ENERGY AND BANDWIDTH-AWARE FEDERATED LEARNING IN WIRELESS INTERNET OF THINGS

By

Sili Wu

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF PHILOSOPHY
AT
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
THE UNIVERSITY OF SYDNEY

JUNE 2024

This research reported in this thesis was supported by the award of a Research Training Program scholarship to the Candidate.

To my loving self

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my two exceptional supervisors for their unwavering support throughout my MPhil journey. Firstly, I would like to acknowledge and express my heartfelt appreciation to Prof. Teng Joon Lim for his invaluable assistance in determining and tackling research challenges. His constructive feedback throughout our meetings has significantly enhanced the calibre of my research and served as a valuable resource in overcoming research obstacles. Next, I would like to acknowledge and express my utmost appreciation to Dr. Phee Lep Yeoh from University of the Sunshine Coast. The feedback he provided me with during our meetings proved invaluable in streamlining my thoughts and enhancing my research abilities.

I am deeply grateful for the unwavering love and support provided by my parents throughout my academic journey. Their profound dedication has been my foundation and inspiration every step of the way. I am also incredibly fortunate to be surrounded by friends whose love and encouragement have enriched this journey immeasurably. To all of you, my heartfelt thanks for being my pillars of strength and for believing in me

Sydney, Australia Sili Wu
June, 2024

Statement of Originality

The work presented in this thesis is the result of original research carried out by myself, in collaboration with my supervisors, while enrolled in the School of Electrical and Computer Engineering at the University of Sydney as a candidate for the Master of Philosophy. These studies were conducted under the supervision of Prof. Teng Joon Lim and Dr. Phee Lep Yeoh. This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Sili Wu
School of Electrical and Computer Engineering
The University of Sydney
30$^{\text{th}}$ June, 2024

Acknowledgment of Authorship

I hereby certify that the work embodied in this thesis contains published papers/ scholarly work of which I am a joint author. I have included as part of the thesis a written declaration endorsed in writing by my supervisor, attesting to my contribution to the joint publications/scholarly work.

Chapter II of this thesis is based on materials published in the conference paper (C1) as listed in Section 1.5. My role involved conducting all simulation experiments, creating the analytical framework, interpreting the results, and drafting the papers.

Chapter III of this thesis contains materials from the published conference paper (C2). My work in this chapter included conducting all simulation experiments, creating the analytical framework, interpreting the results, and writing the paper.

Sili Wu

30$^{\text{th}}$ June, 2024

By signing below I confirm that Sili Wu contributed to all publications embodied in this thesis.

Name of Supervisor: Prof. Teng Joon Lim

Signature of Supervisor:

1$^{\text{st}}$ December, 2024

Abstract

In this thesis, we examine the challenges associated with implementing Federated Learning (FL) in wireless systems that have constraints on energy and bandwidth. First, we investigate decentralized FL in a wireless system, taking into account the transmission energy budget of each client which fundamentally limits the range and bandwidth of data communications. To do so, we propose a model partitioning method that highlights the design choices available within the energy constraint – sharing of a larger partition of the model among clients requires transmission range to shrink and therefore sharing with fewer neighboring nodes. It is non-obvious what the best setting of partition size/transmission range is, and we demonstrate that such a setting can be found for particular deployments.

This thesis delves further into integrating Deep Reinforcement Learning (DRL) into the FL system, offering a smarter approach to bandwidth allocation. We propose a DRL-empowered FL framework for wireless clients that utilizes a Deep Deterministic Policy Gradient (DDPG) agent at the central server to allocate communication bandwidth to each client. The DRL aims to reduce each clients' transmission energy by considering their respective channels and distances to the server. Along with the DRL agent, we perform model partitioning to trade-off the amount of information transmitted by each client and the accuracy of the central model.

Ultimately, this study applies these findings to wireless Internet of Things (IoT) systems, demonstrating the practicality of our FL solutions in distributed systems and how innovative Machine Learning (ML) techniques can foster a balanced and efficient wireless IoT under energy constraints. The main contribution is a novel method for bandwidth management and model partitioning, which is vital for fully realizing the potential of FL in environments limited by resource availability.

Table of Contents

List of Figures

List of Acronyms

DRL	**D**eep **R**einforcement **L**earning
DDPG	**D**eep **D**eterministic **P**olicy **G**radient
FL	**F**ederated **L**earning
IoT	**I**nternet **o**f **T**hings
ML	**M**achine **L**earning
iid	**i**ndependent and **i**dentically-**d**istributed
DFL	**D**ecentralized **F**ederated **L**earning
D-PSGD	**D**ecentralized **P**arallel **S**tochastic **G**radient **D**escent
C-PSGD	**C**entralized **P**arallel **S**tochastic **G**radient **D**escent
PFL	**P**ersonalized **F**ederated **L**earning
RL	**R**einforcement **L**earning
MDP	**M**arkov **D**ecision **P**rocess
DNN	**D**eep **N**eural **N**etworks
DPG	**D**eterministic **P**olicy **G**radient
DQN	**D**eep **Q**-**N**etworks
DoS	**D**enial **o**f **S**ervice
LSTM	**L**ong **S**hort-**T**erm **M**emory
NN	**N**eural **N**etwork
D2D	**D**evice-**t**o-**D**evice
DSGD	**D**ecentralized **S**tochastic **G**radient **D**escent
MLP	**M**ultilayer **P**erceptron

CNN	**C**onvolutional **N**eural **N**etwork
PPP	**P**oisson **P**oint **P**rocess
PCA	**P**rincipal **C**omponent **A**nalysis

Chapter 1

Introduction

In this chapter, we introduce the motivation behind the research work presented in this thesis, followed by the background and literature review. As a technical foundation, we begin with the concept of Federated Learning (FL) and its integration into wireless communication systems. Moving forward, we delve into how Deep Reinforcement Learning (DRL) functions, focusing especially on Deep Deterministic Policy Gradient (DDPG) as utilized in this thesis. Lastly, we discuss model partitioning and its application scenarios. Our research contributions are summarized, and an outline of the thesis and a list of publications are provided at the end of this chapter.

1.1 Motivation

The rise of Industry 4.0 and the Internet of Things (IoT) marks the beginning of a new phase where billions of devices are all interconnected, producing a huge amount of data every day. This level of connectivity and the sheer volume of data bring both exciting possibilities and challenges for Machine Learning (ML), especially in using this data effectively while keeping user information private and making the most out of limited resources. FL is seen as a promising approach that lets multiple

devices work together to train a model without needing to send their data to a central place, which helps keep the data private. An example of a system where FL can be employed is a network of surveillance cameras with the global ML task of anomaly detection or facial recognition. Another example is in hospitals which hold lots of private medical data from their patients and employ ML for diagnosis. They are reluctant to share patient information but, at the same time, would like to cooperate with other organizations to achieve higher accuracy [1]. However, applying FL in situations where devices are connected wirelessly brings up tough challenges, mainly due to the limited energy and bandwidth these devices have to perform complex tasks involving neural networks. There is a pressing need for new methods that can use resources more efficiently and effectively, making it possible for FL to be widely used in real-life settings.

This thesis is motivated by the critical need to address these challenges, aiming to unlock the potential of FL for wireless networks constrained by energy and bandwidth limitations.

1.2 Background and Literature Review

1.2.1 Machine Learning and Federated Learning

Traditionally, centralized ML has been employed to extract meaningful information from large amounts of data, with data collected and stored in a centralized location and models trained on this large data set. However, the centralized approach is limited by communication bandwidth and data storage capacity at the central site. Moreover, it raises privacy concerns when the data contains sensitive information which can be exposed through hacking, misuse and other security breaches. This is particularly

important in application domains such as the medical, financial, and military sectors, where regulations such as HIPAA and GDPR [2] are in place. In such cases, sharing data between different parties is not allowed, and the resulting limited access to disparate data sources makes data aggregation for centralized training infeasible.

Under the emerging paradigms of Industry 4.0 and the IoT, FL is a powerful tool to securely and efficiently perform advanced data analytics on the massive amounts of data arising from large numbers of wirelessly interconnected devices [3]. Since its conception [4], FL has been at the forefront of addressing the dual challenges of privacy preservation and data utilization in distributed systems. There are some distinctions between FL and the traditional distributed learning. In distributed learning, data is distributed to several parties to train a model in order to utilize their computational power. The distributed data is also in an independent and identically-distributed (iid) manner. The objective of distributed learning is to parallelize computing power while FL is to train a global model across a large number of learners [5].

A major advantage of FL is that a network of wireless IoT devices can collaboratively train a central model by only sending their locally trained models and not sharing any of their individual data. This offers attractive benefits to the wireless clients in terms of increased privacy and reduced wireless communication overheads [6].

The original form of FL where a central server aggregates model weights collected from other clients is known as centralized FL [7]. The centralized architecture is shown in figure 1.1(a). In each iteration, the clients and server perform tasks sequentially. Step 1, local devices will act as clients and train their own local ML models with local data. Step 2, these model weights will be transmitted to the central server by

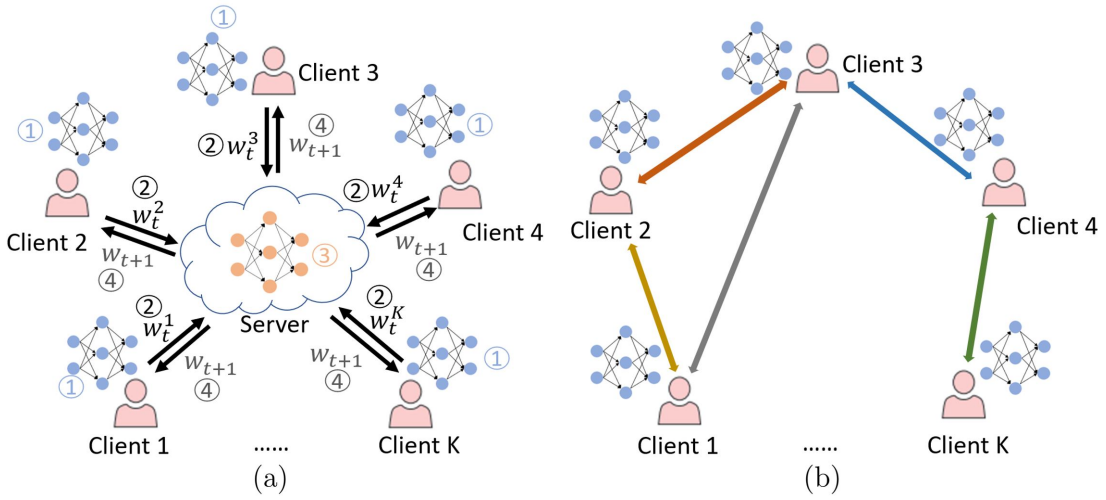

Figure 1.1: (a) Centralized FL versus (b) Decentralized FL

each client. Step 3, the central server, also known as the aggregator, will perform model weight aggregation using aggregating algorithms like FedAvg [4] or FedProx [8] to generate an updated global model. Step 4, the central server will send the aggregated model weights back to all clients. Clients proceed to train their local models initialized with the latest global model and the process of step 1 to 4 will repeat until a stopping criterion is met.

1.2.2 Decentralized Federated Learning

Centralized FL however introduces other problems due to the presence of an aggregator. First, the aggregator presents a single point of failure. If the central server stops working due to e.g. power outage, the entire FL training process will have to be paused. Second, the participants may struggle to reach a consensus on a trustworthy aggregator [9]. Third, even with a trusted aggregator, if a malicious actor gains access to the central server, they could potentially obtain all the personal models and subject the system to an inference attack [10]. We have already mentioned communication

overhead at the aggregator [11] as well. Thus, a central server is not always preferred or even feasible, and a few Decentralized Federated Learning (DFL) solutions have been proposed [9] [11] [12] [13]. All nodes (or workers) in the DFL system will work independently to train their local models as in centralized FL, but most (perhaps all) nodes will also play the role of aggregator at particular points in time, thereby dispensing with a central server. Normally, there will also be a consensus mechanism that tells all workers when to end the training phase. In addition, all nodes need to agree on a common training task, a model architecture, etc. figure 1.1(b) illustrates the architecture of decentralized FL.

FL without a centralized server has been studied quite extensively. Lian et al. [11] studied the Decentralized Parallel Stochastic Gradient Descent (D-PSGD) algorithm on the decentralized computational network and prove that D-PSGD has the same convergence rate as Centralized Parallel Stochastic Gradient Descent (C-PSGD), but it outperforms C-PSGD by avoiding communication traffic congestion at the center node. [12] designed a trust and incentive-based FL called FLChain using blockchain to store gradient parameters to be audited by the public, where incorrect or dishonest updates will be detected by the public and punished. In [9], BrainTorrent was introduced to perform FL on a full peer-to-peer basis without a central server, which allows clients to communicate with each other to exchange model weights directly. However, assumptions were made that a limited number of clients have insufficient data to train on but have strong communication and computation capacity, thus making this solution impractical when we have a large number of IoT devices in the network. Improving on this, [13] proposed a similar decentralized FL idea but without the full peer-to-peer model – each client will randomly select some clients to

communicate with. However, none of these works have considered a limited energy budget, which is a critical limitation in IoT devices. As part of this thesis, we will focus more on this constraint and study the optimization of a wireless DFL system suited to IoT.

1.2.3 Communication Overhead and Model Partitioning

Another problem centralized FL introduces is the communication overhead at the clients. In many IoT applications, devices operate with extremely low power due to their size, cost, and technological limitations. However, in FL, each model update could still require clients to transmit a challenging volume of data to the central server [14]. As such, communication cost and bandwidth allocation are critical variables to optimize in FL, particularly in scenarios involving energy-constrained clients such as personal wireless devices and lightweight IoT sensors.

Initial research focused on optimizing the FL process to reduce communication overhead and improve model accuracy with strategies like model compression and sparsification [15]. Authors in [15] proposed two methods to reduce uplink communication costs in FL, namely structured updates and sketched updates. Structured updates learn an update from a restricted space with fewer variables, which significantly reduces communication costs without sacrificing model accuracy. Other attempts at bandwidth reduction include PerFit, a personalized FL framework [16] for a cloud-edge architecture that enables the learning of a globally-shared model by freezing the lower layers of the model and training the higher layers with clients' personal data. In [17], the authors proposed the FedVF algorithm for personalized FL that updates global and personalized parameters based on layer-wise parameter combinations. Authors in [18] proposed a bisection search algorithm to reduce the time

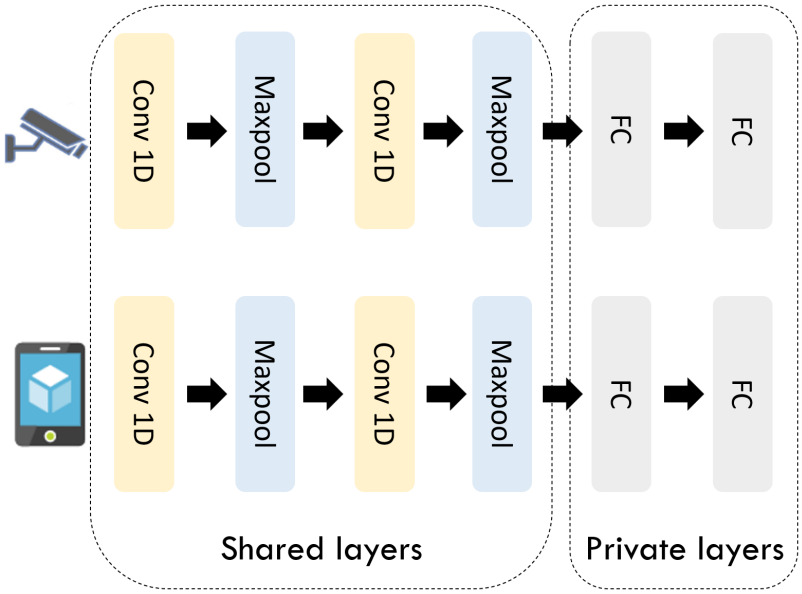

Figure 1.2: Example of model partitioning

delay of FL. Yet, these advancements did not fully tackle the unique constraints faced by wireless networks, where energy consumption and bandwidth allocation become critical bottlenecks.

Among the techniques mentioned above, model partitioning is a common one used in distributed ML systems, including FL and Personalized Federated Learning (PFL). In these systems, model partitioning is used to divide an ML model into smaller sub-models that then enables the sharing of partial (rather than complete) model information. The reason we only share the first partition of the model, also known as the base layers, is that base layers are used to extract features of the dataset, whereas higher layers learn more specific features tailored to the current device [19]. An example is provided in figure 1.2 for a six-layer neural network. We can choose the first 4 layers to share with other clients when performing FL aggregation, and keep the last two layers private. In this way, each client will get information of the first

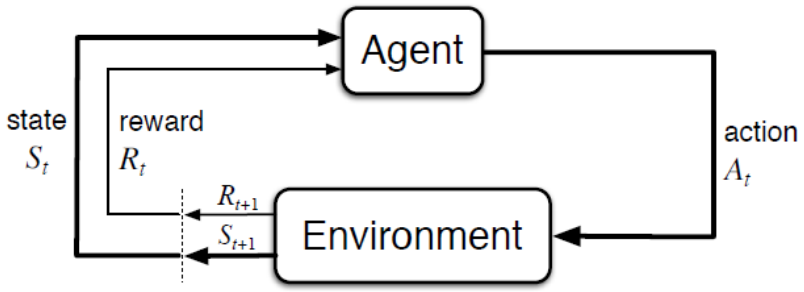

Figure 1.3: The agent-environment interaction

four layers from other clients, and keep the last two layers tailored to its own dataset. Experimental results from [17] show that the algorithm can provide a personalized model with competitive performance and guaranteed global model accuracy.

1.2.4 Deep Reinforcement Learning

The second part of this thesis aims to develop a DRL algorithm along with the FL structure to find an optimal solution for a bandwidth allocation problem. In this section, we will introduce some basics of DRL, and deep deterministic policy gradient (DDPG), a policy-based DRL algorithm employed in this thesis.

RL Preliminaries

Reinforcement Learning (RL) is used to solve problems that satisfy the Markov property, and this process is called a Markov decision process (MDP) [20]. In RL, the agent, or decision-maker, learns from iteration to achieve a goal. The thing it interacts with is called the environment, consisting everything outside the agent. The agent- environment integration example is shown in figure 1.3 [20]. The agent and environment interact continually at each discrete time step, $t = 0, 1, 2, ...$ At each time step t, the agent receives the **state** S_t from the environment as a representation

of the current situation. Based on this, the agent takes an **action** A_t. One time step later, the environment responds to the agent with the next state of environment representation S_{t+1} and a numerical **reward** R_{t+1}.

At each time step, a policy π is used by the agent to implement a mapping from the state to an action. The policy is denoted by π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Methods of reinforcement learning is the process by which an agent adjusts its policy based on what it learns from its experiences.

The goal in RL is to select a policy which maximizes expected return when the agent acts according to it. The most commonly used return in RL is the infinite-horizon discounted return, which is the sum of all rewards ever obtained by the agent, but discounted by how far off in the future they are obtained through a discount factor $\gamma \in (0, 1)$:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1.2.1}$$

where the trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots)$ is a sequence of states and actions, and r_t is the instantaneous reward at time step t. The optimal policy π^* is the policy that maximizes the expected cumulative reward. The central optimization problem in RL can then be expressed by

$$\pi^* = \arg\max_{\pi} E_{\tau\sim\pi}\left[R(\tau)\right], \tag{1.2.2}$$

where $E_{\tau\sim\pi}$ is the expected value with respect to the distribution over trajectories τ induced by following the policy π.

DRL and DDPG

Algorithms for DRL integrate techniques from deep learning, like Deep Neural Networks (DNN), with reinforcement learning algorithms. DNN can be used to approximate functions that are crucial for learning, such as the policy function $\pi_t(a|s)$ (which determines the best action to take in a given state) or the value function $Q^\pi(s, a)$ (which estimates the future reward, or the Q-value, for a given state s and action a) that help guide the agent's decisions.

DDPG is a policy-based algorithm of DRL that is specifically designed for environments with continuous action spaces.. It combines elements from Deterministic Policy Gradient (DPG) and Deep Q-Networks (DQN) [21]. Unlike DQN that only employs a NN as critic to approximate the long-term cumulative return [22], DDPG utilizes an actor-critic architecture. The actor, or the policy network μ, is a NN that determines which action, $\boldsymbol{a} = \mu(\boldsymbol{s}|\theta^\mu)$, to take in a given state, where θ^μ represents the parameters of the actor network. It directly maps states to actions, and allows the algorithm to compute the optimal action for any state without exploring other random actions, making it deterministic. The critic, or the Q-value network, is another NN that estimates the Q-value, $Q(\boldsymbol{s}, \boldsymbol{a}|\theta^Q)$, of that generated action under the given state.

During DDPG training, the critic's evaluation is used to update the actor, and the critic itself is updated by minimizing the difference between the predicted Q-values and the actual returns. In continuous action spaces, the action space is often very large or even infinite. Unlike discrete action spaces where actions can be enumerated, continuous action spaces require selecting an action from a continuum of values. DDPG employs a deterministic policy, wherein the actor network directly outputs

a specific action for a given state, rather than producing a probability distribution over potential actions. This is efficient because it avoids the need to sample from a distribution, which can be computationally expensive and inefficient in continuous spaces. Therefore, DDPG is particularly well-suited for environments with continuous action spaces.

1.2.5 Bandwidth Allocation and DRL in Wireless Networks

The allocation of bandwidth is not just a matter of network optimization; it is intricately linked to energy consumption. Clients with limited battery life must judiciously use their resources for computation and communication. Traditional methods of bandwidth allocation for FL do not account for the dynamic interplay between energy constraints and network conditions [5]. In [23] and [24], to save bandwidth, authors focused on actively selecting a portion of the FL clients' updated training parameters for aggregation at the parameter server. In [25], the authors believed "later-is-better" and developed a heuristic algorithm for joint client selection and bandwidth allocation that achieves a long-term performance guarantee. Most such methods exclude some clients in each aggregation step, resulting in a negative impact on accuracy and convergence speed [26] [27].

Recent studies have begun exploring DRL to optimize decision making in wireless networks. Its ability to learn and adapt to various environments makes it suitable for optimizing bandwidth allocation under energy constraints. Authors in [28] presents a model-free, off-policy algorithm based on DRL for joint sensor scheduling and power allocation in wireless networked cyber-physical systems under Denial of Service (DoS) attacks. In [29], the authors formulated joint client selection and bandwidth allocation as a Markov decision process and developed an algorithm for selecting the best clients

at each round, by considering their data, computing power, and channel gain. Deep-Q learning [30] and Long Short-Term Memory (LSTM) [31] have been used to perform client selection for FL. However, when it comes to bandwidth allocation, DDPG is particularly relevant among DRL algorithms [21]. DDPG, known for its efficiency in continuous action spaces, can be tailored to balance the trade-off between bandwidth usage and energy consumption in real-time. This research stream lays the groundwork for the second part of this thesis, aiming to explore the complexities of applying FL more thoroughly. We will focus on employing DRL techniques to enhance FL in scenarios constrained by energy and bandwidth.

1.3 Contributions

This thesis introduces two novel frameworks designed to overcome the challenges of implementing FL in wireless networks with energy and bandwidth constraints.

The first key contribution is the exploration of wireless decentralized FL for energy-constrained clients. This work innovates the use of a model partitioning strategy that balances the trade-off between the model's information shared and the energy used for transmission. By optimizing the partition size and transmission range, this method demonstrates a novel approach to maximizing FL performance within strict energy budgets. We also found that convergence to the same solution occurs whether only a small portion of the training dataset is used, or the entire dataset, which suggests a low-complexity training method using only a fraction of available data. The detail of this work will be introduced in Chapter 2.

The second significant contribution is a DRL-based FL framework that utilizes the DDPG algorithm for intelligent bandwidth allocation among wireless clients. Using

model partitioning, we quantified the relationship between energy budget, amount of information shared, and allocated bandwidth. We established a DRL method for bandwidth allocation on the uplink of the FL system. Specifically, we deploy a DDPG agent at the central server of the FL and design the state space, action space and reward in a novel DRL-assisted FL framework. This approach not only conserves energy and computational resources but also markedly improves the global model's accuracy. The detail of this work will be introduced in Chapter 3.

Through extensive simulations employing public datasets, the proposed frameworks were validated, showcasing their applicability and efficiency in distributed systems constrained by energy and bandwidth. Together, these contributions address the pressing issues of bandwidth and energy constraints in FL deployment within wireless networks, setting a new direction for research and practical applications of FL in the era of IoT and beyond. Through an in-depth examination of these frameworks, this thesis aims to provide a comprehensive guide for future advancements and implementations of FL in environments where optimizing energy and bandwidth is paramount.

1.4 Thesis Outline

This thesis is organized as follows.

Chapter 1 has introduced the motivation of this thesis. It also provides some theoretical background for Federated Learning, Deep Reinforcement Learning and model partitioning used in Chapter 2 and 3. Existing work on the current problem in FL is presented and the research gaps are explained. In addition, we present the contributions.

Chapter 2 presents a new method to trade-off the model's information shared and the energy used for transmission in DFL. In this chapter, we aim to optimize the partition size and transmission range.

Chapter 3 introduces a novel solution to perform bandwidth allocation for FL using a DRL agent at the central server. In this chapter, we aim to reduce the transmission overhead and improved the global model's accuracy.

Chapter 4 serves as the conclusion of this thesis, summarizing the major contribution and identifying potential areas for future research.

1.5 Related Publications

The following is a list of publications in refereed conference proceedings produced during my MPhil. candidature.

Conference Papers

(C1) S. Wu, S. Shen, P. L. Yeoh and T. J. Lim, "Wireless Decentralized Federated Learning with Energy-Constrained Clients," 2023 IEEE 9th World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, 2023, pp. 01-06, doi: 10.1109/WF-IoT58464.2023.10539451.

(C2) S. Wu, S. Shen, P. L. Yeoh and T. Joon Lim, "Deep Reinforcement Learning-Empowered Federated Learning for Wireless Clients with Energy and Bandwidth Constraints," 2024 IEEE Annual Congress on Artificial Intelligence of Things (AIoT), Melbourne, Australia, 2024, pp. 1-6, doi: 10.1109/AIoT63253.2024.00011.

Chapter 2

Wireless Decentralized FL with Energy-Constrained Clients

In the previous chapter, we introduced the advantages of DFL and some limitation on the current work. Thus, in this chapter, we focus on the study of model sharing strategies in DFL, where in an ad-hoc network, each client acts as a worker when performing local training and as an aggregator when receiving and aggregating model weights from neighbors. For a given transmission energy budget at each client, it may not be optimal to share all learned model parameters because there is an energy cost to sharing each parameter value. Instead, if a client shares only a subset of the model parameters, e.g. certain layers in a Neural Network (NN), it will be able to transmit over a longer range for the same energy consumed in transmission, compared to sharing every layer. In a wireless network, transmitting over a larger range may result in reaching more neighbours, thereby improving the performance of FL. It is not obvious how one chooses the number of layers to share and the transmission range to use in a given network. Hence the main question interrogated in this piece of work is: ***What are the optimal number of shared NN layers and transmission range that achieves the best accuracy in a classification task, for a given***

transmission energy budget at every client?

To address the above question, we consider a novel framework for optimizing DFL settings that accounts for a limited transmission energy budget. To the best of our knowledge, previous works did not consider the trade-off between the number of shared NN layers and the transmission range/number of clients to cooperate with. Thus, we study this trade-off in a DFL setting to maximize the system performance.

2.1 System Model

Consider a set of K devices, denoted as $\mathcal{K} = \{1, ..., K\}$, that can only communicate with their neighboring devices through a wireless Device-to-Device (D2D) network. The network's connectivity is described by a bidirectional graph $\mathcal{G}(\mathcal{K}, \mathcal{E})$, where $\mathcal{K}$ is the set of nodes (vertices), representing entities in the network, and $\mathcal{E}$ is the set of edges (links), representing connections between pairs of nodes. The neighbors of node k are designated as $\mathcal{N}_k = \{j \in \mathcal{K} | (k, j) \in \mathcal{E}\}$. In accordance with the FL framework, each device holds a local dataset $\mathcal{D}_k$, and all devices work together to train an ML model by exchanging information related to the model without sharing data samples directly.

2.1.1 Energy Constraints

Due to limited energy capacity in small IoT devices, the energy budget is a major design constraint in IoT. In this work, we assume that energy at a wireless client is used primarily for information transmission and for local ML training. T denotes the

total number of rounds of aggregation during training in DFL. We consider orthogonal multiple access techniques with total bandwidth B for local model transmission to other clients. At the k^{th} client, let $E_{k\max}$ denote the total pre-defined maximum energy available for both local training and model transmission. The actual energy consumption per round $E_k(d_k, L_k)$ is a function of transmission range d_k and amount of information shared L_k (measured in bits), and is calculated by adding the transmission energy consumption $E_k^{tx}(d_k, L_k)$ and training energy consumption E_k^{tr}, i.e.

$$TE_k\left(d_k, L_k\right) \leq E_{k\max}, \quad \forall k, \tag{2.1.1}$$

$$E_k\left(d_k, L_k\right) = E_k^{tx}\left(d_k, L_k\right) + E_k^{tr}. \tag{2.1.2}$$

As discussed later, E_k^{tr} is not a function of d_k and L_k, thus we will focus on $E_k^{tx}(d_k, L_k)$. Let p_k be the transmit power of user k, and h_k be the channel gain between user k and a receiver at distance d_k. Assuming line-of-sight free-space propagation $h_k \propto d_k^{-2}$ and hence we can write $h_k = \mu/d_k^2$ where μ is assumed to be known (e.g. through pilot symbols) and N_0 is the power spectral density of the complex baseband white Gaussian noise. The achievable transmission rate R_k can be expressed according to Shannon's formula as

$$R_k = B\log_2\left(1 + \frac{p_k h_k}{N_0 B}\right). \tag{2.1.3}$$

Rewriting equation 2.1.3, p_k can be expressed as

$$p_k = \frac{N_0 B}{h_k}\left(2^{\frac{R_k}{B}} - 1\right). \tag{2.1.4}$$

Since L_k bits are transmitted to share node k's model parameters, the time required for transmission to a client at a distance d_k away is $\tau_k = L_k/R_k$. Therefore, the transmission energy consumption is given by

$$E_k^{tx}(d_k, L_k) = p_k \tau_k = p_k \frac{L_k}{B \log_2\left(1 + \frac{p_k \mu}{N_0 B d_k^2}\right)} \tag{2.1.5}$$

From equation 2.1.2, we have $E_{k\max}^{tx} = E_{k\max} - E_k^{tr}$. If we assume that we use up the entire energy budget, i.e. $TE_k^{tx}(d_k, L_k) = E_{k\max}^{tx}$, we can derive the relationship between d_k and L_k as

$$L_k = \frac{E_{k\max}^{tx} B}{T p_k} \log_2\left(1 + \frac{p_k \mu}{d_k^2 N_0 B}\right). \tag{2.1.6}$$

Equation 2.1.6 shows that the amount of information that can be shared, L_k, monotically decreases with increasing d_k. In other words, by choosing to reach more neighboring nodes, we necessarily reduce the amount of information that we share. Since sharing with more neighbors is beneficial to FL while sharing less model information is detrimental, equation 2.1.6 quantifies the trade-off facing DFL designers with energy-constrained nodes such as in IoT.

2.1.2 Trade-off via Model Partition

Having derived the relationship between d_k and L_k, we would like to find the optimal (d_k, L_k) that results in the least total loss/highest accuracy in the system. To reach and hence share model information with more nodes, d_k must be increased by reducing L_k. Therefore, we need to decide where to divide the model weights into two partitions, then clients will only share and aggregate the first partition of their model, which is the layers closer to the input layer. The number of layers shared is denoted by $n \in \mathbb{Z}_+$, where n is the largest integer not greater than the number of NN layers that can be accommodated by L_k bits of coefficients. L_k is derived from

equation 2.1.6. In this network system, the model-simplifying strategy set Π contains several possible strategies π, each of which is a (d, n) pair, calculated using equation 2.1.6 and the NN model structure.

We assume that all clients are aware of other clients' locations, e.g. through periodic updates over a control channel. In addition, it is assumed that any two clients will exchange the same amount of information with each other, i.e. $L_j = L_k$ if nodes j and k are connected. To simplify the scenario, we also assume all clients have the same device type, resulting in the same energy budget across the IoT network.

2.2 Proposed Method

Algorithm 1: Decentralized FL on the k^{th} node

Input: Transmit energy budget E_k^{tx} $\forall k$
Output: The strategy π with the lowest total loss

for *Each strategy* π **do**
  for *Round* t = 1,2,3...T **do**
    for *Each client* k **do**
      Compute neighbor set $\mathcal{N}_k$
      Compute local SGD and model weight w_k^t
      Share partial model weights with all client j = 1,2,...,$|\mathcal{N}_k|$ in $\mathcal{N}_k$
      Perform weight aggregation $w_k^{t+1} = \frac{|\mathcal{D}_k| w_k^t + \sum |\mathcal{D}_j| w_j^t}{|\mathcal{D}_k| + \sum |\mathcal{D}_j|}$
    end
  end
  for *Each client* k **do**
    Compute and share $l_{k,\pi}$ with all $|K|$ clients
    Compute total loss for strategy π: $l_\pi = \sum_{k=1}^{K} l_{k,\pi}$
  end
end

The discussion of transmission scheduling is out of scope in this work so that

we assume a good scheduling scheme already exists among all clients. For example, some Decentralized Stochastic Gradient Descent (DSGD) algorithms were proposed to schedule non-interfering communication links in parallel [32]. Each client has its own dataset $\mathcal{D}_i$ that was collected locally and individually, and not shared with others. All clients reach to a consensus about what the common training task is, what the training model structure is and what the training classes are. Therefore, the energy consumption due to local training E_k^{tr} is the same for all clients. Algorithm 1 provides a heuristic approach designed to approximate a good solution given the constraints, which finds the optimal partitioning strategy based on empirical performance. After each round of local training which contains some local SGD epochs, client k will communicate with all clients in its neighboring client set $\mathcal{N}_k = \{j \in \mathcal{K} | (k, j) \in \mathcal{E}\}$, and aggregate their partial models to continue to the next round of training. This process will repeat until it converges within the maximum number of rounds and the maximum time allowed. The model simplifying strategy π tells all clients how many layers of the NN model to transmit and aggregate. The goal for each client k is to find the model weights:

$$w_{k,\pi}^* = \arg\min_{w_{k,\pi}} f\left(w_{k,\pi}, \mathcal{D}_k\right), \quad \forall k, \tag{2.2.1}$$

where $f\left(w_{k,\pi}, \mathcal{D}_k\right)$ is the local loss function of the model for client k, and $\mathcal{D}_k$ is the local dataset client k has, with each data entry containing a pair of input feature vectors and the ground truth output label. Once the above process is performed for all possible strategies, all clients will broadcast their final local losses $l_{k,\pi}$ to others and all clients will agree on an optimal strategy π^* that minimizes the global loss l_π, which is the sum of the local losses from all K clients.

$$\pi^* = \arg\min_{\pi} \sum_{k=1}^{K} f\left(w^*_{k,\pi}, \mathcal{D}_k\right), \quad \forall k, \tag{2.2.2}$$

$$s.t. \quad TE_k\left(d_k, L_k\right) \leq E_{k\max}, \quad \forall k, \tag{2.2.3}$$

where d_k is the radius of reachable area by client k, L_k is the number of shared coefficients of ML model (in bits), and $E_{k\max}$ is the energy budget client k has for communicating the model information. The simulation results that demonstrate the effectiveness of Algorithm 1 is presented in section 2.3.2.

2.3 Simulation Results

In this section, we simulate decentralized FL to evaluate the trade-off between the number of layers of the model to share, and the number of clients to cooperate with. The goal is to have the highest averaged accuracy in the network system.

2.3.1 Simulation settings

To test our proposed method in simulations, we use the MNIST [33] dataset to train a 4-layer Multilayer Perceptron (MLP) neural network, and CIFAR-10 [34] to train a 6-layer Convolutional Neural Network (CNN). Details of the datasets are listed in Table 2.1. In the simulations, we set 10 clients in total and randomly draw training samples for each client. As a result, the simulation utilizes iid data and each client has 6000 training images for MNIST or 5000 for CIFAR-10 (CIFAR) with all 10 classes. In our simulation, we used batch size of 10, learning rate of 0.01, and communication rounds

Table 2.1: Training and testing datasets

Dataset	# of training images	# of testing images	Input size	Training model
MNIST	60000	10000	28*28*1	MLP
CIFAR-10	50000	10000	32*32*3	CNN

Table 2.2: Strategy set Π for proposed DFL with MNIST (left) and CIFAR (right) datasets

Average number of neighbors	Number of layers shared
0	4
2	3
4	2
9	1

Average number of neighbors	Number of layers shared
0	6
1	5
3	4
6	3
8	2
9	1

of 500. Table 2.2 lists out the strategy set π used in our experiments, calculated using equation 2.1.6.

We consider two types of network topologies in our simulation. The first one is where all clients are spread out evenly on a ring and there is no edge client. The other topology is where clients are randomly located in a finite square area, of which the locations are generated randomly using Poisson Point Process (PPP), with a density of 10 clients/area. To simulate the wireless network, we consider the total bandwidth $B = 1$ MHz. The power spectral density of the complex white Gaussian channel noise is set as $N_0 = 10^{-12}$ W/Hz. Transmit power p_k is 0.5 W. Rayleigh fading parameter μ_k is 0.01. For each client, the transmission energy budget is set as $E = 0.15$ J. It is assumed that when sharing the minimum amount of model weights, all clients remain within communication range of each other.

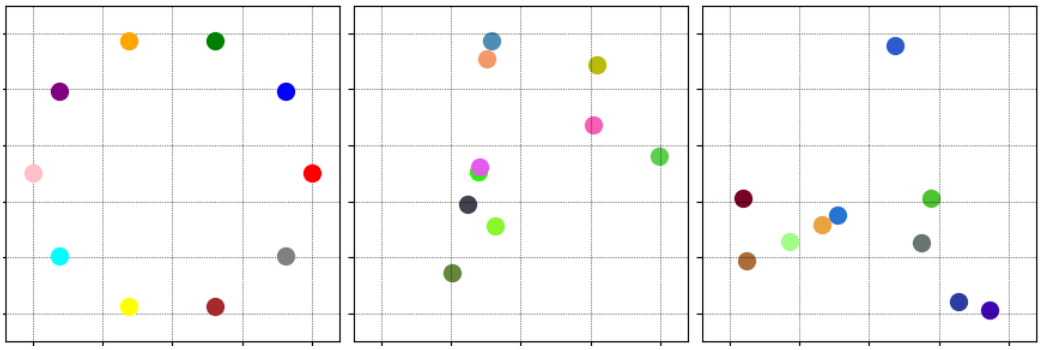

Figure 2.1: Topologies (left to right): (a) Ring topology, (b) Random topology 1, and (c) Random topology 2

2.3.2 Experiment result and analysis

Results under different topologies

Firstly, we have the ring topology where all clients are spaced out on a ring evenly, see figure 2.1(a). In this special topology, clients with the same range for communication will have the same number of neighbors. Additionally, there will be no edge client. We also run the same simulations in two randomly generated topologies, see figure 2.1(b)(c). The result is shown in figure 2.2, which shows the averaged accuracy of 10 clients under different strategies. We can see that, when the amount of shared information is limited by an energy budget, it does not always give better performance to reach out to more clients in the network. This is due to the reduced completeness of the shared model when the communication range increases. In the figure, for both MNIST and CIFAR datasets, there exists a peak marked by a red star, giving the optimal strategy that yields the best performance in terms of accuracy. This peak point depends on the topology itself. For example, in the CIFAR experiments, sharing with one neighbor in ring topology gives the best accuracy, whereas in the two random topologies, sharing with three neighbors is the best strategy. Moreover, we see that

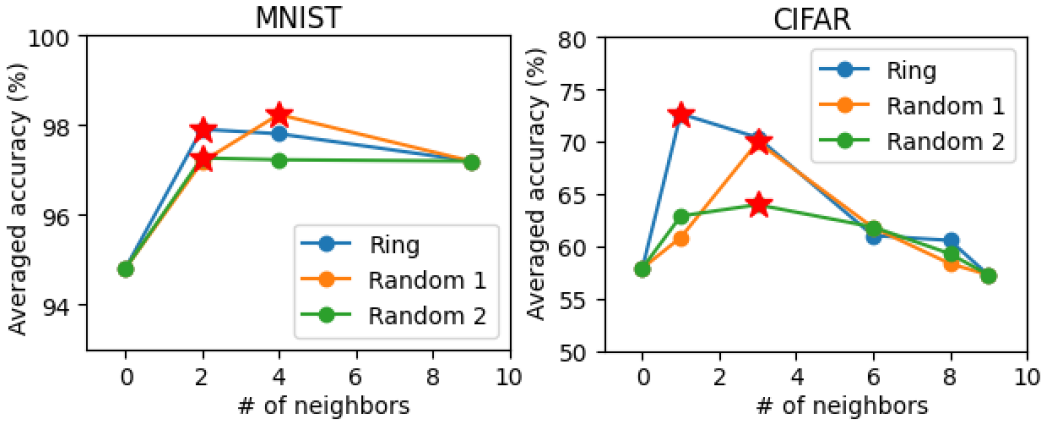

Figure 2.2: Averaged accuracy vs different number of neighbors for different topologies.

random topologies 1 and 2 have the same optimal strategy, but the highest accuracies they get are different. This is because in random topology 2 in figure 2.1(c), there is an obvious outlier within the cluster. Therefore, the accuracy of the optimal strategy depends on how clients are distributed in the network.

Training with small portion of dataset

Another finding in our experiment is that the optimal point appears at the same spot even though we only use a small portion of data to train the neural network, which helps us use this method more efficiently. For MNIST dataset, we perform three sets of same experiments with 3000, 10000 and 60000 training samples respectively. For CIFAR, we do the same with 2000, 10000, 50000 images. The result is shown in figure 2.3. For the same dataset and topology, we can see that the the peak appears at the same number of neighbors no matter whether we train with a small portion of data or the whole set of data. This result shows that, when this method is used in practice, we can find out where the optimal point is with only a small set of data. With the

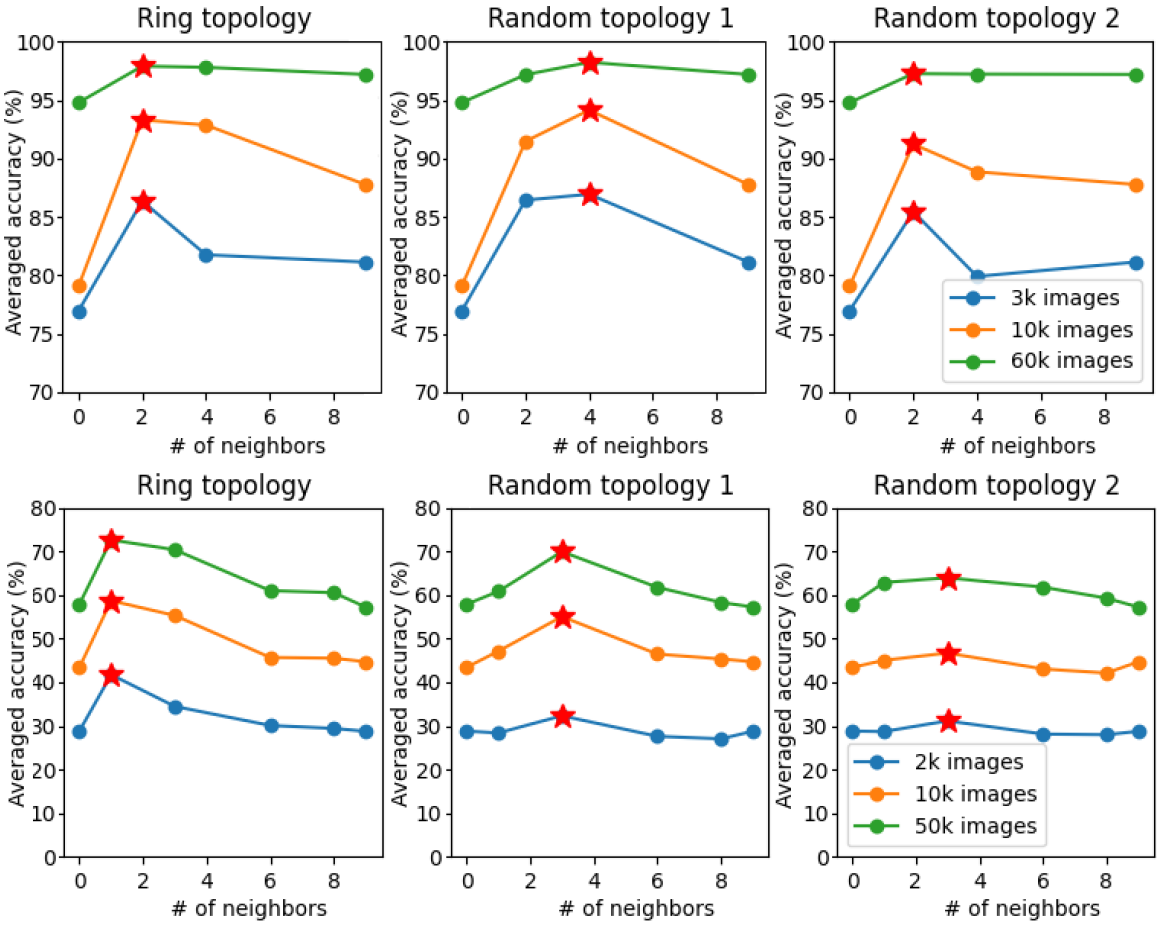

Figure 2.3: Average accuracy vs different number of neighbors for different topologies. MNIST (top) and CIFAR (bottom).

answer in mind, the rest of the training will be done at that optimal point and we can get the best performance more efficiently.

Comparison to centralized FL

To compare the efficiency of our method in decentralized and centralized FL, we conducted experiments in centralized FL with similar settings in random topology 1. The algorithm used for centralized FL in this experiment is in [4] except that one of the ten clients in the network now acts as the central server to perform model aggregation.

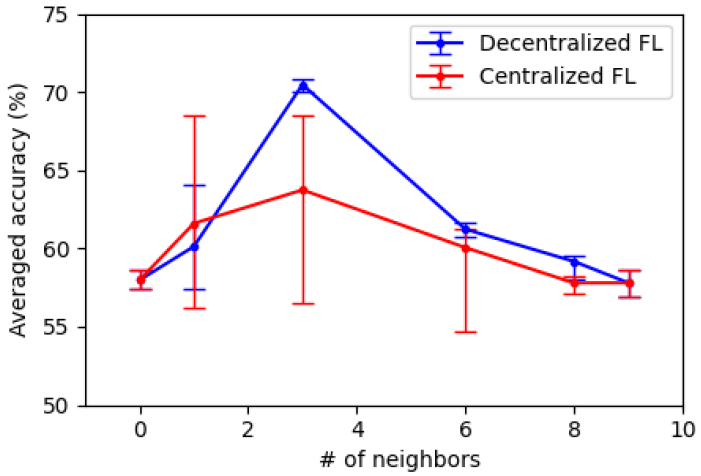

Figure 2.4: Comparison of centralized and decentralized FL in random topology 1.

The central client is the node that can reach the most clients within a specified range. Results are shown in figure 2.4 that the overall performance in accuracy in centralized FL is worse than that in decentralized FL. Under our proposed method, both cases have the same optimal strategy of how many neighbors to share the partitioned model with, but in centralized FL, the accuracy is 6.2% lower than in decentralized FL. The error bar shows us the variability of the ten clients' accuracies. The variation in centralized FL is significantly larger. This is because, in a centralized setting, some clients are far from the center and have limited energy budgets, preventing them from sending their models to the center or receiving the aggregated model. However, the clients that can participate in the aggregation process achieve a similar accuracy to those in decentralized FL.

2.4 Chapter Conclusion

In this chapter, we introduced model partitioning and proposed a method for decentralized network system that finds the optimal model sharing strategy for all clients in the network. Each client has an energy budget for model transmission; therefore, the farther a client is from the center, the less information it can transmit about the ML model. We performed simulations with PyTorch and proved that this method can be used to find the optimal strategy given the network topology and ML task. More importantly, we also found that the optimal strategy can be found using only a small portion of the training dataset, allowing us to apply this strategy during the training of the remaining dataset to achieve the highest accuracy. In the next chapter, we will delve deeper into addressing the research gaps in centralized FL for IoT systems with energy-constrained clients.

Chapter 3

DRL-Empowered FL for Energy-Constrained Wireless Clients

Building upon the insights gained in our previous work in Chapter 2, in this chapter, we consider a novel application of DDPG for optimizing bandwidth allocation in FL utilizing model partitioning, with an emphasis on energy-constrained clients such as cellphones and personal wireless devices. We highlight a specific use case where multiple cellphone cameras in a public area contribute to a global task, such as assessing crowd density or monitoring busy locations, showcasing the practical implications of our research in wireless communications and networking. To the best of our knowledge, no previous works have considered using DDPG to improve FL for bandwidth and energy-constrained wireless clients. Our experimental results indicate a marked improvement in managing bandwidth efficiently. This research aims to bridge the

gap in current FL practices by providing a viable solution for bandwidth management and model partitioning, a crucial step towards realizing the full potential of FL in resource-limited environments.

3.1 System Model

We consider an FL system shown in figure 1.1(a) where K energy-constrained wireless client devices, collectively denoted by $\mathcal{K} = \{1, 2, ..., K\}$ and a central server participate in FL. It is assumed that the locations of these devices and their distances to the server are fixed and known throughout the entire learning epoch. Each device possesses a local dataset $\mathcal{D}_k$, $k \in \mathcal{K}$, and all devices work together with the central server to train an ML model by exchanging their model weights w_k without sharing data samples directly.

3.1.1 Energy Constraints

Similar to 2.1.1, we assume that energy at a wireless client is used primarily for information transmission and for local ML training. T denotes the total number of rounds of FL aggregation during training. We consider orthogonal multiple access techniques with B_k^t being the allocated bandwidth at round t for client k for local model transmission to the server, and the total uplink bandwidth available is B_{max} for each round.

Computing Energy

Let f_k be the CPU frequency of client device k. According to Lemma 1 in [35], the energy consumption required for client k to perform one FL training round using their dataset $\mathcal{D}_k$ is given by

$$E_k^c = \kappa |\mathcal{D}_k| \mathcal{L}_k f_k^2, \tag{3.1.1}$$

where κ is the effective capacitance coefficient of the CPU, and $\mathcal{L}_k$ is the number of required CPU cycles to process one data sample.

Transmission Energy

Let p_k be the transmit power (in Watts) of client k, and h_k be the channel gain between client k and the server located at a distance d_k away. Assuming line-of-sight free-space propagation $h_k \propto d_k^{-2}$[36], i.e., $h_k = \mu_k / d_k^2$ where μ_k is assumed to be known (e.g. through pilot symbols), the achievable transmission rate R_k can be expressed according to Shannon's formula as

$$R_k = B_k \log_2 \left(1 + \frac{p_k h_k}{N_0 B_k}\right). \tag{3.1.2}$$

where N_0 is the power spectral density of the complex baseband white Gaussian noise and B_k is the bandwidth allocated to client k. From (2.1.1), the transmit power can be expressed as

$$p_k = \frac{N_0 B_k}{h_k} \left(2^{\frac{R_k}{B_k}} - 1\right). \tag{3.1.3}$$

We assume a model partitioning approach where α_k denotes the fraction of model weights client k is able to upload to the server at the end of the local training round.

As such, the transmission time for model uploading is given by

$$T_k^U = \frac{\alpha_k z}{R_k}, \tag{3.1.4}$$

where z is the number of bits for the entire trained model w. The corresponding energy consumption for the uplink transmission is

$$\begin{aligned} E_k^U(\alpha_k, B_k) &= p_k T_k^U \\ &= \frac{p_k \alpha_k z}{B_k \log_2\left(1 + \frac{p_k h_k}{N_0 B_k}\right)}. \end{aligned} \tag{3.1.5}$$

Finally, we can derive the relationship between the fraction of the uploaded model weights α_k and the allocated bandwidth B_k as

$$\alpha_k = \frac{E_{k\max}^U B_k}{p_k z} \log_2\left(1 + \frac{p_k h_k}{N_0 B_k}\right), \tag{3.1.6}$$

where $E_{k\max}^U = E_{k\max} - E_k^c$ is the maximum available energy for model transmission with an energy budget of $E_{k\max}$.

Equation (3.1.6) shows that the fraction of information that can be shared, α_k, monotonically increases with B_k. In other words, for a particular energy-constrained client, having more bandwidth allocated will necessarily increase the amount of information that it shares to the server. However, since the total bandwidth is limited, the server will need to decide on how to allocate the bandwidth to each client based on their model weight contribution to the accuracy of the global model, their maximum available energy for model uploading, and their respective distances from the server.

3.1.2 Model Partition

The model partitioning method [19] we used is similar to 2.1.2. After each client trains its local model, we consider model partitioning approximately corresponding

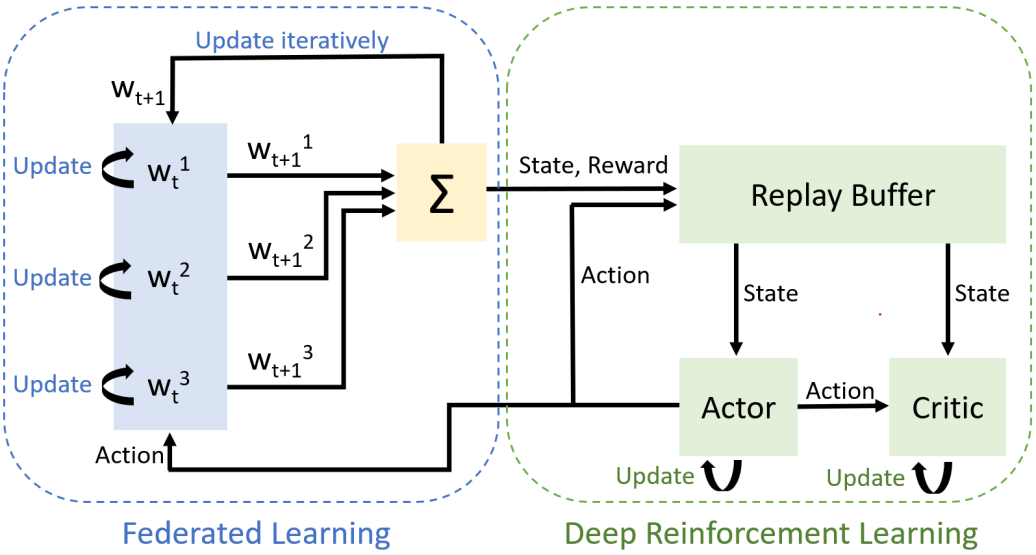

Figure 3.1: Proposed framework: adding a DRL agent at central server

to allow the client to divide its model weights into two partitions according to the allocated bandwidth. The client will then send the first partition (i.e., the layers closest to the input layer) to the server. The number of layers shared is denoted by $n \in \mathbb{Z}_+$, where n is the largest number of NN layers that can accommodate the given fraction α_k of the model weights.

3.2 Proposed Method

Based on the relationship between α_k and B_k derived in (3.1.6), we proceed to find the optimal bandwidth allocation B_k^t for each client k in each FL training round t. In our proposed framework, a DRL agent is employed at the FL server to calculate the optimal bandwidth allocation. We apply the DDPG algorithm as the DRL method because the actions are continuous variables, which may be directly obtained from the continuous output of the actor network.

Algorithm 2: DDPG-assisted FL

Input: Randomly initialize FL NN $\boldsymbol{w}_0$ and send to all clients.
Randomly initialize: actor network parameters θ^{μ}, critic network parameters θ^{Q}, target network parameters $\theta^{\mu'} \leftarrow \theta^{\mu}$ and $\theta^{Q'} \leftarrow \theta^{Q}$.
Initialize replay buffer $\mathcal{R}$

Output: Trained target network parameters $\theta^{\mu'}$ and $\theta^{Q'}$.

for *Round t = 1,2,3...T* **do**
  for *Each client k* **do**
    $\boldsymbol{w}_{t+1}^{k} \leftarrow$ **ClientUpdate**$(\mathcal{D}_k, \boldsymbol{w}_t)$
    Send partial $\boldsymbol{w}_{t+1}^{k}$ to the server
  end
  Observe state $\boldsymbol{s}_{t+1} = (\boldsymbol{w}_{t+1}^{1}, \boldsymbol{w}_{t+1}^{2}, \ldots, \boldsymbol{w}_{t+1}^{K})$
  Obtain reward $r_t = Acc_{t+1} - Acc_t$
  Store transition $\mathcal{T}_t = (\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ in $\mathcal{R}$
  Select action $\boldsymbol{a}_{t+1} = \mu(\boldsymbol{s}_{t+1}|\theta^{\mu}) + N_t$ according to the current policy and exploration noise
  Average all weights $\boldsymbol{w}_{t+1} = 1/K \sum_{k=1}^{K} \boldsymbol{w}_{t+1}^{k}$ Send w_{t+1} and $\boldsymbol{a}_{t+1}$ to clients
  Sample a random minibatch of N transitions $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ from $\mathcal{R}$
  Set $y_i = r_i + \gamma Q(\boldsymbol{s}_{i+1}, \mu'(\boldsymbol{s}_{i+1}|\theta^{\mu'})|\theta^{Q'})$ Compute the target value
  Update the critic network by minimizing the loss:
    $L = (1/N) \sum (y_i - Q(\boldsymbol{s}_i, a_i|\theta^{Q}))^2$
  Update the actor policy using the sampled policy gradient:
    $\nabla_{\theta^{\mu}} J \approx (1/N) \sum \nabla_a Q(\boldsymbol{s}, a|\theta^{Q})|_{\boldsymbol{s}=\boldsymbol{s}_i, a=\mu(\boldsymbol{s}_i)}$
  Update the target networks:
    $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
    $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
end

Figure 3.1 illustrates our proposed framework with 3 clients in the network. The framework consists of the traditional FL processes, and a DDPG agent at the central server as the decision maker/bandwidth allocator. The detailed algorithms at the server and the client are provided in the following pseudo-code listings of Algorithm 2 and Algorithm 3.

Algorithm 3: DDPG-assisted FL

Input: $\mathcal{D}_k, \boldsymbol{w}_t$
Output: $\boldsymbol{w}_{t+1}$
$\mathcal{B} \leftarrow$ split $\mathcal{D}_k$ into batches of size B **for** *each local epoch i from 1 to E* **do**
  for *batch* $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla l(w; b)$
  end
end

FL training

Each client has its own dataset $\mathcal{D}_k$ that was collected locally and individually, and not shared with other clients. The goal for each client k is to find:

$$w_k^* = \arg\min_{w_k} f\left(\omega_k, \mathcal{D}_k\right), \quad \forall k, \tag{3.2.1}$$

where $f\left(\omega_k, \mathcal{D}_k\right)$ is the local loss function of the model for client k. Each data entry in $\mathcal{D}_k$ contains input feature variables and the ground truth output label. The model uploading energy budget, $E_{k\max}^U$, and the distance to the server, d_k, are assumed to be known for all k. The FL training is initiated by the central server and all clients know what the common training task is, what the training model structure is and what the training classes are. To focus on the impact of the device locations, we assume that the energy consumption due to local training E_k^c is the same for all clients given by (3.1.1). After each round of local training which contains some local SGD epochs, all clients will send partial model weights to the server according to their allocated bandwidth. After receiving the aggregated model from the server, they proceed with the next round of training. This process will repeat until the maximum number of rounds or the maximum time allowed is reached.

DRL training

In the implementation of DRL, we specify the following action, state, and reward utilizing only the variables that are present in FL training:

- *Action*: The DDPG agent residing at the server performs bandwidth allocation for the next round of communication of FL for each client. Thus, the agent is trained to produce the action at round t given by $\boldsymbol{a}_t = (B_t^1, B_t^2, ..., B_t^K)$.

- *State Space*: As we have K clients in the network, we define the state at communication round t as $\boldsymbol{s}_t = (\boldsymbol{w}_t^1, \boldsymbol{w}_t^2, ..., \boldsymbol{w}_t^K)$, where $\boldsymbol{w}_t^k$ is the dimensionally reduced model weight vector of client k. Given that each client might share different number of layers of its model to the central server, the DDPG agent will only fetch the first layer as a client's state. In [30], it has been demonstrated that even reduced from 431,080 dimensions to only two dimensions using Principal Component Analysis (PCA), the model weights retain ample information necessary to distinguish itself from others.

- *Reward*: We define the reward to be the difference between the validation accuracy of the FL model at time t and at time $t-1$, $r(t) = Acc_{t+1} - Acc_t$. This value will be calculated by the central server by testing the model performance using a validation dataset that is not shared with any client. As the training proceeds, the reward will be discounted by γ so the long-term accumulative reward [20] is represented by

$$R(t) = \sum_{i=0}^{\infty} \gamma^i r(t+i). \tag{3.2.2}$$

The objective of the DRL agent is to maximize the long-term expected reward

$R(t)$,

$$\begin{aligned} \max_{\mu(.|\theta^{\mu})} \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(t+i)\right], \\ s.t. \sum_{k=1}^{K} B_k \leq B_{\max}, \\ B_k \geq B_{k\min}, \end{aligned} \tag{3.2.3}$$

where $B_{\max}$ is the maximum available total uplink bandwidth that the server can allocate to the clients, and $B_{k\min}$ is the minimum bandwidth required to transmit one layer of client k's model to the server.

During training, DDPG agent initializes the parameters of the actor and critic NNs with random values. At each round of FL, the agent obtains information about state and reward from the server, and generates the action according to $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t|\theta^{\mu}) + N_t$, where μ is the deterministic policy network that maps states to a specific action, and N_t is the exploration noise. After taking the action, the agent observes the instantaneous reward and the state at round $t+1$. The transition $\mathcal{T}_t = (\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ will be stored in the replay buffer $\mathcal{R}$, where a batch of transitions are selected and used as the training sample to optimize the parameters of the actor and critic networks. The weights of the trained actor and critic models will be stored for testing and future use.

3.3 Simulation Results

In this section, we simulate DDPG-assisted FL using PyTorch to evaluate the effectiveness of our DRL agent. Specifically, we use PyTorch to train both the actor and critic networks, optimize their parameters, and handle gradient updates. The

DRL agent is trained using the Adam optimizer with a learning rate of 0.01, and experience replay is employed to stabilize training. The objective is to reduce energy consumption and improve the accuracy of the FL task.

3.3.1 Simulation Settings

To test our proposed method in simulations, we use the public dataset MNIST [33] in FL to train a 4-layer MLP neural network with the first layer being the input layer, the second layer containing 128 neurons, third layer containing 64 neurons and the last layer performing the softmax function. Most deep learning frameworks, including TensorFlow and PyTorch, use 'float32' as the default data type for weights and biases [37]. Therefore, with 32 bits per parameter, the FL neural network's model size is $z = 5.9 \times 10^6$ bits. In the simulations, we set 10 clients in total and utilize non-IID data, where each client has training data of 2 randomly selected labels out of the 10 MNIST labels. In our FL simulation, we used batch size of 10, learning rate of 0.01, and communication rounds of 1000. To conduct DRL agent training, we repeated this training process 5 times, resulting in 1000 rounds per episode for a total of 5 episodes. Each episode is initiated by resetting the FL model weights to random values, simulating a similar reset action within the DRL environment. In the simulation, both the actor and critic networks are structured as 3-layer MLP neural networks.

Figure 3.2 illustrates an example topology used in our simulation. Ten clients are located around the central server, with locations generated randomly as a PPP, with an average density of 10 clients/area. The distance from each client to the server is used to calculate the channel gain $h_k = \mu_k / d_k^2$. To simulate the wireless

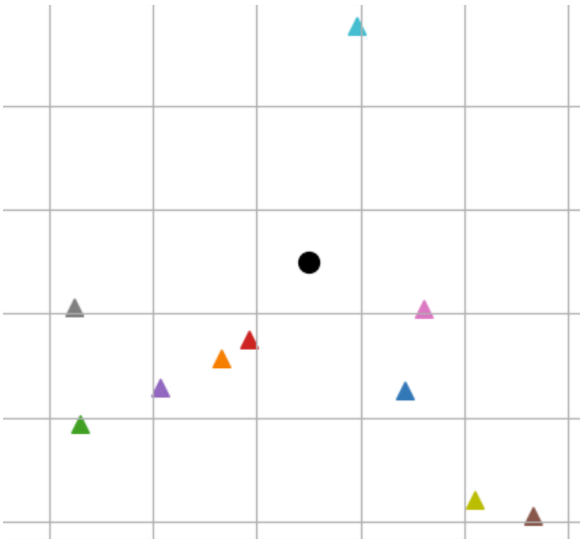

Figure 3.2: Randomly generated location of clients (triangles) around a central server (circle)

network, we consider the total bandwidth $B_{max} = 10$ MHz. The PSD of the complex white Gaussian channel noise is set as $N_0 = 10^{-12}$ W/Hz. Transmit power p_k is 0.5 W. Rayleigh fading parameter μ_k is 0.01. For each client, the transmission energy budget is set as $E^U_{kmax} = 0.15$ J.

3.3.2 Experimental Results and Analysis

During DRL training

Figure 3.3 shows the average accuracy of the FL during 5 episodes of DRL training, with 1000 FL communication rounds per episode. At the end of each episode, the FL model weights are reset to random values but all episodes contribute to the same set of actor and critic networks. The result is depicted as the curve colored in cyan in figure 3.3. We observe that during the initial episode, FL training did not reach convergence. However, as the DRL agent underwent further training, wiser choices on bandwidth

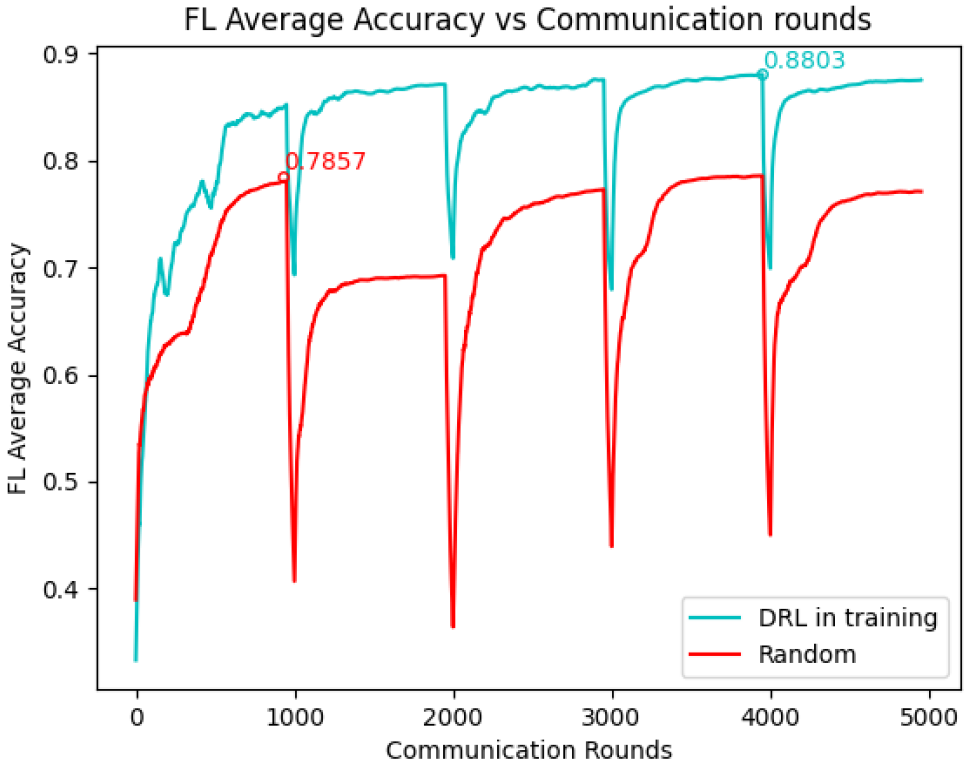

Figure 3.3: Averaged accuracy during DRL training.

allocation were made by the central server. As a result, FL began to converge starting from the second episode and achieved an accuracy of 88.03%. Moreover, there is a trend that FL converged more rapidly as DRL training progressed.

To compare this result to a baseline method, we also ran simulation with random bandwidth allocation, and the result is represented by the red-colored curve in figure 3.3. There is no DRL training in this simulation, but we reset the FL model weights every 1000 rounds for a fair comparison. We can see that, when operating without the aid of the DRL agent, FL did not demonstrate any improvement in convergence accuracy or speed, and was only able to attain a 78.57% accuracy under the same training conditions.

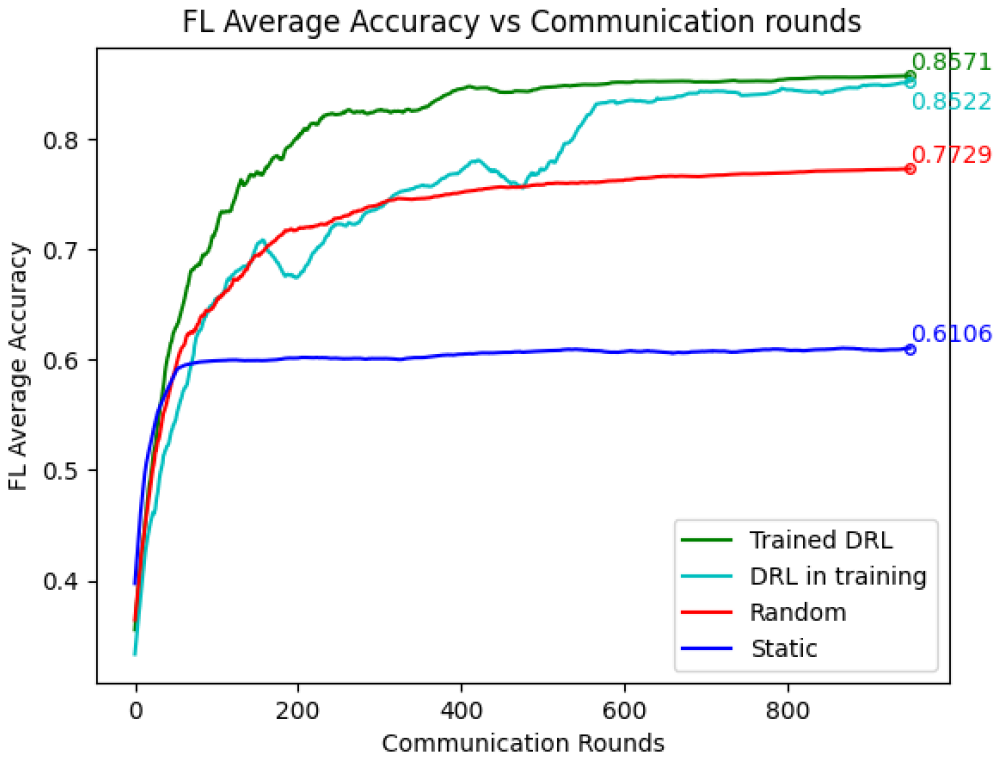

Figure 3.4: Average testing accuracy comparison of proposed method and baselines.

Testing the trained DRL

In figure 3.4, we show the average testing accuracy using the trained DRL compared with two baseline approaches of random bandwidth allocation and static bandwidth allocation for the wireless clients. For random bandwidth allocation, the server allocates bandwidth to clients randomly without the help of a DRL agent. For static bandwidth allocation, each client has the same amount of bandwidth allocated in all FL rounds. Thus the number of layers transmitted by client k will be the same throughout the FL training. The figure shows that the trained DRL agent gives the best FL performance in terms of convergence speed and testing accuracy. Comparing

the trained DRL (green) and DRL in training (cyan), there is an obvious improvement in convergence speed, which means our DRL training is effective and improves over time. Comparing our method to the baselines of random and static bandwidth allocation, it is evident that the proposed DRL-assisted FL outperforms and yields accuracy improvements of 8.43% and 24.65% respectively. Besides, our approach only spent less than half of the communication rounds to reach the same accuracy as the random bandwidth allocation baseline, equivalently saving half of the computation and transmission energy.

Tradeoff between performance improvement and extra cost incurred by the DRL agent

Compared to the two baselines in figure 3.4, we note that our proposed method imposes an additional cost on the central server for training the critic and actor networks, resulting in increased time and computational resource consumption. We assume that the central server possesses significantly greater computational power in contrast to the energy-constrained IoT devices acting as clients. Hence, our analysis will concentrate on the additional time expenditure involved.

In figure 3.3, we see that it takes 5000 FL rounds to train the DRL agent, serving as a preliminary step before the actual task. Nonetheless, this DRL training period can be readily shortened by employing the early stop technique, allowing each episode to conclude once FL converges. In our scenario, we can reset the environment at rounds 1000, 547, 544, 672, and 351 in each episode, effectively reducing the required DRL training rounds to 3114 instead of the original 5000, while still achieving the same outcome.

Furthermore, we observe in figure 3.4 that further time savings could be achieved by using the DRL in training instead of the trained DRL. Comparing the curves of DRL in training (cyan) and random (red) or static (blue), we observed that our method already outperformed the baselines during DRL training phase after 400 communication rounds. As such, we do not necessarily need to complete the entire DRL training to achieve an improved result. However, if we choose to complete the DRL training, we can achieve even better results in terms of convergence speed.

Given that the neural networks used for both actor and critic networks are relatively small in size and simple in structure, there was no noticeable time difference when running simulations with DRL-assisted FL or FL with random bandwidth allocation. Additionally, since the training of actor and critic networks occurs in parallel with the FL task training at clients, the time consumed for this purpose is negligible.

3.4 Chapter Conclusion

In this chapter, we introduced our proposed novel DRL-assisted FL framework for energy constrained clients, based on the model partitioning concept. Specifically, we formulate an optimization problem to improve the performance of FL training. To solve this problem, we deployed a DDPG agent at the central server to help the server perform bandwidth allocation tasks. We presented the algorithm for our proposed method, and our experimental results demonstrated that it outperforms the baselines by improving the accuracy of FL training and reducing the energy consumption.

Chapter 4

Conclusions and Future Work

4.1 Summary

This thesis provides a thorough examination of the challenges associated with implementing FL in wireless networks that have constraints on energy and bandwidth. It utilizes model partitioning during data transmission, which reduces communication overhead and conserves energy. Additionally, it integrates DRL into the FL system, offering a smarter approach to bandwidth allocation. Ultimately, this study applies these findings to a simulated real-world IoT system, demonstrating how innovative communication techniques and FL can foster a balanced and efficient IoT network under energy constraints. This thesis introduces a novel method for managing bandwidth and model partitioning, which is vital for fully realizing the potential of FL in environments limited by resources.

Chapter 2 introduces a new method to balance the amount of model information shared with the energy used for transmission in decentralized Federated Learning. This chapter aims to optimize the partition size and transmission range. We have introduced model partitioning and proposed a method for a decentralized network system that identifies the optimal model-sharing strategy for all clients in the network.

We conducted simulations using PyTorch and proved that this method can determine the optimal strategy based on the network topology and the ML task. Importantly, we also discovered that the optimal strategy can be determined using only a small portion of the training dataset, allowing us to apply this optimal strategy during the remainder of the training to achieve the highest accuracy. The next chapter will explore further research gaps in centralized FL for IoT systems with energy-constrained clients.

Chapter 3 introduces a novel approach to perform bandwidth allocation for FL using a DRL agent at the central server. In this chapter, we aim to reduce transmission overhead and improve the global model's accuracy. We introduced our novel DRL-assisted FL framework for clients with energy constraints, based on the concept of model partitioning. Specifically, we formulated an optimization problem to enhance the performance of FL training. To address this problem, we deployed a DDPG agent at the central server to assist in bandwidth allocation tasks. We presented the algorithm for our proposed method, and our experimental results demonstrated that it outperforms the baselines by improving the accuracy of FL training and reducing energy consumption.

4.2 Future Work

For future work, several intriguing dimensions can be explored to enhance the applicability and robustness of FL in decentralized environments. First, an in-depth investigation into how varying network topologies influence the strategies adopted in decentralized FL is essential. Different network structures, from highly interconnected meshes to sparse trees, may significantly impact both the efficacy and efficiency

of model-sharing and learning outcomes. Additionally, the integration of incentive mechanisms designed to encourage participation of high-quality users while deterring malicious behavior is crucial. Developing economic or reputation-based incentives can help ensure that only reliable and beneficial participants contribute to the learning process, enhancing the overall system integrity and performance.

Expanding the scope to more dynamic network settings is also vital. Future research should consider scenarios where clients are not only mobile and wireless but also display significant diversity in device types, computational capacities, and energy constraints. Moreover, these clients might be tasked with handling various computational tasks of differing complexity levels. This heterogeneity adds layers of complexity in managing resources and scheduling tasks efficiently.

To address these complexities, one could explore adaptive algorithms that adjust learning and communication parameters in real-time based on the current state of the network and devices. Such algorithms could dynamically balance the trade-offs between learning performance and resource consumption, thereby optimizing both individual and collective outcomes.

By pursuing these lines of inquiry, we aim to bridge the gap between theoretical advancements in FL and their practical implementations, ensuring that FL systems are both robust and adaptable to the ever-evolving landscape of networked devices and applications.

Bibliography

[1] R. Garrido-Pelaz, L. Gozalez-Manzano, and S. Pastrana, "Shall we collaborate? a model to analyse the benefits of information sharing," 2016.

[2] M. Goddard, "The eu general data protection regulation (gdpr): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017. [Online]. Available: https://doi.org/10.2501/IJMR-2017-050

[3] G. Aceto, V. Persico, and A. Pescapé, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Commun. Surveys Tutorials*, vol. 21, no. 4, pp. 3467–3501, 2019.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017.

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[6] H. B. McMahan, E. Moore, and D. R. et al., "Communication-efficient learning of deep networks from decentralized data," 2023.

[7] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," 2017.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[9] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," 2019.

[10] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2017, pp. 3–18. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2017.41

[11] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5336–5346.

[12] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLchain: A blockchain for auditable federated learning with trust and incentive," in *2019 5th Int. Conf. Big Data Computing and Commun. (BIGCOM)*, pp. 151–159.

[13] Z. Lian and C. Su, "Decentralized federated learning for internet of things anomaly detection," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1249–1251. [Online]. Available: https://doi.org/10.1145/3488932.3527285

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[15] J. Konečný, H. B. McMahan, F. X. Yu, and P. R. et al., "Federated learning: Strategies for improving communication efficiency," 2017.

[16] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent iot applications: A cloud-edge based framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, 2020.

[17] Y. Mei, B. Guo, D. Xiao, and W. Wu, "Fedvf: Personalized federated learning based on layer-wise parameter updates with variable frequency," in *2021 IEEE Int. Performance, Computing, and Commun. Conf. (IPCCC)*, 2021, pp. 1–9.

[18] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Delay minimization for federated learning over wireless communication networks," 2020.

[19] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, and Y. T. et al., "Continuous control with deep reinforcement learning," 2019.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. 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, pp. 529–533, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:205242740

[23] S. Wang, M. Lee, and S. H. et al., "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *IEEE Conf. on Computer Commun. (INFOCOM)*, 2021, pp. 1–10.

[24] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Commun.*, vol. 39, no. 12, pp. 3606–3621, 2021.

[25] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188–1200, 2021.

[26] K. Y. et al., "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, 2020.

[27] S. U. Stich, "Local sgd converges fast and communicates little," 2019.

[28] K. Wang, W. Liu, and T. J. Lim, "Deep reinforcement learning for joint sensor scheduling and power allocation under dos attack," in *ICC 2022 - IEEE Int. Conference on Commun.*, 2022, pp. 1968–1973.

[29] H. Ko, J. Lee, S. Seo, S. Pack, and V. C. M. Leung, "Joint client selection and bandwidth allocation algorithm for federated learning," *IEEE Trans. on Mobile Computing*, vol. 22, no. 6, pp. 3380–3390, 2023.

[30] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE Conf. on Computer Commun. (INFOCOM)*, 2020, pp. 1698–1707.

[31] T. Zhang, K.-Y. Lam, J. Zhao, and J. Feng, "Joint device scheduling and bandwidth allocation for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.

[32] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *2019 Sixth Indian Control Conf. (ICC)*, pp. 299–300.

[33] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[34] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," *[ONLINE] Available: https://www.cs.toronto.edu/ kriz/.*

[35] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Commun.*, pp. 3590–3605, 2016.

[36] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[37] PyTorch, "torch.set_default_dtype," https://pytorch.org/docs/stable/generated/torch.set_default_dtype.html, last accessed on February 2024.