$$\overline{\mathbf{w}}^t = \overline{\mathbf{w}}^{t-1} - \gamma\frac{1}{K}\sum_{i=1}^{K}\mathbf{g}_i^t, \tag{8}$$

where $\gamma$ is the learning rate. Note that when $t \bmod H = 0, \mathbf{w^t} = \overline{\mathbf{w}}^{\mathbf{t}}$.

Moreover, to further reduce the communication cost and improve computing efficiency, we consider a dynamic batch size in the training process. We gradually increase the batch size in the training stage and propose a time-efficient federated learning approach, called **d**ynamic **b**atch sizes assisted **f**ederated **l**earning (**DBFL**) algorithm, which is described in Alg. 1. Particularly, in each communication round $\tau$, the batch size is different and exponentially increased with an incremental factor $\beta$, i.e., $B_\tau = \lfloor \beta^{\tau-1} B_0 \rfloor$. In Alg. 1, $E$ is the total number of stochastic gradient accesses. The proposed DBFL procedure over future mobile edge devices is briefly described in Fig. 3. The edge server first broadcasts a current global model to the participating mobile edge devices in FL. After receiving the global model, each mobile edge device selects the batch size for local on-device training based on the local data and its computing capability. When a mobile edge device finishes its local training, it will upload its local model via ultra fast wireless transmissions (e.g., WiFi-6, 5G, 6G, etc.) for global model updates. The above steps will repeat until the training converge. The proposed DBFL algorithm can effectively reduce the communication rounds

**Algorithm 1** Dynamic Batch Sizes Assisted Federated Learning Algorithm (DBFL)

**Initialization:** Initialize the global model $\mathbf{w}^0$ and set $\mathbf{w}_i^0 = \mathbf{w}^{\mathbf{0}}, \forall i \in \mathcal{K}$; Set the learning rate $\gamma$, batch size incremental factor $\beta$, local step $H$, and initial batch size $B_0$; Initialize the communication index $\tau = 0$ and the iteration index $t = 1$; Initialize the local step count $lo$

**while** $H \cdot \sum_{\eta=0}^{\tau} B_\eta \leq E$ **do**
  Each device $i$ identifies the batch size $B_\tau = \lfloor \beta^\tau B_0 \rfloor$
  **for** $lo = 1, ..., H$ **do**
    Each device $i$ observes the unbiased stochastic gradients $\mathbf{g}_i^t$ of $f_i(\mathbf{w}_i^{t-1})$ with one batch data with the size $B_\tau$ from the dataset $\mathcal{D}_i$
    Each device $i$ in parallel updates its local model

$$\mathbf{w}_i^t = \mathbf{w}_i^{t-1} - \gamma \mathbf{g}_i^t, \quad \forall i$$

    Update $t \leftarrow t + 1$
  **end for**
  Update the global model $\mathbf{w}^{t-1} = \frac{1}{K} \sum_{i=1}^{K} \mathbf{w}_i^{t-1}$
  Each device $i$ in parallel updates its local model $\mathbf{w}_i^{t-1} = \mathbf{w}^{t-1}$
  Update $\tau \leftarrow \tau + 1$
**end while**

of global updates and benefit from the time efficient large batch training, which had been discussed detailed in Sec. 3.2. Furthermore, the convergence analysis of the proposed DBFL algorithm will be provided in the next section.

From the theoretical points of view, gradually increasing batch sizes is also beneficial for the training. We can interpret the SGD method as integrating a stochastic differential equation (SDE) whose "noise scale" $n \approx \gamma |\mathcal{D}|/B$ [24, 14], where $|\mathcal{D}|$ is the dataset size and $B$ denotes the batch size. For the above non-convex optimization problem, large-scale random fluctuations help to explore the parameter space to avoid trapping in local minima on the initial stage. After that, when we locate a promising region of parameter space, small-scale fluctuations are required to fine-tune the parameters on the later stage. Therefore, exponentially gradually increasing batch sizes is helpful in the training.

### 3.3.2 Convergence Analysis for the DBFL Algorithm

We consider the loss function $f_i$, $\forall i$ in (6) satisfies the following two assumptions:

**Assumption 1** (Smoothness) *The objective function $f_i$ is differentiable and L-smooth, as*

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall i. \tag{9}$$

**Assumption 2** (Bounded variances and second moments) *The variance and the second moments of stochastic gradients evaluated with a mini-batch of size B can be bounded as:*

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right) - \nabla f_i(\mathbf{w})\right\|^2 \leq \frac{\sigma^2}{B}, \forall \mathbf{w}, \forall i \tag{10}$$

$$\text{and } \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right)\right\|^2 \leq \delta^2, \forall \mathbf{w}, \forall i, \tag{11}$$

*where $\sigma$ and $\delta$ are the positive constants.*

We set the total number of iterations as $T$. Under the above assumptions, the following theorem holds, which is a key property to analyze the convergence of the DBFL algorithm.

**Theorem 1** *If we consider the initial batch size $B_0$, batch size incremental factor $\beta > 1$, local step $H$, the number of participating device $K$, and choose the learning rate $\gamma < \frac{1}{L}$, then the convergence rate for all $T > 1$ in Alg. 5 satisfies*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \leq \frac{2\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right)}{\gamma \log_\beta \frac{E(\beta-1)}{B_0}} + 4\gamma^2 H^2 \delta^2 L^2 + \frac{2L\gamma\sigma^2}{\log_\beta \frac{E(\beta-1)}{B_0} K B_0 H} \cdot \frac{\beta}{\beta - 1}, \tag{12}$$

*where $f^*$ is the minimum value of the loss, and $E$ is the total number of stochastic gradient accesses. (Please refer to Appendix A for the proof.)*

For Theorem 1, we can derive the following corollary.

**Corollary 1** *Under Assumptions 1 and 2, if we choose learning rate $\gamma = \frac{\sqrt{K}}{L\sqrt{\log_\beta \frac{E(\beta-1)}{B_0}}}$, local steps $H \leq \left(\log_\beta \frac{E(\beta-1)}{B_0}\right)^{\frac{1}{4}} K^{-\frac{3}{4}}$, the number of participating devices $K < (\log E)^{\frac{5}{3}}$, then the Alg. 5 has the $O\left(\frac{1}{\sqrt{K \log E}}\right)$ convergence rate with $O\left((\log EK)^{\frac{3}{4}}\right)$ communication rounds. (Please refer to Appendix B for the proof.)*

Furthermore, the communication complexity $O(T/H)$ can be considered as $O\left((K\log E)^{\frac{3}{4}}\right)$, which is lower than that of the local SGD method (FedAvg) with a fixed batch size as $O((KE)^{\frac{3}{4}})$ [25].

## 3.4 DBFL over Future Mobile Edge Devices

Both the experimental results obtained in Sec. 3.2 and the convergence analysis derived in Sec. 3.3.2 validate that the proposed dynamic batch sizes scheme can reduce the communication rounds of global updates, and the experimental results also show the efficiency of the large batch computing. Therefore, the total time consumption will be reduced when using the proposed DBFL algorithm. In this section, we employ the DBFL algorithm over future mobile edge devices in Sec. 3.4.1 and develop a control scheme in Sec. 3.4.2 to adjust the batch size incremental factor $\beta$ for further minimizing the total time consumption of FL, including both the communication time and computing time.

### 3.4.1 Problem Formulation

We consider the multi-access edge computing environment in practice and develop the corresponding wireless communication and GPU computing model for future mobile edge devices.

**Communication:** For each future mobile edge device, the average transmission rate over the whole training process can be evaluated as

$$R = W\mathbb{E}_h\left[\log_2\left(1+\frac{P|h|^2}{N_0}\right)\right], \tag{13}$$

where the expectation is taken over the channel fading $h$, and $N_0$ implies the power of additive white Gaussian noise (AWGN). Here, we consider the ideal transmission condition, where we assume the interference can be well-managed and channel conditions are stable. $W$ and $P$ denote the bandwidth and the transmission power, respectively. Afterwards, the transmission delay for transmitting the learning model with the model size $N$ in one

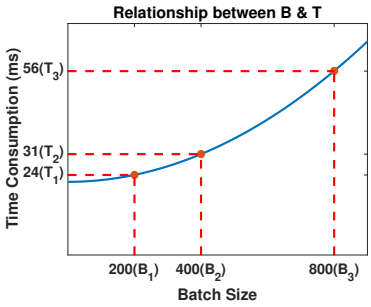



Figure 4: The relationship between batch sizes and time consumption.

communication round can be evaluated as

$$t^{tran} = \frac{N}{R}. \tag{14}$$

**Computing:** Future smart mobile edge devices will be widely armed with high-performance GPUs, which will be powerful in handling sophisticated computations of FL tasks. Consequently, we consider the GPU computing model and formulate the execution time for GPU computing the gradient of one data sample as

$$t_0^{comp} = t_{init} + \frac{u}{f_{mem}} + \frac{v}{f_{core}}, \tag{15}$$

where $f_{core}$ and $f_{mem}$ denote the GPU core frequency and GPU memory frequency, and $t_{init}$ represents the static time consumption [26, 27]. $u$ and $v$ are constant factors that reflect the sensitivity of the task execution to GPU memory and core frequency scaling. Therefore, the time consumption for accessing a batch size of $B$ data can be calculated as

$$t^{comp}(B) = t_0^{comp} \cdot d(B), \tag{16}$$

where $d(B)$ is a function describing the relationship between the batch size and time consumption.

Due to the parallelism property of the GPU, the time consumption for calculating the

gradients with different data sizes does not linearly increase with the batch sizes increasing, as shown in Table 1. The first is that the per data sample time consumption gradually decreases with the batch size increasing. The second one is that the efficiency improvement of the large batch training gradually also decreases with the increasing batch size. Moreover, we find that when the batch size is not extremely, a part of the quadratic function can fit such a nonlinear relationship, as shown in Fig. 4, and have the assumption below. Similar observations can also be found in [23]. Therefore, we choose the quadratic function $d(B) = aB^2+c$ in (16), where $a > 0$ and $c > 0$, to describe the nonlinear relationship, which can well reflect the efficiency of GPU computing for large batch sizes. Note that we use a segment of the quadratic function to represent the time consumption; we can select different values of $a$ and $c$ for different types of GPUs.

**Assumption 3** (Quadratic Relationship) *For the general CNN model, when the batch size is not extremely large, the relationship between the GPU time consumption and batch sizes of gradient calculation follows the quadratic form.*

After establishing the communication and computing model, the next step is to identify the required communication rounds. According to Theorem 1, we have the following corollary on required communication rounds of the DBFL algorithm.

**Corollary 2** *The maximum number of communication rounds $M = \frac{T}{H}$ for achieving $\epsilon$ global convergence accuracy, i.e., satisfying*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \leq \epsilon, \tag{17}$$

*is given by*

$$M_\epsilon = \frac{T}{H} = \frac{A\beta}{\beta - 1} + \chi, \tag{18}$$

*where both $A$ and $\chi$ are positive constants. (Please refer to Appendix C for the proof.)*

When we employ the DBFL algorithm, the total time consumption $\Phi$ can be calculated as shown in (19), which is the summation of the total communication time and the total

$$
\begin{aligned}
\Phi &= M \cdot t^{tran} + H \sum_{\tau=1}^{M} t_0^{comp}(a(B_0\beta^{\tau-1})^2 + c), \\
&= \frac{(A+\chi)\beta - \chi}{\beta - 1} \cdot (t^{tran} + cHt_0^{comp}) + aB_0^2 H t_0^{comp} \frac{1-\beta^{\frac{2(A+\chi)\beta-2\chi}{\beta-1}}}{1-\beta^2} \\
&= \frac{((A+\chi)\beta - \chi)(t^{tran} + cHt_0^{comp})(\beta+1) + aB_0^2 H t_0^{comp}(e^{\ln\beta \cdot \frac{2(A+\chi)\beta-2\chi}{\beta-1}} - 1)}{\beta^2 - 1}.
\end{aligned} \tag{19}
$$

---

computing time. Hence, aiming at minimizing the overall time consumption, we determine the best incremental factor $\beta$ by solving the following optimization problem:

$$
\min_{\beta} \quad \Phi, \tag{20a}
$$

$$
s.t.: \quad \beta \geq \beta_{\min}, \tag{20b}
$$

$$
B_0\beta^M \leq \min_i |\mathcal{D}_i|, \tag{20c}
$$

where $\beta_{\min} > 1$, $|\mathcal{D}_i|$ is the size of the dataset owned by device $i$. The constraint in (20c) limits the maximum value of the selected batch size to be smaller than the local dataset size, which can also be simplified as

$$
\ln\beta((A+\chi)\beta - \chi) \leq Z(\beta - 1), \tag{21}
$$

where $Z = \ln\frac{\min_i |\mathcal{D}_i|}{B_0}$.

We look into some critical parameters of the objective function $\Phi$ with further analysis. The total time consumption includes two parts, that is, communication consumption ("talking") and computing consumption ("working"). The results in (18) indicates that the incremental factor $\beta$ takes impact on communication rounds $M$. Given a target global accuracy, a larger $\beta$ results in a smaller number of communication rounds (less "talking"). However, enlarging the incremental factor leads to the increasing of the calculations each round (more "working"). Therefore, there exists a trade-off between communication cost and computing cost that requires us to find an optimal $\beta$ to determine "work more" or "talk

more" for minimizing the total time consumption. Furthermore, the overall time consumption decreases as the number of devices $K$ increasing, but the relationship between the total time consumption and the local step $H$ is unknown because of the non-monotonicity.

### 3.4.2 Optimal Control Solutions for DBFL

For notational convenience, we define following functions:

$$P_1(\beta) = ((A+\chi)\beta - \chi)(t^{tran} + cHt_0^{comp})(\beta+1), \tag{22}$$

$$P_2(\beta) = aB_0^2 H t_0^{comp}(e^{\ln\beta \cdot \frac{2(A+\chi)\beta - 2\chi}{\beta-1}} - 1), \tag{23}$$

$$P_3(\beta) = \beta^2 - 1, \tag{24}$$

$$\text{and } G(\beta) = \ln\beta((A+\chi)\beta - \chi) - Z(\beta - 1). \tag{25}$$

The objective function $\Phi$ of the optimization problem is in a fractional form. To solve it efficiently, we employ the Dinkelbach [28] method to iteratively find the optimal results, which is widely considered in dealing with fractional programming problem. Introduce an auxiliary variable $q$ and set $q$ as the value of the objective function $\Phi$, that is, $q = \frac{P_1(\beta)+P_2(\beta)}{P_3(\beta)}$. Accordingly, we have $P_1(\beta) + P_2(\beta) - qP_3(\beta) = 0$, and we further define the function $P_1(\beta) + P_2(\beta) - qP_3(\beta)$ as $\Theta(\beta, q)$. Here, $\Theta(\beta, q)$ is strictly monotonic decreasing with the parameter $q$. Notice that solving the problem in (20) is essentially equivalent to finding the root of $\Theta(\beta, q)$. Therefore, we use the parametric approach to represent the objective in (20) and consider the following problem:

$$\min_{\beta} \quad P_1(\beta) + P_2(\beta) - qP_3(\beta), \tag{26a}$$

$$s.t.: \quad \beta \geq \beta_{\min}, \tag{26b}$$

$$G(\beta) \leq 0. \tag{26c}$$

We iteratively solve the problem (26) and update the non-negative variable $q$ in each iteration. Finally, $q$ will be the unique root of the function (26a), which is also the optimal

value of original problem (20). The updating process of variable $q$ is described in Alg. 2. In general, the solution of the original problem (20) can be summarized as follows. We first solve problem (26) and denote the optimal solution as $\beta^*$, which is referred to as the inner loop in Alg. 2. Then, we update the corresponding $q$ value, referred to as the outer loop in Alg. 2. Finally, repeat the above two loops until some specific convergence conditions are reached. We further investigate problem (26) and have the following lemma.

**Lemma 1** *The function $P_1(\beta)$, $P_3(\beta)$, $G(\beta)$ are convex functions in problem (26).*

*Proof.* $P_1(\beta)$ and $P_3(\beta)$ can be easily verified to be convex. For the function $G(\beta)$, we have

$$\frac{\mathrm{d}^2 G(\beta)}{\mathrm{d}^2 \beta} = \frac{\mathrm{d}((A+\chi)(\ln\beta+1))}{\mathrm{d}\beta} - \frac{\mathrm{d}(\chi\beta^{-1}+Z)}{\mathrm{d}\beta} = \frac{A+\chi}{\beta} + \frac{\chi}{\beta^2} > 0, \tag{27}$$

which implies that the function $G(\beta)$ is convex. □

We now first focus on the function $P_2(\beta) = aB_0^2 H t_0^{comp}(e^{\ln\beta \cdot \frac{2(A+\chi)\beta - 2\chi}{\beta-1}} - 1)$. Let $U_1(\beta) = h(g(\beta)) = e^{g(\beta)}$ where

$$h(x) = e^x, \forall x \in \mathbb{R} \tag{28}$$

and

$$g(\beta) = \ln\beta \cdot \frac{2(A+\chi)\beta - 2\chi}{\beta - 1}. \tag{29}$$

Then $P_2(\beta)$ can be rewritten as $P_2(\beta) = aB_0^2 H t_0^{comp}(U_1(\beta) - 1)$. Notice that $g(\beta)$ is concave since

$$\frac{\mathrm{d}^2 g(\beta)}{\mathrm{d}^2 \beta} \leq 2 \cdot \frac{\chi(\beta-1)^3 + A\beta(\beta-1)^2}{(1-\beta)^3\beta^2} < 0, \tag{30}$$

Thus, it is difficult to judge the convexity of the composition function $U_1(\beta)$, as well as $P_2(\beta)$. Besides, the parameter $q$ introduced in the Dinkelbach framework is non-negative, the function $P_1(\beta) - qP_3(\beta)$ has the Difference of Convex (DC) structure since both $P_1(\beta)$

and $P_3(\beta)$ are obviously convex. Hence, we further set this DC structure as $U_2$, i.e., $U_2(\beta) = P_1(\beta) - qP_3(\beta)$.

Therefore, we can re-represent the problem in (26) as:

$$\min_{\beta} \quad aB_0^2 H t_0^{comp}(U_1(\beta) - 1) + U_2(\beta) \tag{31a}$$

$$s.t. \quad \beta \geq \beta_{\min}, \tag{31b}$$

$$G(\beta) \leq 0. \tag{31c}$$

For solving the non-convex objective (31a), we apply the iNner cOnVex Approximation (NOVA) [29] algorithms to linearize the non-convex terms in (31) and find the stationary points iteratively. The main idea is to continuously optimize the approximation of the non-convex objective in (31) and maintain feasibility at each iteration. Hence, we derive a strongly convex approximant of (31a) around each feasible iteration. In order to achieve that, we build an approximation for functions $U_1(\beta)$ and $U_2(\beta)$ as follows:

$$\tilde{U}_1(\beta; \beta^\nu) \triangleq h(g(\beta^\nu) + \nabla g(\beta^\nu)(\beta - \beta^\nu)) + \frac{\kappa}{2}\|\beta - \beta^\nu\|^2, \tag{32}$$

$$\tilde{U}_2(\beta, \beta^\nu) \triangleq P_1(\beta) - qP_3(\beta^\nu) - q\nabla P_3(\beta^\nu)(\beta - \beta^\nu), \tag{33}$$

where $\beta^\nu$ denote the current intermediate $\beta$ obtained in the $\nu$-th iteration. $\nabla g(\beta^\nu)$ and $\nabla P_3(\beta^\nu)$ are the gradients of $g$ and $P_3$ at $\beta^\nu$, respectively, and $\kappa > 0$. To acquire the optimal solution $\beta^*$ in (31), we iteratively compute the solution $\beta(\beta^\nu)$ by taking a step from $\beta^\nu$ with a initial feasible point $\beta^0$. In this way, we build the approximation for $\Theta(\beta)$ in (31) as $\tilde{\Theta}(\beta) = \tilde{U}_2(\beta, \beta^\nu) + aB_0^2 H t_0^{comp}(\tilde{U}_1(\beta) - 1)$ and the approximated problem of (31) can be formulated as:

$$\min_{\beta} \quad \tilde{\Theta}(\beta) \tag{34a}$$

$$s.t. \quad \beta \geq \beta_{\min}, \tag{34b}$$

$$G(\beta) \leq 0. \tag{34c}$$

**Algorithm 2** Batch Sizes Control of DBFL

**Initialization:** Initialize $q = q^{(0)} > 0$, outer-loop iteration index $k = 0$, the stop criteria $\varepsilon$, and the maximum number of iterations $K_{max}$

**repeat**
  Initialize the step size $\zeta^0 \in (0, 1]$
  Set inner-loop iteration index $\nu = 0$ and start with $\beta^0$
  **repeat**
    Calculate $\beta^*(\beta^\nu)$ via (38)
    Set $\beta^{\nu+1} = \beta^\nu + \zeta^\nu(\beta^*(\beta^\nu) - \beta^\nu)$
    Set $\nu = \nu + 1$
  **until** $\beta^\nu$ is a stationary solution of problem (34)
  Set $\beta^* = \beta^\nu$ as the current solution of the primal problem (26).
  Update $q^{(k+1)} = \frac{P_1(\beta^*)+P_2(\beta^*)}{P_3(\beta^*)}$
  **if** $|\Theta(q^{(k)} - \Theta(q^{(k+1)})| \leq \varepsilon$ **then**
    **return** The current solution $\beta^*$.
  **else**
    Set $k = k + 1$
**until** $k = K_{max}$
**return** Optimal solution $\beta^*$.

We can easily observe that the problem in (34) is convex, continuously differentiable, and the Slater's condition is satisfied, implying that strong duality holds. Hence, the problem can be derived using the KKT conditions, and the stationary point $\beta^*$ can be determined accordingly.

The first order derivative of the objective $\tilde{\Theta}(\beta)$ can be calculated as

$$\frac{\mathrm{d}\tilde{\Theta}(\beta)}{\mathrm{d}\beta} = I\beta + AR - 2q\beta^\nu + J(Ye^{\nabla g(\beta^\nu)\beta} + \kappa\beta - \kappa\beta^\nu) = (I + J\kappa)\beta + JYe^{\nabla g(\beta^\nu)\beta} + S, \tag{35}$$

where $R = t^{tran} + cHt_0^{comp}$, $I = 2(A + \chi)R$, $J = aB_0^2Ht_0^{comp}$, $Y = e^{g(\beta^\nu)-\nabla g(\beta^\nu)\beta^\nu}\nabla g(\beta^\nu)$, $S = (AR - J\kappa\beta^\nu - 2q\beta^\nu)$.

We further calculate the first order derivative of $g(\beta)$ and obtain $\nabla g(\beta^\nu) = \frac{2\beta^\nu(A+\chi)-2\chi}{\beta^\nu(\beta^\nu-1)} - \frac{2A\ln(\beta^\nu)}{(\beta^\nu-1)^2}$. As the first order derivative of the function $G(\beta)$ has already shown in (27), we

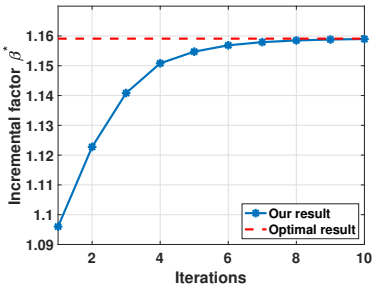



Figure 5: The iterative convergence process of the proposed solution.

have the following equations according to the KKT conditions

$$\begin{cases} \frac{\partial \tilde{\Theta}(\beta)}{\partial \beta} + \mu \frac{\partial G(\beta)}{\partial \beta} = 0, \\ \mu G(\beta) = 0, \\ \mu \geq 0, \\ G(\beta) \leq 0, \end{cases} \tag{36}$$

where $\mu$ is the Lagrange multiplier. Then, we have

$$\beta_1 = -\frac{(I + J\kappa)\mathrm{W}\left(\frac{JY\nabla g(\beta^\nu)\mathrm{e}^{-\frac{\nabla g(\beta^\nu)S}{I+J\kappa}}}{I+J\kappa}\right) + \nabla g(\beta^\nu)\mathrm{S}}{\nabla g(\beta^\nu)(I + J\kappa)}$$
$$\text{and } \beta_2 = e^{\lambda^*}, \tag{37}$$

where $\mathrm{W}(\cdot)$ represents the Lambert function, and $\lambda^*$ satisfies $Ae^{\lambda^*}\lambda^* + e^{\lambda^*}\lambda^*\chi - Ze^{\lambda^*} - \lambda^*\chi + Z = 0$. It should note that $\lambda^*$ can be easily acquired through the bisection method. We further define $\tilde{\beta} = \arg\min\{\Theta(\beta_1), \Theta(\beta_2)\}$, and the optimal solution $\beta^*$ can be given as

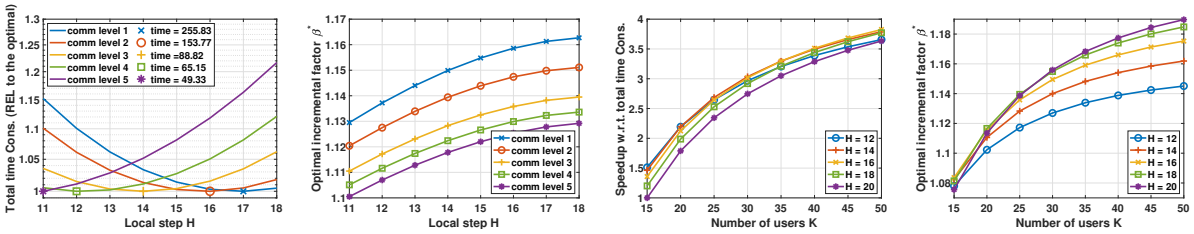
$$\begin{cases} \beta_{\min}, & \tilde{\beta} \leq \beta_{\min}, \\ \tilde{\beta}, & \text{otherwise.} \end{cases} \tag{38}$$

We summarize the procedure of the inner convex approximation as the inner-loop in Alg. 2. Alg. 2 describes how we obtain the optimal incremental factor $\beta^*$ that can control the

batch size in each training round to minimize the total time consumption in MEC networks by well balancing the communication and computing. Note that the main computational workload of the inner convex approximation lies in the calculation of $\beta^*(\beta^\nu)$, i.e., the solution of the problem in (34). We mitigate the workload by computing $\beta^*(\beta^\nu)$ in closed form, which allows us to obtain the unique solutions of the inner-loop problems directly without resorting to any iterative solver that provides approximate solutions only. Thanks to the desirable convergence properties from the Dinkelbach and NOVA algorithms, our batch size control algorithm in Alg. 2 can finish to find $\beta^*$ after a finite number of steps with relatively low computational complexity. Fig. 5 shows one approximation error example of the proposed solution, where we employ the brute force approach to find the optimal results, which verifies the accuracy and efficiency of the proposed solution.

### 3.5 Performance Evaluation

In this section, we conduct extensive simulations to validate the effectiveness of our proposed DBFL algorithm and the corresponding control scheme over future mobile edge devices. To evaluate the performance of our proposed scheme over future mobile edge devices in practice, we need to find the proper model parameters to represent the computing and communication of MEC. First of all, we take some experiments on the edge device to identify the GPU computing model. We use the edge devices occupied with one NVIDIA RTX 8000 GPU to model the edge computing environment of future mobile device in practice, where the GPU core frequency $f_{core}$ and memory frequency $f_{mem}$ equal to $1,400$ MHz and $1,750$ MHz. Through running the ResNet20 model with CIFAR-10 dataset on devices multiple times, we identify that $t_0^{comp}$ nearly equals to 0.024, and $a$ and $c$ can be selected as $9.49 \times 10^{-7}$ and 1 in function $d(B)$, respectively. Next, for the communication model, we consider the ultra fast wireless transmissions (e.g., 5G) to simulate the edge communication environments, where the achievable upload transmission rates $R$ is between 5 Mbps and 125 Mbps depending on the wireless communication conditions.




(a) Impact of comm. level on time (b) Impact of comm. level on $\beta^*$ (c) Speedup over H and K (d) Impacts of H and K on $\beta^*$

Figure 6: Numerical illustration of DBFL algorithm.

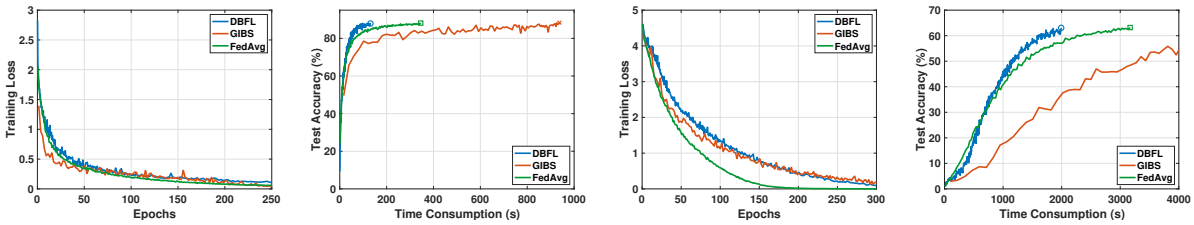
### 3.5.1 Impacts of Communication Conditions, Local Steps, and the Number of Participants on the DBFL Algorithm

After identifying the parameters of the GPU computing model and wireless communication model, the expressions of communication round $M$ and objective $\Phi$ still have some parameters to be estimated. We estimate these parameters with different learning settings by experiments and take $\frac{2b}{\gamma} = 250$, $C = 0.0005$, and $\varrho = 400$ as an example to identify the values of $A$ and $\chi$. We further take $\epsilon = 0.5$ to guarantee $\epsilon > CH^2$ for different potential $H$. The transmission model size $N$ can be estimated as 1.25 MB, and the dataset size for each user equals to $5,000$. The simulation results for different parameter settings of our proposed DBFL algorithm with optimal control are shown in Fig. 6.

We first consider different communication conditions and different local steps $H$ with a fixed number of participating devices $K = 25$, and the simulation results are shown in Fig. 6a and Fig. 6b. Different communication levels represent different communication conditions, where level 1 to 5 correspond to upload transmission rates $R = 5, 10, 25, 50$, and 125 Mbps, respectively. In particular, the level 1 indicates that the communication cost is dominant in total time consumption, while level 5 indicates that the computing cost is dominant. Fig. 6a demonstrates the total time consumption with varying local steps $H$ under different communication conditions, where y-axis represents the value relative to the optimal time consumption for each $H$. In Fig. 6a, under the same communication condition, the local step $H$ has an optimal value in the middle of the candidate ranges. This is because

too small $H$ will lead to inefficient communication. In contrast, if $H$ is too large, the over-fitting problem of local computation also affects the convergence rate. Moreover, it is easy to observe that with the deterioration of communication conditions, the time consumption will increase. It's also notable that the optimal $H$ is gradually decreasing with the rising of the communication level, as the remarked points in Fig. 6a. The reason can be explained as the optimal solutions prefer to reduce the total time consumption by putting more communication rounds (more "talking") when the communication conditions are good, and vice versa. This reflects the trade-off between communication and computing of our proposed scheme. Fig. 6b further shows the relationship between the obtained optimal incremental factor $\beta^*$ and different communication conditions. Here, as the communication conditions degrade, the optimal incremental factor $\beta^*$ increases accordingly. This is another reflection of our algorithm regulating communication and computing. The degraded communication environment indicates that the time consumption of each transmission will increase. In this situation, the incremental factor should be increased for taking more local computing tasks (more "working") in each round to reduce the global updates, so as to minimize total time.

Next, we investigate the impact of different numbers of participating devices $K$ on the total time consumption, and the results are shown in Fig. 6c and Fig. 6d. We simulate the upload transmission rates $R = 25$ Mbps, which indicates the communication cost and computing cost are comparable. Fig. 6c shows that the speedup of the total time consumption grows with the increasing of the participating users under all $H$ situations, but the growth rate gradually slows down. This results from the fact that increasing the number of participating users can help speed up the convergence, and thereby reduce the time consumption. However, when there are enough devices to capture the characteristics of the database $\bigcup_i \mathcal{D}_i$ among all users, adding extra users will only have a slightly positive impact on the training convergence. Fig. 6d depicts the increase of the optimal incremental factor $\beta^*$ when increasing the number of participating FL users. More devices participating in the training will accelerate the convergence, so the optimal incremental factor should be increased to match the change of the convergence rate, thus further minimizing the total




(a) Training accuracy vs. epochs (b) Test accuracy vs. time cons. (c) Training accuracy vs. epochs (d) Test accuracy vs. time cons.

Figure 7: DBFL simulation results on various architectures and datasets. ((a-b): ResNet20 on CIFAR-10; (c-d): VGG19-BN on CIFAR-100.)

time consumption.

### 3.5.2 Convergence Analysis and Comparisons

To demonstrate the effectiveness of our proposed DBFL scheme with optimal incremental factor $\beta^*$ under different learning architectures and varied datasets, we compare it with a typical fixed batch size FL scheme "FedAvg [1]," and a FL scheme with gradually increased batch size "GIBS". In GIBS, all devices communicate with the edge server after every local iteration, and the batch size exponentially increases in each iteration. We consider the scenarios with $K = 10$ participating devices and take local steps $H = 10$ for FedAvg and our DBFL scheme. The experiment results are shown in Fig. 12. All schemes have the same initial batch size $B_0 = 100$ and learning rate 0.2 to ensure comparability. Moreover, We guarantee that our proposed DBFL scheme and GIBS scheme have the same batch size increasing rate in terms of the training epochs, which is also consistent with the decay of the learning rate for the FedAvg scheme.

Figs. 7a and 7b show the learning results of the ResNet20 model on CIFAR-10 dataset with 1 MB parameters, where we consider the training with NVIDIA RTX 8000 GPU and simulate the current 5G network with 50 Mbps upload transmission rate. Fig. 7a demonstrates that the training loss curves of all schemes are nearly identical in terms of the number of gradient calculation operations, i.e., data epochs. Fig. 7b demonstrates the total time consumption to the same target accuracy (e.g., 88%) for the corresponding schemes, and we find that the overall time consumption of our proposed DBFL scheme with optimal

$\beta^*$ can be reduced to one third of that of the FedAvg. In particular, our proposed scheme requires fewer communication rounds than FedAvg, thus reducing the communication time. At the same time, our scheme consumes less computing time due to the large batch training efficiency. Moreover, the optimal $\beta^*$ can trade-off the relationship between communication and computing adapting to the current mobile edge environments to further reduce the overall time consumption. For the GIBS scheme, although the GIBS scheme saves some computing time than the FedAvg, its huge communication overhead makes its convergence speed very slow. We also conduct the experiments of a relatively large model, i.e., the VGG19 model with batch normalization, on CIFAR-100 dataset, where it contains 549 MB parameters. Here, we simulate the 5G network with 1 Gbps upload transmission rate. The results are shown in Figs. 7c and 7d, which reach similar conclusions as those of the "ResNet" one. It is noteworthy that the performance of the FedAvg scheme is slightly better than the others. The reason is that the relatively small batch size of the FedAvg on the initial learning stage speeds up the convergence in this model setting.

From Fig. 7a and Fig. 7c, we also find that the proposed DBFL approach has a similar convergence performance as the typical FL approach, e.g., FedAvg. In other words, both the DBFL and the FedAvg can approach the same target accuracy with similar training epochs. For this reason, there are some potential benefits of our DBFL algorithm when comparing with other time-efficient FL algorithms integrating model compression techniques (e.g., the quantization, sparsification, etc.). On the one hand, unlike the model compression based approaches that usually result in a considerable sacrifice of training accuracy, the DBFL algorithm ensures all the gradient information is aggregated in every global round without distortion, and thereby hardly impairs the learning outcome. On the other hand, our DBFL method can be considered an add-on module integrated with any model compression method for further FL time reduction.

## 3.6 Conclusion

In this paper, we have focused on minimizing the overall time consumption of FL over future mobile edge devices in terms of both communication and computing time. We first developed a time efficient FL algorithm, named DBFL, with the convergence guarantee, where the batch size exponentially increases during FL training. In particular, gradually increasing the batch sizes in DBFL can fully adapt to the GPUs' parallelism property and reduce the required communication rounds for global updates. To minimize the total time consumption of deploying DBFL in MEC networks, we have further designed a batch size control scheme, which helps balance the time consumption of wireless communication and that of local computing in practice. Extensive simulations have shown the impacts of different communication conditions, local steps, and the number of participants on the proposed DBFL algorithm. Furthermore, the results have validated the effectiveness of our proposed control scheme, which saves nearly $3\times$ time than the state-of-the-art approaches. Appendix proof is available for this work at https://github.com/shidian117/DBFL/blob/main/Appendix.pdf.

# 4 Energy and Spectrum Efficient Federated Learning via High-Precision Over-the-Air Computation

## 4.1 Introduction

With the development of mobile communications and Internet-of-Things (IoT) technologies, mobile devices with built-in sensors and Internet connectivity have proliferated and generated huge volumes of data at the network edge. These data can be collected and analyzed to build increasingly complex machine learning models. To avoid raw-data sharing among the untrustworthy parties and leverage the ever-increasing computation capability of mobile devices, the emerging federated learning (FL) framework allows participating mobile devices to collaboratively train a machine learning model under the orchestration of a centralized server by just exchanging the local model updates with others via wireless communications. With such good properties, FL over mobile devices has inspired a wide utilization in a large variety of intelligent services, such as the keyword prediction [1], voice classifier [30], and e-health [8], etc.

Although only model updates instead of raw data are transmitted between mobile devices and the FL server, such updates could contain hundreds of millions of parameters with complex neural networks. That makes the uplink transmissions from mobile devices to the FL server for model updates particularly challenging, resulting in a huge burden on both wireless networks and mobile devices. On the one hand, the spectrum resource that can be allocated to each device decreases proportionally as the number of devices increases, hampering the scalability of FL, i.e., FL over a large number of mobile devices, if there are limited spectrum resources. On the other hand, transmitting a large volume of model updates periodically and executing heavy local on-device computing tasks can quickly drain out the energy of batter-powered mobile devices. Such a mismatch restricts mobile devices or makes them reluctant to participate in FL.

Over-the-air computation (AirComp) provides a promising solution to address the aforementioned spectrum challenge by achieving scalable and efficient model update aggregation

in FL. Unlike the conventional orthogonal multiple access techniques, where each user is restricted to its allocated spectrum band, AirComp allows all the users to utilize the whole spectrum for transmissions simultaneously. By applying AirComp to FL, all the participating devices can transmit their model updates on the same channel. Due to the fact that MAC inherently yields an additive superposed signal, the signals of all the participating devices are aligned to obtain desired arithmetic computation results directly over the air, thus significantly improving the spectrum efficiency. However, most works in the literature employ the analogy modulation to design their over-the-air FL schemes, which is not compatible with commercial off-the-shelf digital mobile devices and thus hinders their deployment in current/future communication systems, such as LTE, 5G, Wi-Fi 6, and 6G, etc. Besides, most existing efforts focus on one iteration transmission performance of the Aircomp FL [31, 32], and the effects of AirComp on overall FL training performance, especially the FL convergence, are rarely discussed.

Therefore, in this work [33], we design a novel multi-bit Aircomp (M-AirComp) FL scheme, named ESOAFL, which is compatible with the most common Quadrature Amplitude Modulation (QAM) to transmit the model updates, so that we do not need to modify the modulation protocols manufactured within commercial off-the-shelf mobile devices. Specifically, gradient quantization is incorporated into the ESOAFL scheme to facilitate the digital modulation, and only part of the gradients are selected to transmit to cope with the channel fading. In addition to handling the spectrum issue in FL over wireless networks, our scheme is battery-friendly to the participating mobile devices. Here, the energy consumption is considered from the long-term learning perspective where local computing (i.e., "working") and wireless communication (i.e., "talking") are two main focuses. Our M-AirComp FL scheme only requires updated gradients with good channel conditions to transmit, which further saves the communication energy compared with other AirComp schemes. Moreover, we theoretically analyze the convergence property of our ESOAFL approach, based on which we quantify the number of communication rounds needed for

achieving the convergence, and the overall long-term energy consumption is further modeled. Finally, we develop a joint transmission probability and local computing control approach to balance "working" and "talking," thus minimizing overall energy consumption. Our salient contributions are summarized as follows.

- We propose an energy and spectrum efficient AirComp FL approach (ESOAFL) with high precision M-AirComp design, where updated gradients are quantized into multi-bit adapting to the digital modulation settings. Additionally, we adopt an energy efficient power control policy to facilitate the M-AirComp, where only updated gradients with good channel conditions are selected to participate in the FL training.

- To help minimize the overall energy consumption of the proposed ESOAFL approach, the corresponding convergence analysis is derived, which quantitatively indicates the impacts of the M-AirComp on FL. Besides, the gradients transmission probability and local computing iterations are further optimized to achieve the overall energy efficiency.

- We conduct extensive simulations and verify the effectiveness of our proposed ESOAFL scheme and the corresponding control approach under various learning models, datasets, and multiple wireless environmental settings. Compared with other schemes, our proposed method shows significant spectrum utilization and energy efficiency superiority. The ESOAFL approach has the potential to improve spectral efficiency dozens of times and save at least half of the energy consumption.

## 4.2 Preliminaries of AirComp FL

During the FL training process, all users need to transmit their local updates to the server for aggregation, which causes severe transmission congestion and requires a lot of communication resources, especially with massive participating users. As one of the advanced wireless techniques, over-the-air computation (AirComp) enables all users to simultaneously transmit the local gradients over the same wireless medium without interference

and aggregates gradients together in the transmission process, significantly improving the spectrum utilization and saving communication resources.

Let $\mathcal{X} := \{x_1, x_2, ..., x_K\}$ be the input set and $\tilde{y}$ be the output objective of the system. In other words, $x_k$ denotes gradients to be transmitted for user $k$ in FL, and $\tilde{y}$ denotes the aggregation result received at the server with AirComp. Specifically, with AirComp, a wireless communication system usually adopts precoding and amplification at transmitters, while receivers often have equalization blocks for signal detection. Therefore, AirComp computes the aggregated objective at each time slot as

$$\tilde{y} := \text{Air}\left(\mathcal{X}\right) = \frac{a}{K}\left[\sum_{k=0}^{K} h_k p_k x_k + n\right], \tag{39}$$

where $h_k \in \mathbb{C}$ is the channel coefficient between user $k$ and the aggregator, $h_k$ satisfies the Rayleigh distribution, i.e., $h_k \sim CN(0, \sqrt{\lambda})$. $n \sim N(0, \sigma_z^2)$ is the additive white Gaussian noise (AWGN) at the receiver. The Tx-scaling factor $p_k \in \mathbb{C}$, a.k.a. power control policy, compensates the phase shift posed by the channel and jointly amplifies the transmitted data. The goal of the Tx-scaling is to ensure that each distributed user contributes equally at the receiver antenna and the superposed signal is proportional to the ideal summation. Note that the ideal summation is defined as the average operation over the input set without the AirComp, i.e., $y := \frac{1}{K}\sum_{k=1}^{K} x_k$. Accordingly, the Rx-scaling factor $a \in \mathbb{R}$ acts as a equalizer, and rescales the sampled analog result to its expected value.

## 4.3 M-AirComp Design and M-AirComp based FL

### 4.3.1 The Design of Multi-Bit Over-the-Air Computation

Different from the most existing AirComp approaches with an analogy modulation scheme, we implement the digital modulation scheme for the AirComp to pair with the commercial transmit devices and design a Multi-bit AirComp scheme (M-AirComp). In this situation, the Rx-scaling factor $a$ acts as a digital domain equalizer, and the division operation in Eq. 39 to calculate the arithmetic average is also in the digital domain. In

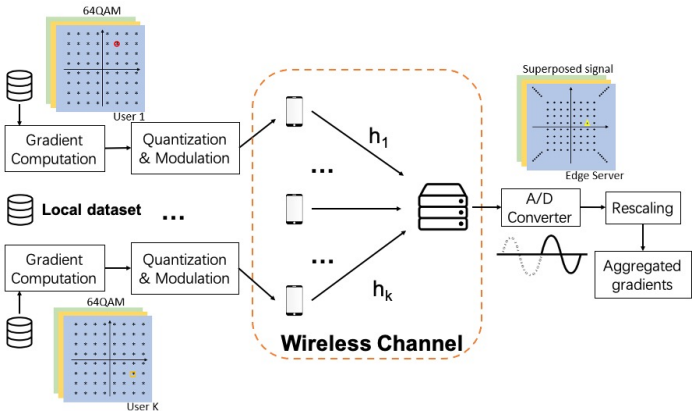



Figure 8: Multi-bit Over-the-Air computation design.

order to eliminate the burden of redesigning the modulation scheme, we tend to integrate the gradient quantization to the most common Quadrature Amplitude Modulation (QAM) in LTE, 5G, and Wi-Fi 6 standard. Instead of transmitting arbitrary values, each gradient is clipped and quantized as the Multiple Amplitude Shift Keying (MASK) symbol, which is compatible with modern digital devices. In the following, two MASK modulated gradients can be transmitted orthogonally using in-phase (I) and quadrature (Q) channel simultaneously. We notice that it is equivalent to map two separate gradients onto a symbol from the square $M^2$ QAM constellation. We limit $M$ between 2 to $2^b$. For example, when $b$ is set as 3, the user will use 64QAM to transmit two gradients shown in Fig. 8. In this way, altering the value M at the transmitter allows full digital data transmission while preserving $b$-bit resolution, according to the estimated channel gain.

We assume the edge server will equip with a high-resolution analog-to-digital converter (ADC) (e.g., 16-bit). While receiving, multiple QAM symbols superpose at the sampling instance, which can be viewed from (a part of) a higher-order rectangular QAM constellation diagram (when the number of mobile devices is odd) or a zero-centered constellation diagram (when the number of users is even). However, since the biggest possible value after aggregation can be obtained from user feedback, we can utilize this value as the ADC reference voltage. In order to alleviate the detection complexity, we directly use the quantized samples followed by Rx-scaling defined in Eq. 39 in the digital domain. In this way, the transmission module is implemented in a digital manner, which enables the M-AirComp to

have better compatibility compared with traditional AirComp. The process is also illustrated in Fig. 8. This result can be viewed as the desired computational result added by quantization error and channel noise, whose impacts on federated learning performance are analyzed in the following section.

During the transmission process, each device is constrained by an average transmitting power budget $P^0$. Thus, assume all devices have the same power budget, and the average transmission power constraint is

$$\mathbb{E}[|p_k|^2] \leq P^0, \forall k. \tag{40}$$

Due to the transmit power constrain, some users facing the poor fading channels cannot completely align their amplitude, which means the Tx-scaling factor $p_k \in \mathbb{C}$ cannot be infinitely enlarged to meet the amplitude alignment requirement. Therefore, we adopt an energy efficient power control policy that the gradients of users with poor channel conditions are not transmitted. In other words, we set the transmit power of these users as 0, thus saving the transmission energy. Let $g_{\text{th}}$ denote as a channel threshold, and the power control policy $p_k$ can be represented as

$$p_k = \begin{cases} \frac{\sqrt{\varrho} h_k^\dagger}{|h_k|^2}, & |h_k|^2 \geq g_{\text{th}} \\ 0, & |h_k|^2 < g_{\text{th}} \ . \end{cases} \tag{41}$$

For the above equation, $\varrho$ is a scaling factor to guarantee the desired SNR. Under the above power control policy, only gradient elements facing channel gain larger than the threshold $g_{\text{th}}$ can be allowed to transmit. Note that the threshold $g_{\text{th}}$ can be adjusted to control the gradient transmission. Due to the power constraint in Eq. 40, the threshold $g_{\text{th}}$ can be set as an arbitrary value larger than a minimum value $g_{\text{th}}^{\min} := h^2 = \frac{\varrho}{p^2} = \frac{\varrho}{P^0}$. Specifically, under a certain communication environment, the greater the threshold $g_{\text{th}}$ we set, the larger the number of allowable transmitting gradients. By changing the threshold $g_{\text{th}}$, our M-AirComp design has the potential to only involve gradients with good

**Algorithm 3 E**nergy and **S**pectrum Efficient **O**ver the **A**ir **F**ederated **L**earning Algorithm (ESOAFL)

**Initialization:** Initialize the global model $\mathbf{w}^0$ and set $\mathbf{w}_k^{0,0} = \mathbf{w^0}, \forall k \in \mathcal{K}$; Set the learning rate $\gamma$ and $\eta$, local computing iterations $H$, and the channel gain threshold $g_{\text{th}}$

Initialize the communication index $r = 0$ and the local computing iteration count $h = 0$

**while** $r < R$ **do**
  **for** $h = 0, ..., H-1$ **do**
    Each device $k$ computes the unbiased stochastic gradients $\nabla F_k(\mathbf{w}_k^{r,h})$ of $f_k(\mathbf{w}_k^r)$ with one batch size of data from the dataset $\mathcal{D}_k$
    Each device $k$ in parallel updates its local model: $\mathbf{w}_k^{r,h+1} = \mathbf{w}_k^{r,h} + \eta \nabla F_k(\mathbf{w}_k^{r,h}), \quad \forall k$
  **end for**
  Each device $k$ calculates the accumulated gradients with gradient quantization as $Q\left(\eta \sum_{h=0}^{H-1} \nabla F_k(\mathbf{w}_k^{r,h})\right)$
  Each device $k$ transmits the quantized accumulated gradients if the observed channel gain larger than the pre-selected threshold $g_{\text{th}}$ , i.e., $|h_k|^2 \geq g_{\text{th}}$ ; otherwise, no transmission
  All transmitted gradients are aggregated over the air and the global model is updated as in Eq. (43)
  Update $r \leftarrow r + 1$
  Each device $k$ updates its local model $\mathbf{w}_k^{r,0} = \mathbf{w}^r$
**end while**

channel conditions, which just require low transmit power in an energy efficient manner. Thus, we define a long-term average transmission probability $p_b$ to indicate the degree of gradient participation. To be specific, each specific threshold corresponds to a transmission probability. Since the channel coefficient is Rayleigh distributed $h_k \sim CN(0, \sqrt{\lambda})$, the channel gain is an exponential distribution. Therefore, the transmission probability $p_b$ corresponding to the threshold $g_{\text{th}}$ can be calculated as

$$p_b = \int_{g_{\text{th}}}^{\infty} \lambda e^{-\lambda x} \mathrm{d}x = e^{-\lambda g_{\text{th}}} . \tag{42}$$

If the probability of keeping these gradient elements to transmit is $p_b$, the Rx-scaling factor $a$ will be set as $\frac{1}{\sqrt{\varrho} p_b}$ to rescale the received signal. Due to the property of the Rayleigh fading channel and the power constrain of the local user devices, the achievable highest transmission probability $p_b^{\max}$ is calculated as $p_b^{\max} = e^{-\lambda g_{\text{th}}^{\min}} = e^{-\lambda \frac{\varrho}{P_0}}$.

### 4.3.2 M-AirComp Based FL

To improve spectrum efficiency and reduce energy consumption, we design an **E**nergy and **S**pectrum Efficient **O**ver the **A**ir **F**ederated **L**earning (**ESOAFL**) approach with gradient quantization technique, which is shown in Fig. 8 and described in detail in Alg. 3. All mobile devices start the training procedure with the initialized model parameters. Specifically, each user executes $H$ local computing SGD steps with mini-batch size data of its own dataset. After the local training, we adopt the uniform gradient quantization operator $Q(\cdot)$ to quantize the updated gradients with low bits, i.e., 4-bit or 8-bit. Taking $b$-bit quantization as an example, the gradients of all participants are quantized to $2^b$ levels with a specific maximum/minimum value, catering to the digital wireless transmission scheme. Next, for the transmission process, every 2 gradient element is modulated into one digital symbol over the sub-channel according to our M-AirComp design. We assume the symbol-level synchronization among all the mobile devices that ensures coherent and concurrent transmission. This assumption can be realized by dedicating the bandwidth for mobile device synchronization, e.g., 1.08 MHz primary synchronization channel (PSCH) and secondary synchronization channel (SSCH) in LTE system [34], or the AirShare[35] for distributed MIMO synchronization. Then we employ the M-AirComp operator Air$(\cdot)$ and apply the proposed energy efficient power control scheme. The threshold $g_{\text{th}}$ is determined firstly, and then gradient element whose corresponding channel gain larger than this threshold can be allowed to transmit. In this way, the long-term transmission probability $p_b$ can be calculated as $e^{-\lambda g_{\text{th}}}$. Because M-AirComp integrates wireless transmissions and aggregation over the air, the server receives only the aggregated updated gradients. Finally, the server updates the global model with the aggregated updated gradients, which can be represented as

$$\mathbf{w}^{r+1} = \mathbf{w}^{r} - \text{Air}\left(\left\{Q\left(\eta \sum_{h=0}^{H-1} \nabla F_k(\mathbf{w}_k^{r,h})\right)\right\}_{\mathcal{K}}\right). \tag{43}$$

After updating the global model, the server will broadcast the global model to all devices for continuing training. We repeat the above procedure for $R$ rounds until the model converges to a stationary point. Particularly, the convergence requirement can be represented as $\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f^r\|_2^2 \leq \epsilon$, where $\epsilon$ denotes the target training loss and $\nabla f^r$ is the global function gradient at round $r$.

## 4.4 Spectrum and Energy Efficient FL: Formulation and Solutions

In this section, we first formulate an overall energy minimization problem and establish the communication and computation energy models of the proposed ESOAFL approach. Based on the derived convergence analysis, we then find the optimal control policy in terms of the transmission probability $p_b$ and local computing iterations $H$ to minimize the overall energy consumption.

### 4.4.1 Energy Minimization Problem Formulation

Deploying energy-hungry FL training on mobile devices is challenging due to the limited battery capacity of mobile devices. Hence, in this work, we aim to minimize the total energy consumption of FL training via local computing iterations $H$ and transmission probability $p_b$ control. The average energy consumption per communication round of mobile device casts as $E = E^{comm}(p_b) + E^{comp}H$. Here, $E^{comm}(p_b)$ is the communication energy to transmit the updated gradients, which is related to the transmission probability $p_b$, and $E^{comp}$ is the computing energy of performing one local iteration. The goal is to minimize the overall energy consumption in FL training while guaranteeing the model convergence, denoted as

$$\begin{aligned} \min \quad & \mathbb{E}\left[E_{tot}\right] \triangleq \mathbb{E}\left[RE^{comm}(p_b)\right] + \mathbb{E}\left[RE^{comp}H\right], \\ \text{s.t.:} \quad & \frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\left[\|\nabla f^r\|_2^2\right] \leq \epsilon. \end{aligned} \tag{44}$$

### 4.4.2 Communication and Computation Energy Models

**Communication model** If we consider the M-AirComp power control policy with transmission probability $p_b$ which should be smaller than $p_b^{\max}$, the threshold channel gain is mapped as $g_{\text{th}} := -\frac{1}{\lambda}\ln p_b$. In this way, the average power consumption among all users and time slots will be

$$P^{comm} = p_b \varrho \int_{g_{\text{th}}}^{\infty} \lambda \frac{1}{x} e^{-\lambda x} \mathrm{d}x = -p_b \varrho \lambda \mathrm{Ei}\left(-\lambda g_{\text{th}}\right) = -p_b \varrho \lambda \mathrm{Ei}\left(\ln p_b\right), \tag{45}$$

where $\mathrm{Ei}(x)$ is the exponential integral function denoted as $\mathrm{Ei}(x) = \int_{-\infty}^{x} \frac{e^t}{t}\mathrm{d}x$. Due to the fact that $-\ln p_b$ is positive, we have $\mathrm{Ei}\left(\ln p_b\right) = -\mathrm{E}_1\left(-\ln p_b\right)$, where $\mathrm{E}_1(x) = \int_x^{\infty} \frac{e^{-t}}{t}\mathrm{d}x$. Therefore, we get $P^{comm} = -p_b \varrho \lambda \mathrm{Ei}\left(\ln p_b\right) = p_b \varrho \lambda \mathrm{E}_1\left(-\ln p_b\right)$. For positive real values of the $x$, $\mathrm{E}_1(x)$ can be bracketed by elementary functions as follows

$$\mathrm{E}_1(x) < e^{-x} \ln\left(1 + \frac{1}{x}\right). \tag{46}$$

Due to $-\ln p_b > 0$, we have

$$P^{comm} \approx p_b \varrho \lambda e^{\ln p_b} \ln\left(1 + \frac{1}{-\ln p_b}\right) = \varrho \lambda {p_b}^2 \ln(1 - \frac{1}{\ln p_b}). \tag{47}$$

After receiving the full precision gradients, each device is required to quantize the gradient into low-bit precision for digital transmission. Then, we adapt MASK to modulate the gradients, which means the magnitude of each symbol is sufficient to decode the transmission gradient. Let $T_s$ denote the symbol duration, which is in inverse proportion to sub-channel bandwidth $B$. Therefore, for transmitting the model with the number of $d$ gradients, $d/2$ symbol is required according to the M-AirComp design. Thus, the transmission time can be represented as $T^{comm} = \frac{d}{2M_s} T_s$, where $M_s$ symbols are transmitted in parallel.

Accordingly, the communication energy consumption for each device in each communication round is the product of the average transmission power and the transmission time,

as

$$E^{comm} = P^{comm} \times T^{comm}. \tag{48}$$

**Computational model**

With massive data are generated or collected on mobile devices, local on-device computing can naturally be treated as computation-hungry tasks. Luckily, most modern smart devices are equipped with high-performance GPUs and can handle such heavy training tasks efficiently. Therefore, we consider the GPU computational energy model here. We model the energy consumption for processing a mini-batch of data in one iteration as a product of the runtime power and the execution time, i.e.,

$$E^{comp} = P^{comp} \times T^{comp}, \tag{49}$$

where $P^{comp}$ and $T^{comp}$ are runtime power and execution time of the edge devices, respectively. Both of them are related to the GPU core frequency/voltage and the memory frequency [26], denoted as

$$P^{comp} = P^0 + af^{mem} + b(v^{core})^2 f^{core}, T^{comp} = T^0 + \frac{u}{f^{mem}} + \frac{v}{f^{core}}. \tag{50}$$

$P_0$ and $T_0$ are the static power and static time consumption; $f^{core}/v^{core}$ and $f^{mem}$ represent the core frequency/voltage and memory frequency, respectively. $a$, $b$, $u$, and $v$ are constant coefficients that reflect the sensitivity of the task execution to GPU memory and core frequency/voltage scaling [26, 27]. Given a specific FL task, i.e., a neural network model and the corresponding dataset, such coefficients can be accurately estimated according to the hardware experiments. Since there are $H$ local computing iterations between two sequential communication rounds, the energy consumption of local computing in one communication round can be calculated as the product of the energy consumption of one iteration and the local iteration number, i.e., $E^{comp} \times H$.

### 4.4.3 Impacts of Control Variables on ESOAFL Convergence

In this subsection, we derive the convergence analysis of the ESOAFL approach, where we theoretically analyze the impacts of control variables $p_b$ and $H$ on training convergence. Firstly, we have the following model assumptions.

**Assumption 4** (Smoothness) *The objective function $f_i$ is differentiable and L-smooth, as*

$$\|\nabla f_k(\mathbf{x}) - \nabla f_k(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall k. \tag{51}$$

**Assumption 5** (Bounded variances and second moments) *The variance and the second moments of stochastic gradients evaluated with a mini-batch can be bounded as:*

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right) - \nabla f(\mathbf{w})\right\|^2 \leq \sigma^2, \forall \mathbf{w}, \forall i \tag{52}$$

$$\textit{and } \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right)\right\|^2 \leq \delta^2, \forall \mathbf{w}, \forall i, \tag{53}$$

*where $\sigma$ and $\delta$ are positive constants.*

**Assumption 6** (Quantization bounded variances) *The output of the quantization operator $Q(x)$ is an unbiased estimator of its input $x$, and its variance grows with the squared of L2-norm of its argument, i.e., $\mathbb{E}[Q(x)] = x$ and $\mathbb{E}[||Q(x) - x||^2] = q||x||^2$.*

In our work, we consider the Rayleigh channel and employ the power control policy with a transmission probability $p_b$. Thus, the following assumption is obtained.

**Lemma 2** (M-AirComp bounded variances) *The output of the M-AirComp operator $Air(\mathcal{X})$ with the proposed power control scheme is an unbiased estimator of its input set $\mathcal{X}$, and its variance decreases with the increasing of the transmission probability and grows with the squared of its argument, i.e., $\mathbb{E}[Air(\mathcal{X})] = y$ and $\mathbb{E}[||Air(\mathcal{X}) - y||^2] = \frac{1}{K^2}(\frac{1}{p_b} - 1)\sum_{x_k \in \mathcal{X}} x_k^2 + \frac{\sigma_z^2}{K^2 {p_b}^2}$.*

*Proof.* Let $\mathcal{X}$ be the input set of the M-AirComp operator, and we further define $\bar{\mathcal{X}}$ as the successful transmit set to help the proof. Accordingly, the mean and the mean of the square

values can be expressed as:

$$\begin{aligned}
&\mathbb{E}[\mathrm{Air}(\mathcal{X})] \qquad (54)\\
=&\mathbb{E}\left[\frac{1}{p_b K}\left[\sum_{x_k\in\bar{\mathcal{X}}} x_k + \sum_{x_k\notin\bar{\mathcal{X}}} x_k + n\right]\right]\\
=&\frac{1}{p_b K}\left[\sum_{x_k\in\bar{\mathcal{X}}} x_k\cdot p_b + \sum_{x_k\notin\bar{\mathcal{X}}} 0\cdot(1-p_b) + \mathbb{E}\left[n\right]\right] = y\\
\text{and } &\mathbb{E}[(\mathrm{Air}(\mathcal{X}))^2] \qquad (55)\\
=&\mathbb{E}\left[\frac{1}{{p_b}^2 K^2}\left(\sum_{x_k\in\mathcal{X}} x_k + n\right)^2\right]\\
=&\mathbb{E}\left[\frac{1}{{p_b}^2 K^2}\left(\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} x_i x_j + 2\sum_{x_k\in\mathcal{X}} x_k n + n^2\right)\right]\\
=&\frac{1}{{p_b}^2 K^2}\left[\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}, i\neq j} x_i p_b x_j p_b + \sum_{x_k\in\mathcal{X}} x_k^2 p_b\right] + \frac{\sigma_z^2}{K^2{p_b}^2}\\
=&\frac{1}{{p_b}^2 K^2}\left[{p_b}^2\left((\sum_{x_k\in\mathcal{X}} x_k)^2 - \sum_{x_k\in\mathcal{X}} x_k^2\right) + p_b\sum_{x_k\in\mathcal{X}} x_k^2 + \sigma_z^2\right]\\
=&\frac{1}{K^2}\left((\sum_{x_k\in\mathcal{X}} x_k)^2 + (\frac{1}{p_b}-1)\sum_{x_k\in\mathcal{X}} x_k^2\right) + \frac{\sigma_z^2}{K^2{p_b}^2}.
\end{aligned}$$

Thus, the variance is equal to the mean of the square value minus the square of the mean value, which is represented as

$$\begin{aligned}
&\mathrm{Var}(\mathrm{Air}(\mathcal{X}))\\
=&\mathbb{E}[(\mathrm{Air}(\mathcal{X}))^2] - \mathbb{E}[\mathrm{Air}^2(\mathcal{X})]\\
=&y^2 + \frac{1}{K^2}(\frac{1}{p_b}-1)\sum_{x_k\in\mathcal{X}} x_k^2 + \frac{\sigma_z^2}{K^2{p_b}^2} - y^2 \qquad (56)\\
=&\frac{1}{K^2}(\frac{1}{p_b}-1)\sum_{x_k\in\mathcal{X}} x_k^2 + \frac{\sigma_z^2}{K^2{p_b}^2}.
\end{aligned}$$

□

**Theorem 2** *For the proposed ESOAFL approach, under the above assumptions, if learning*

*rates $\theta$ and $\eta$ satisfy*

$$1 \geq L^2\eta^2H^2 + HL\theta\eta\frac{q(2-p_b)+Kp_b}{Kp_b}, \tag{57}$$

*and with considering the gradient quantization $q$, the M-AirComp transmission probability $p_b$, and the local computing iterations $H$, the convergence rate after $R$ communication rounds can be bounded as*

$$\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f^r\|_2^2 \leq \frac{2(f(\mathbf{w}^0)-f(\mathbf{w}^*)}{\eta\theta HR} + \frac{\eta\theta L}{K}\frac{(p_b+q)}{p_b}\sigma^2 + \eta^2L^2H\sigma^2 + \frac{\theta\eta L}{HK^2{p_b}^2}\sigma_z^2, \tag{58}$$

*where $f(\mathbf{w}^*)$ is the minimum value of the loss. (Please refer to the Appendix for the proof.)*

The proof of Theorem 2 can be derived based on the L-smoothness gradient assumption on global objective [36]. After expanding the inequality of the global objective, we first bound the inner product between the stochastic gradient and full batch gradient, while we can also bound the distance between the global model and the local model. Next, we can bound the updated gradients with M-AirComp and quantization operators. Finally, by integrating the derived results above, we can obtain the convergence analysis of the ESOAFL approach.

**Corollary 3** *To achieve the linear speedup, we need to have $\theta\eta = O\left(\frac{\sqrt{K}}{\sqrt{RH}}\right)$. If we further choose $\theta\eta = O\left(\frac{1}{L}\sqrt{\frac{Kp_b}{RH(p_b+q)}}\right)$, the convergence rate can be represented as*

$$\begin{aligned}
\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f^r\|_2^2 &\leq \frac{2L(f(\mathbf{w}^0)-f(\mathbf{w}^*)\sqrt{(p_b+q)}}{\sqrt{KRHp_b}} + \frac{\sqrt{p_b+q}}{\sqrt{KRHp_b}}\sigma^2 + \frac{K}{R\theta^2}\sigma^2 + \sqrt{\frac{1}{K^3RH^3(p_b+q){p_b}^3}}\sigma_z^2 \\
&\overset{(a)}{=} O\left(\frac{\sqrt{p_b+q}}{\sqrt{KRHp_b}}(2L(f(\mathbf{w}^0)-f(\mathbf{w}^*)+\sigma^2)) + \frac{K}{R\theta^2}\sigma^2\right) \\
&\overset{(b)}{=} O\left(\frac{\chi}{\sqrt{KRH}}\right) + O\left(\frac{K}{R}\right),
\end{aligned} \tag{59}$$

*where $(a)$ is due to the fact that $O(\sqrt{\frac{1}{K^3R}})$ decays faster than $O(\sqrt{\frac{1}{KR}})$, and we set $\chi = \sqrt{\frac{p_b+q}{p_b}}$ in $(b)$.*

**Algorithm 4** JCP Control Algorithm

**Initialization:** $\epsilon, \xi, \iota = 10^{-5}$; $\gamma^0 \in (0, 1]$; $\kappa = 0$

1: **repeat**
2: Solve (65) and set the optimal value as $\boldsymbol{\phi}^*(\boldsymbol{\phi}^\kappa)$
3: Set $\boldsymbol{\phi}^{\kappa+1} = \boldsymbol{\phi}^\kappa + \gamma^0(\boldsymbol{\phi}^*(\boldsymbol{\phi}^\kappa) - \boldsymbol{\phi}^\kappa)$
4: Set $\kappa = \kappa + 1$
5: Set $\gamma^\kappa = \gamma^{\kappa-1}(1 - \xi\gamma^{\kappa-1})$
6: **until** $||\boldsymbol{\phi}^\kappa - \boldsymbol{\phi}^{\kappa-1}||_2^2 \leq \iota$
7: Round the current $H$ to the nearest integer in $\mathcal{H}$
8: **return** The current solutions of $p_b$ and $H$.

After establishing the communication and computing energy models, another key component to formulate the overall energy consumption problem is to obtain the required communication rounds. Accordingly, we can obtain it from the derived convergence analysis.

**Corollary 4** *From the Corollary 3, the required maximum number of communications for achieving the $\epsilon$ target training loss, i.e., satisfying $\epsilon = \frac{1}{R}\sum_{r=0}^{R-1} \|\nabla f^r\|_2^2$, is given by*

$$\begin{aligned} R &= O\left(\frac{2\epsilon\sigma^2 HK^2 + \chi^2(\delta+\sigma^2)^2\theta^2}{2\epsilon^2\theta^2 HK}\right) + O\left(\frac{+\chi(\delta+\sigma^2)\theta\sqrt{4\epsilon\sigma^2 HK^2 + \chi^2(\delta+\sigma^2)^2\theta^2}}{2\epsilon^2\theta^2 HK}\right) \\ &= O\left(K\right) + O\left(\frac{\chi^2}{HK}\right) + O\left(\frac{\chi}{\sqrt{H}}\right), \end{aligned} \tag{60}$$

*where $\chi = \sqrt{\frac{p_b+q}{p_b}}$ and $\delta = 2L(f(\mathbf{w}^0) - f(\mathbf{w}^*))$.*

### 4.4.4 Overall Energy Minimization Reformulation and Solution

Aiming at minimizing the energy consumption during the entire training process, we reformulate the Joint local Computing and transmission Probability (JCP) control problem as

$$\min_{p_b, H} \quad R \times (E^{comm} + HE^{comp}) \tag{61a}$$

$$= \left(\frac{A_0(p_b+q)}{p_b H} + \frac{B_0\sqrt{p_b+q}}{\sqrt{p_b H}} + C_0\right) \times \left(\varrho\lambda {p_b}^2 \ln(1 - \frac{1}{\ln p_b})T^{comm} + HE^{comp}\right),$$

$$s.t.: \quad 0 < p_b \leq p_b^{max}, \tag{61b}$$

$$H \in \mathcal{H}, \tag{61c}$$

where $A_0$, $B_0$, and $C_0$ are constants used to approximate the big-$O$ notion in Eq. 60. From the above formula, we observe that increasing the local computing iterations $H$ reduces the needed communication rounds $R$ ("talking"), but increases the computing energy consumption per round ("working"). Similarly, adjusting $p_b$ also affects the required communication rounds and the energy consumption of each round. Thus, it is necessary to optimize $H$ and $p_b$ to balance the "working" and "talking", thus minimizing the overall energy consumption.

For notational simplicity, we define $\boldsymbol{\phi} = \{p_b, H\}$ and represent the objective function as $\Theta(\boldsymbol{\phi}) = \Theta_1(\boldsymbol{\phi}) \times \Theta_2(\boldsymbol{\phi})$, where

$$\Theta_1(\boldsymbol{\phi}) = \frac{A_0(p_b + q)}{p_b H} + \frac{B_0\sqrt{p_b + q}}{\sqrt{p_b H}} + C_0 \tag{62}$$

$$\text{and } \Theta_2(\boldsymbol{\phi}) = \varrho\lambda {p_b}^2 \ln(1 - \frac{1}{\ln p_b})T^{comm} + HE^{comp}. \tag{63}$$

Noticing the simple and decoupled constraints in (61b-61c), we relax the constraint in (61c) as $H_{min} \leq H \leq H_{max}$ where $H_{min}$ and $H_{max}$ are the minimum and the maximum integer in $\mathcal{H}$, respectively. Moreover, we can easily observe that both function $\Theta_1(\boldsymbol{\phi})$ and $\Theta_2(\boldsymbol{\phi})$ are positive and convex after calculating the first and second-order partial derivative of these two functions.

Capturing such the "product-of-convexity" property of the objective function $\Theta(\boldsymbol{\phi})$, we can use the inner convex approximation method [29] to solve the relaxed JCP control problem by optimizing a sequence of strongly convex inner approximations of $\Theta(\boldsymbol{\phi})$ in the form: given $\boldsymbol{\phi}^\kappa \in \Phi$

$$\Theta(\boldsymbol{\phi}, \boldsymbol{\phi}^\kappa) = \Theta_1(\boldsymbol{\phi})\Theta_2(\boldsymbol{\phi}^\kappa) + \Theta_1(\boldsymbol{\phi}^\kappa)\Theta_2(\boldsymbol{\phi}), \tag{64}$$

where $\boldsymbol{\phi}^\kappa = \{H^\kappa, {p_b}^\kappa\}$ refers to the intermediate $\boldsymbol{\phi}$ obtained in the $\kappa$-th iteration. Obviously, the approximated objective function in (64) is strongly convex with the fixed $\boldsymbol{\phi}^\kappa$. With the surrogate function above, we are actually required to efficiently compute the optimal solutions of the following convex optimization problem in each iteration, while preserving

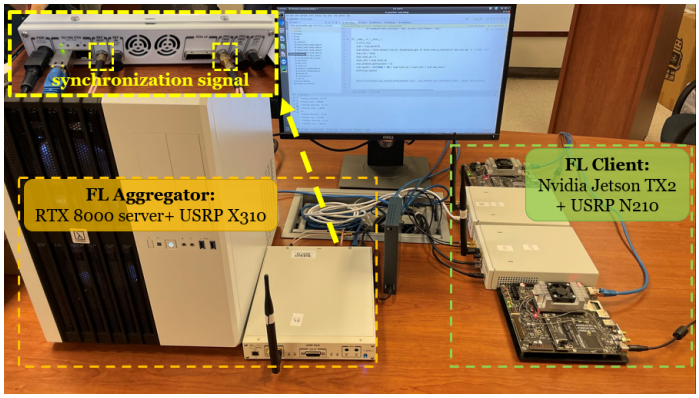



Figure 9: Federated learning via M-AirComp testbed in the lab.

the feasibility of the iterates to the original problem (61).

$$\min_{p_b, H} \quad \Theta(\boldsymbol{\phi}, \boldsymbol{\phi}^{\kappa}), \tag{65a}$$

$$s.t.: \qquad 0 < p_b \leq p_b^{max}, \tag{65b}$$

$$H_{min} \leq H \leq H_{max}. \tag{65c}$$

Notice that the problem (65) can be solved by various commercial solvers, e.g., IBM CPLEX optimizer [37]. The formal description of the Joint Power and Aggregation Control Algorithm is presented in Alg. 4. Starting from a feasible point $\boldsymbol{\phi}^0$, the method consists in iteratively computing the solution $\boldsymbol{\phi}^*(\boldsymbol{\phi}^{\kappa})$ to the surrogate problem (65), and then taking a step from $\boldsymbol{\phi}^{\kappa}$ towards $\boldsymbol{\phi}^*(\boldsymbol{\phi}^{\kappa})$. The process is repeated until it meets the termination criterion, and the value of $H$ is rounded afterward to ensure its feasibility.

## 4.5 Performance Evaluation

### 4.5.1 Implementation of M-AirComp

As shown in Fig. 9, we first set up experiments to elaborate the usage of M-AirComp for FL testbed in the lab. The system comprises one edge server as well as two edge devices. We let one RTX-8000 server with one USRP X310 play the role of the over-the-air FL aggregator. Besides, each FL client consists of the NVIDIA Jetson TX2 as the computation unit and USRP N210 as the wireless transmitter. We also employ WBX 50-2200 MHz

Rx/Tx USRP daughterboards, with up to 200 mW output power. The synchronization is provided by USRP X310 REF and PPS output ports through cable connection. In the end, all USRPs are connected to an internet switch. We run MATLAB codes from the Communication Toolbox Support Package for USRP Radio to control the transmitting and receiving in different sessions on the RTX-8000 server.

We first show the feasibility of M-AirComp by the in-lab experiments. In our M-AirComp demo, incorporated with quantization, two clients are transmitting QAM symbols, for example, 16 QAM for 4-bit quantization given in Fig. 10. From the constellation, the receiving symbol set is expanded into a constellation for higher-order modulations, which explains the addition carried in the over-the-air computation from the communication point of view. The aggregated symbol will be further decoded as a quantized model update, with a certain probability of bit error with regards to the signal-to-noise ratio (SNR).

### 4.5.2 Some Observations of the ESOAFL

AirComp can dramatically improve the spectrum efficiency in the FL training process. In addition, if the communication environment (i.e., channel condition) is extremely poor, our proposed ESOAFL approach can still retain the performance in the case of many participating devices. We consider a severe communication environment with a SNR = 5dB over different numbers of participants(e.g., K = 10, 20, and 30). Here, we train the ResNet-18 model with the CIFAR-10 dataset. As shown in Fig. 11a, with the increasing number of participating devices, the convergence gap between the ESOAFL approach and its ideal case (i.e., FedAvg without channel noise) gradually decreases. This verifies that the AirComp variance is decreasing with the number of participating devices $K$. Moreover, especially with a large number of participants (e.g., K=30), the training curve of the ESOAFL approach is similar to its ideal case (i.e., FedAvg) in terms of training epochs, which also exhibits the strong anti-interference ability of the proposed ESOAFL approach.

One difficulty in the problem-solving process is to estimate the values of $A_0$, $B_0$, and $C_0$, which are related to the specific learning model and dataset. Here, we conduct the

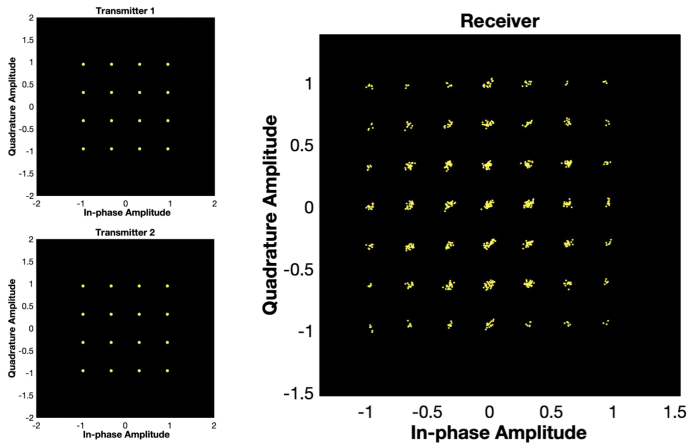



Figure 10: Constellation diagram of M-AirComp demo (left: transmitter; right: receiver).

sampling-based methods to estimate these parameters, where we empirically sample different combinations ($H$, $p_b$) and employ the derived bound in (60) to approximate these values. Note that the estimation overhead is marginal. Take the ResNet-18 model with the CIFAR-10 dataset as an example. We first implement various local computing iterations $H$ and different transmission probabilities $p_b$ for the training task. Then we set the target training loss and record the corresponding number of communication rounds. After receiving these records, we utilize the Non-linear least squares curve fitting algorithm to estimate values of $A_0$, $B_0$, and $C_0$, and the estimation results are shown in Fig. 11b. From Fig. 11b, obviously, with the increase of local computing iterations $H$ and transmission probability $p_b$, the number of required communication rounds required is decreasing, but this effect is gradually weakened. At the same time, the computing energy consumption of each round increases linearly with the incremental of local computing iterations $H$. Thus, the energy trade-off between local computing and wireless communications has to be considered to minimize the overall energy consumption, where $H$ and transmission probability $p_b$ are required to be carefully selected.

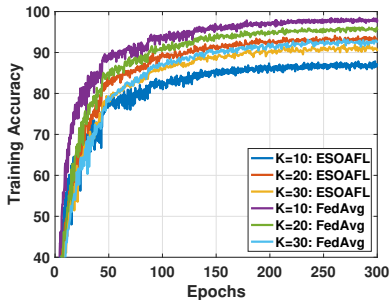
### 4.5.3 Spectrum and Energy Efficiency of the ESOAFL

After finishing the estimation process and perceiving the above observations, we implement the proposed JCP control scheme to find the optimal local computing iterations $H$

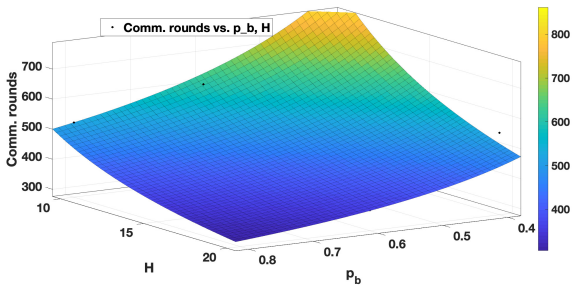
and transmission probability $p_b$. Here, we consider two different image classification models and datasets to verify the effectiveness of our proposed approach, where the LeNet model on the MNIST dataset is relatively light, and ResNet-18 on CIFAR-10 is relatively complex. Both datasets consist of 50000 training images and 10000 test images in 10 classes, and we set batch size as 128 and 32 for ResNet and LeNet, respectively. In each round of FL, we set $K = 10$ participating mobile devices executing $H$ steps of SGD in parallel, and the maximum transmission probability $p_b^{\max}$ is set to 0.77 according to the simulated communication environment and the power constraint. The initial learning rate is $\eta = 0.2$ with a fixed decay rate. We consider several popular FL schemes as baseline approaches compared with our proposed ESOAFL-OPT approach (i.e., ESOAFL with optimal JCP control).

- FedAvg [1]: the FL approach without AirComp, where the ideal transmission is taken without channel noise.
- FedPAQ [38]: all participants transmit the quantized version of model updates to the edge server.
- OBDA-ADV [39]: a modified version of the OBDA (one-bit digital AirComp), where we improve the original scheme without considering the quantization at the receiver to preserve the learning precision.
- ESOAFL-MAX: the proposed ESOAFL scheme without the transmission control, where we adopt the maximum transmission probability $p_b^{\max}$ to transmit gradients.

We assume all schemes can utilize the same amount of communication bandwidth. Furthermore, We utilize the Nvidia TX2 as the mobile device and deploy Jtop [40] tool to measure the computing energy, where the LeNet model consumes 0.03J, and the ResNet model consumes 0.5J for one training iteration. For example, training the ResNet model for one iteration consumes 130ms, and the GPU power is nearly 4W. We assume the AirComp can be deployed in the commercial LTE system for wireless transmissions. The resource block is 180 kHz, and we can obtain the transmission time of local updates with the specific model size accordingly. Moreover, we assume the average maximum transmit power




(a) Training performance under poor channel conditions.

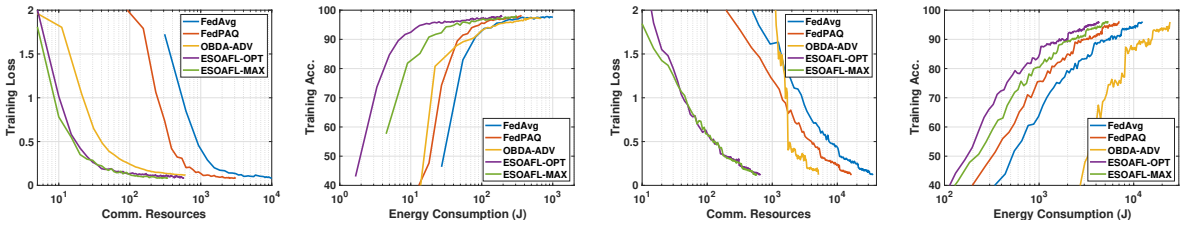



(b) Comm. rounds vs. $p_b$ and H.

Figure 11: Observations of the ESOAFL.

is 0.2W. Thus, the transmission energy consumption can be calculated as the product of transmit power and transmission time. In all schemes, We set the average SNR=15dB for participants, whose channel quality can be reflected by the CQI (Channel Quality Indicator) category 11. In this case, the modulation scheme, code rate, bits per resource element are 64QAM, 0.8525, 5.115, respectively, in FedAvg and FedPAQ for a fair comparison.

Fig. 12a and 12b show the simulation results for LeNet on MNIST. Here, we set the target training loss $\epsilon$ as 0.07 and assume the data samples are independent and identically distributed (IID). For the OBDA-ADV scheme, we bring the local SGD method (i.e., taking several training steps among the sequential communication rounds) into the original scheme. Let the spectrum resource consumed in each communication round of the ESOAFL approach as the unit communication resource. We set the gradient quantization level as 4-bit in ESOAFL and FedPAQ. Fig. 12a illustrates the communication resources consumption during the training procedure, and we can obviously find that the proposed ESOAFL significantly improves the spectrum efficiency compared with FedAvg, FedPAQ. One reason is that FedAvg and FedPAQ consume more communication resources in each round because all devices cannot take the concurrent transmission with the same bandwidth. Another reason is that each pair of gradients can be transmitted orthogonally using in-phase (I) and quadrature (Q) channels simultaneously in the proposed ESOAFL scheme. However, according to the LTE protocol, each resource element can only carry several bits of a gradient in FedAvg and FedPAQ. In the meantime, Fig. 12b presents the energy consumption during




(a) Training loss vs. comm. resources (b) Training acc. vs. energy cons. (c) Training loss vs. comm. resources (d) Training acc. vs. energy cons.

Figure 12: ESOAFL simulation results on various architectures and datasets. ((a-b): LeNet on MNIST; (c-d): ResNet-18 on CIFAR-10.)

FL training, where $H = 3$ and $p_b = 0.29$ is obtained for optimal controlling of ESOAFL. The results show that our ESOAFL scheme consumes the least energy among all schemes. Specifically, when achieving the same target training loss, the energy efficiency of ESOAFL-OPT is twice and three times higher than that of FedPAQ and OBDA-ADV, respectively. This is because the energy efficient power control policy and the digital modulation scheme in the M-AirComp design save both the transmit power and time. Moreover, since the optimized transmission probability is much lower than the maximum value, our ESOAFL-OPT approach only consumes nearly half of the ESOAFL-MAX approach's energy, which demonstrates the necessity of the JCP control scheme. Noted that the low-precision OBDA-ADV approach cannot reach the target training loss we set, and thus we consider the training loss $\epsilon = 0.12$ for the OBDA-ADV approach.

Fig. 12c and 12d demonstrate the performance comparison of all schemes with ResNet-18 model on CIFAR-10 dataset. We set the target training loss $\epsilon$ as 0.12, and obtain the optimal control strategies $H = 11$ and $p_b = 0.51$. Like the conclusions described above, the proposed ESOAFL approach dramatically improves the spectrum efficiency and reduces the required energy. In this situation, our proposed ESOAFL-OPT saves hundreds of times of communication resources compared with FedAvg and FedPAQ. It also saves more than $8\times$ of communication resources compared with the OBDA method. Accordingly, our proposed ESOAFL-OPT scheme saves nearly one-third and two-thirds of energy consumption than FedPAQ and FedAvg schemes. Furthermore, the OBDA-ADV approach has relatively

poor convergence performance compared with other approaches due to the high precision requirement of the complex ResNet-18 model.

## 4.6 Related Works

Recently, much attention has been paid to the energy-efficient FL over mobile devices, where several advanced techniques are utilized to save energy during the FL training [41]. On the one hand, gradient sparsification [42] and gradient quantization [43] techniques can compress model updates in the transmission process, significantly reducing the communication burdens [15]. On the other hand, some researchers consider applying a weight quantization scheme to reduce the required computing energy [44]. Although these methods can effectively reduce the energy cost, they are mainly considered from the perspective of learning algorithms and widely ignore the communication components, especially with the physical layer aspects of communication. Realizing the above problem, some pioneering works exploit the waveform superposition property of the wireless medium and propose the AirComp FL [45]. Cao et al. in [46], and Amiri and Gündüz in [47] apply AirComp to solve the communication bottleneck when a large number of participants aggregate the data together, where power allocation schemes are derived to satisfy the mean square error (MSE) requirements. Additionally, the works in [31] propose joint device selection and communication scheme design methods to improve the learning performance for AirComp FL. All these works take analogy modulation schemes for wireless transmission, which are difficult to be implemented on commercial devices. In addition, the convergence analysis for the whole FL training procedure is not discussed in these works. Noticing the limits above, Zhu et al. [39] applies the 1-bit digital modulation and derives the convergence analysis accordingly. However, 1-bit based scheme tremendously scarifies precision without considering the energy consumption during the training process. Different from the existing approaches, our design targets the general digital modulation scheme with multiple bits, where the convergence-guaranteed FL approach integrates both the AirComp and the gradient quantization techniques. Moreover, the energy consumption issue, including both

computing and communication, is well studied accordingly.

## 4.7 Conclusion

In this paper, we proposed the ESOAFL scheme for energy and spectrum efficient FL over mobile devices, where M-AirComp was applied for model updates transmission in a joint compute-and-communicate manner. A high-precision digital modulation scheme with multi-bit gradient quantization was designed for the participating devices to upload their model updates during FL. With the theoretical convergence analysis of the modified FL algorithm, we further developed a joint local computing and transmission probability control approach aiming to minimize the overall energy consumed by all devices. Extensive simulations were conducted to verify our theoretical analysis, and the results showed that the ESOAFL scheme effectively improves the spectrum efficiency with the learning precision guaranteed. Besides, it also saved at least half of energy consumption compared with other FL methods. Appendix proof is available for this work at https://github.com/shidian117/ESOAFL/blob/main/proof.pdf.

# 5 Make Smart Decisions Faster: Deciding D2D Resource Allocation via Stackelberg Game Guided Multi-Agent Deep Reinforcement Learning

## 5.1 Introduction

With the ever-increasing population of cellular users and their traffic requirements, enabling device-to-device (D2D) communication in cellular networks has recently been identified as an effective solution to improve the spatial frequency reuse and boost the throughput of cellular networks. D2D-enabled networks allow direct communication between two mobile users in close proximity without traversing the base station (BS). With the shared underlay spectrum assignment, D2D users are allowed to reuse the licensed spectrum of the cellular network, which improves resource utilization, enhances user experience, and expands the application of cellular communications.

However, using the shared spectrum for D2D communications makes the resource allocation more complicated due to the intertwined interference environment. Specifically, frequency reuse among cellular and D2D users may degrade the signal-to-interference-plus-noise ratio (SINR) for the ongoing transmissions over the shared channel, and thus limit the network performance. In this regard, it is indispensable to design an efficient resource allocation scheme in terms of channel allocation and power control in D2D-enabled networks, so that the users under two communication modes can coordinate with each other to improve the overall performance. Moreover, to collect accurate full channel state information (CSI) is inefficient and even impractical for a controller due to the mobility of D2D users and the time-varying communication environments. Hence, it is necessary to achieve the D2D resource allocation in distributed manner, while how to coordinate the strategies among users is very challenging.

Most existing D2D resource allocation schemes in the literature are based on optimization or game-theoretical methods, where the channel allocation coupled with power control problem is usually formulated as a mixed integer non-linear programming that is highly

non-trivial to solve [48, 49, 50]. Besides, they mainly focus on one time slot and have limited consideration about the time-varying channel conditions of mobile networks. As a result, every time the channel conditions change, they have to solve the resource allocation problem again, which incurs significant computing overheads to the negotiation phase for obtaining the optimal interference management strategy. In particular for some D2D scenarios with highly dynamic environments, e.g., vehicle-to-vehicle (V2V) communications, the existing methods are extremely time-consuming and no longer applicable. Although resource allocation with dynamics can be modeled as a Markov game, it is very difficult to formulate and solve the problem due to the unknown transition probability and the curse of dimensionality [51, 52]. As one of the most popular machine learning techniques, multi-agent deep reinforcement learning (MADRL) has recently attracted a lot of attention. In addition, modern smart mobile devices and future intelligent devices are widely armed with high-performance integrated processors (e.g., GPUs) to undertake intensive computations of heavy learning tasks. Some pioneering research efforts employ the MADRL technique to tackle the D2D resource allocation problem because of its excellent performance in coping with dynamic environments and making sequential decisions under the uncertainty for multiple agents [53, 54]. For illustrative purposes, as shown in Fig. 13, we compare the accumulated time consumed by the Stackelberg game (SG) [55] approach and the MADRL approach [56] to provide the resource allocation (power control and channel allocation) strategy in a simplified scenario. Both approaches contain 4 D2D pairs, and other settings are the same as those of the proposed method in our paper. We set the channel condition to vary every time period. Under this situation, the SG approach needs to solve the problem every period repeatedly. In contrast, the MADRL approach can train the model to converge in the initial periods and implement the trained model to make resource allocation decisions accordingly. Therefore, there exists a turning point in the curve of the MADRL approach. It is observed that the time consumption of the MADRL approach can be significantly reduced after the initial training process, while that of the SG approach for each time period stays the same and the accumulated time consumption grows much faster than

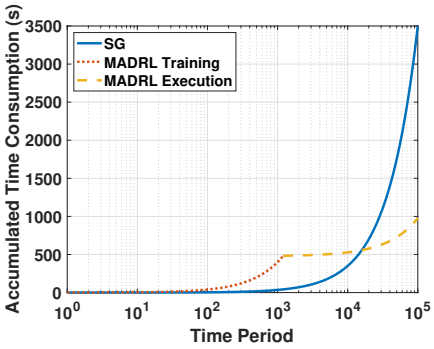



Figure 13: Accumulated time consumption between SG and MADRL (4 D2D pairs, and each time period has a new channel condition).

that of MADRL. However, MADRL solutions cannot always guarantee the performance since the learning process is usually not stable enough and difficult to converge. This makes the MADRL-based approaches still challenging to be applied in practice due to the long training time computation and limited performance improvement.

To address the issues above, this work [57] develops a faster distributed D2D resource allocation decision making approach, named **ST**ackelberg game guided multi-agent **D**eep **R**einforcement **L**earning (STDRL), by effectively integrating Stackelberg game (SG) to MADRL. Here, the BS and D2D pairs are considered as the leader and followers in the SG module, while all D2D pairs are individual agents in the MADRL module as well. Specifically, we employ the Stackelberg equilibrium in each stage game to derive a Stackelberg Q-value (ST-Q), which can provide explicit learning directions for agents in MADRL. We can also guarantee that there exists a unique equilibrium for each stage game, and the Q-value of the MADRL approach will eventually converge to the equilibrium. Benefiting from the guidance of SG, our STDRL approach, compared with the general MADRL, yields a reduced number of iterations, and thus makes the learning process much faster. Furthermore, if the time consumption can be reduced, the energy consumption that is significant for mobile devices will also be reduced. Accordingly, satisfactory and fast resource allocation strategies can be adaptively developed based on the STDRL in a distributed implementation to handle the dynamic environment well. Our salient contributions are summarized as follows:

- We develop an STDRL approach to enable interference-mitigated D2D resource allocation, which integrates SG and MADRL to achieve quick decision making in a distributed manner. Based on this approach, D2D pairs can learn to cooperate with others and make smart power control and channel allocation decisions by themselves in time-varying communication environments.

- We define the Stackelberg Q-value (ST-Q) in our STDRL to guide the learning process of agents and employ the Stackelberg equilibrium obtained from the game model to calculate ST-Q. With the guidance of the ST-Q, the learning process of our proposed STDRL approach can converge much faster than the general MADRL one. As a result, the total time consumption for learning can be significantly reduced, which helps D2D pairs take actions timely.

- We conduct a series of numerical simulations to validate the efficacy and efficiency of our STDRL approach. The results demonstrate that the proposed STDRL approach can provide proper D2D resource allocation strategies to improve network performance in a time-efficient way.

## 5.2 Related Works

### 5.2.1 Resource Allocation in D2D Networks

Recently, much attention has been paid to resource allocation and interference management problems [58], which are the critical and fundamental parts of D2D communications. For instance, Abdallah et al. [59] proposed a resource allocation framework with three power control schemes to mitigate interference in a D2D underlaid cellular system using stochastic geometry technique. Ibrahim et al. [50] applied the Bender's decomposition algorithm [60] to find the optimal power control and routing strategy in D2D cellular systems. Moreover, an enormous amount of research effort goes into a game-theoretical model for the D2D resource allocation issue [49], particularly applying the Stackelberg game [55]. Specifically, Sawyer et al. [61] defined different utility functions for different types of followers in a Stackelberg

game and proposed a distributed algorithm to find the equilibrium in D2D communications. Zhang et al. [62] proposed a hierarchical game framework to analyze the individual's distributive strategies on the visible light communication (VLC) network. All these works targeted at deriving the optimal strategy for one time slice under an optimization or game framework while not considering the dynamic channel conditions and time-varying environments of the mobile networks. Furthermore, some DRL based approaches are developed to solve the resource allocation problems in dynamic communication environments, especially for V2V communications [53, 63, 54]. However, these RL-based approaches only consider the discrete action space. Further, unstable training process and difficulty to converge are some other challenges of these models.

### 5.2.2 Multi-Agent Reinforcement Learning with Game Theory

Multi-agent reinforcement learning is a natural modeling method for distributed control problems like the resource allocation issue in D2D communications. How to coordinate the relationship among agents has always been a difficulty of MARL. Accordingly, some research employs the Nash equilibrium obtained from the game theory into the MARL framework, which can provide a clear target for the learning process to converge. The study proposed by Hu & Wellman[64][65] implemented the general Nash equilibrium to formulate the Q-value in stochastic games. Moreover, some works target on employing the Stackelberg game to find the optimal policies in asymmetric MARL [66, 67] and improving the convergence rate with the help of game theory [68, 69]. However, these Q-learning based approaches have high computational complexity and are hard to implement in practice. Besides, these frameworks cannot handle problems with extremely large state-action spaces, such as the considered D2D resource allocation problem. Nowadays, the rapid development of deep learning techniques increases by jointly considering multi-agent deep reinforcement learning and game theory. Li et al. [70] proposed a robust MADRL framework with a zero-sum game by reviewing the worst behaviors of the agents. Zhang et al. [71] treated agents unequally and defined the bi-level reinforcement learning problem. Moreover, some other works [72, 73]

introduced the mean field game in RL to tackle the issue with a large number of agents and achieved the excellent performance in some applications [74, 75]. Unlike the existing frameworks that cannot handle large state-action spaces or do not include game theoretic solutions, we consider a specific Stackelberg game solution and adopt neural networks as the functional approximator in the MARL. Specifically, we employ the Stackelberg equilibrium obtained from the game to guide the learning process of MARL, which makes the learning process more stable and converge faster. Moreover, we implement the proposed framework to the D2D communication system with the continuous state and action space of the RL model, where we can reduce the time cost in the negotiation process and achieve the time-efficient resource allocation.

## 5.3 System Model

### 5.3.1 Problem Description

We consider a single-cell D2D underlaid cellular network where the BS is located at the center of the cell, as illustrated in Fig. 14. There are two types of users in the system, i.e., D2D users and cellular users, where D2D users are in pairs to form the D2D communication links. Assume that there are $M$ cellular users and $N$ D2D pairs in the system. We denote the sets of all the cellular users and D2D pairs by $\mathcal{M}$ and $\mathcal{N}$, respectively. Further, we consider the scenario that all D2D pairs will reuse the uplink channels occupied by cellular users. In the real communication environments, the uplink resources are less utilized than downlink resources, and the BS has more power to handle the interference than mobile devices. Another reason is that the transmit power of the BS is dominant, which may cause serious interference to the D2D links and make the reuse of downlink spectrum resources very difficult, if not impossible. Therefore, we consider D2D pairs to reuse the uplink resources in our problem.

Suppose that there are $M$ orthogonal channels preassigned to the cellular users where the $i$-th channel is assigned to the $i$-th cellular user. Orthogonal frequency division multiplexing (OFDM) is exploited to convert the frequency-selective wireless channels into multiple

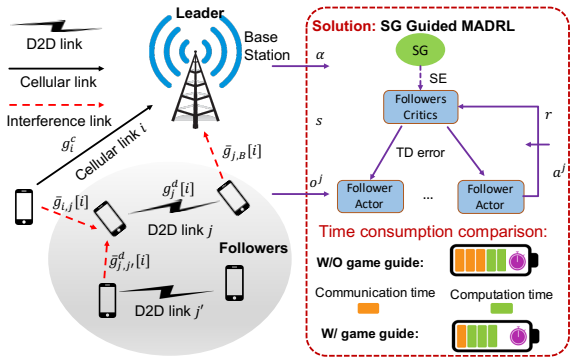



Figure 14: Deciding D2D resource allocation via SG guided MADRL.

parallel flat channels over different subcarriers. In this way, each D2D pair can reuse these available channels occupied by cellular users for better channel utilization. However, the channel reusing leads to a non-negligible problem; namely, the severe interference exists between the cellular link and the D2D link or even among D2D links. Therefore, the resource allocation strategy, including how the D2D pair can select the channel and transmit power for the communication, needs to be carefully developed to guarantee the communication quality in the system.

A set of binary variables $\{x_{ij}\}_{\forall i \in \mathcal{M}, j \in \mathcal{N}}$ are defined to indicate the status of channel reusing in the network. Here, if D2D pair $j$ reuses channel $i$, $x_{ij} = 1$; Otherwise, $x_{ij} = 0$. Note that each D2D pair is allowed to reuse only one channel, i.e., $\sum_i x_{ij} = 1$. In our problem setting, $p_i^c$ and $p_j^d[i]$ denote the transmit power for cellular user $i$ and D2D transmitter $j$ under the $i$-th channel, respectively. Moreover, we consider different channel models, e.g., different path loss models, for D2D links and cellular links. $g_i^c$ is the channel gain from cellular user $i$ to the BS, and $\bar{g}_{j,B}[i]$ is the interference channel from D2D transmitter $j$ to the BS under the $i$-th channel. Similarly, $g_j^d[i]$ denotes the channel gain between the $j$-th D2D transmitter and the $j$-th D2D receiver under the $i$-th channel. $\bar{g}_{i,j}[i]$ and $\bar{g}_{j',j}^d[i]$ denote the interference channel from cellular user $i$ and D2D transmitter $j'$ to D2D receiver $j$ over channel $i$, respectively. All of these channel gains are affected by the frequency dependent small-scale and large-scale fading, which will be changed among different coherence time

periods.

Therefore, the received signal-to-interference-plus-noise ratio (SINR) of the $i$-th cellular link and the $j$-th D2D link under the $i$-th channel can be computed as

$$\gamma_i^c = \frac{p_i^c g_i^c}{I_i^c + \mathcal{N}_0} \text{ and } \gamma_j^d[i] = \frac{p_j^d[i] g_j^d[i]}{I_{ij}^d + \mathcal{N}_0}, \tag{66}$$

where $I_i^c = \sum_j x_{ij} p_j^d[i] \bar{g}_{j,B}[i]$ represents the interference to the $i$-th cellular link from other D2D links reusing the same channel. $I_{ij}^d = p_i^c \bar{g}_{i,j}[i] + \sum_{j',j' \neq ,j} x_{ij'} p_{j'}^d[i] \bar{g}_{j',j}^d[i]$ represents the interference to D2D link $j$. $\mathcal{N}_0$ is the additive white Gaussian noise (AWGN) power. Further, the channel capacity of the $i$-th cellular link and the $j$-th D2D link under channel $i$ are given by

$$C_i^c = W \log_2(1 + \gamma_i^c) \text{ and } C_j^d[i] = W \log_2(1 + \gamma_j^d[i]), \tag{67}$$

where $W$ is the channel bandwidth.

### 5.3.2 Design Objective

To improve the network performance, a fast decision-making approach for D2D resource allocation based on SG and MADRL will be presented. As described above, the interference is incurred by the channel sharing among the D2D links and cellular links, which will inevitably degrade the network performance. Therefore, in the following sections, we first will introduce a distributed power control and channel allocation framework for the D2D-enabled network based on the Stackelberg game, which can provide the proper resource allocation strategies for one fixed period. However, every time the channel conditions change, we need to solve the Stackelberg game problem from scratch, which is time-consuming and inefficient. To tackle this issue, we then introduce a multi-agent reinforcement learning approach, where the training process can be accelerated with the guidance of the Stackelberg equilibrium, to make the sequential decisions in a distributed manner. Specifically, a game theory module helps to calculate a Stackelberg Q-value (ST-Q) in each training step, which

will be fed in the machine learning module then to contribute to the update of the learning model. In other words, this ST-Q calculated based on the equilibrium can be seen as the direction of the RL model updates, and the Q-value in the RL module gradually approaches the ST-Q with our design. Therefore, the policy (or Q-value) obtained from the RL module will converge to the equilibrium. By jointly considering the SG and the MADRL, D2D pairs can make fast resource allocation decisions to improve the network performance in dynamic communication environments.

## 5.4 Stackelberg Game Approach for D2D Resource Allocation

In this section, we employ the Stackelberg game approach to coordinate the interference among D2D links and cellular links for high resource utilization in a fixed network setting. There is a single leader (BS) and multiple followers (D2D pairs) in our game scenario, and we assume that the BS owns all of the communication resources. The D2D pairs need to pay the reuse fee for reusing the uplink channels occupied by cellular users, which are managed by the BS. Consequently, the BS charges these reuse fees as the revenue for sharing the channel resources. Therefore, under this Stackelberg game framework, the BS is willing to share its communication resources, which makes the better resource utilization. The BS sets the unit reuse price according to the current communication environment firstly. Next, each D2D pair will choose to reuse the specific uplink channel and manage the communication power based on the unit reuse price announced by the BS. In this way, every D2D pair is enabled to make both the channel allocation and power control strategies in a distributed way, while the equilibrium can be reached in the end.

### 5.4.1 Utility Functions

In this section, we introduce utility functions of the leader and the follower, namely $U^l$ and $U^f$. It should be noted that in 5G networks, end users may have various types of service requests, which leads to a slightly different formulation of utility functions for different services (e.g., short messaging service, video calling or gaming). Here, we consider a general utility function formulation to capture the ordinary service requests, and other

types of utility functions for the specific service requests are also suitable for the proposed framework. In the Stackelberg game, the BS is the leader and needs to set the prices to maximize its utility firstly. The utility of the BS can be defined as the throughput of all the cellular links and the profits obtained by sharing the uplink channels with D2D pairs. We set the profits proportional to the interference the BS observed. Thus, the utility function for the leader (BS) can be described as

$$\begin{aligned} U^l(\alpha_{ij}, p_j^d[i]) &= \sum_{i\in\mathcal{M}} C_i^c + \beta^l \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} \alpha_{ij} x_{ij} p_j^d[i] \bar{g}_{j,B}[i] \\ &= \sum_{i\in\mathcal{M}} W \log_2(1 + \frac{p_i^c g_i^c}{\sum_j x_{ij} p_j^d[i] \bar{g}_{j,B}[i] + \mathcal{N}_0}) + \beta^l \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} \alpha_{ij} x_{ij} p_j^d[i] \bar{g}_{j,B}[i], \end{aligned} \tag{68}$$

where the first term represents the throughput performance of the cellular systems, and the second term represents the reuse fee gained by the BS from sharing the uplink channel resources. Specifically, $\alpha_{ij}$ is the unit reuse price of channel $i$ sharing with D2D pair $j$, and $\beta^l$ is a scalar coefficient to balance the system's throughput gain and the reuse fee obtained from all D2D pairs. Therefore, we need to set the proper unit reuse price $\alpha_{ij}$ to maximize the leader's (BS) utility function in (68), and the optimization problem can be defined as

$$\begin{aligned} \max_{\alpha_{ij}} \quad & U^l(\alpha_{ij}, p_j^d[i]), \\ s.t.: \quad & \alpha_{ij} > 0, \forall i \in \mathcal{M}, j \in \mathcal{N}. \end{aligned} \tag{69}$$

In the above question, $p_j^d[i]$ can be selected by the D2D pair $j$. In our Stackelberg game, we consider the multi-follower scenario, where the followers are all D2D pairs. In order to mitigate the interference in the system, each D2D pair needs to choose the satisfactory reused channel and the corresponding transmit power, $p_j^d[i]$. Moreover, the transmit power for each D2D pair is bounded by maximum and minimum values. Each D2D pair has the partial observations of the whole system, which means all the D2D pairs will make the decisions at the same time in a distributed manner.

We consider the scenario that D2D users can share the content to the other users directly. The utility function of each D2D pair is determined by other D2D pairs sharing the same channel resource. For D2D users, they are sensitive to the quality of the contents, e.g., definition of video or the latency of the games. So these D2D pairs require higher achievable data rates, which have a direct impact on the quality of the transmitted contents. Meanwhile, the interference also has a impact on the reuse fee consumption. Hence, reducing the interference is another goal for D2D pairs. As a result, with the unit reused price announced by the BS, the utility function of each follower (D2D pair) $j$ under reused channel $i$ can be given as

$$U_j^f(p_j^d[i], \alpha_{ij}) = C_j^d[i](p_j^d[i]) - \beta^f \alpha_{ij} p_j^d[i] \bar{g}_{j,B}[i], \tag{70}$$

where $\beta^f$ is a positive scalar coefficient to ensure both parts have the same magnitude.

For each follower (D2D pair), the strategy is to find the proper reused channel and the proper transmit power, $p_j^d[i]$, to maximize its own utility function as defined in (70). Therefore, by considering the leader's unit reuse price, the optimization problem for D2D pair $j$ can be defined as

$$\begin{aligned} \max_{p_j^d[i]} \quad & U_j^f(\alpha_{ij}, p_j^d[i]) \quad \forall j \in \mathcal{N}, \\ s.t.: \quad & p_{min} \leqslant p_j^d[i] \leqslant p_{max}. \end{aligned} \tag{71}$$

In the next parts, we will analyze the utility functions for the leader and followers proposed in this Stackelberg game. We will find the best response, i.e., transmit power and unit reuse price, of followers and the leader when considering each other's strategies. Moreover, we will prove that there exists a unique Stackelberg equilibrium in each stage of the game.

**Algorithm 5** Resource Allocation in D2D communications by Stackelberg Game

**Input:** Given CSI of communication environments at time $t$

**Output:** Stackelberg equilibrium strategies. BS: the optimal unit reuse price $\alpha_{ij}^*$. D2D pair $j$: the reused channel $i^*$ and the corresponding transmit power $p^{*d}_j[i^*]$

**Initialization:** The unit reuse price $\alpha_{ij}$ and the transmit power strategy $p_j^d[i]$ for last time step $(t-1)$, $\forall i \in \mathcal{M}$ and $\forall j \in \mathcal{N}$ .

If time step is 0, we initially select the unit reuse price and the transmit power.

1: **for** Each time slot $t$ **do**
2: The BS calculates the optimal unit reuse price $\alpha_{ij}^*$, $\forall i \in \mathcal{M}$, $j \in \mathcal{N}$ using (75) and announce them.
3: **for** Each D2D pair (follower) $j$ **do**
4: Calculate the utility function as described in (70).
5: Calculate the optimal transmit power, using (73), given the unit reuse price announced by the leader.
6: Select the reused channel $i$ according to the policy described in (76).
7: **return** The optimal unit reuse price $\alpha_{ij}^*$, $\forall i \in \mathcal{M}$, $j \in \mathcal{N}$. For each follower $j$, $\forall j \in \mathcal{N}$, the reused channel $i^*$ and its corresponding optimal transmit power $p^{*d}_j[i^*]$.

### 5.4.2 Follower Analysis

Given the unit reuse price $\alpha_{ij}$ announced by the leader, each D2D pair determines the transmit power to maximize its utility function. The utility $U^f$ will be higher with the increasing of $p_j^d[i]$. However, if the transmit power is too large, the utility will decrease since the improvement of power consumption and reuse fee cost more significant than the increase of the channel capacity. Therefore, each follower aims to choose the proper transmit power.

**Proposition 1** *Given the unit reuse price $\alpha_{ij}$, the utility of each D2D pair $j$ under $i$-th cellular channel is continuous and strictly concave with respect to the D2D transmit power $p_j^d[i]$. (Please refer to Appendix A for the proof.)*

The best response can be obtained by setting the first derivative of the utility equal to 0, since the second-order derivative is smaller than 0, i.e., $\frac{\partial^2 U_j^f(\cdot)}{\partial p_j^{d^2}[i]} < 0$. Thus, the best transmit power can be solved as

$$\frac{\partial U_j^f(\cdot)}{\partial p_j^d[i]} = W \cdot \frac{g_j^d[i]}{\ln 2(1+\gamma_j^d[i](p_j^d[i]))(I_{ij}^d + \mathcal{N}_0)} - \beta^f \alpha_{ij} \bar{g}_{j,B}[i] = 0, \tag{72}$$

with the solution of

$$p^{*d}_j[i] = \frac{W}{\ln 2\beta^f \alpha_{ij} \bar{g}_{j,B}[i]} - \frac{I^d_{ij} + \mathcal{N}_0}{g^d_j[i]}. \tag{73}$$

According to the above equation, the transmit power will be too large or too small when the unit reuse price is extremely high or low. Therefore, to guarantee the quality of the communication service, we set the upper and lower bound of the transmit power $p^d_j[i]$, which means the best transmit power will be selected in $\{p_{min}, p^{*d}_j[i], p_{max}\}$.

### 5.4.3 Leader Analysis

The Stackelberg game has a hierarchical framework. The leader needs to consider the reaction of followers when setting strategies, which means the BS should take the corresponding best response of followers into account when setting unit reuse price.

When considering the followers' corresponding strategies, the leader's utility function will represent the impact of the leader's unit reuse price and follower's best response on the cellular system. If the unit reuse price is higher, the followers will decrease their transmit powers or select other channels. In this case, the obtained reuse fee may decrease, but the cellular system's throughput will increase due to the decrease of the interference. If the unit reuse price is lower, it will go to the opposite. Thus, we try to find the proper unit reuse price and set the upper and lower bounds of the unit reuse price as $\alpha^{max}$ and $\alpha^{min}$, respectively.

By Substituting the followers' best transmit power described in (73) into the leader's utility function, we have

$$U^l(\alpha_{ij}, p^d_j[i]) = \beta^l \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{N}} \frac{x_{ij} W}{\ln 2\beta^f} - \beta^l \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{N}} x_{ij} \left(\frac{\alpha_{ij}(I^d_{ij} + \mathcal{N}_0)\bar{g}_{j,B}[i]}{g^d_j[i]}\right) +$$
$$\sum_{i\in\mathcal{M}} W \log_2\left(1 + \frac{p^c_i g^c_i}{\sum_j x_{ij}\left(\frac{W}{\ln 2\beta^f \alpha_{ij}} - \frac{\bar{g}_{j,B}[i](I^d_{ij}+\mathcal{N}_0)}{g^d_j[i]}\right) + \mathcal{N}_0}\right). \tag{74}$$

**Proposition 2** *Given the followers' best response transmit power $p^d_j[i]$, the optimal unit*

*reuse price $\alpha_{ij}$, $\forall i \in \mathcal{M}, j \in \mathcal{N}$, can be uniquely obtained. (Please refer to Appendix B for the proof.)*

We set $A_i = p_i^c g_i^c$, $B = \frac{W}{\ln 2\beta^f}$, and $C_{ij} = \frac{-(I_{ij}^d + \mathcal{N}_0)\bar{g}_{j,B}[i]}{g_j^d[i]}$. Further, we set $D_{ij} = \sum_k x_{ik} C_{ik} + \sum_{k/j} x_{ik} \frac{B}{\alpha_{ik}} + \mathcal{N}_0$. From the solutions shown in Appendix B, the best response unit reuse price $\alpha_{ij}^*$ can be uniquely selected in

$$\begin{cases} \{\alpha_{ij}^{min}, -\frac{B}{A_i} - \frac{\beta^f B}{\beta^l C_{ij}}, \alpha_{ij}^{max}\} & D_{ij} = 0 \\ \{\alpha_{ij}^{min}, \alpha_{ij}^{max}\} & A_i + D_{ij} = 0 \\ \{\alpha_{ij}^{min}, \frac{-B(2D_{ij}+A_i)+\sqrt{\Delta}}{2D_{ij}(D_{ij}+A_i)}, \alpha_{ij}^{max}\} & D_{ij} > 0 \\ \{\frac{-B(2D_{ij}+A_i)-\sqrt{\Delta}}{2D_{ij}(D_{ij}+A_i)}, \alpha_{ij}^{\max}\} & D_{ij} < 0, A_i + D_{ij} > 0 \\ \{\alpha_{ij}^{\min}, \alpha_{ij}^{\max}\} & D_{ij} + A_i < 0, \end{cases} \tag{75}$$

where $\Delta = A_i^2 B^2 (1 - \frac{4\beta^f}{\beta^l C_{ij}} (\frac{D_{ij}^2}{A_i} + D_{ij}))$.

### 5.4.4 Joint Channel Allocation and Power Control

In the D2D underlay communication system, the D2D pair can reuse the channel occupied by cellular users to improve resource utilization. Each D2D pair needs to control its transmit power and select the proper channel to reuse so that the intra-cell interference can be significantly reduced. Based on the analysis of the BS and D2D pairs, we can set the appropriate unit reuse price for the leader and the optimal transmit power for each D2D pair. Further, each D2D pair should find a suitable channel to reuse.

After the leader announces the unit reuse price, D2D pairs will select the channel to reuse with the optimal transmit power computed in (73). Furthermore, the channel allocation policy for each D2D pair $j$ can be defined as

$$i^* = \arg\max_i (U_j^f (p^{*d}{}_j[i], \alpha_{ij}^*)). \tag{76}$$

Channel $i^*$ is the selected channel for D2D pair $j$ which can maximize its utility function, meaning $x_{i^*j} = 1$. Accordingly, the distributed allocation procedure in a fixed network

setting including power control and channel allocation is described in Algorithm 5. For instance, if multiple D2D pairs first select the same channel, the interference of that channel will increase. In this case, the BS may announce a new unit reused price for that channel, after which D2D pairs may avoid selecting that channel or reduce transmit power according to our proposed mechanism. Through the interaction among D2D pairs in a few iterations, even if each D2D pair is selfish, the system will finally reach an equilibrium to improve the system performance.

## 5.5 Stackelberg Game Guided Multi-Agent Reinforcement Learning Approach

The proper resource allocation strategies for one communication stage can be obtained according to the SG approach above. However, the communication tasks need to be executed over several coherence time slots nowadays, where the communication environment changes dynamically. If we still only adopt the game approach, we need to repeat the above process to obtain the strategy, which is time consuming. To address this difficulty, we further formulate the resource allocation issue as an MARL problem with the help of equilibrium obtained in game model to adapt to the time-varying communication environment, where each D2D pair acts as the individual agent to make smart D2D resource allocation decisions. To this end, the agents will make sequential decisions at each coherence time to handle the dynamic communication environments. According to interact with the unknown communication environments, D2D pairs obtain the local observations and take their actions to maximize their revenues in a distributed manner.

### 5.5.1 Reinforcement Learning Based Formulation

The resource allocation problem can be modeled as a Markov game (stochastic game), where each D2D pair is an individual agent. At the current state $s_t$, each agent $j$ can obtain its own local observations $o_t^j$. The local observations $o_t^j$ are the part of the global state $s_t$. Next, each agent $j$ will take the action $a_t^j$ and receive the reward $r_t^j$. All of the individual

actions can form a joint action $a_t$. Then the environment moves to the next state $s_{t+1}$ with a probability. The key components of the RL problem are described below.

**State Space and Local Observation:**

The state, $s_t \in \mathcal{S}$, can be defined as the set of all the environment components, e.g., channel conditions and the agent's behavior. However, each agent can only observe a part of the state, which can be defined as the local observation of the agent and is aligned with the realistic communication environments. For the D2D agent, the local observation of D2D agent $j$ contains the channel information such as channel gain for the $j$-th D2D link under channel $i$, $g_j^d[i]$, for all $i \in \mathcal{M}$; the interference channel from other D2D links, $\bar{g}_{j',j}^d[i]$, for all $j' \neq j$ and $i \in \mathcal{M}$; the interference channel from the cellular users, $\bar{g}_{i,j}[i]$, for all $i \in \mathcal{M}$. Here, we assume that the receiver of the D2D pair can accurately estimate the channel information and feed it back to the transmitter. In practice, exploiting vector quantization or codebook-based techniques in the feedback process makes it possible to reduce the feedback overhead and improve the reconstruction quality simultaneously. As a result, the feedback delay can be much smaller than the time slot duration, and the transmitter can be assumed to obtain the channel information instantaneously and accurately. Further, the local observation of the D2D agents also contains the interference power, $I_{ij}^d$, for all $i \in \mathcal{M}$, defined in (66). Moreover, because the BS (leader) will broadcast the unit reuse price firstly, we take $\alpha_{ij}$ for all $i \in \mathcal{M}$ also into the local observation of the D2D agent $j$. Therefore, the local observation of D2D agent $j$ at coherence time step $t$ can be described as

$$o^j(t) = \{g_j^d[i](t), \bar{g}_{j',j}^d[i](t), \bar{g}_{i,j}[i](t), I_{ij}^d(t-1), \alpha_{ij}(t)\}, \forall\ i \in \mathcal{M}. \tag{77}$$

The global state describes the current communication environment, which contains all the local observations from the D2D agents and some environment information perceived by the BS. The perceived information of the BS contains the channel gain between the cellular user $i$ and the BS, $g_i^c$, for all $i \in \mathcal{M}$; the interference channel from other D2D links, $\bar{g}_{j,B}[i]$, for all $i \in \mathcal{M}$ and $j \in \mathcal{N}$; the interference power, $I_i^c$, for all $i \in \mathcal{M}$ and $j \in \mathcal{N}$. These elements represent the dynamics of the cellular users. Therefore, the global state at

coherence time step $t$ can be described as

$$s(t) = \{g_j^d[i](t), \bar{g}_{j',j}^d[i](t), \bar{g}_{i,j}[i], g_i^c(t), \bar{g}_{j,B}[i](t), I_i^c(t-1), I_{ij}^d(t-1), \alpha_{ij}(t)\}, \forall\ i \in \mathcal{M},\ j \in \mathcal{N}. \tag{78}$$

Such the local observations and states can capture the real-time communication environments. At different coherence time periods, the actions will be determined to adapt to dynamic communication environments.

**Action Space:**

The resource allocation problem for D2D agents can be divided into power control and channel allocation issues. Therefore, the action for D2D agent $j$ at time step $t$ can be defined as $a^j(t) = \{p_j^d[i](t)\}$, which consists of two parts: how much transmission power in $\{p_{min}, p_{max}\}$ to choose and which channel $i$ to reuse. Specifically, we represent the selected channel in a binary-encoded manner. So the dimension of the D2D agent's action space is $1 + \lceil \log_2(M) \rceil$. We consider the joint action of all agents as $\mathbf{a(t)} = \{a^1(t), ..., a^N(t)\} \in \mathcal{A}$.

**Reward Function:**

For the proposed resource allocation problem, we aim to maximize the utility functions defined in (70). Therefore, the instant reward function for D2D agent $j$ can be defined as

$$r^j(t) = C_j^d[i](a^j(t)) - \beta^f \alpha_{ij}(t) a^j(t) \overline{g}_{j,B}[i](t). \tag{79}$$

### 5.5.2 Reinforcement Learning with Stackelberg Equilibrium

In the MARL problem, the agents select their actions according to their policies. The policy for agent $j$ can be defined as how to choose the action at each state, i.e., $\pi^j : \Upsilon(\mathcal{A}^j) \leftarrow \mathcal{S}$, where $\Upsilon$ is a probability distribution under the action space $\mathcal{A}^j$. Moreover, the joint policy $\boldsymbol{\pi}$ can be described as $\boldsymbol{\pi} = \{\pi^1, ..., \pi^N\}$ in our resource allocation problem. The goal for each agent $j$ is to maximize the expected long term accumulated reward, which is called

the value function. The value function for agent $j$ can be formulated as

$$V_{\boldsymbol{\pi}}^{j}(s) = V^{j}(s, \boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^{t} \mathbb{E}[r^{j}(t) | \boldsymbol{\pi}, s_0 = s], \tag{80}$$

where $s_0$ is the initial state. $\gamma$ is the discount factor, which can balance the instant reward and the future rewards.

For the value based RL problem, the Q-value (action-value function) can be defined following the Bellman equation. The Q-value represents the value from state $s$ and action $a$ over policy $\pi$, which can be described for agent $j$ as

$$Q_{\boldsymbol{\pi}}^{j}(s, \mathbf{a}) = r^{j}(s, \mathbf{a}) + \gamma \mathbb{E}_{s'}[V_{\boldsymbol{\pi}}^{j}(s')], \tag{81}$$

where $s'$ is the state of the next time step. The objective is to find the optimal policy $\boldsymbol{\pi}_*$ to maximize the value function.

Considering the game theory is a natural way when figuring out the MARL problem. In this situation, the Nash equilibrium in game theory can be seen as the optimal policy in the MARL problem. The Nash equilibrium can represent each agent's best response to others. We use the $\boldsymbol{\pi}_* = \{\pi_*^1, ..., \pi_*^N\}$ to describe the Nash equilibrium in our resource allocation problem over all agents. For each agent $j$ and for $s \in \mathcal{S}$, it satisfies

$$V^{j}(s, \pi_*^{j}, \boldsymbol{\pi}_*^{-j}) \geqslant V^{j}(s, \pi^{j}, \boldsymbol{\pi}_*^{-j}), \tag{82}$$

where $\boldsymbol{\pi}_*^{-j}$ donates the optimal policy for all the agents except agent $j$.

To adapt the Nash equilibrium to the MARL, Hu et al.[64] proposed a Multi-agent Nash Q-learning algorithm. They define a two-step iterative training procedure where they first obtain the Nash equilibrium from the current stage game for all the agents, and then update the Q-value according to that Nash equilibrium. Moreover, by implementing the

Nash equilibrium, the Nash Q-value function for each agent $j$ can be defined as

$$Nash\ Q_*^j(s, \mathbf{a}) = r^j(s, \mathbf{a}) + \gamma \mathbb{E}_{s'}[Nash\ V_{\boldsymbol{\pi}_*}^j(s')]. \tag{83}$$

In the above equation, the optimal policy $\boldsymbol{\pi}_*$ can be obtained from the solution of the game theory, and the Nash $V$ is the value function calculated based on the optimal policy. Through many iterations, the Q-value will eventually converge to the optimal Q-value, which means the value obtained according to the Nash equilibrium in the game.

In our resource allocation problem, we formulate the problem by the Stackelberg game and obtain the Stackelberg equilibrium. Similar to the Nash Q-learning, we design a two-step learning procedure for our problem. We first obtain the Stackelberg equilibrium from the SG solution of each stage game. Then, we apply that equilibrium to update the components defined in the RL problem, i.e., the value function and policy. Moreover, the order of the Stackelberg itself is considered in the learning procedure design, where the leader takes action first, followed by followers. Similar to (83), we define the Stackelberg value function for agent $j$ as $ST\ V_*^j(s, \mathbf{a})$, which is the value obtained under the Stackelberg equilibrium. Therefore, the Stackelberg Q-value (ST-Q) can be represented as

$$ST\ Q_*^j(s, \mathbf{a}) = r^j(s, \mathbf{a}) + \gamma \mathbb{E}_{s'}[ST\ V_*^j(s')]. \tag{84}$$

During the learning phase, when considering the Stackelberg value function defined above, the Q-value of each agent $j$ can be updated in the following manner

$$Q^j(s, \mathbf{a})^{t+1} = (1 - \epsilon) Q^j(s, \mathbf{a})^t + \epsilon[r^{jt} + \gamma\ ST\ V_*^j(s')], \tag{85}$$

where $\epsilon$ is the learning rate and

$$ST\ V_*^j(s') = \sum_{a^j} \pi_*^j(a^j | s', \mathbf{a}^{-j}) \mathbb{E}_{\mathbf{a}^{-j} \sim \boldsymbol{\pi}_*^{-j}}[Q_*^j(s', \mathbf{a})]. \tag{86}$$

The optimal policy $\boldsymbol{\pi}_*$ can be derived from the Stackelberg equilibrium in each stage

game. Different from the general MARL, we replace the target Q-value in general MARL with the ST-Q obtained from the Stackelberg game. The ST-Q can guide the updating procedure of the Q-value and the learning process of the agent. Moreover, the Q-value will converge to the Stackelberg equilibrium ST-Q with some assumptions[76], and the detailed proof can be found in Appendix C.

Our goal is to design a distributed framework and let each D2D agent learn the policy by itself. In this manner, agents cannot access the global state when taking actions in the execution stage. Therefore, the policy functions are determined based on the local observations of the agents. Hence, the policy function of the follower $j$ can be defined as $\pi^j(o^j)$. The leader (BS) determines the unit reuse price according to (75) firstly, then each D2D agent will take the action with the policy function $\pi^j$ by itself.

In the learning process, we need to solve the Stackelberg game and the RL problem alternately. With the guidance of the proposed ST-Q, the learning process is more stable and converge faster than the general one. In our case, the additional information is provided due to the Stackelberg equilibrium. Thus, the target Q-value can be calculated based on the Stackelberg game solution, not based on the existing action patterns in general MARL. Based on this analysis, our proposed framework will converge better and faster than the general MARL framework, which means the number of required iterations will be reduced.

### 5.5.3 Fast D2D Resource Allocation Decision via STDRL

In our proposed D2D resource allocation model, the state space is high dimensional and the action space is continuous. Accordingly, we introduce the function approximators for the Q-function and the policy function, where both the Q-value defined in (84) and the policy $\pi$ can be modeled as the neural networks. We implement our proposed system based on a state-of-art multi-agent deep reinforcement learning framework, multi-agent deep deterministic policy gradient (MADDPG) [56]. Note that MADDPG needs the global information in the training stage and can be distributed executed by each agent, where each D2D agent has one actor and one critic network. The framework is shown in Fig. 14, and

the detailed descriptions are shown as follows.

Each agent maintains a Q-value to evaluate the policy performance, which can be modeled as the **critic network** with parameter $\omega$. For each agent, the critic network is established based on the global state and all agents' actions. Therefore, all agents' critic networks are trained on the BS. We introduce the critic network with the parameters $\omega$ into the updating function as described in (85). Similar to the DQN algorithm proposed by DeepMind [77], each agent $j$ updates the Q-value by minimizing the following loss function

$$\mathbb{L}(\omega^j) = (y^j - Q_{\omega^j}(s, \mathbf{a}))^2, \tag{87}$$

where $y^j = r^j(s, \mathbf{a}) + \gamma \; ST \; V_{*\bar{\omega}^j}(s')$ is the target Stackelberg value modeled with the parameter $\bar{\omega}^j$. The target value can be calculated according to the (86), which can be derived from the Stackelberg game, and then the stochastic gradient descent method is applied to minimize the loss function of the critic network.

For the policy part, we establish the **policy network** $\Omega$ with the parameter $\theta$ for each agent. Each agent can generate a policy based on its policy network. In this case, for D2D agents, they will locally train the policy network on its own and develop the policy based on its own observation. Additionally, we transform the continuous outputs related to the selected channel into discrete binary outputs. Therefore, the policy network for D2D agent $j$ can be defined as $\Omega_{\theta^j}(o^j)$. It should be noticed that the leader will take action first in our Stackelberg game, followed by followers. Consequently, the followers' (D2D pairs) observations contain the action of the leader (unit reuse price). The goal of the policy network for each agent is to maximize the objective function $J$. Similar to the deep deterministic policy gradient (DDPG) algorithm [78], we apply the deterministic policy and train the network by the sampled policy gradient method. The gradient can be calculated as

$$\nabla_{\theta^j} J(\theta^j) = \nabla_{a^j} Q_{\omega^j}(s, \mathbf{a})|_{a^j = \Omega_{\theta^j}} \nabla_{\theta^j} \Omega_{\theta^j}(o^j). \tag{88}$$

**Algorithm 6** Stackelberg Game Guided MADRL (STDRL) for D2D Resource Allocation

**Input:** For all D2D agents $j$, $j \in \{1, ..., N\}$, randomly initialize the critic network $Q_\omega$ and actor network $\Omega_\theta$ with parameter $\omega$ and $\theta$, respectively. Initialize the replay memory $\mathbb{M}$ and set the target network by copying the parameters from $Q_\omega$ as $Q_{\bar{\omega}}$.

**Output:** the selected action for all of the agents.

1: **for** episode $= 1, ..., K$ **do**
2: Initialize the noise process $\mathcal{N}$ for action exploration.
3: Initial state $s_0$ for the communication environments.
4: **for** $t = 1, ..., T$ **do**
5: The leader BS broadcasts the unit reuse price calculated with (75).
6: Each D2D agent $j$ takes action $a^j(t) = \Omega_{\theta^j}(o^j(t)) + \mathcal{N}_t$ based on the policy network and the exploration noise locally.
7: Each D2D agent gets the immediate reward $r^j(s, \mathbf{a})$, and the next state $s(t+1)$ can be observed.
8: Each D2D agent uploads its own information to the BS, e.g., $r^j(s, \mathbf{a})$ and $a^j(t)$. BS stores transition $(s(t), \mathbf{a}(t), \mathbf{r}(t), s(t+1))$ in memory.
9: **for** Each agent $j$, $j \in \{1, ..., N\}$ **do**
10: Randomly sample mini-batch size $M$ of transitions $(s(t), \mathbf{a}(t), \mathbf{r}(t), s(t+1))$ in memory.
11: Set target $Q$ value as $y^j = r^j(s, \mathbf{a}) + \gamma\ ST\, V_{*\bar{\omega}^j}(s')$ by (86), where the optimal policy can be calculated as the Stackelberg Equilibrium from the Algorithm 5.
12: The Critic Network can be trained by minimizing the Loss function $\mathbb{L}(\omega^j) = (y^j - Q_{\omega^j}(s, \mathbf{a}))^2$ according to the Adam Algorithm in BS.
13: D2D agents download $Q$ functions. The Actor Network can be trained by sampled policy gradient, as $\nabla_{\theta^j} J(\theta^j) = \frac{1}{M} \sum \nabla_{a^j} Q_{\omega^j}(s, \mathbf{a})|_{a^j = \Omega_{\theta^j}} \nabla_{\theta^j} \Omega_{\theta^j}(o^j)$.
14: Update the parameter $\bar{\omega}$ in target networks with "soft" way: $\bar{\omega} \leftarrow \tau\omega + (1-\tau)\bar{\omega}$

Our proposed Stackelberg game guided multi-agent deep reinforcement learning (STDRL) approach for D2D resource allocation is summarized in Algorithm 6. We have two stages in STDRL, i.e., the training and execution stages. For the training part, at the initial state, the BS and D2D pairs take the actions in a sequential according to the policy networks and the noise process $\mathcal{N}$ to balance the exploration and exploitation [78] in step 5 and 6. After executing the actions, each agent can observe the reward, and all D2D pairs upload the local information, e.g., reward and action, to the BS. Next, all critic networks are trained on the BS based on the observed global state and joint actions in steps 11 and 12. The target value is defined based on the Stackelberg equilibrium, which can be obtained from Algorithm. 5. Then all critic networks can be trained on each agent locally. Moreover, D2D pairs need to download the critic networks from the BS for updating the policy networks in

step 13. Finally, we update the parameters of the target network in a "soft" way [78]. $\tau$ is the soft parameter, which can make the target value change slowly. Since the game theory module we added has the closed-form solutions, our proposed STDRL does not introduce additional computational complexity compared with the general MADRL algorithm [56]. When the training converges, we will get a general model which can be deployed in dynamic communication environments. For the execution stage, we only need to employ actor networks to make the resource allocation decisions in a distributed manner as described in steps 5 and 6. In this situation, the execution stage has a linear time complexity of $O(n)$, where $n$ is the length of the decision steps.

Moreover, since we introduce the game module to compute the equilibrium, we push more computing workload into each training iteration and reduce the workload incurred by wireless transmission/information exchange among the agents. In this way, we dramatically reduce the required number of iterations to converge and thereby improve the time efficiency of the training process. We compare the training time consumption of our proposed STDRL with the ordinary MADRL approach, including the computation time and communication delay. In each iteration, interactions among the agents are needed in both approaches for sharing some RL components, i.e., actions. Accordingly, the communication time linearly increases with number of training iterations. For the computing time, the additional computing loads are introduced in our proposed STDRL approach due to computing the Stackelberg equilibrium. Therefore, the computation time of our proposed approach is slightly larger than that of the general MADRL one in each iteration, which can be seen as a sacrifice. On the plus side, our approach requires many fewer iterations to converge than the ordinary MADRL, which results in a remarkable reduction of total training time consisting of communication time and computing time. Accordingly, the energy consumption will also be reduced when the mobile devices have a fixed running power.

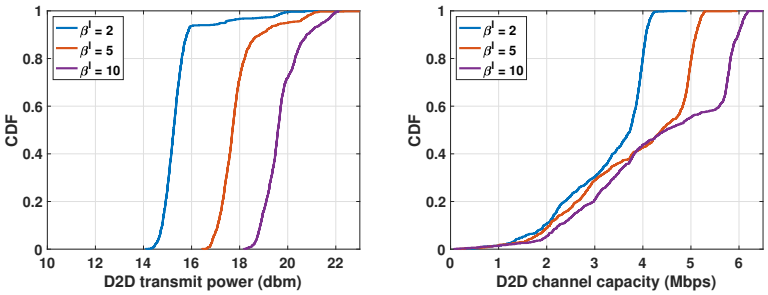
Table 2: STDRL Simulation Parameters.

| **Parameter** | **Value** |
|---|---|
| Carrier frequency | 2 GHz |
| Bandwidth | 1.4 MHz |
| Edge device antenna gain | 3 dBi |
| Cellular UE Tx power | 23 dBm |
| D2D UE Tx power | 5-23 dBm |
| Cell radius | 500 m |
| Max D2D communication distance | 50 m |
| Path loss model of cellular links | $128.1 + 37.6 \log_{10} d$ |
| Path loss model of D2D links | $148 + 40 \log_{10} d$ |

### 5.6 Performance Evaluation

In this section, abundant numerical simulations are executed to evaluate the performance of the proposed STDRL approach for the D2D resource allocation issue. We consider a cellular-based communication scenario with 4 cellular users and 4 D2D communication pairs. The main simulation parameters of the communication system are summarized in Table 2. Specifically, we consider different path loss models for D2D links and cellular links. Moreover, to describe the dynamic communication environments, we set that the small-scale fading (fast fading) is updating every coherence time, and the large-scale fading (pass loss and shadowing) is changing every 100 coherence times. The step size of our proposed approach is set as the same as the fast-fading updating unit. For the learning part, the actor and critic neural networks for each agent consist of 3 fully connected hidden layers, with 512, 256, and 128 nodes, respectively. The batch size is selected as 32, and the learning rate is 0.0001. We set the experience replay memory size as 400 or 200 for different simulations. Moreover, the discount factor $\gamma$ is 0.99 and "soft" parameter $\tau$ is 0.01 in the RL part. Particularly, all of the parameters of the learning approach are derived from the parameter tuning.

We first evaluate the effectiveness of our proposed game theoretical model and identify the scalar coefficients $\beta$. According to the scale of simulation parameters of communication environments, the scalar coefficients $\beta^f$ is chosen as 40. Note that if $\beta^f$ is smaller, the optimal D2D pair transmit power will be larger with the fixed $\beta^l$. In Fig. 15, we plot

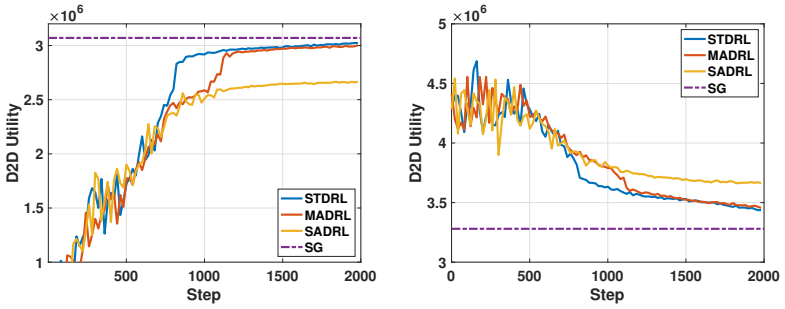



(a) D2D pairs' power distribution (b) D2D pairs' channel capacity distribution

Figure 15: Performance comparison under different $\beta^l$.

cumulative distribution function (CDF) of average D2D pairs' transmit power and channel capacities under different scalar coefficient $\beta^l$ with $\beta^f = 40$, respectively. The performance is displayed over 10000 times Monte Carlo simulation experiments. For a larger $\beta^l$, the unit reused price for the follower is relatively lower. Therefore, the follower will select relatively larger transmit power, as shown in Fig. 15a. In this way, the channel capacity of the D2D pair will be increased as well, which is described in Fig. 15b. Note that when the D2D user's location is very close to the cellular user's location, the D2D pair will be significantly interfered with, causing low channel capacity. In these situations, the D2D pair's channel capacity is mainly affected by the nonadjustable environmental settings (e.g., location) rather than the adjustable optimization variable (e.g., transmit power, unit reuse price). Therefore, the value of the $\beta$ will have a slight impact on the D2D user's channel capacity in these situations, which is why the two lines ($\beta^l = 5$ and $\beta^l = 10$) are basically coincident before the point "4Mbps" In additional, the larger transmit power of the D2D pair causes the increased interference of cellular users, causing a decrease in the channel capacity of cellular users.

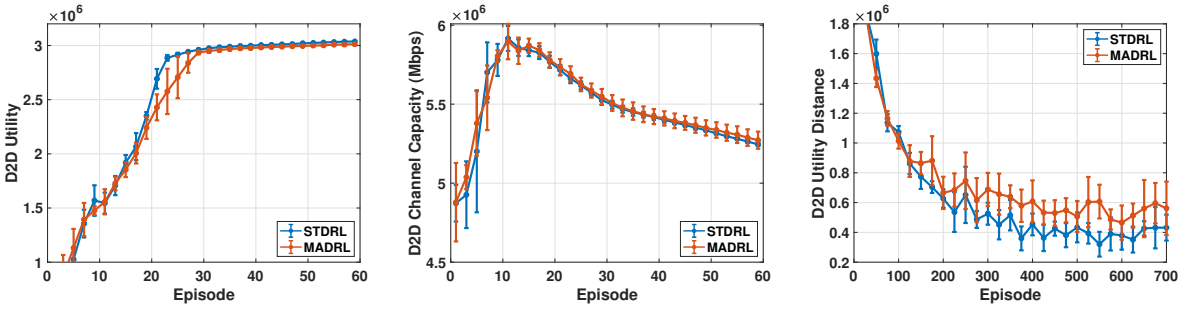
To show the performance of our proposed STDRL approach, we consider three baselines of our proposed approach, termed the classic Stackelberg game (SG) approach [79], the single-agent deep reinforcement learning (SADRL) approach [80], and the general multi-agent deep reinforcement learning (MADRL) approach [81]. The SG approach can be regarded as a near-optimal solution. The SADRL approach allows each agent (D2D) pair to




(a) The average D2D pairs' utility (b) The base station's utility

Figure 16: Performance comparison in relatively static environments.

make the resource allocation decisions dependently. For the MADRL approach, we adopt the state-of-art MADRL framework, the multi-agent deep deterministic policy gradient (MADDPG) approach. For convenience, we take $\beta^l = 5$ and $\beta^f = 40$ in our experiments. First of all, we consider relatively static communication environments, where we fix the large-scale fading and only take the fast fading updating into account. We train our proposed STDRL approach and MADRL approach until they converge, and the results are shown in Fig. 16. Because the SG approach needs to solve the problem for every step, we take the average of the SG results as the SG baseline. Fig. 16a. depicts the average D2D agents' utility (reward) during training. On the one hand, the SADRL approach underperforms the multi-agent DRL based approaches since there is no obvious coordination among the agents. Moreover, our proposed STDRL achieves a little higher utility than the MADDPG approach, which is similar to the SG approach. On the other hand, with the help of the equilibrium, our STDRL approach converges faster than the general MADRL approach. Specifically, the STDRL approach needs about 900 steps (iterations) instead of 1200 steps in the MADRL approach, thanks to the equilibrium providing additional information in the training process. The corresponding BS's utility is shown in Fig. 16b, which has a similar behavior to that of the D2D's utility. In particular, the fluctuation of the training curve at the initial stage is due to the exploration of the RL approach. Since the SADRL approach does not need communication among agents and performs worse than other approaches, we ignore it in the following parts. Furthermore, we record the time consumption of the proposed framework. For each step, the communication delay and the computation time of




(a) The average D2D pairs' utility (b) The average D2D pairs' channel capacity (c) The average D2D utility distance between SG scheme and learning scheme

Figure 17: Performance comparison with dynamic communication environments.

the learning part are both close to 10 ms, and the computation time of an additional SG part is about $1-2$ ms. Therefore, our proposed STDRL approach can save approximately 17% time than the general MADRL one.

Next, we deploy our proposed approach in the dynamic communication environments, which means we consider both the large-scale fading and small-scale fading updating. We run each simulation (train the model until it converges) 10 times and display the mean value and the standard deviation results (error bars) in the corresponding figures. Each training episode contains 20 training steps. Note that we fix the large-scale fading for each training episode and let the small-scale fading change over each step. We firstly consider that the location of users is relatively fixed. The average D2D pairs' utility is shown in Fig. 17a. According to Fig. 17a, we can find that our proposed STDRL can still achieve a little higher utility for D2D agents than the general MADRL approach with fewer episodes in dynamic communication environments. In additional, in the initial training stages, the agent may take bad actions due to the exploration noise and limited learned information. To better understand the performance differences between our proposed approach and the MADRL approach, we present the average D2D pairs' channel capacity in Fig. 17b. The average D2D pairs' channel capacity of the STDRL approach is slightly or neglectable higher than that of the MADRL approach, but the average D2D pairs' utility of the STDRL approach is higher than that of the MADRL approach. This indicates that our proposed STDRL approach can achieve better interference management than the MADRL approach, which

can also be validated with the BS's utility.

Our goal is to make fast resource allocation decisions in dynamic communication environments. We further consider the users' mobility in our proposed STDRL approach training. We still change the large-scale fading every episode and the small-scale fading every step. Similar to the setting in the above simulations, we take 40 iterations in each episode. Due to the dynamic nature of the communication environments, it is not suitable to consider the average instant reward in each episode as an indicator of training performance. So we calculate the utility difference between the learning approach and the SG approach and use it as a measure of the learning performance. As shown in Fig. 17c, we calculate the average D2D pairs' utility distance between the SG scheme and the learning scheme for each episode in the training process. Fig. 17c shows that our proposed STDRL approach needs nearly 370 episodes to converge, which is less than the 500 episodes for the MADRL approach. Moreover, compared to the general MADRL approach, STDRL can converge to the point closer to the near-optimal SG solution. In particular of the dynamic environment, the performance improvement of STDRL is obvious, since the equilibrium introduced provides explicit learning directions.

We employ actor networks to make resource allocation decisions in a distributed implementation and summarize the performance comparison results in Table 3, where the "random" approach means we randomly select the resource allocation strategies. We compare the average D2D pairs' utility, channel capacity, BS's utility, and the time consumption for the training, including time consumption of communication and computation, for these four approaches. According to Table 3, we can say that our proposed STDRL can achieve higher utility and larger channel capacity with less time consuming than the general MADRL approach. Furthermore, we can save about 20% training time compared with the general MADRL approach. This means the time-efficient resource allocation strategies can be made based on our STDRL approach. Further, the result of our STDRL approach is close to that of the SG approach, while our approach can tackle the resource allocation issue

Table 3: STDRL Performance Comparison.

| | Random | MADRL | STDRL | SG |
|---|---|---|---|---|
| Avg. D2D Utility ($\times 10^6$) | 1.24 | 2.49 | 2.78 | 3.12 |
| Avg. D2D Ch. Cap. (Mbps) | 2.01 | 3.58 | 4.11 | 4.94 |
| BS Utility ($\times 10^6$) | 2.72 | 2.92 | 2.96 | 3.26 |
| Comm. Time Cons. (s) | – | 200 | 148 | – |
| Comp. Time Cons. (s) | – | 200 | 170 | – |
| Total Time Cons. (s) | – | 400 | 318 | – |

in dynamic communication environments. Specifically, when the communication environment changes, we can provide convincing resource allocation strategies in a few inference iterations, where the SG approach needs to solve this problem again.

## 5.7 Conclusion

To improve the throughput of D2D-enabled networks, we have developed a Stackelberg game guided multi-agent deep reinforcement learning (STDRL) scheme in a distributed way to make the smart D2D resource allocation decisions in dynamic communication environments. With the guidance of our proposed ST-Q calculated by the Stackelberg equilibrium derived from the game approach, the learning process of STDRL is more stable and converges faster than that of the general MADRL approach. Furthermore, we can guarantee that there exists a unique equilibrium in each stage game, and the Q-value in the learning process will converge to the equilibrium in the end. Multiple simulations have shown that our STDRL approach can provide satisfactory resource allocation strategies to enhance system utility. Moreover, our proposed STDRL approach can be achieved in a time-efficient way, where we can save about 20% learning time compared with the general MADRL approach. Appendix proof is available for this work at https://github.com/shidian117/STDRL/blob/master/Appendix.pdf

# 6 Deep Q-Network Based Route Scheduling for Transportation Network Company Vehicles

## 6.1 Introduction

The emergence of Transportation Network Company (TNC) (e.g., Uber, Lyft, etc.) services that offer transport-as-a-service has solved the dilemma of the "first-mile/last-mile" problem of public transit users and traffic congestion problem in urban areas. The "first-mile/last-mile" problem is severe especially for public transit users since their starting locations/final destinations are relatively far away from existing public transportation options. The TNC mobility service providers have evolved as the best choice for addressing such concerns since TNCs use software to effectively pair passengers with vehicles (drivers) by collecting data from passengers and vehicles for route scheduling and optimization.

Indeed, TNCs have been playing a vital role in providing mobility services to users. The major goal for TNC is to effectively schedule cruising (vacant) vehicle routing to quickly match passengers with vehicles. In urban areas, vacant TNC vehicles usually roam around here and there so as to pick up passengers timely called "TNC cruising vehicles". In practice, there exists a certain delay while collecting passengers' data, since TNCs can only obtain the data when a passenger requests the service. Therefore, the cruising vehicles always only seek passengers based on drivers' own experience and may take longer time to meet the passengers especially if they are far away from the passengers. Besides data collection, rush hours or extreme weather conditions also account for long waiting times. Since TNC vehicles cruising significantly affects the TNC drivers' long-term revenue, it is critical to develop efficient vehicle scheduling strategies to improve services to users and benefits of drivers. To solve the TNC cruising vehicle scheduling problem, we propose a scheduling method based on deep Q-network, a deep reinforcement learning algorithm.

Deep Q-Network (DQN) was proposed by Deepmind in 2013 [82], which is a Deep Reinforcement Learning (DRL) method combining Reinforcement Learning with Deep Neural

Networks [83]. It has recently found several applications as it outperforms the reinforcement learning methods, and some of the existing works in the literature solve the vehicle route scheduling problems. Qu et al. [84] developed a cost-effective recommender system for taxi drivers to maximize their profits when finding the passengers. Powell et al. [85] proposed a grid-based method to reduce the number of cruising miles while simultaneously increasing the number of live miles. These methods rely on historical data for scheduling, which cannot be adapted to dynamic urban environments. Du et al. [86] adopted lstm to predict the traffic flows based on the dataset of traffic history, and the predicted traffic benefits the drivers to determine the optimal route with efficient travel time. Beyond that, some researchers also study reinforcement learning based vehicle route scheduling. Verma in [87] used Monte Carlo method to study optimal route planning of cruising taxis, and Han in [88] demonstrate that reinforcement learning algorithm of the Q-learning family can learn optimal actions for routing autonomous taxis in a real city scenario. Even though these reinforcement learning methods consider dynamic urban environments, they use only a limited number of states such as only the location of vehicles and do not consider the competition between vehicles.

To address the above issues, in this work [89, 90], we propose "big" TNC service data based scheduling to intelligently schedule the routes of TNC cruising vehicles, and develop an effective DRL-based online dynamic framework for "last-mile/first-mile" services. The proposed approach has the potential to reduce the waiting time of passengers, searching time of drivers, and maximize every individual TNC drivers' long-term revenue. Our salient contributions are summarized as follows:

- We firstly employ the deep Q-network to learn the urban environment, and guide TNC cruising vehicles to pick up passengers while maximizing the long-term revenue of an individual TNC driver.

- Different from the existing scheduling works, we consider high-dimensional components of the urban environment during scheduling that include different spaces, times

and especially the competition between TNC cruising vehicles which significantly affects TNC drivers' searching time and revenue.

- We illustrate our proposed dynamic model based on both big real-time data and historical data that can schedule TNC cruising vehicles' routes during the model training, and that can be applicable to an actual dynamic urban environment.

## 6.2 TNC Cruising Vehicle Scheduling Model with Reinforcement Learning

In this section, we first introduce the TNC cruising vehicle scheduling model and the reinforcement learning. We then elucidate the reinforcement learning based TNC cruising vehicles scheduling model.

### 6.2.1 Model Configuration

The goals of the TNC cruising vehicle route scheduling problem are to minimize waiting and searching time of passengers and drivers, and maximize the long-term revenue of the drivers. We consider an urban area divided into multiple regions. We assume the shape of the city to be a rectangular grid that can be divided into $M \times N$ cells uniformly, where each cell represents an area in the city, as shown in Fig. 18. The TNC cruising vehicle scheduling problem is described as follows.

A TNC vehicle in a cell is cruising and searching for passengers, without any prior passenger data since a passenger's data is available only after the passenger requests the service. Therefore, the TNC cruising vehicle interacts with TNC service center and informs its status. The TNC service center guides this TNC vehicle to a neighboring cell. Then the TNC vehicle searches the new cell for passengers and constantly reports its status to the TNC service center as either vacant or occupied. If its status is occupied, the guiding process of TNC service center ends and the next request to a new guiding process resumes after the TNC vehicle drops the passengers at the destination cell and reports the revenue. If the status is vacant, the requests and responses between the TNC cruising vehicle and

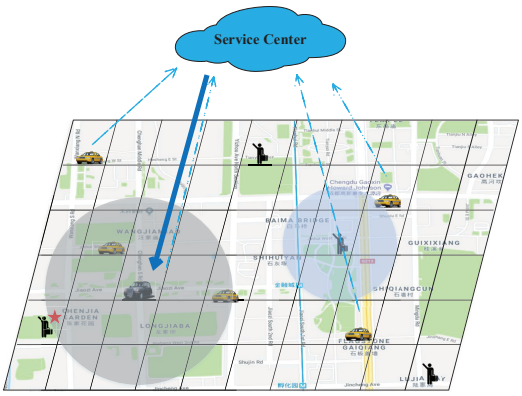



Figure 18: TNC cruising vehicle scheduling model

the service center continue until the vehicle finds passengers. Note that the competition among TNC vehicles also needs to be considered while processing these requests.

### 6.2.2 Model Description

The TNC cruising vehicle scheduling model configuration can be formulated as the Markov decision process, but the concern is that transition probability is unknown in such case. The idea of model-free reinforcement learning is more suitable for such situation. For the model-free reinforcement learning method, the objective of the agent (TNC service center) is to minimize the searching time of the TNC cruising vehicle (driver) and maximize the long-term revenue of the individual driver. With the Markov decision process, the major components of the model-free reinforcement learning method for the TNC cruising vehicle scheduling model are described as follows:

**State $S$:** TNC cruising vehicle's situation at a certain moment

Our dynamic model can be applied to timely environments, rather than purely being dependent on historical data. It considers the scheduling of TNC cruising vehicles under different times and spaces. On one hand, different situations during weekdays and weekends and different time periods of the day that have significant impact need to be considered in the model. The above is the urban environment that we need to consider before scheduling. At the beginning of each step, the agent observes the urban environment and collects some related parameters as the input state. On the other hand, each TNC cruising vehicle is not

a separate entity and the distribution of other TNC vehicles in the surroundings can also significantly impact the guiding policy, since they compete with each other. Taking all the above factors into consideration, the parameters of input states are defined as follows:

- $X_m$: The horizontal position of the TNC vehicle, $m \in (1, 2, ...M)$.
- $Y_n$: The vertical coordinate of the TNC vehicle, $n \in (1, 2, ...N)$.
- $T_k$: Day of the week, classified as weekday and weekend, $k \in (1, 2)$.
- $P_l$: Different time periods of one day, such as morning peak, evening peak, etc., $l \in (1, 2, ..., L)$
- $C = \{C_s, C_n, C_e, C_w\}$: The number of other TNC vehicles that compete with the vehicle being guided from all the four directions. Component $C$ can be divided into eight parts, and it contains the number of competitors close to the TNC cruising vehicle around two layers in four directions.

Therefore the input state $S(k) = \{X_m, Y_n, T_k, P_l, C\}$ at each time step $k$ is represented the set of all the above parameters.

**Action** $A$**:** For each state, the agent will guide the TNC cruising vehicle to take one of the actions below to reach to the next state

TNC service center has four alternative actions $A = \{a_s, a_n, a_e, a_w\}$ at each state to guide the TNC cruising vehicle, which means that the service center can guide the TNC cruising vehicle to one of the four adjacent cells in the south, north, east and west.

**Reward** $R$**:** The instant reward for TNC cruising vehicle at each time step after taking an action

Only if the TNC vehicle picks up a passenger, the instant reward is profit for that order. Otherwise the reward is zero. Because the competition from other nearby TNC vehicles is considered, there is a certain probability of failure to pick up passengers by guiding. When the vehicle being guided successfully picks up the passenger, we define the instant reward $r$ to be the profit $r_p$ for that order times the success rate $\rho$. So the instant reward $r(k)$ at

time step $k$ can be expressed as

$$r(k) = \begin{cases} \rho \cdot r_p & \text{when guided one picks passenger,} \\ 0 & \text{otherwise.} \end{cases} \tag{89}$$

After picking up the passenger $i$, the TNC driver makes a profit. The profit $r_p$ obtained by an individual TNC cruising vehicle driver for an order can be expressed as

$$r_p(i) = e(i) - c(i), \tag{90}$$

where $e(i)$ and $c(i)$ are earnings obtained and costs spent for picking up the $i$-th passenger. For effective scheduling, we define the earnings, success rate and the cruising vehicles in a cell based on the strategies considered in real-world TNCs similar to the one considered in this work.

Based on different ride distances, we define the earnings as

$$e(i) = \begin{cases} B & L(i) \leq D_1, \\ B + (L(i) - D_1) * u_1 & D_1 < L(i) \leq D_2, \\ B + (L(i) - D_1) * u_1 + (L(i) - D_2) * u_2 & D_2 < L(i), \end{cases} \tag{91}$$

where $B$ is the base price, $D_1$ and $D_2$ are the distance standards for dividing the earnings, $L(i)$ is the distance traveled by the TNC vehicle to drop the passenger $i$ at its final destination, and $u_1$ and $u_2$ are dynamic pricing factors that depend on driver's supply and the ride distance. We define the costs $c(i)$ for the ride based on the cost for the fuel and the cost to the company. Thus, $c(i)$ can be computed as

$$c(i) = L(i) \cdot f + p \cdot e(i), \tag{92}$$

where $f$ is the price of fuel required per unit distance and $p$ is the percentage of TNC company's interest for each order.

Another factor that needs to be considered while picking up the passengers is the competition among TNC vehicles. There is a possibility that other TNC vehicles might pick up the passengers first and the TNC cruising vehicle being guided by the TNC service center might fail. Specifically, to quantify this situation, the success rate $\rho$ in each state at each time step is introduced to measure the success rate of TNC cruising vehicle being guided by the service center in picking up the passenger successfully. The success rate $\rho$ in picking up a passenger is defined as

$$\rho = \frac{P_s}{C_0 \cdot p_k + \sum_{n=1}^{N} C_n}, \tag{93}$$

where $P_s$ is the number of passengers in that position at that time step, $p_k$ is the probability that the TNC vehicle searches for the passenger in the cell it is present, and $(1 - p_k)$ represents the probability that the TNC vehicle searches for the passenger in other cells. $C_0$ is the number of TNC cruising vehicles in that position at that time step, and $C_n$ represents the equivalent number of TNC cruising vehicles which have $n$ step distances from the passengers' location. For example, $C_n$ for 1 and 2 steps can be computed as

$$C_1 = H_1 \cdot (1 - p_k) \cdot \frac{1}{4} \text{ and } C_2 = (H_{21} + 2H_{22}) \cdot (1 - p_k)^2 \cdot (\frac{1}{4})^2, \tag{94}$$

where $H_1$ is the total number of TNC cruising vehicles at an unit distance from passengers. $H_{21}$ and $H_{22}$ are the number of TNC cruising vehicles that are two distance units away from passengers under two different situations. Based on this assumption, equivalent number of competing TNC cruising vehicles for one specific passenger can be obtained.

## 6.3 Problem Formulation and TNC Cruising Vehicle Scheduling

In this section, we formulate TNC cruising vehicle scheduling problem and introduce the algorithm of deep reinforcement learning to achieve the best policy for TNC cruising vehicle scheduling.

### 6.3.1 Deep Q-Networks

The main idea of deep Q-network is using function approximation techniques to approximate the state action value function. For this non-linear function, neural network is a great approximator.

Different from general Q learning, DQN uses a deep neural network to replace the traditional *Q-table*, so that the state action value can be represented as a function by using neural network, which means that the process of updating *Q-table* is the process of training deep neural network. In the process of updating the neural network, we need to consider the $Q$ value at this moment and the next moment. Hence, we use two separate networks to represent two $Q$ values respectively, and we define two types of neural networks of $Q$ value as

$$Q(s_k, a_k) = Q(s_k, a_k; w) \text{ and } Q(s'_k, a'_k) = \hat{Q}(s'_k, a'_k; w^-), \tag{95}$$

where $w$ is the parameter of evaluation $Q$ network, and $w^-$ is the parameter of the target $Q$ network. Due to the non-uniformity between high-dimensional state space and low-dimensional action space, both neural networks only use the state as input and the output is the $Q$ value of each possible action.

In order to get the optimal policy according to $Q$ network, this $Q$ network can be trained by using supervised learning. We treat the term including target $Q$ value $[r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; w^-)]$ as the label. The loss function of neural network is given as

$$L(w) = \mathbb{E}[r_k + \gamma \max_{a'} \hat{Q}(s_{k+1}, a'; w^-) - Q(s_k, a_k; w)]^2. \tag{96}$$

### 6.3.2 Overall Architecture

In the training process, we minimize the loss function by using gradient descent with respect to the parameter $w$ at iteration $i$. For using neural network as an approximator

**Algorithm 7** DRL for TNC Cruising Vehicle Scheduling Algorithm

**Input:** replay memory size $M$, discount factor $\gamma$, greedy $\epsilon$, exploration increment $\delta$, neural network learning rate $\alpha$ the observations of the urban environment.

**Output:** the selected action for TNC cruising vehicle.

Initialize replay memory $M$, evaluation Q-network with parameter $w$ and target Q-network with parameter $w^-$.
**for** episode $=1, ..., N$ **do**
    The TNC cruising vehicle start at the state $s_1$.
    **for** $k = 1, ..., K$ **do**
        TNC cruising vehicle with adaptive probability $\epsilon + \delta$ takes a random action $a_k$; otherwise selects action with greedy policy.
        The TNC cruising vehicle takes the action $a_k$, gets the instant reward $r_k$ and moves to the next state $s_{k+1}$.
        Store transition $(s_k, a_k, r_k, s_{k+1})$ in memory $M$ of the center.
        Randomly sample mini-batch size of transitions $(s_j, a_j, r_j, s_{j+1})$ in $M$.
        Set target $Q$ value as $r_j$, when $s_{j+1}$ is the final state; Otherwise, target $Q$ value is $r_j + \gamma \max_{a'} \hat{Q}\left(s_{i+1}, a^{'}; w^{-}\right)$.
        The TNC center uses GD to minimize the loss function and update $w$.
        Every $L$ steps use $w$ replace $w^-$ in $\hat{Q}$.

in reinforcement learning to replace the *Q-table*, this method always tends to diverge. In order to avoid divergence, two tricks are introduced, which are 1) *Experience replay and* 2) *Fixed Target Network*. Thus the derivative of the loss function can be expressed as

$$\nabla_w L(w) = \mathbb{E}_{s_j, a_j, r_j, s_{j+1} \in \mathbb{M}}[(r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; w^-) - Q(s_j, a_j; w)) \nabla_w Q(s_j, a_j; w)], \tag{97}$$

where $\mathbb{M}$ is the experience replay memory, which can store the transitions and let the agent randomly pick up the $j-th$ transition for the training process later. Experience replay memory can break the relationship between training data. At every fixed step, using the parameter of evaluation network $w$ to replace the parameter of target network $w^-$ is called the fixed target network.

The total process of scheduling TNC cruising vehicle to pick up passengers is shown in Algorithm 7. After collecting real time data, we use deep Q-network with experience replay and fixed Q-targets to learn and interact with the urban environment at a certain moment, and then schedule the individual TNC cruising vehicle to reduce searching time of

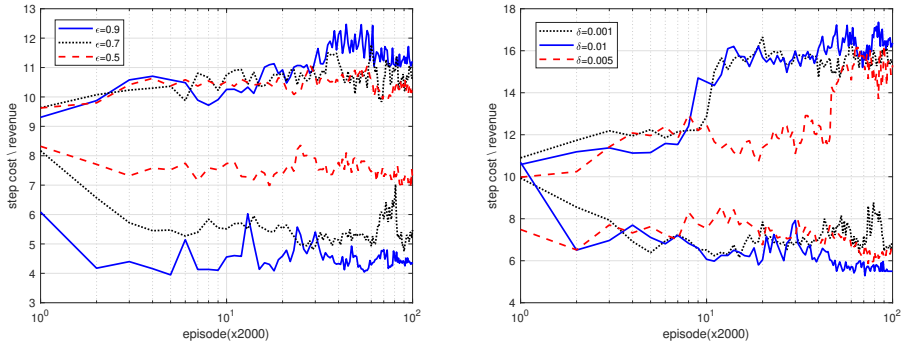
Table 4: Example of DiDi Database.

| **Lable** | **Example** |
|---|---|
| pick up_time | 1477964797 |
| drop off_time | 1477966507 |
| pick up_longitude | 104.09464 |
| drop off_longitude | 104.08927 |
| pick up_latitude | 30.703971 |
| drop off_latitude | 30.65085 |

TNC drivers. Specifically, as shown in Algorithm 7, in state $s_k$, according to the guidance from the TNC service center, the TNC cruising vehicle can take action $a_k$ into a new cell with $\epsilon$-greedy policy which selects a random action $a_k$ with adaptive probability $\epsilon = \epsilon + \delta$, where $\delta$ is the increment at each time step, or selects $a_k = \arg\max_a Q(s_k, a; w_i)$. The TNC cruising vehicle executes an action $a_k$, receives a reward $r_k$, and enters into the next state $s_{k+1}$. After executing action $a_k$, the TNC vehicle will report the transition $(s_k, a_k, r_k, s_{k+1})$ to the service center, and the TNC server center will store this transition in its memory. If the passenger is picked up by this guided TNC vehicle in this cell, after reporting the instant earnings, the episode will end and start a new one; otherwise, the center will continue to guide this TNC cruising vehicle for picking up passengers. Further, when the collected transition samples are abundant enough, the TNC center will randomly choose mini-batch size of transitions $(s_i, a_i, r_i, s_{i+1})$ to train the neural network. For the training process, the service center uses gradient descent to minimize the mean square errors (MSE) between Q-target and Q-evaluation, and resets $\hat{Q} = Q$ every $L$ steps. The training evolves with dynamic reports from the TNC cruising vehicles. When a TNC cruising vehicle sends a query with its current location information to the center for the next step, the center will put forward its guidance based on the DQN learning results.

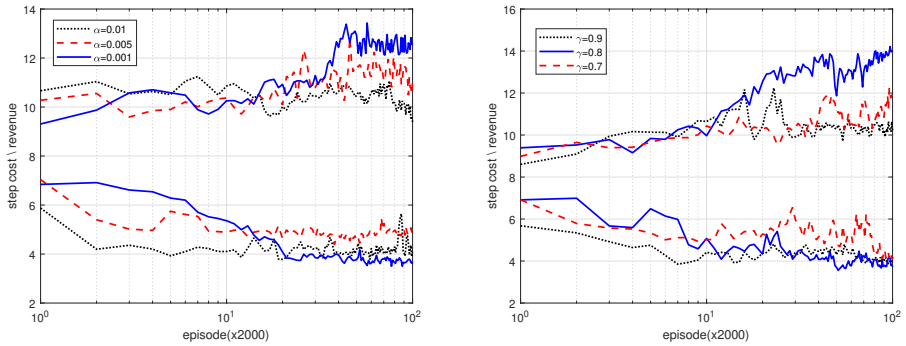
### 6.4 Performance Evaluation

We use real-world public taxi data [91] from Didi Chuxing, a Chinese TNC. The database is a collection of all orders for a given month (11/2016) in Chengdu, a Chinese city, including the time and locations of boarding and leaving the passengers. An example of the data can




(a) Varying $\epsilon$, fixed $\delta = 0.01$, $\alpha = 0.001$, $\gamma = 0.8$

(b) Varying $\delta$, fixed $\epsilon = 0.9$, $\gamma = 0.8$, $\alpha = 0.001$




(c) Varying $\alpha$, fixed $\epsilon = 0.9$, $\delta = 0.01$, $\gamma = 0.8$

(d) Varying $\gamma$, fixed $\epsilon = 0.9$, $\delta = 0.01$, $\alpha = 0.001$

Figure 19: TNC vehicle scheduling parameters tunning

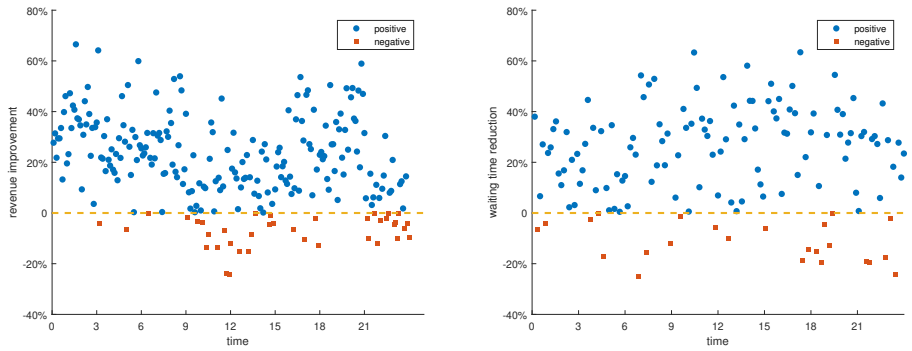
be found in Table 4.

According to the geographical features of Chengdu, we consider a $30 \times 30$ grid representing main downtown area of Chengdu city, where each cell represents 0.25 square kilometers (500 m$\times$500 m). For considering the competition between TNC cruising vehicles, we simulate passengers' distribution and vacant rate $\xi$ of TNC cruising vehicles by using pickup positions. Assume that a TNC cruising vehicle spends the same amount time (e.g., 1min) for each step. Although this assumption doesn't include all of the traffic factors, it is enough to evaluate the performance of our algorithm. The dynamic model we proposed is applicable to different time and space conditions in city. The distance standard is defined as $D_1 = 5$ km, $D_2 = 12$ km and base price equals to $B = 9$ CNY and dynamic price factors $u_1 = 1.6$ CNY/km, $u_2 = 2.4$ CNY/km. The percentage of TNC company's interest for each order $p$ is 10% and unit fuel cost $f$ is 0.5 CNY/km. In our implementation, we choose a

5-layer full connected feed-forward neural network as our DNN construction [92] where each hidden layer has 20 neurons. We set the experience replay memory size as $M = 2000$, and mini-batch size is 32. In order to choose the best parameters of DRL algorithm, we evaluate four parameters: the probability of exploration $\epsilon$, the discount factor $\gamma$, the learning rate $\alpha$ and the exploration increment parameter $\delta$ under a fixed-time urban environment.

As shown in Fig. 19, the average moving step, i.e. the time cost for TNC cruising vehicle to find the passenger and the average revenue per order, which are computed every 2000 episodes from the beginning to the end of training process at one moment, changes quickly at first and then stabilizes on a certain value. The above three lines represent the changes in revenue, the following three lines are the changes in the step cost. The smaller the step cost and the higher the revenue, the better the performance. Fig. 19a shows the dependence of step cost and revenue on $\epsilon$. It can be seen that an optimal $\epsilon$ exists so that we choose $\epsilon = 0.9$ in all subsequent simulations. To balance the exploration and exploiting, and let agent have more exploration in the early episode, we adjust the exploration increment parameter $\delta$ slightly. Fig. 19b shows that $\delta = 0.01$ is the optimal choice so that the model has both the smallest step cost and largest revenue. In Fig. 19c, when $\alpha = 0.001$, the performance of the neural network is the best. Fig. 19d plots step cost under different discount factor $\gamma$. We find that $\gamma = 0.8$ is the best choice. All of the figures in Fig. 19 show that the revenues of TNC drivers gradually increases and the searching time decreases with training process going by.

To evaluate the learning performance, we select one specific day to test the dynamic scheduling model. We compare average step cost per TNC cruising vehicle to pick up the passenger and average revenue per order between with scheduling and without scheduling in the same time period. Specifically, without scheduling means the TNC drivers will search the passenger and earn the revenue randomly and based on their own experience. We divide the whole day into seven periods. We randomly choose some guiding results compared with the unguided one. The model performance improvement ratio is shown in Fig. 20. Fig. 20a plots the increase of drivers' revenue after guidance, and each dot is the average of 100




(a) revenue improvement (b) waiting time reduction

Figure 20: TNC vehicle scheduling final results

data. In the most of time, the revenue increases and the average improvement ratio is 19.2%. Fig. 20b illustrates that the driver's searching time significantly reduces in most cases, and the average shortening rate is 21.7%.

To sum up, the experiments show that the dynamic TNC cruising vehicle route scheduling model could help TNC vehicle find better routes when they are cruising so as to reduce searching time and improve TNC drivers' revenue significantly.

## 6.5 Conclusion

This paper develops an optimization routes scheduling model for TNC cruising vehicles based on deep reinforcement learning. The model can guide TNC cruising vehicles by the continuous interaction between TNC service center and urban environments. Especially, different times and spaces and the competition between TNC cruising vehicles are all factors that we consider. According to the practical big data of Chengdu TNC vehicles, the evaluation results provide empirical support of the effectiveness of deep Q-Network for the TNC cruising vehicle scheduling problem. When TNC cruising vehicles are guided, the searching time is significantly shorter than without guiding, and the long-term revenues of TNC drivers also dramatically increase.

# 7 No One Left Behind: Avoid Hot Car Deaths via WiFi Detection

## 7.1 Introduction

Every summer, there are some children who have been forgotten in the car and died due to heatstroke [93]. Some careless parents have forgotten to take the children out, and finally the irreparable tragedy happened. Just in the United States, 52 children died in 2018 and 43 children died in 2017 because of heatstroke in vehicle [93], and the majority of these in-car heatstroke deaths are accidents that are preventable. In recent years, the number of heatstroke deaths of children in vehicles is still increasing. How to remind these careless parents is an urgent problem to solve. An effective solution to these preventable tragedies is to effectively detect rear seat children, and send a prompt and accurate alert to their parents.

There exist some techniques in the market or in the literature to remind parents of the children left in the rear seat. The most common way is to install a pressure sensor under the car seat [94], but it cannot distinguish between heavy stuff (e.g., grocery items) and the child in the rear seat. Thus, it may send annoying false alarms. Another option is to install a 2-D surveillance camera in the vehicle [95]. According to the image captured by the camera, it can easily analyze whether the children are left unattended in a parked car or not. Even though with high accuracy for alerts, all of the activities in the car will be exposed to the camera, which may lead to the privacy leakage of the driver and passengers. Some other works try to detect the baby by using the infrared cameras. However, it cannot detect the baby when the interior environment is already heated in hot summer. Recently, Diewald et al. in [96] propose an RF-based baby detection system, but additional expensive devices such as CW-radar are needed.

To address the above issues, we rely on the WiFi signal to detect if there is a child in the rear seat of the car. On the one hand, by using the WiFi signal, it can avoid the privacy problems caused by the surveillance camera, and it will not trigger the alarm device

incorrectly because of placing heavy stuff. On the other hand, parents don't need to layout additional devices for the detection system. The proposed design only needs the commercial off-the-shelf WiFi devices as the transmitter and receiver. Additionally, with the prosperity of mobile communications, some luxury cars has already provided WiFi interface, and most vehicles will have that in the near future. These car embedded WiFi devices can easily be set as the transmitter for WiFi signals, and the parent's mobile phones can be set as the receiver. Thus, the child detection system using WiFi signal can be easily implemented in cars being used in daily life.

Therefore, in this paper [97], we propose the rear seat children detection via the WiFi signals using machine learning techniques [90, 98]. The vehicle is not required to equip any other devices, but just the WiFi cards to implement our detection system. There are two steps in our detection system. The first one is to identify if there is a pet, a child or some other sundries in the rear seat of the car by using the static Channel State Information (CSI) of WiFi signals. Next, we try to distinguish between the pets and children according to the CSI signals over time. Additionally, CSI is captured by Network Interface Card (NIC), e.g., Intel 5300 [99]. Our salient contributions are summarized as follows:

- We firstly employ WiFi signal to detect children in the rear seat. Different from existing child detection works, our method is device-free system, and cannot leakage the privacy information for the users.

- We apply both the phase and amplitude measurement of CSI for WiFi Signal. Moreover, we calibrate the phase information and reconstruct the CSI with the adjusted phase.

- A two-step deep learning based child detection method is proposed with CSI. The child and pet can be roughly distinguished with other sundries placed in the rear seat. Furthermore, movements for the child and pet can be captured in CSI radio images, and convolutional neural network is applied to detect children according to these CSI radio images.

Figure 21: WiFi detection experiments: child and dog in rear seat, respectively.

- Our experiment results show the advantages of the proposed deep learning based system compared to the KNN based method in terms of performance. Meanwhile, experiment results show that our child detection system is capable of high precision, and the accuracy is higher than 95%.

## 7.2 Preliminaries and System Design

In this section, we introduce the background of the WiFi channel state information (CSI), and establish the detection system based on CSI.

### 7.2.1 WiFi Channel State Information

Orthogonal Frequency Division Multiplexing (OFDM) technology has attracted great attention because it can effectively resist intersymbol interference (ISI) and improve system capacity [100]. It has been widely implemented in WiFi standards. CSI can be obtained according to a channel estimation process in the 802.11n/g WiFi system. It can represent the signal transmission proprieties such as a combined effect of absorption, scattering, reflection and refraction for these OFDM subcarriers. Compared to traditional received signal strength (RSS) information which only represents received signal strength, CSI could provide us more detailed fine-grained physical information, which is sensitive to the surrounding environments. When the WiFi environment surrounding the transmitter and receiver changes, CSI can represent the changes very well.

As WiFi technology continues to mature, CSI has a wide range of applications in various

fields. Wang et al. [101] achieve the suspicious object detection according to the CSI complex value and machine learning technique. Wang et al. [102] present a novel deep-learning-based indoor localization system using channel state information. Zhang et al. [103] track human breath status by using commodity WiFi device. CSI has also achieved good performance in detecting dynamic changes of human body, such as person recognition and activity recognition [104].

According to installing the driver to the off-the-shelf NICs, e.g., the Intel 5300 NIC, we can extract the channel state information from the commercial WiFi devices. CSI can be represented as $Y = CSI \cdot X + N$, where $X$ and $Y$ are the transmitted and the received signal, respectively. $N$ is the noise in the surrounding areas. The channel state information $CSI$ can be estimated according to the communication link between transmitter and receiver, as shown in Fig. 21. $CSI$ is a $N_{tx} \times N_{rx} \times N_{sub}$ matrix, where $N_{tx}$, $N_{rx}$ and $N_{sub}$ represent the number of transmitter antennas, receiver antennas and the subcarriers, respectively. Considering the case of one transmitted antenna and one received antenna, the CSI value for the $i$-th subcarrier can be obtained from the matrix $CSI$, which is defined as

$$CSI_i = |CSI_i| \exp\{j\phi_i\}. \tag{98}$$

$CSI_i$ is the complex value, which contains both the amplitude $|CSI_i|$ and the phase $\phi_i$ measurement for the channel proprieties of the $i$-th subcarrier. The WiFi OFDM system contains 56 subcarriers according to a 40MHz channel, and the driver of Intel 5300 NIC used in our work can extract the information of 30 subcarriers.

### 7.2.2 CSI Phase Information Analysis

Due to hardware defects of the Network Interface Card itself, the measured phase usually contains lots of errors. As the black and blue points shown in Fig. 22, the measurement phase is randomly distributed between $[0, 2\pi]$. According to [105], there are two kinds of distortions for the measurement phase: one is the Carrier Frequency Offset (CFO). The center frequencies of the transmitter and receiver are not perfectly synchronized. Even though

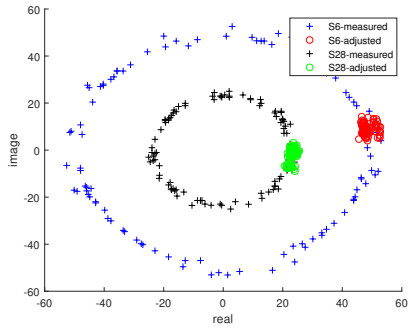



Figure 22: Measured CSI & Adjusted CSI.

the CFO is compensated by the receiver's CFO corrector, signal still carries the residual CFO, which causes the CSI phase offset constantly change according to the subcarriers. The other one is Sampling Frequency Offset (SFO) generated by ADC. The transmitter and receiver sampling frequencies are generally slightly mismatched, which leads to the CSI phase offset, too. According to the distortions mentioned above, the measurement phase can be defined as

$$\tilde{\phi_i} = \phi_i + 2\pi \frac{m_i}{K} \delta + \beta + \mathbb{N}, \tag{99}$$

where $\tilde{\phi_i}$ and $\phi_i$ represent the measured phase and true phase for the $i$-th subcarrier, respectively. $\delta$ is the timing offset due to SFO, $\beta$ is an unknown offset due to CFO, and $\mathbb{N}$ represents the measurement noise. In equation 99, $m_i$ is the subcarrier index of the $i$-th subcarrier, and $K$ is the fast Fourier transform (FFT) size, which equals to 64. It's difficult to get the true phase information due to the unknown $\delta$ and $\beta$. In this work, we unwrap the raw phase information, and then adjust the unwrapping phase information by linear transformation similar used in [106] to get the estimated phase information.

We find that the measured phase is folded in the range of $[-\pi, \pi]$ with the increasing of the subcarrier index $i$. Thus, we compute the true measured phase by subtracting multiple $2\pi$. According to the unwrapped phase, the linear transformation method is applied to

remove the phased errors. The slope of phase $k$ and the offset $b$ can be estimated as

$$k = \frac{\tilde{\phi_{30}} - \tilde{\phi_1}}{m_{30} - m_1} \text{ and } b = \frac{1}{30}\sum_{i=1}^{30} \tilde{\phi}_i. \tag{100}$$

Then, the adjusted phase $\hat{\phi}_i$ can be got by subtracting $km_i + b$, which can be represented by

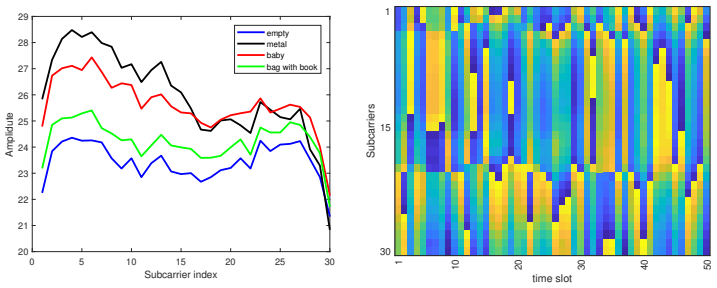
$$\hat{\phi}_i = \tilde{\phi}_i - km_i - b. \tag{101}$$

After we get the adjusted phase, we reconstruct the Channel State Information according to the adjusted phase as $\hat{CSI}_i = |CSI_i|\, exp\{j\hat{\phi}_i\}$. $\hat{CSI}_i$ is the reconstructed Channel State Information, and both the amplitude and the phase information are the constant at this time. For instance, in Fig. 22, the green and red points represent the adjusted phase information for 6 and 28 subcarriers, respectively. Our detection system can capture the properties of different objects based on the stable reconstruction CSI of different objects.

### 7.2.3 System Overview

To detect if there is a child sitting in the rear seat of the car, we set a pair of WiFi transmitter and receiver near the rear door of the car, as shown in Fig. 21. By using channel state information from these communication links, our system is able to identify the classification of objects placed in the middle of these WiFi devices. In most of cases, people always put children, pets, i.e., dogs and other stuff in the rear seat. The basic idea is to capture the changes in the wireless channel when different objects are placed. Different materials and different sizes for the different objects lead to the differences between the responded CSI.

Based on that, we establish our two-step detection system. In the first step, we roughly classify different objects by using the static CSI. Due to the different effects of different objects on the wireless channel, we separate children and pets from other sundries placed in the rear seat via learning algorithms. We cannot distinguish children and pets at this step,




(a) Static CSI (b) CSI Radio Image

Figure 23: Responding CSI for different objects.

due to the similar effects of children and pets on the wireless channel. In the second step, according to the different movements of the pets and children, children can be detected via CSI signals over time. We apply the convolutional neural network (CNN) to classify these series time signals. After recognizing the left child in the rear seat, the alarm will be sent to parents. This two-step child detection system can remind parents remember their children left in the back seat.

## 7.3 Child Detection via Deep Learning

In this section, we introduce a two-step child detection system by deep learning techniques. The children, pets and other objects can be distinguished in the first step. Moreover, according to the different movements for the child and pet, the child can be detected according to the convolutional neural network in the second step.

### 7.3.1 Roughly Child Detection

WiFi signal is very sensitive to the environments surrounding the transmitter and receiver, especially for the area in the middle of these two devices. Due to different materials and different size objects have the different effects of the WiFi signal, the classification of the objects placed in the middle of the devices can be obtained. As shown in Fig. 23a, based our experiments, the fiber and plastic such as bags, books or clothes have the less effects on the wireless channel. Theoretically, most of the wireless signal will penetrate directly through the objects without reflection and scattering. The metal products like laptops or

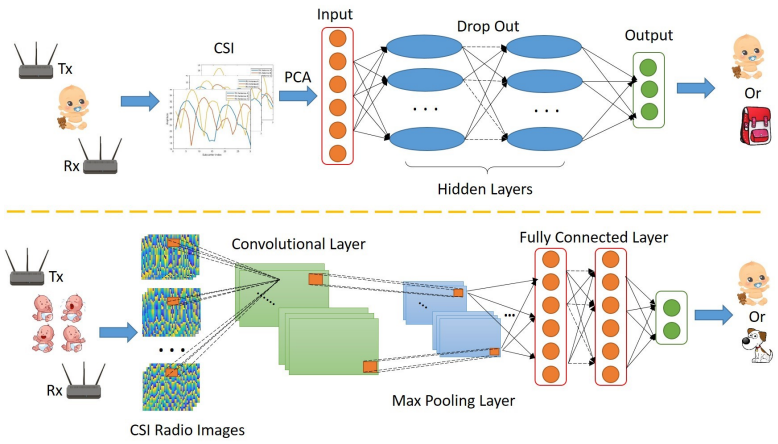



Figure 24: System architecture for baby detection system.

vacuum cup have the strong effects for the wireless channel. In this case, a large portion of WiFi signal bypass the metal objects, and reflection and scattering are happening. For the babies, small children and dogs, the blocking properties to wireless signals by the body made up of blood and skin is between metal and fiber.

Our first step can be achieved based on these differences. We layout three antennas for transmitter and receiver, respectively. Then a nine-pair channel state information can be obtained at one time, and each pair of CSI contains 30 subcarriers. According to the unwrapping and linear transformation introduced in Section 7.2.2, 270 CSI complex values can be gotten each time. Different from others only using part of the subcarriers' information, we adopt the rich information of all subcarriers. Each CSI sample contains $2 \times 270$ elements. Such a high-dimensional data brings the computational burden of the neural networks. To reduce the dimension of the original data, principle component analysis (PCA) is adopted. We apply PCA on the normalized training data set $X^O_{Train}$. Firstly, compute the covariance matrix $C$ of $X^O_{Train}$. Then, calculate the eigenvalues and eigenvectors of covariance matrix $C$. By sorting the eigenvalues $V = (\lambda_1, \lambda_2, ..., \lambda_{540})$ in descending order as $\lambda_1 > \lambda_2 >, ..., > \lambda_{540}$, the corresponding eigenvectors can be gained as $E = [e_1 e_2, ..., e_{540}]$. When the sum of first $k$ eigenvalues is greater than 95% of the sum of all eigenvalues, we select the corresponding eigenvectors $E_c = [e_1 e_2, ..., e_k]$ for calculating the principle components. At this time, the first $k$ eigenvectors that capture 95% of the total variance, which

means the principle components $X_{Train}^{PCA} = X_T^O E_c$ contain more than 95% original information. To guarantee the same size between the training data and test data, the principle components of the test data can also be gained by the eigenvectors $E_c$ as $X_{Test}^{PCA} = X_{Test}^O E_c$.

Based on the principle component analysis, the dimension of the data can be reduced to $k$. Another reason we adopt PCA is that it can eliminate correlations between different features, which means it can remove the environment noise. In our work, CSI streams of different subcarriers are highly correlated, because the fixed surrounding environment has the same impact on different transceiver pairs and different subcarriers. Thus, the reconstructed data can break the relationship between the original features, and eliminate interference from environmental noise. Moreover, PCA-based data can represent the different characteristics of different objects more clearly.

$X_{Train}^{PCA}$ and $X_{Test}^{PCA}$ are the reconstructed data sets. We apply these reconstructed data sets as the input of the neural network. According to the multi-layer fully connected neural networks, the output layer represents the classification of the objects. Based on different objects have different effects on the wireless channel, we define three type of classifications of the objects. One is children and pets, one is metal products, and the other is other plastic or fiber stuff. Thus, the number of nodes for the output layer is 3. According to the SoftMax function, the output values $\hat{y}$ are between $(0, 1)$. We use cross entropy as the loss function in this problem, and the cost function $L$ can be shown as

$$L = \frac{1}{B} \sum_{i=1}^{B} [y ln\hat{y} + (1 - y) ln(1 - \hat{y})], \tag{102}$$

where $B$ is the batch size in training. $y$ stands for the label of the object. $\hat{y}$ stands for the output value of the neural network, which is a probability between 0 and 1. We use Adaptive Moment Estimation to minimize the loss function. Additionally, the dropout technique is applied in the framework to reduce overfitting in neural networks.

### 7.3.2 Child Detection with CSI Radio Image

The children and pets can be recognized as one classification in the first step, and they cannot be distinguished separately due to the similar effects on the wireless channel according to the static channel state information. Thus, we detect the children according to the different movements between children and pets. For the children placed in the car seat, since their bodies are tied by a seat belt, they can only do some movements like shaking their arms and legs. For the pets, such as dogs, they can move freely in the rear seat. Based on these differences, we apply deep learning framework to extract the features of the movements. We collect CSI over the time to do the analysis, which contains features of the movement. Inspired by features extraction ability of CNN in image processing tasks such as image classification or object detection, we apply CNN framework to extract the features of the CSI radio image and do the classification.

CSI radio image is a CSI matrix, and we use the image to represent that matrix, as shown in Fig. 23b, different colors represent the different values. It is composed of the amplitude information of CSI from multiple antennas and multiple channels in a period of time. According to Intel 5300 NIC, channel state information for 30 channels in one transceiver pair can be gained, hence the CSI amplitude measurement vector $A_t$ at the $t$-th instant can be defined as

$$C_t = [C_t^1, ..., C_t^n, ..., C_t^N]^T, \tag{103}$$

where $N$ equals to 30, $C_t^n$ is the CSI amplitude measurement for $n$-th channel at $t$-th instant. We collect the amplitude measurement vectors while a child or a pet is taking some actions. If we record the movement for a while, the number of $T$ vectors can be got. The CSI radio image can be defined as

$$C = [C_1, ..., C_t, ..., C_T]. \tag{104}$$

For one image C, the abscissa is the time axis, and $Y$ axis is the index number of 30

channels. If we set one antenna for transmitter and three antennas for receiver, then we have 3 TX-RX pairs and get three CSI radio images at one instance. These radio images are seen as the input data for the convolutional neural network. Moreover, CNN extracts the features from these images and identify the classification of these images. The proposed framework of CNN is shown in Fig. 24. It contains convolutional layers, max pooling layers and fully connected layers. The first few layers of the neural network are alternating combinations of convolutional layers and max pooling layers. The subsequent layers are the fully connected layers and the results of the classification.

First of all, by vertical and horizontal sliding, the kernel portion is convolved over the entire three CSI radio images in the convolutional layer and the dot product of the image and kernel weights is computed. Each radio image is a feature map, and the calculation result of the $l$-th convolutional layer is shown as

$$X_j^l = \sigma\left(\sum_{i \in M_j} X_i^{l-1} * K_{ij}^l + b_j^l\right), \tag{105}$$

where $i$ and $j$ are the feature map index of the convolutional layers, $b_j^l$ is the bias, and $K_{ij}^l$ is the responding kernel. $*$ represents the convolutional operation and $\sigma$ is the activation function. According to the convolutional layer, we can extract the movement features from CSI radio images. Next, before delivering the results to the next convolutional layer, we apply the data batch normalization layer and max pooling layer to speed up the training process and reduce the sensitivity to the initial network configuration. Finally, two fully connected layers are connected to the convolutional layers, dropout is applied to prevent overfitting problem. SoftMax function converts the output value from fully connected layers into the probability value between $(0, 1)$ for the classification.

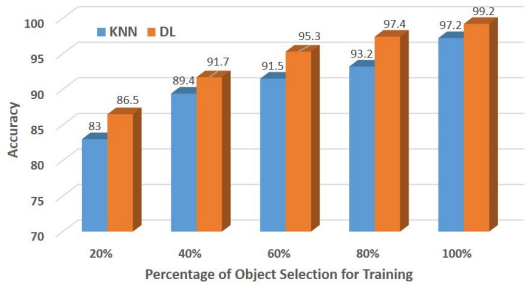
## 7.4 Experiment Results

We apply our detection system on a pair of desk computers as transmitter and receiver, both of which are equipped an Intel 5300 NIC and three 9dBi omni-directional WiFi dual

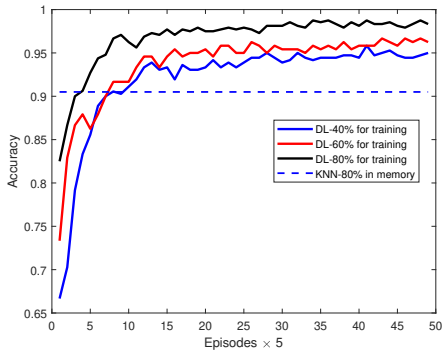
band antennas. The Linux 802.11n CSI Tool [99] is installed on both desk computers. We choose channel 64, which the center frequency is 5.32GHz with 40 MHz bandwidth, as our wireless communication parameter. For both of the two steps, the transmission rate for transmitter is $100pkt/sec$. We place the entire experimental platform in the rear seat of the car.

First of all, for the first step, we conduct lots of experiments to verify that the proposed system can effectively separate children and pets from other stuff. We experiment with twenty different objects, which can be classified into three categories. One is metal, such as laptop in the bag or metal bottle. One is other stuff, like the fiber backpack and plastic bag with cookies. The other is children and dogs. For each experiment, we slightly change the target object to collect static channel state information. For the PCA part, we take the first $k = 90$ eigenvectors which can capture more than 95% of the total variance. This 90 eigenvectors are principle components, which can contain more than 95% original information. We adopt three hidden layers fully connected neural network to solve the classification problem, with 256,128 and 64 nodes for each hidden layer, respectively. To evaluate the performance of the proposed system, we choose different percentages of objects' experimental data as training data, and set the rest part as the test data. For instance, if we choose 60% data as the training data, which means we select the number of 12 objects as the training objects and set the remaining 8 new objects as the test objects. K-nearest neighbors algorithm (KNN) is used as our baseline model. The results are shown in Fig. 25a. As the percentage of training objects used increases, the performance of KNN based method and our deep learning based method increase accordingly. Moreover, our performance is always better than KNN based method. More specifically, if we use enough training samples, i.e. apply more than 60% objects as training objects, accuracy will be above 95%.

After the children and pets are distinguished from other objects in the first step, CSI radio images which contain the movements of children and pets are captured in the second step. We take the experiment with different children and dogs at different times. The size of each CSI radio image is $52 \times 30$. Our CNN framework contains two convolutional




(a) Accuracy for recognizing babies and pets.




(b) Accuracy for baby identification from pets.

Figure 25: Detection Accuracy with different percentages of training data used.

layers which have 32 and 64 kernels, respectively. Two max pooling layers and two fully connected layers are implemented in the network. Similar to step 1, we collect lots of the CSI radio images, randomly select part of the experimental data as the training data set, and set the rest part as the test data set. Fig. 25b shows the test accuracy of different percentages data using in training. Test Accuracy increases as the percentage of training data increases. When we have enough training data, such as 80% of the experimental data, the accuracy will reach 98% or more. Additionally, comparing to the KNN based method, the performance improves significantly.

Based on our two-step child detection system, parents can effectively remember the children who have forgotten in the back seat of the car, and the tragedy of children dying from heat stroke can be avoided.

## 7.5 Conclusion

To avoid hot car deaths, we have developed a two-step rear seat child detection system based on commercial WiFi devices using deep learning method. Based on the different responses of channel state information with different objects, we can distinguish children and pets from other objects by applying deep learning framework in the first step. Meanwhile, the phase information for the CSI is adjusted according to linear transformation method. Furthermore, children can be detected more accurately by the difference between

the movements of child and pet, which are included in CSI radio images. Accordingly, convolutional neural network is adopted to analyze these CSI radio images in the second step. The multiple experiments show that our deep learning approach has a significant performance improvement over KNN based detection method, and the recognition accuracy is above 95%. In the future, we will directly apply the NIC in the mobile phones or vehicles and use CSI signal to detect the vital signs to cope with babies sleeping.

# 8 Future Work

In my future research, I will continue my investigation on intelligent mobile edging, where high efficiency, low cost, and good decision are still the main concepts. The works in this dissertation have shown the feasibility and efficiency of how mobile networks make AI better, how AI makes mobile networks better, and how AI and mobile networks can efficiently enable various intelligent applications. But there are still some problems that need to be solved or further explored. For example, we assume the communication and computing environments are static in current energy/time efficient FL works. However, such ideal cases limit the implementation of the proposed algorithms. In addition, with the ever-increasing computing and communication capabilities of mobile devices, tremendous new applications will continue to be generated and implemented on edge devices. With my extensive experience in deep learning and wireless communications, I plan to conduct future research in the following directions:

- **DRL for Task Placement in Multi-Access Edge Computing:** Cooperative Multi-Access Edge Computing (MEC) is a promising paradigm for next-generation mobile networks. However, when the number of users explodes, the computational complexity of existing optimization or learning based task placement approaches in the cooperative MEC can increase significantly, which leads to intolerable MEC decision-making delay. Therefore, we plan to propose a mean field game (MFG) guided deep reinforcement learning (DRL) approach for the task placement in the cooperative MEC, which can help servers make timely task placement decisions, and significantly reduce average service delay. Instead of applying MFG or DRL separately, we jointly leverage MFG and DRL for task placement and let the equilibrium of MFG guide the learning directions of DRL. We also want to ensure that the MFG and DRL approaches are consistent with the same goal. Thus two approaches can be optimized iteratively to improve the MEC networks' performance.

- **Efficient FL Training with DRL:** The energy and time consumption of mobile devices in FL training can be significantly reduced with the proposed algorithms. However, some practical issues still need to be carefully considered in the implementation of FL, such as data and device heterogeneity, communication and computing run-time variance, etc. Due to the high-dimensional optimization space and time-varying characteristics, such challenges cannot be easily optimized by the traditional methods. Thus, we plan to apply the RL techniques to solve the aforementioned challenges. In DRL, both the communication and computing environments, system architectures, and model parameters can be considered the state. The optimal control actions, such as the selected participating users and the model compression strategies, will be determined accordingly in each FL round. With the help of DRL, the control strategies of the FL training can be made automatically in each round, thus reducing the total time and energy consumption with dynamic communication and computing environments.

- **Extensive Applications of Efficient FL:** With the massive growth of personal data with end users and the rapid popularization of the power-efficient mobile edge devices, FL over mobile devices can be applied to a large number of applications and can be involved in every area of daily life. With energy-efficient training strategies, FL over mobile devices is perfectly compatible with lifelong on-device learning that requires a constant training process and is battery-driven. For example, it can be applied to some real-time assisting services like the voice UI, keyboard prediction, and some low-latency control scenarios such as gaming and automated guided vehicles. Moreover, along with the exponential improvement in on-device AI capabilities, more sensing data from smart sensors like the cameras, microphones, and compass, can be effectively utilized in Industrial IoT, e-health, finance, and social networks, etc.

# Bibliography


[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, April 2017.

[2] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE network*, vol. 34, no. 3, pp. 134–142, February 2019.

[3] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6G," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96–103, April 2020.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, March 2017.

[5] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv:1811.03604*, June 2018.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1.

[7] Qualcomm, "5G+AI: The ingredients fueling tomorrow's tech innovations," https://www.qualcomm.com/news/onq/2020/02/04/5gai-ingredients-fueling-tomorrows-tech-innovations, accessed January, 2021.

[8] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, April 2018.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, Januray 2019.

[10] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. International Conference on Learning Representations*, Vancouver, CANADA, April 2018.

[11] H. Yu and R. Jin, "On the computation and communication complexity of parallel sgd with dynamic batch sizes for stochastic non-convex optimization," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, Long Beach, VA, June 2019.

[12] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE journal on selected areas in communications*, vol. 35, no. 6, pp. 1201–1221, April 2017.

[13] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, October 2020.

[14] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour,"

*arXiv:1706.02677*, April 2018.

[15] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, Virtual Conference, May 2021.

[16] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conference on Computer Communications*, Paris, France, April 2019.

[17] C. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *arXiv:1910.13067*, October 2019.

[18] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, New Orleans, LA, May 2020.

[19] R. Chen, L. Li, K. Xue, C. Zhang, M. Pan, and Y. Fang, "Energy efficient federated learning over heterogeneous mobile devices via joint design of weight quantization and wireless transmission," *arXiv:1406.2661*, December 2021.

[20] D. Shi, L. Li, M. Wu, M. Shu, R. Yu, M. Pan, and Z. Han, "To talk or to work: Dynamic batch sizes assisted time efficient federated learning over future mobile edge devices," *Submit to IEEE Transactions on Wireless Communications*, 2022.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, June 2016.

[22] N. Zhang, Y. Chen, and J. Wang, "Proc. image parallel processing based on gpu," in *2010 2nd International Conference on Advanced Computer Control*, vol. 3, March 2010, pp. 367–370.

[23] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," in *Proc. International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020.

[24] S. L. Smith and Q. V. Le, "A bayesian perspective on generalization and stochastic gradient descent," in *Proc. International Conference on Learning Representations*, Vancouver, CANADA, April 2018.

[25] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. the AAAI Conference on Artificial Intelligence*, Honolulu, HI, January 2019.

[26] X. Mei, X. Chu, H. Liu, Y.-W. Leung, and Z. Li, "Energy efficient real-time task scheduling on CPU-GPU hybrid clusters," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, May 2017.

[27] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and performance characterization and modeling of GPU-accelerated systems," in *Proc. IEEE international parallel and distributed processing symposium*, Phoenix, AZ, August 2014.

[28] S. Schaible, "Fractional programming. ii, on dinkelbach's algorithm," *Management science*, vol. 22, no. 8, pp. 868–873, April 1976.

[29] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—part i: Theory," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1929–1944, December 2016.

[30] Apple's Siri Team, "Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant," https://machinelearning.apple.com/research/hey-siri, accessed June, 2020.

[31] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[32] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Joint optimization of communications and federated learning over the air," *arXiv:2104.03490*, April 2021.

[33] D. Shi, C. Huang, L. Li, H. Wang, X. Zhou, M. Shu, and M. Pan, "Energy and spectrum efficient federated learning via high-precision over-the-air computation," *Submit to IEEE Transactions on Mobile Computing*, 2022.

[34] M. Sriharsha, S. Dama, and K. Kuchi, "A complete cell search and synchronization in lte," *EURASIP Journal on Wireless Communications and Networking*, vol. 111, no. 1, pp. 1–14, March 2017.

[35] O. Abari, H. Rahul, D. Katabi, and M. Pant, "Airshare: Distributed coherent transmission made seamless," in *IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, April 2015.

[36] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. International Conference on Artificial Intelligence and Statistics*. Virtual Conference: PMLR, April 2021, pp. 2350–2358.

[37] IBM, "Ibm cplex optimizer," https://www.ibm.com/analytics/cplex-optimizer, accessed April 4, 2021.

[38] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. International Conference on Artificial Intelligence and Statistics*, virtual, August 2020.

[39] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Transactions on Wireless Communications*, November 2020, doi:10.1109/TWC.2020.3039309.

[40] R. Bonghi, "Jetson stats," https://github.com/rbonghi/jetson_stats, accessed March, 2021.

[41] D. Shi, L. Li, R. Chen, P. Prakash, M. Pan, and Y. Fang, "Towards energy efficient federated learning over 5g+ mobile devices," *accepted by IEEE Wireless Communications*, 2021.

[42] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Proc. of Advances in Neural Information Processing Systems*, Montréal, Canada, December 2018.

[43] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient SGD via gradient quantization and encoding," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, December 2017.

[44] F. Fu, Y. Hu, Y. He, J. Jiang, Y. Shao, C. Zhang, and B. Cui, "Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript," in *Proc. International Conference on Machine Learning.* virtual: PMLR, July 2020, pp. 3304–3314.

[45] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, October 2020.

[46] X. Cao, G. Zhu, J. Xu, and K. Huang, "Optimized power control for over-the-air computation in fading channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7498–7513, August 2020.

[47] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, February 2020.

[48] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, November 2019.

[49] Z. Han, D. Niyato, W. Saad, and T. Başar, *Game Theory for Next Generation Wireless and Communication Networks: Modeling, Analysis, and Design.* Cambridge University Press, 2019.

[50] A. Ibrahim, T. M. N. Ngatched, and O. A. Dobre, "Using bender's decomposition for optimal power control and routing in multihop D2D cellular systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5050–5064, August 2019.

[51] M. I. Ashraf, C. Liu, M. Bennis, W. Saad, and C. S. Hong, "Dynamic resource allocation for optimized latency and reliability in vehicular networks," *IEEE Access*, vol. 6, pp. 63 843–63 858, October 2018.

[52] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, January 2018.

[53] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2282–2292, October 2019.

[54] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, April 2019.

[55] F. Wang, L. Song, Z. Han, Q. Zhao, and X. Wang, "Joint scheduling and resource allocation for device-to-device underlay communication," in *Proc. IEEE wireless communications and networking conference (WCNC)*, Shanghai, China, April 2013.

[56] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances in neural information processing systems*, Long Beach, CA, December 2017.

[57] D. Shi, L. Li, T. Ohtsuki, M. Pan, Z. Han, and V. Poor, "Make smart decisions faster: Deciding d2d resource allocation via stackelberg game guided multi-agent deep reinforcement learning," *IEEE Transactions on Mobile Computing*, June 2021, DOI:10.1109/TMC.2021.3085206, © 2021 IEEE. Reprinted, with permission.

[58] G. Yu, L. Xu, D. Feng, R. Yin, G. Y. Li, and Y. Jiang, "Joint mode selection and resource allocation for device-to-device communications," *IEEE Transactions on Communications*, vol. 62, no. 11, pp. 3814–3824, October 2014.

[59] A. Abdallah, M. M. Mansour, and A. Chehab, "Power control and channel allocation for D2D underlaid cellular networks," *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 3217–3234, March 2018.

[60] L. Li, D. Shi, R. Hou, X. Li, J. Wang, H. Li, and M. Pan, "Data-driven optimization for cooperative edge service provisioning with demand uncertainty," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4317–4328, March 2021.

[61] N. Sawyer and D. B. Smith, "Flexible resource allocation in device-to-device communications using stackelberg game theory," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 653–667, October 2019.

[62] H. Zhang, W. Ding, F. Yang, J. Song, and Z. Han, "Resource allocation in heterogeneous network with visible light communication and D2D: A hierarchical game approach," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7616–7628, August 2019.

[63] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Deep reinforcement learning based mode selection and resource allocation for cellular v2x communications," *IEEE Internet of*

*Things Journal*, December 2019, DOI:10.1109/JIOT.2019.2962715.

[64] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. International Conference on Machine Learning (ICML)*, Madison, WI, July 1998.

[65] J. Hu and M. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, pp. 1039–1069, November 2003.

[66] V. Könönen, "Asymmetric multiagent reinforcement learning," *Web Intelligence and Agent Systems: An international journal*, vol. 2, no. 2, pp. 105–121, October 2004.

[67] C. Cheng, Z. Zhu, B. Xin, and C. Chen, "A multi-agent reinforcement learning algorithm based on stackelberg game," in *Proc. Data Driven Control and Learning Systems (DDCLS)*, Chongqing, China, May 2017.

[68] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. International Conference on Machine Learning (ICML)*, Williamstown, MA, July 2001.

[69] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, WA, August 2001.

[70] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. AAAI Conference on Artificial Intelligence*, Honolulu, HI, January 2019.

[71] H. Zhang, W. Chen, Z. Huang, M. Li, Y. Yang, W. Zhang, and J. Wang, "Bi-level actor-critic for multi-agent coordination," in *Proc. AAAI Conference on Artificial Intelligence*, New York, NY, February 2020.

[72] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018.

[73] D. Mguni, J. Jennings, and E. M. de Cote, "Decentralised learning in systems with many, many strategic agents," in *Proc. Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, February 2018.

[74] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. ACM The World Wide Web Conference*, San Francisco, CA, May 2019, pp. 983–994.

[75] D. Shi, H. Gao, L. Wang, M. Pan, Z. Han, and H. V. Poor, "Mean field game guided deep reinforcement learning for task placement in cooperative multi-access edge computing," *IEEE Internet of Things Journal*, March 2020, DOI:10.1109/JIOT.2020.2983741.

[76] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural computation*, vol. 11, no. 8, pp. 2017–2060, October 1999.

[77] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, February 2015.

[78] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[79] Y. Han, X. Tao, and X. Zhang, "Power allocation for device-to-device underlay communication with femtocell using stackelberg game," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, April 2018.

[80] T. Zhang, K. Zhu, and J. Wang, "Energy-efficient mode selection and resource allocation for d2d-enabled heterogeneous networks: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1175–1187, February 2021.

[81] Z. Li and C. Guo, "Multi-agent deep reinforcement learning based spectrum allocation for d2d underlay communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1828–1840, February 2020.

[82] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv:1312.5602*, December 2013.

[83] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[84] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong, "A cost-effective recommender system for taxi drivers," in *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, New York, NY, August 2014.

[85] J. W. Powell, Y. Huang, F. Bastani, and M. Ji, "Towards reducing taxicab cruising time using spatio-temporal profitability maps," in *International Symposium on Spatial and Temporal Databases (SSTD)*, Minneapolis, MN, August 2011.

[86] X. Du, H. Zhang, H. Van Nguyen, and Z. Han, "Stacked LSTM deep learning model for traffic prediction in vehicle-to-vehicle communication," in *Vehicular Technology Conference (VTC-Fall)*, Canada, September 2017.

[87] T. Verma, P. Varakantham, S. Kraus, and H. C. Lau, "Augmenting decisions of taxi drivers through reinforcement learning for improving revenues," in *Proc. of International Conference on Automated Planning and Scheduling*, Pittsburgh, PA, June 2017.

[88] M. Han, P. Senellart, S. Bressan, and H. Wu, "Routing an autonomous taxi with reinforcement learning," in *Proc. International on Conference on Information and Knowledge Management (CIKM)*, Indianapolis, IN, October 2016.

[89] D. Shi, J. Ding, S. M. Errapotu, H. Yue, W. Xu, X. Zhou, and M. Pan, "Deep q-network based route scheduling for tnc vehicles with passengers' location differential privacy," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7681–7692, March 2019.

[90] D. Shi, J. Ding, S. M. Errapotu, H. Yue, X. Zhou, and M. Pan, "Deep q-network based route scheduling for transportation network company vehicles," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Abu, Dhabi, December 2018, © 2018 IEEE. Reprinted, with permission.

[91] D. COMPANY, "Gaia initiative," https://outreach.didichuxing.com/research/opendata/en/, accessed April 4, 2019.

[92] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[93] O. Kids and Cars, "Heatstroke deaths of children in vehicles," https://www.kidsandcars.org/how-kids-get-hurt/heat-stroke/, accessed April 1, 2019.

[94] E. Gold, "Sensorsafe securemax smart infant car seat," https://www.evenflo.com/gold/, accessed April 1, 2019.

[95] B. Security, "Dual view car camera," https://www.brickhousesecurity.com/car-cameras/, accessed April 1, 2019.

[96] A. R. Diewald, J. Landwehr, D. Tatarinov, P. Di Mario Cola, C. Watgen, C. Mica, M. Lu-Dac, P. Larsen, O. Gomez, and T. Goniva, "Rf-based child occupation detection in the vehicle interior," in *2016 17th International Radar Symposium (IRS)*, May 2016.

[97] D. Shi, J. Lu, J. Wang, L. Li, K. Liu, and M. Pan, "No one left behind: Avoid hot car deaths via wifi detection," in *Proc. IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020, © 2020 IEEE. Reprinted, with permission.

[98] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3072–3108, Fourthquarter 2019.

[99] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11 n traces with channel state information," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 53–53, Jan 2011.

[100] W. Xu, X. Li, C. Lee, M. Pan, and Z. Feng, "Joint sensing duration adaptation, user matching, and power allocation for cognitive ofdm-noma systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 1269–1282, Feb 2018.

[101] C. Wang, J. Liu, Y. Chen, H. Liu, and Y. Wang, "Towards in-baggage suspicious object detection using commodity wifi," in *2018 IEEE Conference on Communications and Network Security (CNS)*, May 2018.

[102] X. Wang, L. Gao, S. Mao, and S. Pandey, "Csi-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, Jan 2017.

[103] D. Zhang, Y. Hu, Y. Chen, and B. Zeng, "Breathtrack: Tracking indoor human breath status via commodity wifi," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3899–3911, April 2019.

[104] X. Ma, Y. Zhao, L. Zhang, Q. Gao, M. Pan, and J. Wang, "Practical device-free gesture recognition using wifi signals based on metalearning," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 228–237, Jan 2020.

[105] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Liu, and M. Liu, "Phaseu: Real-time los identification with wifi," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015.

[106] X. Wang, L. Gao, and S. Mao, "Csi phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1113–1123, Dec 2016.