TOWARDS EFFICIENT AND PERSONALIZED COLLABORATIVE EDGE LEARNING ON HETEROGENEOUS ENVIRONMENT

TAO GUO

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Computing

Towards Efficient and Personalized Collaborative Edge Learning on Heterogeneous Environment

Tao Guo

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

Jun 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

____________________________ (Signed)

Tao Guo (Name of student)

Abstract

With the development of artificial intelligence and the corresponding application by-products, Internet of Things (IoT) devices, e.g, smart watches, cell phones has engaged in people's daily life. Such proliferation has resulted in the silos and isolation of local data. To make the full use of the long range personal data and ensure data privacy, a new computing paradigm arised, ,i.e, collaborative edge computing. Collaborative edge computing allows training a large neural model over a wide range based on their isolated datasets, e.g, Federated Learning (FL) and Split Learning (SL).

As the diversified IoT edge devices prospered and thrived in recent years, several challenges present in today's collaborative training process, including data heterogeneity, model heterogeneity and resource heterogeneity. Furthermore, as the model sizes become larger, foundation models, e.g, BERT, DALL-E, GPT-3, emerges as an assistant to the adaptation of of a wide range of downstream tasks. However, existing framework e.g, FL and SL, can not fully satisfy the heterogeneous requirements and keep up with the state-of-the-art models. Thus, there is a need to explore efficient collaborative edge learning frameworks for better performance. In this thesis, we explore novel frameworks and propose efficient and effective method to address the heterogeneous challenges above.

First, we focus on tackling the model and resource heterogeneity across different IoT devices. Existing FL requires all users to store the entire model locally and employs an iterative update mechanism by exchanging model parameters repeatedly with the server. Such behavior has high demand for local computation and memory capabilities and can cause significant communication overhead. SL on the

other hand address the resource by reallocating the most of the neural networks on the server, hence alleviate the majority of computation burden. However, the inherent training mechanism of SL, i.e, sequential training order between the edge and cloud, impede the optimal training efficiency. Thus, we come up with a novel collaborative learning framework, i.e, Tree Learning, to optimize the training efficiency across clients. Specifically, We allocate different layers for heterogeneous clients according to their different computation capacities and successfully facilitate all the participants to achieve the minimum synchronization overhead via a global level parallelism scheme. We further provide rigorous theoretical analysis of our framework and conduct extensive experiments across various datasets to validate the effectiveness.

Second, we propose an innovate paradigm in FL to solve the existing challenges from a different perspective. Artificial intelligence (AI) nowadays has shown its success to train on broad data and produce large pretrained models (e.g., BERT, DALL-E, GPT-3) that can help human with timely and properly decisions. Recently, a paradigm shift arise when the pretrained models are utilized to adapt to the downstream tasks. And here we rename the aforementioned models the foundation models (FM). Inspired by the adaptation of FM in the centralized manners, we revisits the question of how FL mines the distributed data in iterative training rounds, and exploit the emerging foundation model (FM) to optimize the FL training. We propose PromptFL, other than training the whole model parameters, our framework works with the prompt vectors instead. Specifically, FL clients train prompts instead of a model, which can simultaneously exploit the insufficient local data and reduce the aggregation overhead. Experiments show the superiority of PromptFL from system feasibility, model performance and privacy preserving.

Third, we consider to address the statistical heterogeneity in existing PromptFL to achieve a better personalization for local user modeling. Given the lightweight nature of prompt learning, researchers have migrated the paradigm from centralized to decentralized system to innovate the collaborative training framework of Federated Learning (FL), which we called PromptFL. However, current PromptFL mainly

focuses on modeling user consensus and neglects the adaptation of local edge devices, leaving the personalization of PromptFL largely under-explored. Here we leverage the the unique advantage of multimodality in vision-language models by learning user consensus from linguistic space and adapting to user characteristics in visual space collaboratively. We also survey the personalization techniques in traditional pFL and reform them in current PromptFL scenario. Experiments show the superiority of our *p*`FedPrompt` against the alternative approaches with robust performance.

Finally, we focus on the data utilization challenges on local client in existing PromptFL. Although PromptFL offers significant in benefiting computation, communication, and privacy over the existing frameworks, none of the researches analyze it from the data utilization manners. During the experiments, we found that federated prompting is a data-efficient but data-sensitive paradigm, and therefore, it is crucial to select data carefully for participation in the process. This work presents a local data selection strategy based on *informative vectors* that specify the most informative direction in the weight space of a vision-language model. Moving in this direction steers the behavior of pre-trained neurons precisely and improves performance on the local task. Experiments show that informative vectors offer promising robustness, making it a simple yet effective way to enhance the performance of federated prompting.

In summary, this thesis aims to design efficient and personalized collaborative framework for edge devices on the heterogeneous environment. We identify challenges in the collaborative learning for edge devices and provide solutions from different perspectives to overcome the communication, computation, statistics and resource challenges. Extensive experiments show the effectiveness of our methods.

Publications arsing from the thesis

1. **Tao Guo**, Song Guo, Feijie Wu, and Wenchao Xu, Jiewei Zhang, Qihua Zhou, Quan Chen and Weihua Zhuang, "Tree Learning: Towards Promoting Coordination in Scalable Multi-Client Training Acceleration", *IEEE Transactions on Mobile Computing (TMC)*, 2023.

2. **Tao Guo**, Song Guo, and Junxiao Wang, "*p*`FedPrompt`: Learning Personalized Prompt for Vision-Language Models in Federated Learning", *Proceedings of the ACM Web Conference 2023 (WWW '23)*, May 1–5, 2023, Austin, TX, USA.

3. Qihua Zhou, Song Guo, Yi Liu, Jie Zhang, Jiewei Zhang, **Tao Guo**, Zhenda Xu and Zhihao Qu, "Hierarchical Channel-spatial Encoding for Communication-efficient Collaborative Learning", *Advances in Neural Information Processing System 35 (NeuIPS 2022)*, Nov 28 - Dec 9, 2022, New Orleans, LA, USA

4. Qihua Zhou, Song Guo, Zhihao Qu, Jingcai Guo, Zhenda Xu, Jiewei Zhang, **Tao Guo**, Boyuan Luo and Jingren Zhou, "Octo: INT8 Training with Loss-aware Compensation and Backward Quantization for Tiny On-device Learning", *USENIX Annual Technical Conference*, July 14-16, 2021, Virtual Event.

5. **Tao Guo**, Song Guo, Junxiao Wang and Wenchao Xu, "PromptFL: Let Federated Participants Cooperatively Learn Prompts Instead of Models – Federated Learning in Age of Foundation Model", *IEEE Transactions on Mobile Computing (TMC)*, 2023.

6. **Tao Guo**, Song Guo, Junxiao Wang, Jiewei Zhang and Wenchao Xu, "Efficient Attribute Unlearning: Towards Selective Removal of Input Attributes from

Feature Representations", submitted and under review.

7. **Tao Guo**, Song Guo, and Junxiao Wang, "Explore and Cure: Unveiling Sample Effectiveness with Context-Aware Federated Prompt Tuning", submitted and under review.

Acknowledgements

Time flies, and finally we have arrived at this moment. I am honored to express my deepest gratitude and appreciation to all the people that assist me during the Ph.D. studying period.

First and foremost, I would like to thank my Ph.D. supervisor, Prof. Song Guo, for his precious guidance and full support throughout these three years journey. Prof. Song Guo is the mentor I deeply respect, not only for his remarkable academic accomplishment, but also for his conscientiousness and responsibility towards us. During the whole journey, I have benefited from his constructive criticism, incisive perspective, insightful feedback and profound expertise. Without Prof. Song Guo's meticulous guidance guidance, I can not reach this far. His invaluable guidance is a lesson that I shall not forget throughout my life.

I also would like to thank my collaborators for their support and discussions. Specially, I would to thank most Dr. Junxiao Wang, who was a postdoc when he was in the Hong Kong Polytechnic University. Dr. Junxiao Wang is a very helpful and responsible senior fellow of me, his sincere dedication to me and my work and his insightful opinion make our collaboration makes our collaboration smooth and enduring. Second, I would like to thank RAP. Wenchao Xu, another senior fellow of our group. RAP. Wenchao Xu is also very supportive and diligent during our collaboration and makes our cooperation enjoyable. Third, I would like to thank Prof. Quan Chen, and Prof. Weihua Zhuang for their help at the emergency time. Finally, I would like to thank my colleagues, Feijie Wu, Jiewei Zhang and Qihua Zhou for their contributions.

Last but not least, I would like to thank my parents, who give me all the full support and understanding for the entire time. Without their support, I could not achieve this goal and accomplishment. I am so grateful for their endless love, patience, understanding and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Collaborative edge learning enables users with personal data to learn a global or personalized model in a private and distributed way, which has permeated in to people's daily life and played a significant part in many sectors like finance, medical and education system. However, computation and communication burdens are usually the bottleneck of the collaborative learning, with the heterogeneity of local edge devices, resources and statistical challenges manifest. This thesis explores effective and efficient collaborative training framework towards the personalization on heterogeneous environment. In this chapter, we first introduce the overview of our research problems in section 1.1. Next, we describe the challenges of this research topic in section 1.2. Then, we present the sketch of our research framework in section 1.3. After that, we present the main contributions of this thesis in section 1.4. Finally, we give the organization of the thesis in section 1.5.

1.1 Overview

With the advancement of hardware technique and machine learning, diverse edge devices, e.g, laptops, cell phones and smart phones emerge and play an important part in people's daily aspects. However, as the computation and memory capability is restricted on edge, previous researchers dedicate to find a more efficient way to train

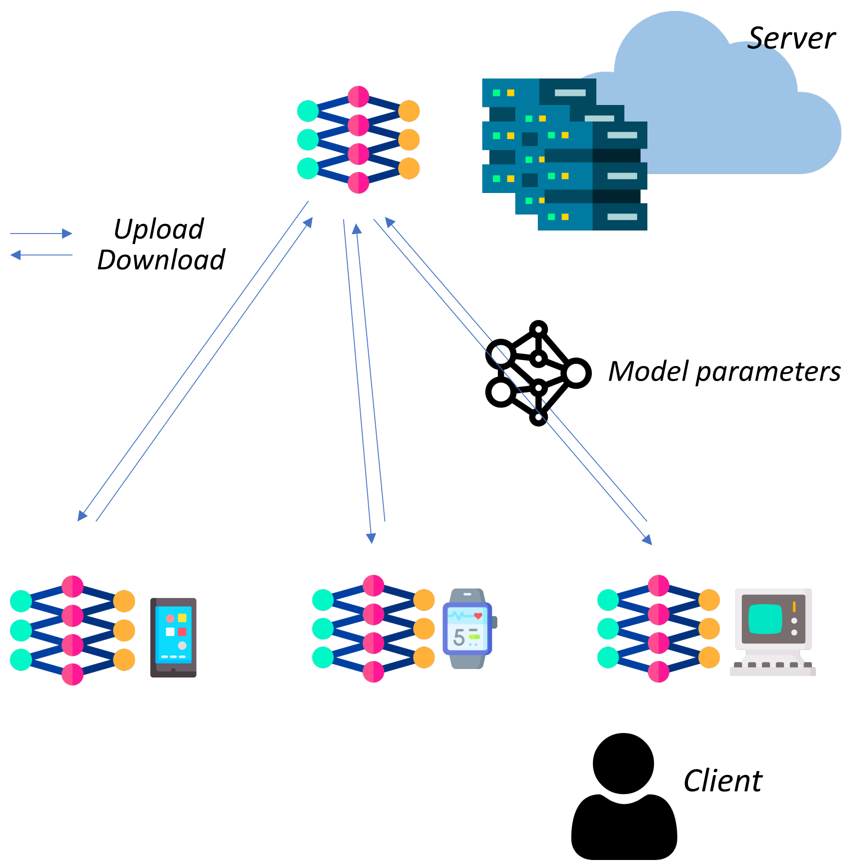

Figure 1.1: Illustration of federated learning system. In federated learning system, each client owns a complete model as the server. Each client first use their own data to train on edge for a few epochs. After that some clients are selected to upload their model parameters for aggregation. And the new model parameters will be distributed after the aggregation each round. After several rounds, the collaborative training process is complete.

for the data on edges. Driven by the development of big data, researchers have developed a collaborative training paradigm that enables local users to collaboratively learn a shared prediction model while keeping all the training data on device with the help of secure and robust cloud infrastructures. With the collaborative training, isolated data can be utilized, which is beneficial for both global administrator and local users. As time progresses, two main paradigms developed, Federated Learning (FL) [65, 74, 6] and Split Learning (SL) [45, 103, 133].

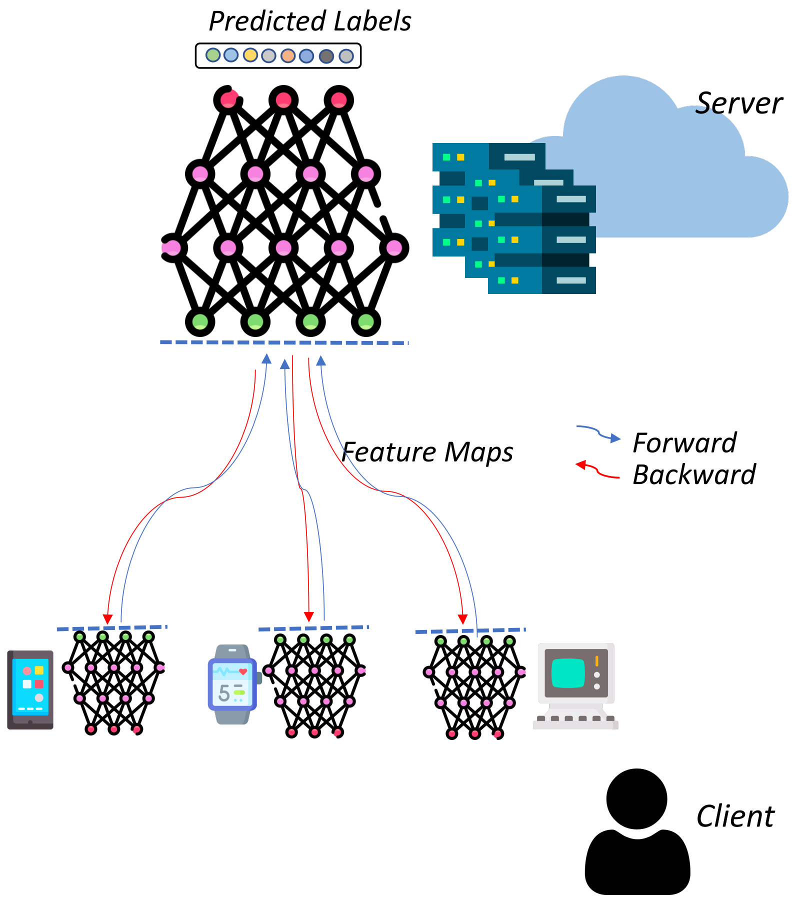

Figure 1.2: Illustration of split learning system. In split learning system, each client owns part of the neural network model, and the remaining part is placed on the cloud. During the training process, local model first perform forward propagation with the local data. After that, local client transmit feature maps to the server. The remaining model on the server then takes the feature maps as input and then continues the forward propagation. Backward propagation likewise. Combining the previous steps, constitute the whole process.

Federated learning [6, 145, 71] is a relative mature collaborative paradigm proposed by Google in recent years. Such paradigm enables multiple clients in training a shared neural network model using their local data without sharing with the server. Each client shares the same model with the server, and train the model by updating the model parameters. In each training round, all the clients first update their weights locally with their local private data, and then transmit the parameters to

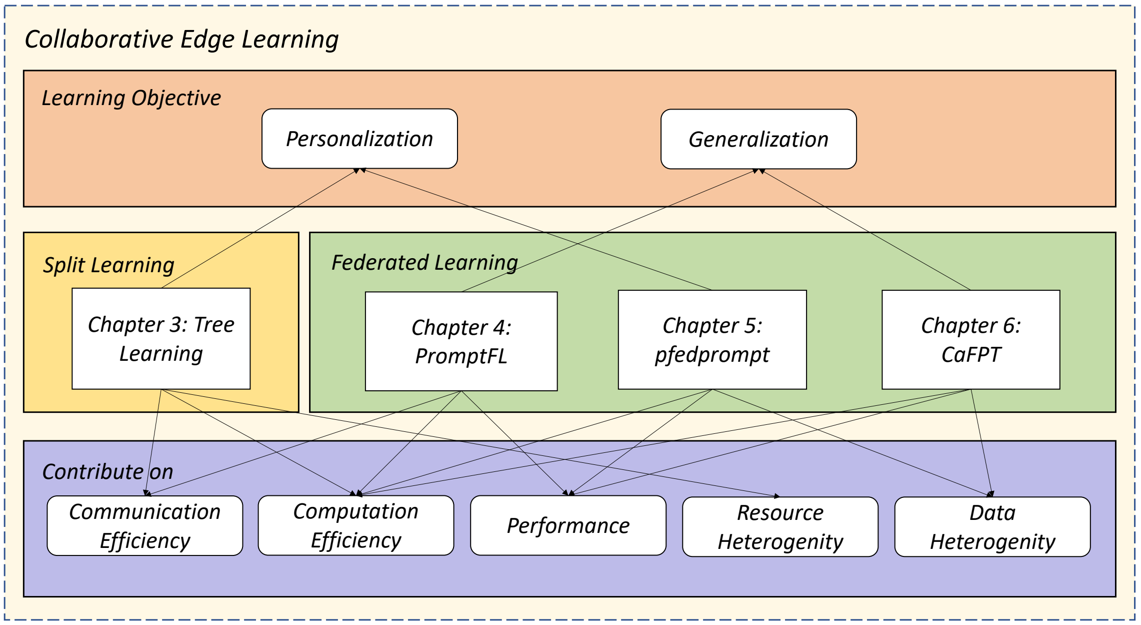

Figure 1.3: Research Framework of this thesis. We organize the positioning of this thesis within the field of collaborative edge learning and connect the learning objective and the contributions we focus on for each chapter.

the server for aggregation. After that, the averaged model will be distributed to all clients and substitute the original weights. As the model size becomes larger, not only it would be difficult to accommodate the whole model locally, the communication burden increases. We show the operation framework of federated learning in 1.1.

Unlike federated learning, split learning [45, 103, 133] splits the neural network into two parts and places each part on client and server respectively. Split learning transmits feature map instead of whole model parameters to constitute a complete feed forward process with the server. Since the client-side needs to train only parts of the model, SL can significantly reduce the computation overhead for clients during the training phase. As only the feature map needs to be transmitted to the server, the communication cost can also be significantly reduced. We show the operation framework of split learning in 1.1.

1.2 Challenges

Although federated learning and spit learning are able to provide significant benefit towards collaborative learning and edge devices, several challenges and limitations arise. First of all, collaborative edge learning requires the the communication between the edges and server. Large transmitted size results in heavy communication burden and subsequently lead to poor efficiency. Second, edge devices own limited computational resources. So it would be unbearable to train a large model locally. Third, collaborative learning leverages local data collected from distributed resources, which makes it reasonable that data on each client does not share identical distribution. Forth, different edge devices posse various hardware resource. For example, smart watch may have relative smaller computation capacity compared to the laptop. In summary, it is indispensable to explore novel personalized paradigm with high communication and computation efficiency.

1.3 Research Framework

Our thesis on the other hand aims to solve the above challenges and propose new frameworks for collaborative edge learning. The structural outline of my thesis is shown in Fig. 1.3.

As shown in Fig. 1.3, we categorize our works into two base categories in current collaborative edge learning, i.e, federated learning and split learning, and indicate the origin and mutation of which category. Further more, we assign the learning objective and the challenges that have been addressed for each work.

Tree Learning raised to deal with the resource heterogeniety of edge devices during the collaborative edge learning. For example, smart watches, cell phones and laptops may have distinct training capabilities. The aim of this work is to achieve high communication and computation efficiency for the whole framework. This work targets the personalization problem, especially obtain the model personalization on local.

PromptFL raised as the product of the time of foundation models. To better leverage the benefits of foundation models, we shift the paradigm of prompt leaning from centralized to decentralized manner. This work aims to obtain a balance between efficiency, computation and performance with the help of large vision-language models. This work brings a novel framework and perspective to the FL society improves a general ability of federated learning.

Since *PromptFL* is a newly emerged framework, many challenges regarding the paradigm is under developed. *p*`FedPrompt` was proposed to address the data heterogeneity problem under the federated setting. This work aims to achieve high local performance for each client with few computational cost. This work attains a comprehensive research to discuss the non-iid data distribution problems on different clients in federated prompting scenarios and come up with a unique solution. Thus, this work targets on the peronalization problem with the data distribution heterogenity problem.

Existing federated prompting randomly choose few samples to instruct the downstream tasks, however, noisy data may exists and be chosen. *Ensure your data for federated prompting* was proposed to filter out the bad examples and choose the representative one for federated prompting. This work aims to achieve better performance with fewer high quality data. Thus, this work targets on the generalization ability with the data quality heterogenity problem.

1.4 Thesis Contributions

We briefly summarize the contribution of this thesis as follows:

1. **Tree Learning: Towards Promoting Coordination in Scalable Multi-**

Client Training Acceleration.

To tackle the resource heterogeneity for rapidly growing edge devices, we propose an efficient collaborative training framework to promote coordination in scalable multi-client training. We adopt a tree-aggregation structure with an adaptive partition and ensemble strategy to achieve optimal synchronization and fast convergence at scale. To find the optimal split point for heterogeneous clients, we also design a novel partitioning algorithm by minimizing the idleness during communication. In addition, a parallelism paradigm is proposed to unleash the potential of optimum synchronization between the clients and server. Furthermore, we theoretically prove that our framework can achieve better convergence rate than state-of-the-art CL paradigms. We conduct extensive experiments and show that our framework is 4.6$\times$ in training speed as compared with the traditional methods, without compromising training accuracy.

2. **PromptFL: Let Federated Participants Cooperatively Learn Prompts Instead of Models — Federated Learning in Age of Foundation Model.**

   Recent FL inherently entailing numerous rounds and data for training. Such behavior suffers even more with the combined effect of a long training process and unfavorable factors such as non-IID data, limited communication bandwidth Foundation models (FMs) are large models trained on massive amounts of data in a self-supervised manner, often out-of-the-box or with minimal effort, which are largely used for downstream tasks in the centralized manner. We rethink the question of applying FMs to FL to address the above challenges and propose PromptFL, a framework that replaces existing federated model training with prompt training. PromptFL ships an off-the-shelf public CLIP to the user device and applies prompting as the adaptation technique to unleash the power of the FM in FL. We analyze the PromptFL empirically and experimentally, and show its qualifications in terms of system feasibility, privacy, and performance competitiveness.

3. ***p*FedPrompt: Learning Personalized Prompt for Vision-Language Models in Federated Learning.**

   Current prompt training in FL mainly focuses on modeling user consensus and lacks the adaptation to user characteristics, leaving the personalization of prompt largely under-explored. To address the personalization problems in federated prompting, we adapt personalized FL (*p*FL) approaches over the past few years to prompt training for heterogeneous users. Unfortunately, we find that with the variation of modality and training behavior, directly applying the *p*FL methods to prompt training leads to insufficient personalization and performance. To bridge the gap, we present *p*`FedPrompt`, which leverages the unique advantage of multimodality in vision-language models by learning user consensus from a linguistic space and adapting to user characteristics in visual space in a non-parametric manner. Through this dual collaboration, the learned prompt will be fully personalized and aligned to the user's local characteristics. We conduct extensive experiments across various datasets under the FL setting with statistical heterogeneity. The results demonstrate the superiority of our *p*`FedPrompt` against the alternative approaches with robust performance.

4. **Explore and Cure: Unveiling Sample Effectiveness with Context-Aware Federated Prompt Tuning.**

   Existing federated prompting has shown great potential in collaborative learning by offering significant benefits in computation, communication, and privacy over existing frameworks. However, existing researches overlook the internal mechanisms underlying federated prompt tuning and comply with the traditional context-unaware tuning mechanism. Our experiments, on the other hand, demonstrate that federated prompting is a data-efficient but data-sensitive paradigm, and therefore, the samples that involved in the prompt tuning process hold significant importance. To address the above issue, we propose Context-aware Federated Prompt Tuning (CaFPT), which facilitates the retrieval process by conditioning on the examples capable of activating the most pertinent knowledge inside the pre-trained models with information theory. Moving in this direction steers the behavior of pre-trained neurons precisely and improves performance on the local task. Informative vectors are built by pruning clients' training data based on their $\mathcal{V}$-*usable* information. The study shows that these vectors can be updated and combined through operations like FedAVG, and the resulting model's behavior is steered accordingly on multiple clients' tasks. Extensive experiments have demonstrated that informative vectors offer promising robustness, making it a simple yet effective way to enhance the performance of federated prompting.

1.5 Thesis Organization

The rest of the thesis consists of six chapters and is organized as follows. Chapter 2 gives a brief introduction of the background of collaborative edge learning in the heterogeneous environment and study the existing literature in this field. Chapter 3 introduces *Tree Learning*, an acceleration framework in promoting coordination in scalable multi-client training. Chapter 4 presents PromptFL, a commutation and communication efficient framework by replacing existing federated model training with prompt training. Chapter 5 discuss the the personalization of federated prompting and propose *p*`FedPrompt`, which learns user consensus from linguistic space and adapts to user characteristics in visual space to achieve personalization. Chapter 6 focuses on the data sensitivity of federated prompting and propose a data efficient approach by selecting the representative examples and specifying the most informative direction in the weight space of a vision-language model. Chapter 7 concludes the thesis and discusses future research directions.

Chapter 2

Background and Literature Review

This chapter reviews the background of current framework of collaborative edge learning. Chapter 2.1 introduces the existing framework category from macroscopic perspective. Chapter 2.2 discloses the problems in heterogeneity environment. Chapter 2.3 presents foundation models and prompt training.

2.1 Collaborative Edge Learning Framework

With the proliferation of edge devices such as smart phones and wearable-devices, as well as the increasing volume of user data, collaborative learning over multiple parties without aggregating their private data has attracted growing research attentions. There are two mainstream CL schemes, i.e., *Federated Learning* [65, 74, 6] and *Split Learning* [45, 103, 133]. In this section, we introduce two kinds of collaborative edge learning framework: 1) federated learning and 2) split learning.

2.1.1 Federated Learning

Federated Learning [6, 145, 71] is an emerging CL paradigm, which enables multiple clients in training a shared neural network model using their local data without sharing with the server. Each client owns the complete model as the one on the server. In each training round, all the clients first update their weights locally with their local private data, and then transmit the parameters to the server for aggregation. After that, the averaged model will be distributed to all clients and substitute the original weights. As the size of model increases, accommodating the whole model may be unaffordable for resource constrained devices.

2.1.2 Split Learning

Split learning is another CL method that trains machine learning models among client(s) and a server [45, 103, 133]. SL splits the neural network into two parts and places one part with fewer layers on the client side, and the remaining part on the server side. As the whole model is not shared by all entities, and the raw data can be kept locally, the user privacy can be better protected than in federated learning. During the training process, the clients first upload feature maps after local training and then aggregate them on the server for the following propagation. Since the client-side needs to train only parts of the model, SL can significantly reduce the computation overhead for clients during the training phase. As only the feature map needs to be transmitted to the server, the communication cost can also be significantly reduced.

In this thesis, we improve and reform the split learning framework in chapter 3 and revamp based on the federated learning framework in chapter 4, chapter 5 and chapter 6.

2.2 Heterogeneity on Edge

Collaborative learning enables local users to collaboratively train a neural network model without sharing their local data. However, clients are not identical and are heterogeneous in different aspects, e.g, communication bandwidth, computation capability and statistical heterogeneity. Thus, the fundamental federated or split learning framework is not applicable in the heterogeneous environment, so it is more practical to come up with more comprehensive solutions. In this section, we mainly describe two heterogeneity: 1) Resource Heterogeneity, as shown in [**?**] and 2) Statistical Heterogeneity, as shown in [**?**].

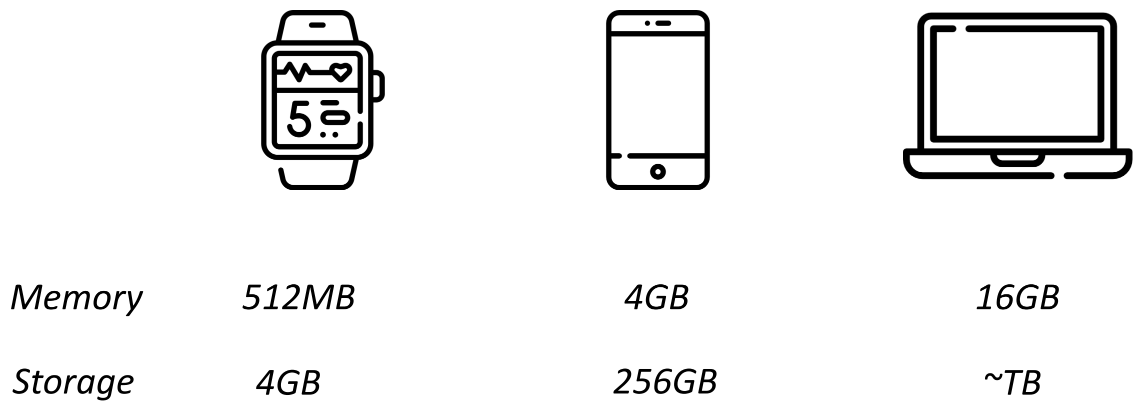

Figure 2.1: Resource heterogeneity. Computation capacity varies as the memory and storage differs among various edge devices. Common smart watches only has 4 GB storage, while that of laptop owns to the order of magnitude in terabytes. Thus, if we want to train the two tyeps of edge devices together, challenges regarding resources should be addressed.

2.2.1 Resource Heterogeneity

Artificial intelligence has gradually infiltrated into people's daily life and has engaged into the decision and routine tasks. With the combining development of both hardware and software technologies, edge devices become diverse. For example, smart watches, cell phones and laptops are the most common edge devices around us. Although collaborative edge learning allows to train neural network models jointly with their own data across distances, resource heterogeneity of the local devices may influence the model performance lies in the following aspects. 1) Computation capacities are different; Different edge devices own different memory and storage, so it would be unwise to allocate the same neural network models on different clients. Researchers achieve personalization by customizing model design for each client [4, 79, 45, 133, 128]. 2) Communication bandwidth differs; Communication

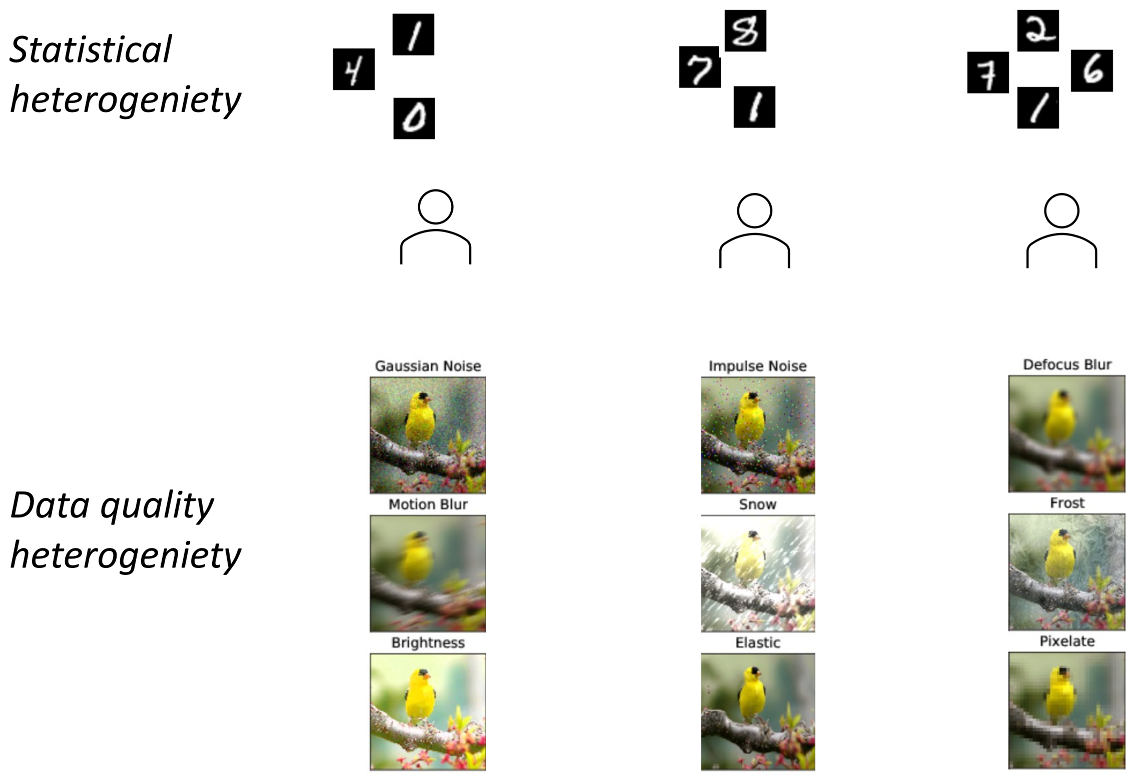

Figure 2.2: Data quality and statistical heterogeneity. The bottom of the figure shows the data quality heterogeneity on each client. Different corruption may appears on some of the data, thus, not all data are equal. The top shows the statistical heterogeneity on each client, where not all labels appears for a single client.

cost between clients and server serves as an important factor in affecting the training efficiency. Furthermore, communication bandwidth differs regarding hardware and network situations. Thus, researches dedicate to alleviate the transmission size, e.g, compression of model updates by either quantization [3, 110, 153] or sparsification [12, 85]

2.2.2 Statistical Heterogeneity

Other than resource challenges, local devices always suffer from statistical challenges since data is gathered from various origins. Data heterogeneity lies in the 1) data quality heterogeneity and 2) data distribution heterogeneity. 1) Current collaborative learning framework assumes and depends on high-quality data available on the clients. However, low-quality data can easily appears with wrong labels or blurring images. Thus, recent researchers committed to eliminate the bad influence caused by these noisy samples [131, 141, 16, 144]. 2)Stochastic gradient descent is widely used in training deep neural networks with good performance and assuming that data on clients follow the iid sampling. However, it is not always the case in practice, data on different clients may own different labels assembly or have different distribution within the same labels. Recent researches propose strategies to improve performance on non-iid data [123, 57, 94, 125].

In this thesis, we address the resource heterogeneity in chapter 3 and chapter 4 and address the statistical heterogeneity in chapter 5 and chapter 6.

2.3 Age of Foundation model

2.3.1 Foundation Model

AI is going through a paradigm shift with the rise of models (*e.g.*, BERT, GPT-3, CLIP, DALL-E·2 [25, 9, 105, 108]) trained on broad data using self-supervision at scale that can be adapted to a wide range of downstream tasks. Researchers call these models foundation models (FMs) to emphasize their key core. From a technical standpoint, FMs are not new. Foundation models have first taken shape in the field of natural language processing, which follows the idea to leverage the knowledge from the pretrained task to the another downstream tasks [25, 9]. However, the sheer size and scope of FMs over the past few years has expanded our imagination of what is possible. In terms of the representative GPT family, GPT-3 grows up to 175 billion parameters compared with 1.5 billion parameters of GPT-2. Furthermore, foundation models have developed from the language field to the vision-language field by superving the visual models from the language manner the with tremendous data pairs, i.e, CLIP and ALIGN [105, 59]. For example, the CLIP model trained on 400,000,000 labeled images. The training process took 30 days across 592 V100 GPUs. This would have cost $1,000,000 to train on AWS on-demand instances. FMs are scientifically interesting for their impressive performance and capabilities, but what makes them critical to research is that they are rapidly being integrated into real-world deployments of AI systems, with profound implications for users.

2.3.2 Prompt Training

Prompt learning has become an emerging paradigm with the rise of pretrained models, which origins from natural language processing by directly adapts the pretrained language models with 'cloze' style prompt to another task. As the development of foundation models in vision, techniques of prompt also moves from language to vision. The pretrained vision-language models like CLIP consist of an image encoder and a text encoder to predict the pairing relationship between images and texts. Therefore, these models can be converted to an image classifier. The users may convert all [class] to prompt such as "this is a photo of [class]" and predict the caption class the model estimates the best pairing with the given image. Previous research has involved prompt engineering [35, 77, 118, 146], in which human engineers or algorithms search for the best template for the classes.

Chapter 3

Tree Learning: Towards Promoting Coordination in Scalable Multi-Client Training Acceleration

3.1 Introduction

There have been growing interests regarding the collaborative machine learning paradigm, especially for mobile edge devices [114, 1, 39]. Compared with traditional centralized machine learning [22], collaborative learning (CL) [136, 147] allows training a large neural model over a wide range of clients based on their isolated datasets, e.g., Federated learning (FL) [74, 6, 145] and Split Learning (SL) [100, 134, 2], which have emerged as promising mechanisms to conduct distributed learning over multiple parties.

However, neither FL nor SL work well for resource-constrained edge devices. First, FL employs an iterative update mechanism that requires all users to update the entire model repeatedly via exchanging the parameters with the server, which can cause significant communication overhead. Second, users have to accommodate the entire model locally that requires non-negligible memory space and computing

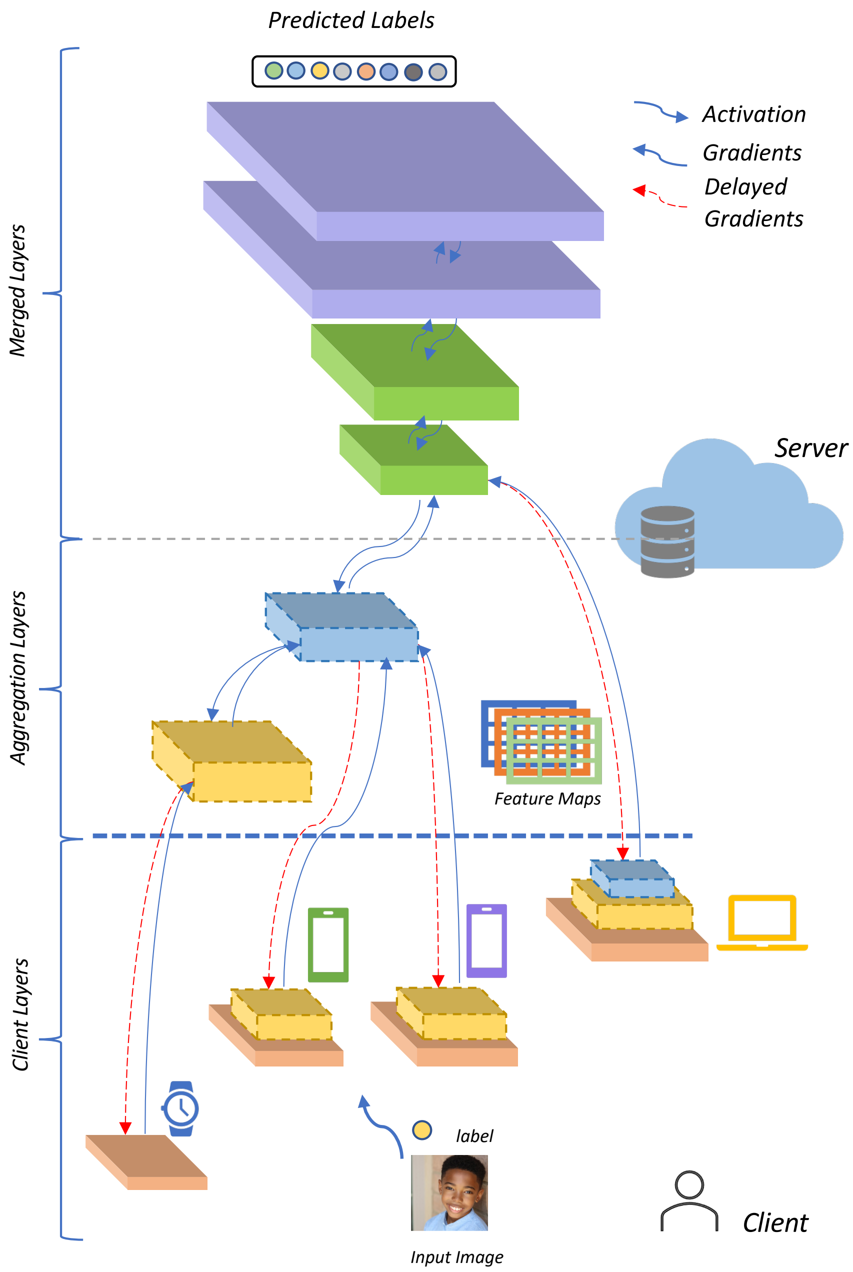

Figure 3.1: Tree-Aggregation Structure of *Tree Learning*. Three segments have been divided though the whole framework: Clients Layers, Aggregation Layers and Merged Layers. Dashed lines indicate parallelism between clients and server using delayed gradients.

capacity for local training, and thus limits the further application of FL to a wide range of resource constrained mobile devices, e.g., smart watches, cell phones, and

Internet of things, IOT devices which often have limited computing and communication resources, yet are expected to support various artificial intelligence (AI) applications [65, 74, 6]. To achieve lightweight overhead for resource constrained clients while still maintain efficient collaborative learning, existing works have proposed to improve the FL performance, e.g, compression of model updates by either quantization [3, 110, 153] or sparsification [12, 85]. However, the efficiency is improved often at the cost of compromising the model accuracy and introducing extra computing overhead.

Split learning is considered as an alternative solution for distributed and collaborative learning over edge clients. Although SL can deal with the resource scarcity problem by offloading most computing of the neural networks to the server, there are some inherent limitations. First, clients have to follow a sequential order to interact with the server in each iteration, i.e., neither elements of the network can be updated until their dependencies have been executed. Such a locking mechanism allows only one element to stay active at a time and prevents the whole system from updating the model in parallel, leading to under-utilization of the distributed computing resources. Furthermore, when aggregating multiple feature maps from different clients, the server has to wait until all of them are received before the next iteration. Efficient and effective training methods for multiple clients with limited resources are missing from status quo. [45, 133, 117].

In this paper, we present *Tree Learning*, a systematic framework that can accelerate the collaborative learning among resource constrained devices and the server by splitting the neural network model with optimal synchronization pattern and exaggerating simultaneously training with full parallelism mode. Compared with status quo CL paradigms, e.g., peer-to-peer FL or traditional SL, which may suffer from a dragged training process when clients have disparate computing capacities, *Tree Learning* employs a tree-aggregation structure to optimize the aggregation process with minimum overhead. Besides, the backward propagation is unlocked to allow simultaneous training between clients and server sides. The clients rely on delayed gradients from the previous round, instead of the one from the server-side in current round, such that the idle computing resources can be better utilized. For two typical scenarios, i.e., the horizontal mode with data samples distributed across different clients, the vertical mode, with features of data samples vertically partitioned among multiple clients [133, 134, 45], we design corresponding pipeline mechanisms to parallelize the clients and server's training process. To adapt to the various client resource levels, a dynamic partitioning algorithm is proposed to find the optimal split layer for each client considering both the clients' computing capacity and the model structure.

To the best of our knowledge, *Tree Learning* is the first framework to improve the training efficiency of collaborative training by minimizing the synchronization overhead with automatic pipeline parallelism without sacrificing any model accuracy. And we are the first to propose the hierarchical aggregation scheme to comply with the novel tree structure. The main contributions of this paper are summarized as follows:

- We propose *Tree Learning*, a system-algorithm co-design framework for collaborative training acceleration, which adopts a novel aggregation scheme with a tree structure and dynamically allocates different layers to heterogeneous clients according to their different computation capacities, via the self-adapting partition and ensemble strategy;

- We propose a general solution for this framework by selecting the optimal split layers for heterogeneous clients and successfully facilitate all the participants to achieve the minimum synchronization overhead via a global level parallelism scheme;

- We theoretically prove the effectiveness of *Tree Learning* by providing the rigorous theoretical analysis of convergence and accuracy guarantee, which shows that the training process can be enhanced without compromising the original model performance;

- We conduct extensive experiments across various datasets and models to evaluate the performance of *Tree Learning*, which shows that *Tree Learning* can achieve 4.6x in training speed as compared with existing frameworks, without compromising model performance.

3.2 Preliminaries

3.2.1 Configuration Categories

Considering practical scenarios of collaborative machine learning, there are two main configurations based on the dataset partition criterion and type, i.e., horizontal learning and vertical learning.

Traditional federated learning [74] and split learning [45] are usually referred to as horizontal learning, where each client share different identity of data samples but with complete feature space. For example, two banks from different regions may have different groups of users, but they all share the same group features, e.g., name, bank account and deposit amount, such that the collaboration of more clients will enhance the model training by providing mode isolated data samples. The training architecture is illustrated in Fig. 3.2(a).

Vertical learning [82, 14] appears later as an orthogonal complement to the traditional horizontal learning [145, 133]. In the vertical configuration, instead of owning the entire feature group, each client shares exclusive features, which is common in practical cases. Consider a hospital case where hospital A holds records of MRI scan images of patients, hospital B holds the blood test results of the same group of patients, and both hospitals have common information such as the names and social security numbers but are unwilling to share sensitive data with each other. In this scenario, collaborative training of vertical learning can help to capture the complete features of the target subject without sharing sensitive information with one another, as illustrated in Fig. 3.2(b).

3.2.2 Limitations of current collaborative learning paradigms

Current collaborative learning, i.e., FL and SL, first emerges as an innovative approach to solving the challenges of fully utilizing isolated data across spacial regions while protecting local data privacy. Though highly effective, each of the paradigm has its own limitations. The intrinsic definition of FL [111] requires that each client downloads the current model from the cloud and collaboratively learns a shared model, while keeping all the training data on devices. Such approach highly limits the potential of training a large neural network, restricting the model performance. SL, on the other hand, can subtly address the problem, by offloading the majority of model compares to the server. It preserves the former part on the client, which can achieve privacy protection to some extent by not sharing raw data while alleviating the hardware resource pressure on clients. However, it suffers from two limitations. First, due to the inherent sequential procedure of forward and backward propagation, only one party can stay active at a time. This backward locking constrains the model updates from parallelism and underutilizes the computation resources. Second, given the circumstance of disparity in hardware resources of different clients, identical model or layers located in all clients may result in a prolonged waiting time for aggregation. Thus, it is necessary to develop a new solution to address the limitations. The collaborative learning solution should aim at higher efficiency, better performance, and adaptive personalization for each client.

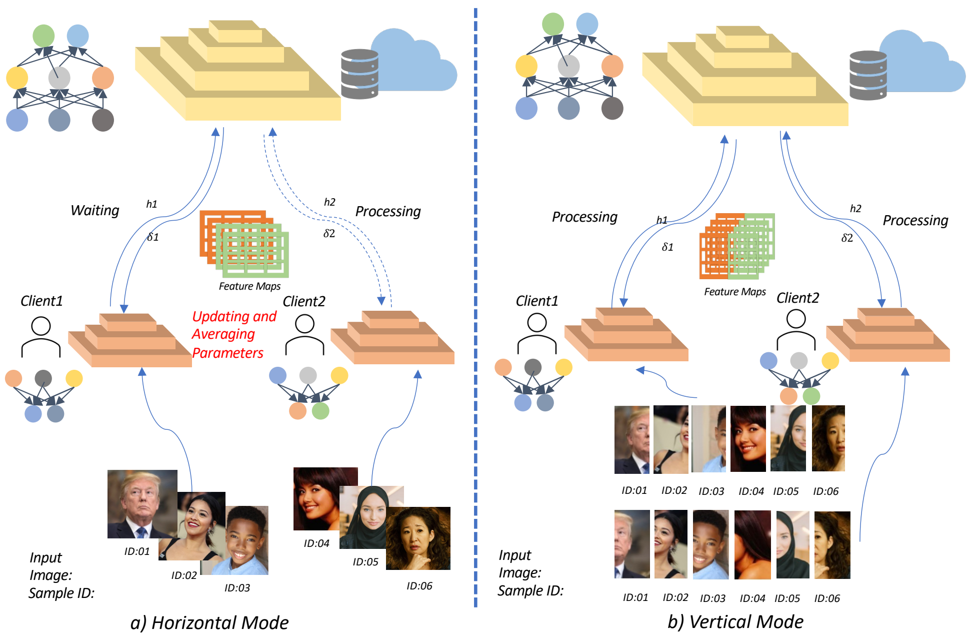

Figure 3.2: **Horizontal mode v.s. vertical mode**: Horizontal: Each client has a set of sample batches. Vertical: Each client has a unique set of features.

3.3 *Tree Learning* Design

In this section, we present the design of *Tree Learning*, to address the above-mentioned limitations of th e existing CL frameworks. First, we introduce the training scenario of *Tree Learning*, then we elaborate the design of the Tree Aggregation Scheme and Pipeline Parallelism of *Tree Learning*.

3.3.1 Training scenario

Definition 3.1 (*Tree Learning*)**.** *Consider n edge clients that collaboratively train a model with their distributed datasets. The data samples of client i is denoted by d_i, so that $D = d_1, d_2, \ldots, d_n$ denotes the total data space. The entire model, ϕ, has l layers and can be splitted into two parts with $\phi = \phi^C \circ \phi^S$, where ϕ^C denotes the part at client side and ϕ^S the part at the server side. Tree Learning can adjust the split position and thus can adaptively accommodate the client part with different layers* *according to the capacity of each client. The model from the perspective of client i is*

$$\phi_i = \phi^C_{[0-k]} \circ \phi^S_{(k-l]}, k \in (0, l),$$

where $\phi_1 = \phi_2 \cdots = \phi_i \cdots = \phi_n$ holds for the proposed Tree Learning.

3.3.2 Tree aggregation scheme

Traditional collaborative training methods have two stages in each iteration, i.e., one at the client side and the other at the server side, while the aggregation operation is required for each iteration. Such mechanism requires the server to synchronize the updates from all clients, and thus can cause an excessive waiting time if clients are of different computation capacity and communication bandwidth, i.e., a potential straggler would greatly slow down the training process. The proposed *Tree Learning* applies a novel tree-aggregation architecture with multiple meeting points during the aggregation process, which can adapt to different resource levels of the edge clients. When the feature maps reach the meeting point, they aggregate together and continue to the following layers on the server.

The structured view of the *Tree Learning* aggregation process is shown in Fig. 3.1. Due to different choices of the cut layer, n clients may have an identical or partially different model structure for the collaborative training. The entire model is partitioned into two parts, i.e., the client part and the server part. At the server side, there are two categories of layers, i.e., aggregation layers and merged layers. The training process consists of the following four steps:

- **Step 1** Profile and select the best partition layer for each client according to its computation and communication levels with the partition algorithm;
- **Step 2** Clients locally perform forward propagation with local data and send the intermediate feature maps to the server;

- **Step 3** When clients approach the first meeting point, they aggregate the feature maps and go through the following layers. After that, the feature map of the following layer will join the activation of some other clients at the next meeting point. This process will continue to the last meeting point at the bottom of the joint model of server;

- **Step 4** The server sends back and splits gradients to all individual participants and for clients' local model updating.

We integrate a pipeline parallelism for step 4 to further improve the training efficiency, as shown in the following section 3.3.3. Furthermore, to avoid the fluctuation due to huge communication jam or client disruption, we propose the Adaptive Substitute Mechanism in section 3.3.3 to address the situation. Furthermore, we can monitor and recalculate the optimal split point for given epochs with algorithm 2, according to different tasks and training status. And continue for the remaining training process. Since we use the branch-and-bound method for early exit and utilize the latest best solution for initialization, the solution space is largely reduced. Compared with the training overhead, the cost for searching the optimal split point is negligible.

3.3.3 Pipeline Parallelism

Delayed Gradients Mechanism: The chain-rule in the back propagation process of the training iteration can be the bottleneck in CL and can limit the training efficiency significantly. It is because either side can continue until the preceding layers is updated, e.g., the clients have to wait until the gradients are acquired from the server. We apply the delayed gradients policy to break the lock of backward propagation by decoupling the dependency relationship.

As shown in Fig. 3.3, after a client finishes the forward propagation and sends

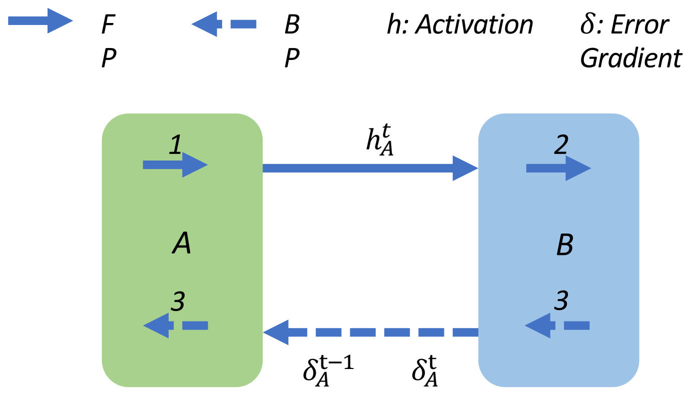

Figure 3.3: Delayed Gradients: A use the delayed gradients from last round when B performing Forward Propagation. After receiving the gradients from B in this round, A save it for the usage of next iteration.

activation to the server, it continues forward propagation on the remaining layers. In the meantime, the client executes backward propagation with delayed gradients updated from the server in the last round, instead of waiting for the gradients of this round from the server. In each iteration, a client uses the delayed gradients from the last round and saves the gradients coming around for the next round, while the server still uses the real-time data. In this way, the training efficiency is highly improved, without sacrificing accuracy. We show that the model convergence of the delayed gradients policy is theoretically guaranteed, which is given in Section 3.5.

Pipeline Parallelism: Normally, the server has abundant computation resources, while edge clients may suffer from resource insufficiency. As a result, more layers should be placed at the server side, with less layers at the client side. We discuss the optimal partition layer selection in Section 3.6.

To apply the preceding mechanism in *Tree Learning*, we illustrate the pipeline in Fig. 5.2 and Fig. 3.5. In the original case, as shown in Fig. 5.2, we present the two cases in baseline Split Learning. Specifically, (a) represents the horizontal mode where the interaction between clients and server are in the round-robin manner. (b) represents the vertical mode where features from each clients are required to wait

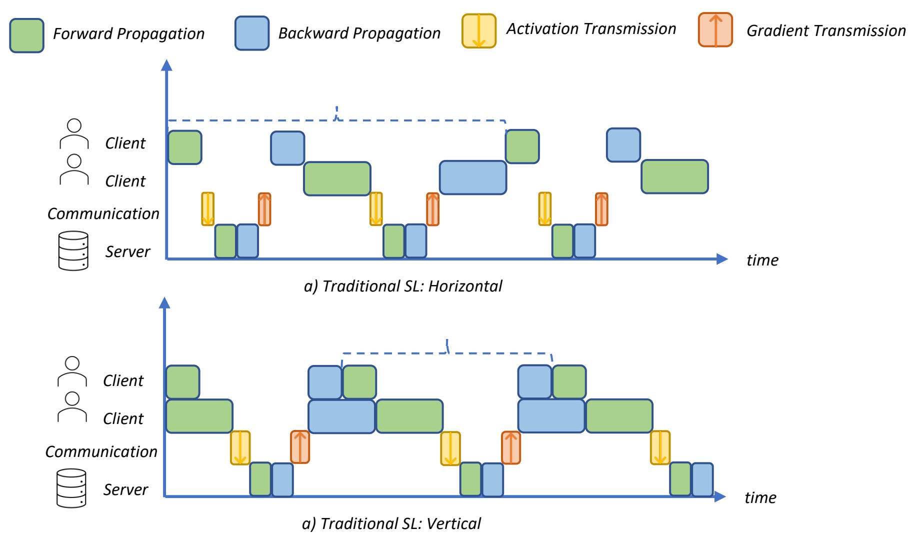

Figure 3.4: Split Learning Baseline Training: Sequential training order performing between clients and server. (a)Round-robin manner is performed in horizontal mode with clients take turns to interact with the server. (b)Server waits to receive feature maps for all clients to aggregate and then continue to perform forward and backward propagation on server.

for each other in each iteration before the aggregation process on server. In both cases, only one client or the server is activated as discussed in the previous section, due to the dependencies in backward pass. After breaking the lock of backward propagation, pipeline parallelism aims to maximize the number of active entities at any given time slot.

From the pipeline parallelism below, we enable the clients to execute forward propagation and send activation to the server. When the server receives feature maps and continues forward propagation, the clients execute back propagation with its delayed gradients at the same time, which enables the parallelism of both the server and clients.

Furthermore, the transmission of the activation and gradients leads to multiple bubbles in both client and server sides during the training. We propose to use a multi-thread method to overlap the activation transmission with backward propagation and gradient transmission with forward propagation, respectively. Fig. 3.5

shows the improvement of Tree Learning from both (a)the optimal split point effect only and (b)the overall effect with parallelism. And as shown in Fig. 3.5 (a), when the client executes forward pass, it can also receive the gradients of last iteration from the server. When the server executes backward pass, it can also receive the activation of next iteration from the client. With the overlapping of computation and communication, the time utilization of both server and client is improved.

Adaptive Substitute Mechanism During the training process, dropout of clients may happen due to communication instability or other reasons. Hence, we propose an adaptive substitute mechanism to handle the dropout situations. During each iteration between clients and server, clients locally perform forward propagation with local data and send the intermediate feature maps to server, as in Step 2. Server records the intermediate feature maps for the corresponding batch and update the value for each round. Server also maintains an average waiting time for each client by recording and updating the average waiting time each round. For each round, server will check the current waiting time. If current waiting time is larger than the current average waiting time, server will use the latest stored value to continue the following step, instead of waiting for the arrival of the new value. Otherwise, server will wait for all input to arrive.

3.4 Partition Algorithm for *Tree Learning*

In order to find the optimal spilt points in the complex structure with the interaction of tree aggregation and pipeline parallelism, we need to minimize the overall time of the entire training process across clients and the server. The synchronization procedure of *Tree Learning* is given in Algorithm 1. We have c clients $i_1, i_2, ..., i_c$, and a server, s, each client having its own dataset d_i. In each round, every client passes through its own client model M_i and sends its intermediate output h_i to the

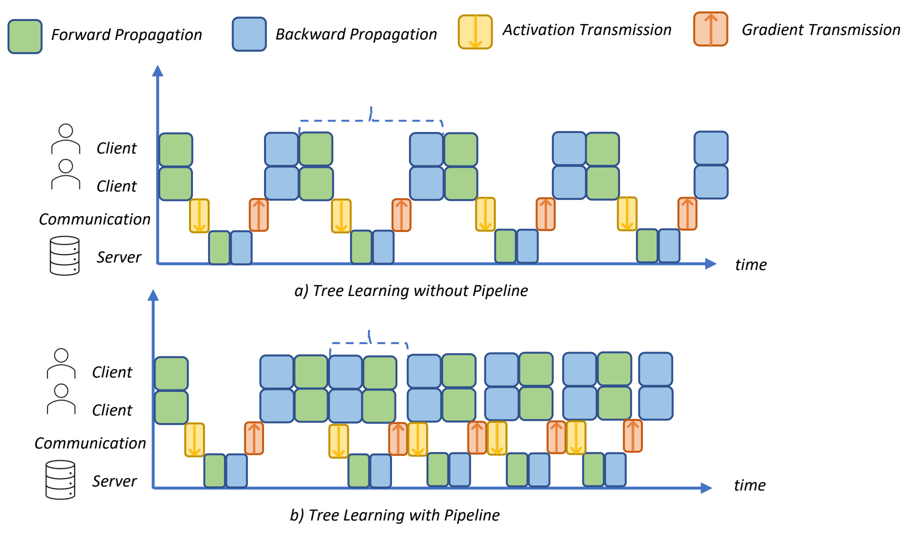

Figure 3.5: *Tree Learning* Training: Parallelism training with optimal split points selected with algorithm 1. (a)Tree Learning enabled with only optimal split point assigned. (b)Tree Learning enabled with both optimal split point as well as parallelism between clients and server.

server. After uploading the feature maps, clients can perform backward propagation and use gradients g_{t-1}^i from the last round, which realizes parallelism between clients and server. As in Fig. 3.1, the server side consists of two parts. The first part is the aggregation layers $l_k \in [l_{first}, l_{last}]$, where each layer has several inputs either from previous layers or directly from some clients with corresponding models. After aggregating all the intermediate features as input, forward propagation continues by passing through all the aggregation layers with the same manner. The second part is merged layers shown as the top component in Fig. 3.1, where forward propagation is carried out layer by layer.

Since the whole training model remains intact regardless the selection of split layers, the partition strategy has little impact on model accuracy. Thus, we can optimize the whole framework performance by improving the training efficiency and minimizing the total training time. Given heterogeneity in hardware resources of the clients, we propose a partition algorithm to assign different layers at each client in

Algorithm 1 *Synchronization Mechanism in Tree Learning.* Client model M_c, Server model M_s, Layer index i, Client index c, Intermediate output h.

```
1: initialize $\omega_c$, $\omega_s$. Stepsize sequence $\gamma_t$
2: for each round $t = 1, 2...$ do
3:     Forward propagation in each client c, get $h_c = M_c(d_c)$.
4:     Send $x^i$ to server.
5:     if round t is not first round then
6:         Use previous gradient $g^c_{t-1}$ to update parameters:
7:         $\omega^c_{t+1} \leftarrow \omega^c_t - \gamma_t \cdot g^c_{t-1}$
8:     else
9:         Use current gradient $g^c_t$ to update parameters:
10:        $\omega^c_{t+1} \leftarrow \omega^c_t - \gamma_t \cdot g^c_t$
11:    end if
12:    for each aggregation layer $k \in [k_{first}, k_{last}]$ do
13:        if aggregation layer k receives all input demand in this layers then
14:            Aggregate $h_k = concat(h^1_{k-1}, h^2_{k-1}, ...)$
15:        else
16:            Wait until all input arrive
17:        end if
18:    end for
19:    Forward propagation from $k_last$ layer to the final classification layer and
       complete $y = M_s(h_c)$
20:    Back propagation from the final classification layer to $k_{last}$ layer and backup
       the corresponding gradients:
21:    $g^c_t$.backup()
22:    for each aggregation layer $k \in [k_{last}, k_{first}]$ do
23:        Split gradients to nodes in $k-1$ layers
24:    end for
25: end for
```

order to minimize the overall training time of the overall model. We have objective function and the constrains from Eq. 3.1 to Eq. 3.6. Consequently, we can find the optimal solution to this problem using the branch and the bound method.

The objective function, T_{min}, depends on the training time from the clients to the last layer of the aggregation layers, denoted by T^{last}, and the training time after aggregation layers until the final classification layer of the server model, i.e., the merged layers, which is decided by the calculation amount of each layer Q^{last} and the calculation speed V_s of the server. $last \in [2, n]$ represents the index of last aggregation layer ranges from the second layer to the final layer, which means at least one layer should be placed on the clients. Hence, we have the **objective function**:

$$T_{min} = \min_{last \in [2,n]} (T^{last} + \frac{\sum_{last}^{N} Q^k}{V_s}) \tag{3.1}$$

where T^{last} is decided by two components, the part that directly comes from the clients and the part from the previous layers. The latter part can be represented by the minimum training time from $(last - 1)$ layer plus the time go through the *last* layer. That is,

$$T^{last} = \max(t^{last}, (\frac{Q^{last-1}}{V_s} + T^{last-1})) \tag{3.2}$$

t^{last} represents the maximum training time within c clients with $last-1$ layers which directly join the *last* layer on the server, given by

$$t^{last} = \max_{i \in [0,c]} (\frac{\sum_{1}^{last-1} Q^k}{V_i} + \frac{F^{last-1}}{B_i}) \times x_i^{last}, \tag{3.3}$$

where the first term is the computation time on clients, decided by the calculation amount of each layer Q^{last} and the calculation speed V_s of the server. And the second is the communication time between clients and the *last* layer on the server, decided by forward feature transmitted size and current communication speed; x_i^{last-1} represents whether this layer is split point or not, equal to 1 if it is a split layer and otherwise 0.

We can see that the equation forms a recursion which is defined in terms of itself or of its type. So we need a terminating scenario that does not use recursion to produce an answer. Thus, we have the **initialization** for T^2 to calculate maximum training time for the last aggregation layer on the second layer, given by

$$T^2 = t^2 = \max_{i \in [0,c]} (\frac{Q^1}{V_i} + \frac{F^1}{B_i}) \times x_i^2 \tag{3.4}$$

$$x_c^i = \begin{cases} 0 & i = 1, c \in [0, C] \\ 0 \text{ or } 1 & i > 1, c \in [0, C] \end{cases} \tag{3.5}$$

$$\sum_{i=0}^{N} x_c^i = 1, \forall c \in [0, C] \tag{3.6}$$

Let x_c^i be an indicator for each client c. If the value equals 1, the i^{th} layer is the split layer of client c. Layers below the split layer are placed on clients, while the split layer and the above layers remain on the server. Since the first layer of the model must be placed on clients, $x_c^0 = 0$. Also, as only one split point is allowed for each client, so the sum of x_c^i equals 1.

To solve the problem in (1) - (6), we apply the Branch and Bound method for various client numbers. In general the Branch and Bound algorithm is to find a value, x, that maximizes or minimizes the value of an objective function, $f(x)$, among some set of search space S. Here, we have an 0-1 integer model with variable x_c^i selected between 0 and 1, with the objective function in (1) and constraints in (5) - (6). We also set bounds for an early exit to eliminate candidate solutions on branches that cannot provide an optimal solution. Obviously, a client with weaker computation capacity cannot handle more layers than a stronger one. Thus, we accelerate the searching speed by skipping the layers lower than the split layer for the client with slower calculation speed. Also, during the searching process, node with larger training time than current optimal training time should be early exited without further growing. The algorithm of solving the optimal split partition problem is given in Alg. 2.

During the solving process, a solution space tree will be generated. Each of the tree level represents the client starting from the weakest calculation capacity to the strongest one, as shown in line 2 in Alg. 2. To begin with, we first initiate an activeset of $\{\varnothing\}$, which maintains the nodes for each level of the tree. We have C clients and $N-1$ possible split points for each client. Obviously, clients with weaker calculation capacity can not process more layers than the stronger one, thus the start point of child node can not be larger than it's predecessor, which greatly narrow down the range of possible split points for each client. As the algorithm begins, we obtain a current best solution as a bound for early exit. And as the tree grows, if the current solution results more time than the current given best time, we terminate the process and kill the node. Otherwise, we add the node into the activeset to continue the search process. Furthermore, if the node reach to the last level of the tree, i.e, split points for all clients have been explored, we will update the current best outcome and the current best solution as in line 12 to 14. Though the whole process, we manage to find the optimal solution to incur the least training time.

3.5 Theoretical Interpretation

To evaluate the efficiency of our proposed algorithm, in this section, we theoretically analyze the convergence rate of *Tree Learning*. First, we formulate the problem and define the annotations throughout the section. Next, under the assumptions made, we present the theoretical guarantee under the non-convex objective functions.

3.5.1 Preliminary

Problem Formulation In this paper, the optimization problem targets to minimize the sum of expected loss among all clients through non-convex objectives, i.e.,

$$\min_{\omega \in \mathbb{R}^d} F(\omega) = \frac{1}{C} \sum_{i=1}^{C} F_i(\omega) \tag{3.7}$$

where M refers to the number of clients, and the data among clients are heterogeneous such that $\mathbb{E}[F_i(\omega)] \neq F(\omega)$.

Algorithm 2 *Solving Optimal Partition Strategy in Tree Learning.* C clients, $N-1$ possible choices for each client. *activeset* contains nodes in the tree that are active. $time_{min}$ is the optimal training time now. *clientset* contains all clients.

```
1: Initialize $activeset = \{\varnothing\}$, $time_{min} = \infty$, $currentbest = NULL$
2: Sort from the slowest client speed to the highest client speed, each client corresponding to the level of tree from top to the bottom, $clientset_{sorted} = sort(clientset)$
3: Denote $currentbest = \{l_0, l_1, \ldots, l_C\}$ where $T_s(l_i) = T_s(l_i)$, $i \in [0, C]$
4: Let $time_{min} = time(currentbest)$
5: while $activeset$ is not empty do
6:     Choose a branching node, $k \in activeset$
7:     Remove node k from $activeset$
8:     Generate the children of node k, child $i \in [l, n]$ represents the possible partition layer for each client    ▷ Each child comes from the next element of $clientset_{sorted}$
9:     for $i = l, ..., n-1$ do
10:        if $time(current)$ is larger than $time_{min}$ then
11:            Kill the child i
12:        else if Child i is a complete solution then
13:            Update $time_{min}$ as the time for this solution
14:            Record $currentbest$ as the path to child i from root
15:        else
16:            Add child i to the $activeset$
17:        end if
18:    end for
19: end while
```

Annotations,

- $\nabla F(w)$, $\nabla_c F(w)$ and $\nabla_s F(w)$ refer to the entire, client and server part of the gradient (i.e., partial derivative of F with respect to w), respectively;
- $w_t^{(i)}$ and $w_t^{(s)}$ refer to the weights on client $i \in \{1, ...M\}$ and the server at time t, respectively;
- $\|\cdot\|_2$ indicates ℓ_2 norm of a vector.

Assumptions. The following assumptions are commonly adopted in the previous studies [78, 58] to analyze the convergence rate:

Assumption 1. *For any $i \in \{1, ..., M\}$, loss function $F_i(\cdot)$ is differentiable and with L-Lipschitz gradients, where*

$$\|\nabla F_i(w) - \nabla F_i(v)\|_2 \leqslant L \|w - v\|_2 \tag{3.8}$$

Based on Assumption 1, objective function F in Equation 3.7 is an L-smooth function. In addition, following Cauchy–Schwarz inequality, we have

$$\begin{aligned}\|\nabla F(w) - \nabla F(v)\|_2^2 &= \left\| \frac{1}{M} \sum_{i=1}^{M} (\nabla F_i(w) - \nabla F_i(v)) \right\|_2^2 \\ &\leqslant \frac{1}{M} \sum_{i=1}^{M} \|\nabla F_i(w) - \nabla F_i(v)\|_2^2 \\ &\leqslant L \|w - v\|_2^2\end{aligned} \tag{3.9}$$

Accordingly, we have the following lemma.

Lemma 3.1. *Under Assumption 1, the following inequality holds:*

$$F(w) - F(v) \leqslant \langle \nabla F(v), w - v \rangle + \frac{L}{2} \|w - v\|_2^2$$

Proof. Based on Assumption 1, we have

$$\begin{aligned}F(w) &= F(v) + \int_0^1 \frac{\partial F(v + t(w - v))}{\partial t} dt \\ &= F(v) + \int_0^1 \nabla F(v + t(w - v)) \cdot (w - v) dt \\ &= F(v) + \nabla F(v)(w - v) \\ &\quad + \int_0^1 (\nabla F(v + t(w - v)) - F(v)) \cdot (w - v) dt \\ &\leqslant F(v) + \nabla F(v)(w - v) \\ &\quad + \int_0^1 L \|t(w - v)\|_2 \|w - v\|_2 dt\end{aligned}$$

$$\leqslant F(v) + \nabla F(v)(w-v) + \frac{L}{2}\|w-v\|_2^2$$

Therefore, the inequality holds. □

Assumption 2. *There exists a scalar, $G \geqslant 0$, that is the bound of the second norm of the gradient of $F_i(\cdot)$, where*

$$\|\nabla F_i(\boldsymbol{W})\|_2^2 \leqslant G^2 \tag{3.10}$$

3.5.2 Convergence Analysis

With Algorithm 1, the following analysis is based on the hypothesis that the server does not simultaneously receive the gradients from more than one client. We construct the mathematical model in accordance with the flow of server update. When server parameters are updated from $t-1$ to t, the recursive functions for client and server are given by:

$$\begin{aligned} \textbf{Client: } \omega_t^{(i_t)} &= \omega_{j(t)}^{(i_t)} - \eta_{i_t} \nabla_c F_{i_t}\left(\omega_{j(j(t))}^{(i_t)}, \omega_{j(t)-1}^{(s)}\right) \\ \textbf{Server: } \omega_t^{(s)} &= \omega_{t-1}^{(s)} - \eta \nabla_s F_{i_t}\left(\omega_{j(t)}^{(i_t)}, \omega_{t-1}^{(s)}\right) \end{aligned} \tag{3.11}$$

where i_t indicates which client sends the feature maps to server at time $t-1$, $j(t)$ is a function indicating the last time that client i_t sends its feature maps to the server. Using $\bar{\omega}_t$ to represent the averaged client models, i.e., $\bar{\omega}_t^{(c)} = \left(\sum_{i=1}^M \omega_t^{(i)}\right)/M$, the recursive function between two successive iterations is:

$$\begin{aligned} \omega_t :=& \left(\bar{\omega}_t^{(c)}, \omega_t^{(s)}\right) \\ =& \omega_{t-1} - \frac{\eta_{i_t}}{M} \nabla_c F_{i_t}\left(\omega_{j(j(t))}^{(i_t)}, \omega_{j(t)-1}^{(s)}\right) \\ & - \eta \nabla_s F_{i_t}\left(\omega_{j(t)}^{(i_t)}, \omega_{t-1}^{(s)}\right). \end{aligned} \tag{3.12}$$

According to the update between ω_t and ω_{t-1}, we derive the following convergence guarantee.

Theorem 1. *Denote the optimal model and the initial model by ω_* and ω_0, respectively. Let stepsize η be $O\left(1/\sqrt{G^2T}\right)$. The average ℓ_2 norm of the gradients using Algorithm 1 is*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla F(\omega_t)\right\|_2^2 \leqslant O\left(\frac{G}{\sqrt{T}}\right)+O\left(\frac{\tau^2}{T}\right) \tag{3.13}$$

where τ is the maximum staleness, i.e., $\tau = \max_{t\in\{0,\ldots,T-1\}}(t-j(t))$. When T is sufficiently large, i.e., $T \geqslant \tau^4/G^2$, the convergence rate is dominated by $O\left(\frac{G}{\sqrt{T}}\right)$.

Proof. According to [32], the expectation value for $(\omega_t - \omega_{t-1})$ can be expressed as follows:

$$\begin{aligned}&\mathbb{E}_{i_t}\left(\nabla_c F_{i_t}\left(\omega_{j(j(t))}^{(i_t)},\omega_{j(t)-1}^{(s)}\right)+\nabla_s F_{i_t}\left(\omega_{j(t)}^{(i_t)},\omega_{t-1}^{(s)}\right)\right)\\ &=\nabla_c F\left(\omega_{j(j(t))}^{(c)},\omega_{j(t)-1}^{(s)}\right)+\nabla_s F\left(\omega_{j(t)}^{(c)},\omega_{t-1}^{(s)}\right)\end{aligned} \tag{3.14}$$

Since the partial derivative of loss function $F(\cdot)$ follows L-Lipschitz, we have the following inequality based on Lemma 3.1:

$$\begin{aligned}&\mathbb{E}\left(F(\omega_t)\right)-F(\omega_{t-1})\\ &\leqslant \mathbb{E}\left\langle\nabla F(\omega_{t-1}),\omega_t-\omega_{t-1}\right\rangle+\frac{L}{2}\mathbb{E}\left\|\omega_t-\omega_{t-1}\right\|_2^2\\ &=-\eta\mathbb{E}\left\langle\nabla F(\omega_{t-1}),\nabla_c F\left(\omega_{j(j(t))}^{(c)},\omega_{j(t)-1}^{(s)}\right)+\nabla_s F\left(\omega_{j(t)}^{(c)},\omega_{t-1}^{(s)}\right)\right\rangle\\ &\quad+\frac{L\eta^2}{2}\mathbb{E}\left\|\nabla_c F_{i_t}\left(\omega_{j(j(t))}^{(i_t)},\omega_{j(t)-1}^{(s)}\right)+\nabla_s F_{i_t}\left(\omega_{j(t)}^{(i_t)},\omega_{t-1}^{(s)}\right)\right\|_2^2\\ &=-\frac{\eta}{2}\left\|\nabla F\left(\omega_{t-1}\right)\right\|_2^2\\ &\quad-\frac{\eta}{2}\left\|\nabla_c F\left(\omega_{j(j(t))}^{(c)},\omega_{j(t)-1}^{(s)}\right)+\nabla_s F\left(\omega_{j(t)}^{(c)},\omega_{t-1}^{(s)}\right)\right\|_2^2\\ &\quad+\frac{\eta}{2}\mathcal{Q}_1+\frac{L\eta^2}{2}\mathcal{Q}_2\end{aligned} \tag{3.15}$$

where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are given by:

$$\mathcal{Q}_1 = \mathbb{E}\left\|\nabla F\left(\omega_{t-1}\right) - \nabla_c F\left(\omega^{(c)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right) - \nabla_s F\left(\omega^{(c)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right\|_2^2$$

$$\mathcal{Q}_2 = \mathbb{E}\left\|\nabla_c F_{i_t}\left(\omega^{(i_t)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right) + \nabla_s F_{i_t}\left(\omega^{(i_t)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right\|_2^2$$

Next, we find the bound for $\mathcal{Q}_1$ and $\mathcal{Q}_2$, respectively. For $\mathcal{Q}_1$, we have

$$\begin{aligned}
\mathcal{Q}_1 &= \mathbb{E}\left\|\nabla F\left(\omega_{t-1}\right) - \nabla F\left(\omega^{(c)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right.\\
&\qquad \left. + \nabla_c F\left(\omega^{(c)}_{j(t)}, \omega^{(s)}_{t-1}\right) - \nabla_c F\left(\omega^{(c)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right)\right\|_2^2\\
&\leqslant 2\mathbb{E}\left\|\nabla F\left(\omega_{t-1}\right) - \nabla F\left(\omega^{(c)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right\|_2^2\\
&\quad + 2\mathbb{E}\left\|\nabla_c F\left(\omega^{(c)}_{j(t)}, \omega^{(s)}_{t-1}\right) - \nabla_c F\left(\omega^{(c)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right)\right\|_2^2\\
&\leqslant 2L^2\mathbb{E}\left\|\omega^{(c)}_{t-1} - \omega^{(c)}_{j(t)}\right\|_2^2 + 4L^2\left\|\omega^{(c)}_{j(t)} - \omega^{(c)}_{j(j(t))}\right\|_2^2\\
&\quad + 4L^2\left\|\omega^{(s)}_{t-1} - \omega^{(s)}_{j(t)-1}\right\|_2^2\\
&\leqslant 6L^2\eta^2(t-j(t))^2G^2 + 4L^2\eta^2(j(t)-j(j(t)))^2G^2\\
&\leqslant 10L^2\eta^2\tau^2G^2
\end{aligned} \tag{3.16}$$

Besides, under Assumption 2, we have

$$\begin{aligned}
\mathcal{Q}_2 &\overset{(a)}{\leqslant} 2\left\|\nabla_c F_{i_t}\left(\omega^{(i_t)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right)\right\|_2^2\\
&\quad + 2\left\|\nabla_s F_{i_t}\left(\omega^{(i_t)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right\|_2^2\\
&\leqslant 2\left\|\nabla F_{i_t}\left(\omega^{(i_t)}_{j(j(t))}, \omega^{(s)}_{j(t)-1}\right)\right\|_2^2 + 2\left\|\nabla F_{i_t}\left(\omega^{(i_t)}_{j(t)}, \omega^{(s)}_{t-1}\right)\right\|_2^2\\
&\leqslant 2G^2 + 2G^2 = 4G^2
\end{aligned}$$

where (a) follows for $\|a+b\|_2^2 \leqslant 2\|a\|_2^2 + 2\|b\|_2^2$. Therefore, plugging $\mathcal{Q}_1$ and $\mathcal{Q}_2$ back to Equation 3.15, we have

$$\begin{aligned} &\mathbb{E}\left(F(\omega_t)\right) - F(\omega_{t-1}) \\ &\leqslant -\frac{\eta}{2}\left\|\nabla F\left(\omega_{t-1}\right)\right\|_2^2 + 5L^2\eta^3\tau^2G^2 + 2L\eta^2G^2 \end{aligned} \tag{3.17}$$

Given the initial weights of model ω_0 and the last model ω_T, where T is the number of iterations, we can obtain the following inequality by summing up the all Equation 3.17 from $t = 1$ to T:

$$\begin{aligned} &\mathbb{E}(F(\omega_T)) - F(\omega_0) \\ &\leqslant \sum_{t=1}^{T}\mathbb{E}\left(F(\omega_t)\right) - F(\omega_{t-1}) \\ &\leqslant -\frac{\eta}{2}\sum_{t=1}^{T}\left\|\nabla F\left(\omega_{t-1}\right)\right\|_2^2 + 5L^2\eta^3\tau^2G^2T + 2L\eta^2G^2T \end{aligned} \tag{3.18}$$

Denoting by ω_* is the optimal solution to the objective function F, we have $F(\omega_*) - F(\omega_0) \leqslant \mathbb{E}(F(\omega_T)) - F(\omega_0)$. Therefore, by setting appropriate learning rate, i.e., $O\left(\sqrt{\frac{F(\omega_0)-F(\omega_*)}{G^2LT}}\right)$, the convergence boundary holds:

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla F(\omega_t)\right\|_2^2 \leqslant O\left(G\sqrt{\frac{L(F(\omega_0)-F(\omega_*))}{T}}\right) \\ + O\left(\frac{L\tau^2(F(\omega_0)-F(\omega_*))}{T}\right) \end{aligned} \tag{3.19}$$

□

3.6 Evaluation

In this section, we demonstrate the effectiveness of *Tree Learning* by conducting the experiments with different cluster settings over multiple backbone models. First, we introduce the Testbed and experiments setting for evaluation. Then, we analyse the experiment results in terms of training speed, accuracy, convergence analysis with various scenarios and settings. Furthermore, we analyse the advantage of *Tree Learning* over existing collaborative machine learning and conduct an ablation study to demonstrate the effectiveness of *Tree Learning*.

3.6.1 Methodology

Testbed. In order to evaluate the performance of *Tree Learning*, we perform our experiments by simulating multiple clients with a lower calculation speed and a server with much higher computation capacity. We use two different clusters for our experiments. we use NVIDIA RTX3080 GPU with 10GB device memory for the server's configuration, and pure CPU for clients. Furthermore, we use NVIDIA RTX3090 GPU with 24GB device memory and A100 GPU with 40GB device memory

Table 3.1: **Performance Comparison between *Tree Learning* and Split Learning**: The table shows the comparison of performance on four representive backbone models, MobileNet, VGG, AlexNet and ResNet with different settings and client numbers. All the settings are trained towards convergent and we record the per-epoch training time for *Tree Learning* and Split Learning. Best split point and Speed ratio are given. We can observe that the acceleration ratio can reach up to at most 4.61x and the average ratio is around 3x regardless the clients number and scenarios.

Models	User	Horizontal				Vertical			
		Split Learning	Tree Learning	Best split point	Ratio	Split Learning	Tree Learning	Best split point	Ratio
MobileNet	1	192.1	41.8	[2]	4.61x	192.1	41.8	[2]	4.61x
	2	196.2	55.1	[2,2]	3.56x	278.8	103.3	[3,4]	2.70x
	4	203.7	74.7	[2,2,3,4]	2.73x	500.1	233.8	[2,3,3,4]	2.14x
	6	260.7	84.1	[1,1,1,2,2,3]	3.10x	771.48	357.9	[1,1,1,1,2,2]	2.16x
VGG	1	116.6	57.4	[2]	2.03x	116.6	57.4	[2]	2.03x
	2	130.4	65.2	[2,3]	2.01x	242.4	86.7	[2,4]	2.80x
	4	177.1	78.6	[1,2,2,4]	2.25x	638.9	168.0	[2,3,2,4]	3.80x
	6	249.5	87.1	[1,1,1,2,2,4]	2.87x	817.4	249.6	[1,1,1,2,2,2]	3.28x
AlexNet	1	240.1	116.2	[3]	2.06x	240.1	116.2	[3]	2.06x
	2	253.5	123.9	[2,3]	2.04x	545.3	301.1	[2,2]	1.81x
	4	533.6	124.6	[2,2,3,3]	4.28x	3805.7	1581.5	[1,2,2,3]	2.41x
	6	545.8	142.3	[1,2,2,3,3,3]	3.84x	4940.3	2075.6	[1,2,2,2,3,3]	2.38x
ResNet	1	1025.6	536.4	[1]	1.91x	1025.6	536.4	[1]	1.91x
	2	1197.7	364.9	[1,2]	3.28x	2990.2	1205.7	[1,1]	2.48x
	4	1204.8	555.2	[1,1,2,2]	2.17x	13650.3	6757.5	[1,1,1,2]	2.02x
	6	1296.1	638.5	[1,1,1,2,2,3]	2.03x	15782.9	7971.2	[1,1,1,2,2,3]	1.98x

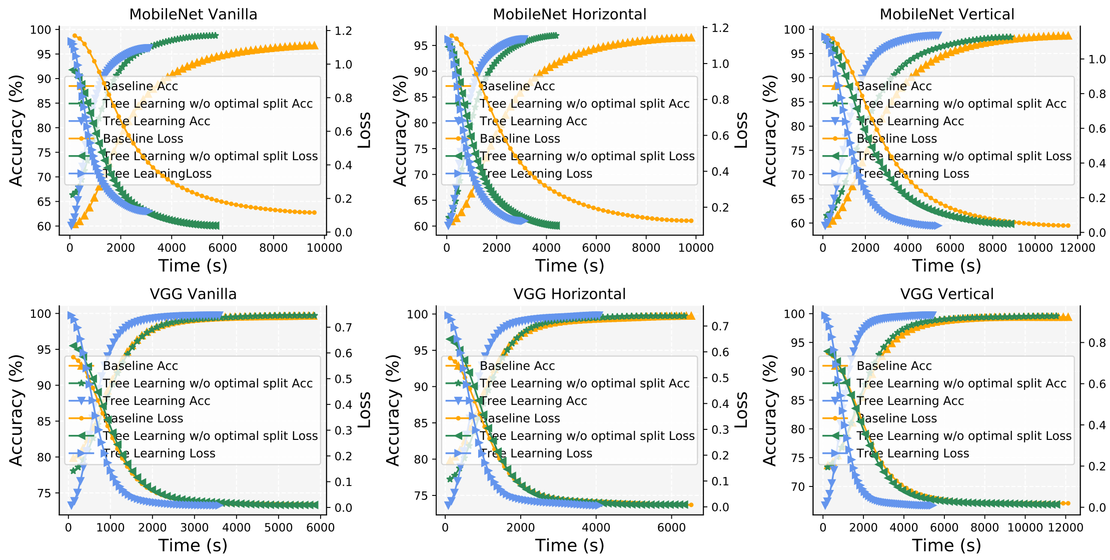

Figure 3.6: **Time versus Accuracy of *Tree Learning* Training**: Blue lines represent TREE LEARNING, green lines represent TREE LEARNING without best split point, and yellow lines represent the baseline SL. From the figure we can observe that TREE LEARNING converges fastest against the other two baselines, and can reach above 3x faster than Split Learning.

as our server's configurations for more clients experiments. Both the servers run on on Ubuntu 18.04 LTS OS. For client setting, we simulate clients with different calculation speeds to imitate a practical scenario with heterogeneous edge devices.

Benchmark Models. We use Cifar10 [66] as our dataset, since the performance improvement of experiments is hardly relevant with input data and the resource constraints on edge devices when using large scale datasets such as ImageNet [23]. Two different DNN models are used in our experiments: 1) MobileNet [53], which is a typical model suitable for mobile and edge devices; 2) VGG [116], which is one of the classic models used for deep learning. 3) Alexnet [67], one of the first deep learning model to improve the ImageNet contest. 4) Resnet [50], one of the first deep learning model with residual blocks. We train the all models using SGD with a momentum of 0.9 and an initial learning rate at 0.001.

Baselines. We evaluate two configurations, the horizontal one and the vertical one.

The horizontal one separates training data in batch perspective, while the vertical one splits features of data samples between clients but keeps the total amount of samples invariant, as introduced in Sec. 3.2. For each category, we select the existing SL as our baseline and choose clients with different calculation speeds to simulate the practical resource diversity of edge devices, i.e., smart watch, mobile phone and personal laptop. We compare *Tree Learning* against baseline and *Tree Learning* without optimal split layer. For baseline, split points for all the clients are the same, we place around half of the partition layers on the client and half of them on the server. We use a mini-batch size of 16 for MobileNet, 18 for VGG and 12 for AlexNet and ResNet. In the experiments, we measure the time when the advertised validation accuracy is achieved. We also show the advantage of *Tree Learning* from the perspective of model complexity and extra computation burden incurred by comparing with existing collaborative learning works [145].

3.6.2 Training Speed

To evaluate the training speed of *Tree Learning*, we conduct experiments for both our *Tree Learning* and the baselines, with different calculation speeds for each clients. Furthermore, to ensure the current split point is optimal, we monitor the training status and re-calculate the best split point according to current computation and communication speed of each client and perform step one every fixed epoch, depends on the task difficulty. As shown in Table. 3.1, we implement *Tree Learning* in the scenarios of horizontal and vertical configuration and each of them trained with the four backbone models with the server of RTX3080. We record the best split point as the most frequently combination, and as we observed, optimal split point rarely changed given the stable local computation speed and adaptive substitute mechanism for communication turbulence. We list 4 client setting given to space constraints and server capacity in Tab. 3.1, and we can observe that most of the settings can reach over 2x or 3x speed up and some settings can even obtain over 4x speed up with superiority, which largely demonstrates the effectiveness of our method. We also explore more clients scenarios within the following paragraph. We compare the optimal partition configuration of *Tree Learning* with the partition strategy of the baseline model over the corresponding training time with similar accuracy when they converge. We can see that, without compromising any accuracy, *Tree Learning* can achieve up to 361% speed-up in training speed as compared with existing model parallelism. Another observation is that *Tree Learning* has robust performance in all the circumstances, regardless of the neural network model, scenario configuration, and number of clients.

3.6.3 Acceleration Ratio across clients and cluster

To further validate the scalability effectiveness of *Tree Learning* of different configurations, we carry out our experiments on three different machines, RTX3080, RTX3090 and A100 with different amount of clients, ranging from 1 to 20, as shown in Fig. 3.7. For a small number(1-2) of clients, we use RTX3080 as the server. To accommodate more clients, we use more powerful machine as the server. We deploy 4-10 clients with RTX3090 and 16-20 clients with A100. We simulate clients with different levels of computing speed for each setting. For example, we assign 4 clients with a relatively low speed, 2 clients with a medium speed and 2 clients with a high speed in the setting of 8 clients in order to simulate smart watches, phones and personal laptops. Fig. 3.7 shows that *Tree Learning* can reach 4.61x with only one client and 16.07x with 20 clients. More clients result in exaggerated increase in the training time in traditional framework, but lead to consistent or a slow increase of the training time in *Tree Learning*.

3.6.4 Convergence Analysis

Despite the significant improvement in training efficiency, one of the common assumptions for a training accelerating algorithm is the cost of diminishing accuracy. However, instead of compromising any accuracy during training, *Tree Learning* can achieve the same accuracy even with less time. Fig. 3.6 illustrates the results of three scenarios and two models , with x axis as the training time and y axis as the accuracy. From each subplot, we can see that among the configurations of vanilla, horizontal and vertical, *Tree Learning* converges faster and achieves superior time-to-accuracy as compared with the others.

3.6.5 Resource Efficiency

As we proved the outstanding performance and training efficiency in the previous evaluation, We further conduct a comparison of *Tree Learning* against Federated Learning to validate the advantage of resource efficiency of our proposed method. We leverage mobilenet as backbone in both methods and observe several indicators. As shown in Tab. 3.2, we share more comparisons on computing resources in GFlops and communication bandwidth in MB required by these techniques. We also record the learnable parameters on client for each method. We can see that *Tree Learning* requires a lower bandwidth, fewer GFlops and learnable parameters on client, which largely alleviates the resource overhead compared with Federated Learning. In real case, the overhead depends on the dataset size and the partition layer.

3.6.6 Ablation Study

We conduct an ablation study to show the experiment results of separate improvements with different components against the complete *Tree Learning* and baseline model in terms of acceleration ratio.

Efficacy of Real Embedded Device Environment. In order to evaluate the

Table 3.2: **Resource Efficiency Comparison between *Tree Learning* and Federated Learning**: We observe the resource efficiency of Tree Learning in terms of GFlops, communication bandwidth needed, and learable parameters on the client. The results show the outstanding efficiency against the Federated Learning framework.

Method	GFlops	Communication Bandwidth(MB)	Learnable Parameters
Tree Learning	847.77	1.5	1344
Federated Learning	28218.70	21.88	3228170

performance of *Tree Learning* on real edge devices, we implement the proposed framework via commercial off-the-shelf devices and compare it with simulation. As in a practical environment of embedded devices, we deploy *Tree Learning* on four NVIDIA Jetson Nano with 4GB device memory and 16GB storage, Rasparry Pi 4 Model B with 8G device memory and 64G storage as heterogeneous clients with different computing capacity. We use a 1 Gbps bandwidth Ethernet to connect the clients with the server. All the devices are operated with Ubuntu 18.04LTS with ARMv8 Processor. We can see from Table. 3.3 that *Tree Learning* shows superior performance in the testbed environment as well as in simulation settings.

Efficacy of Partition Algorithm. To further validate the effectiveness of Dynamic Partition Algorithm, we conduct the experiments on the static partition, employing our partition algorithm with agents of different computation capacity. We also validate their individual performance on the testbed environment with epoch of 50 and different configurations. Both of the experiments result show that Partition Algorithm significantly accelerates the training process. As shown in Fig. 3.8, in spite of satisfying acceleration made by *Tree Learning* without optimal split layers, the full *Tree Learning* can further expedite the training process.

Efficacy of Overlapping between Communication and Computation. We conduct experiments to show the execution overlapping of the client and server when

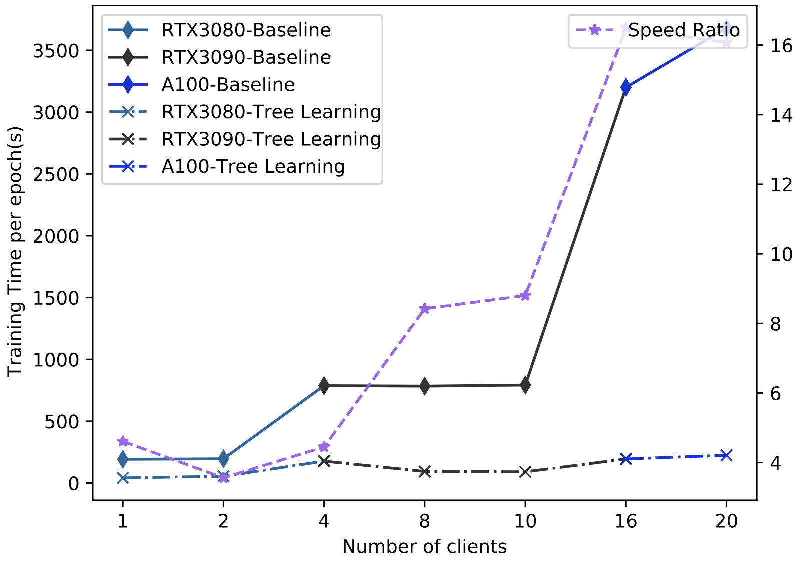

Figure 3.7: **Training speed acceleration across different clusters and clients**: We also implement on various clients and clusters and observe that more clients need more training time on SL but similar time on Tree Learning, which result in a higher speed-up ratio.

Table 3.3: **Performance Comparison between simulation and testbed environment**: We validate the effectiveness of *Tree Learning* by employing the experiments on the testbed environment and compare the performance against the simulation one with the epoch of 50. We can observe that the performance of Tree Learning on real testbed environment can achieve similar performance as simulation.

Models	Scenarios	Setting	Baseline	Tree Learning	Ratio
MobileNet	Horizontal	Simulate	9809.8	2753.3	3.56x
		Testbed	15597.8	4485.6	3.47x
	Vertical	Simulate	13942.1	5164.3	2.69x
		Testbed	27794.2	8944.4	3.10x
VGG11	Horizontal	Simulate	8857.4	3931.2	2.25x
		Testbed	46001.2	19848.7	2.32x
	Vertical	Simulate	31942.8	8401.4	3.80x
		Testbed	62512.2	20432.3	3.06x

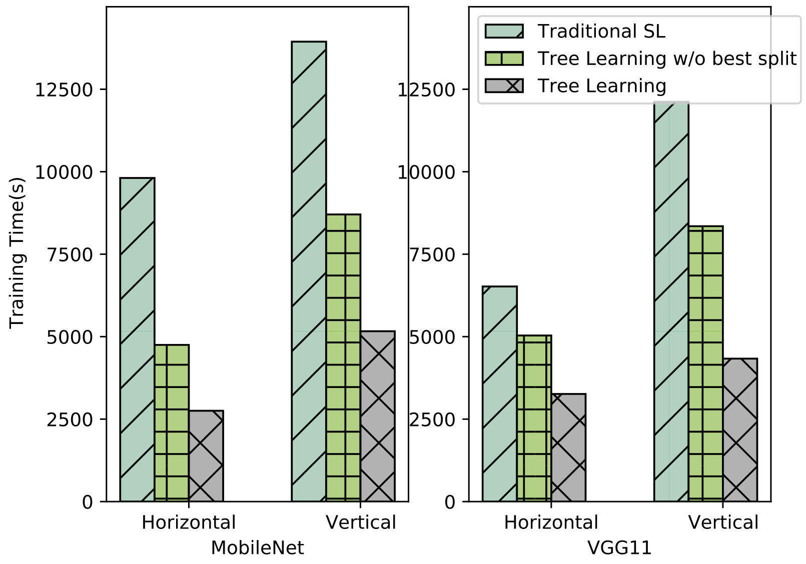

Figure 3.8: **Ablation study of parallelism and optimal split point**: Comparison of training time between *Tree Learning*, *Tree Learning* w/o partition strategy and SL. The difference between SL and *Tree Learning* w/o partition strategy represents the effectiveness of parallelism, and the difference with *Tree Learning* further shows the superiority of proposed partition strategy.

using *Tree Learning* versus using the traditional SL methods after the partition algorithm. The two configuration use the same batch size and split point to control variables. Fig. 3.9 shows the execution time of client and server, the communication time and the overall execution time when training the models with 1 epoch. We observe that, the overall execution time approximately equals to the sum of the other three times because there is no overlapping during the training. On the contrary, in *Tree Learning*, in spite of the similar client execution time, server execution time and communication time, the overall execution time is largely reduced because of two reasons: (1) the overlapping of the execution times between the clients and the server; (2) the overlapping of the computation and communication times. Furthermore, we notice that small discrepancy of training time on clients and server will result in better performance of *Tree Learning*.

Efficacy with Dropouts To further validate the effectiveness of our method when encountering dropout situation, we conduct experiments to simulate the dropout situation. We randomly select 10% of clients to drop out of the system for 5 rounds during the training process, as shown in Fig. 3.10. We employ on both horizontal and vertical scenarios for comparison, and observe that even with the dropout of clients, the performance of our system remains outstanding.

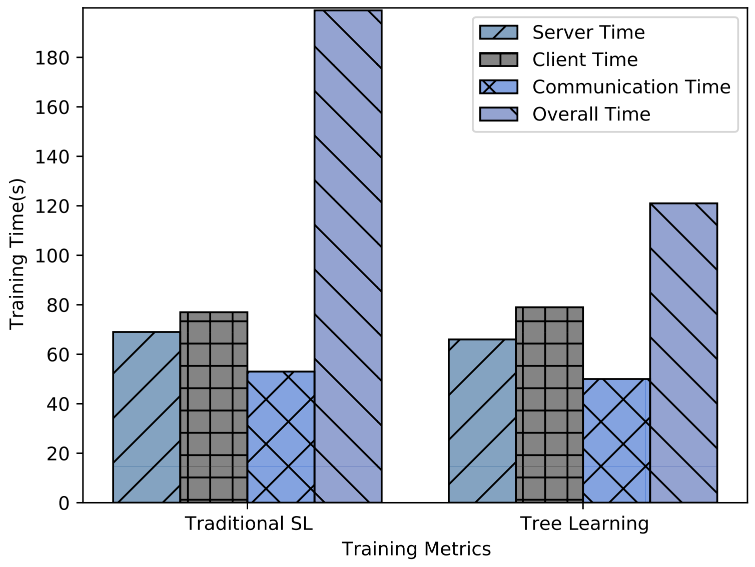

Figure 3.9: **Time Breakdown per-epoch of the Training**: The figure shows the comparison of breakdown of time spend on each period between *Tree Learning* and SL. We can observe that although the breakdown seems similar in *Tree Learning* and SL, the overall time of *Tree Learning* is much smaller than that of Sl.

3.7 Chapter Summary

In this chapter, we have proposed *Tree Learning*, a novel system-algorithm co-design framework which can accelerate the collaborative training over resource constrained devices while maintaining training accuracy. We have applied a tree-aggregation scheme to optimize the synchronization overhead and a parallelism mechanism be-

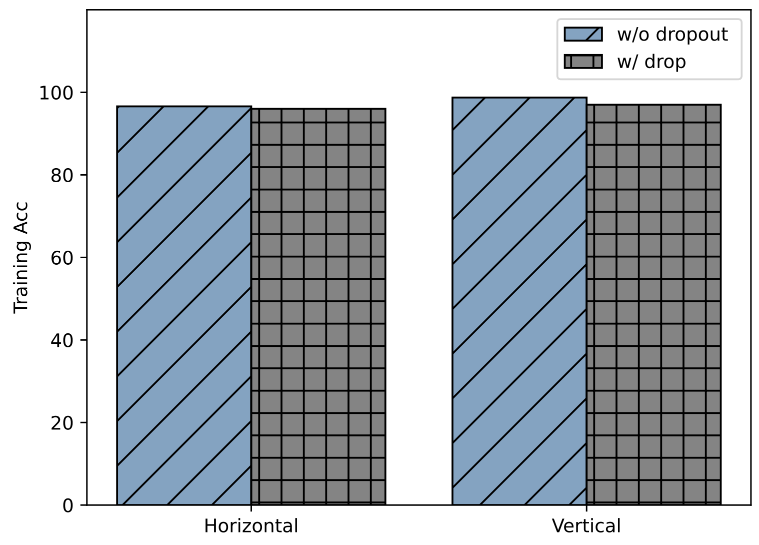

Figure 3.10: **Ablation Study with Dropout Scenario**: The figure shows the performance comparison of *Tree Learning* with or without the dropout situation. We can observe that although encountering dropout of some clients, the performance of *Tree Learning* still remains good in both the horizontal and vertical scenarios, which validate the robustness of our proposed system.

tween the clients and server to realize the fully parallelism of the system. We have also proposed a general solution for the *Tree Learning* framework to find the optimal aggregation layers through an optimization formulation. The effectiveness of *Tree Learning* is demonstrated via extensive experiment result, not only from simulations, but also using real embedded devices. This work has established a constructive towards a diversified collaborative machine learning paradigm, which provides some insights for future collaborative learning research and development.

Chapter 4

PromptFL: Let Federated Participants Cooperatively Learn Prompts Instead of Models — Federated Learning in Age of Foundation Model

4.1 Introduction

The ever-growing edge devices, e.g., smart phones, autonomous vehicles, etc., have become the dominant computing platforms today. These devices generate vast amounts of valuable data while providing hidden insights into the human world. Artificial intelligence (AI) nowadays has shown its success to mine the big edge data and produce accurate models that can replace human decisions timely and properly. However, analyzing large amounts of data using sophisticated machine learning algorithms requires significant computing power. Therefore, traditional AI paradigms require to gather all raw data to a cloud center for centralized training, which can incur significant communication overhead and potential privacy leakage, and thus are not desirable for edge users [154, 114, 1].

Federated learning (FL) [80, 6, 145]has emerged to conduct distributed machine learning that allows multiple edge users to jointly train a shared model without sharing their raw data, which has been demonstrated great success in many edge applications, e.g., input word prediction, voice assistant, etc. [49, 79], that can mine massive distributed data without exposing users' privacy, and thus are widely applied in various edge scenarios. The FL training process comprises of two iterative phases, i.e., local training and global aggregation. Thus the learning performance is determined by both the effectiveness of the parameters from local training and smooth aggregation of them. However, these two requirements are not easy to satisfy in edge environment, i.e., edge users often have limited bandwidth and insufficient data, which can cause inefficient parameters aggregation, excessive training time and reduced model accuracy. Despite the rich opportunities offered by FL, fundamental research problems still need to be addressed before FL can be readily applied to real-world deployments.

Existing research efforts have focused on improving the FL optimization process [74, 149] or refining model architectures [104], but this does not change that FL inherently entails a large number of communication rounds and a large amount of labeled data for training, which are often unavailable for edge users. Such challenges are particularly salient under the combined effect of a long training process and unfavorable factors such as non-IID and unbalanced data [73], limited communication bandwidth, and unreliable and limited device availability. Therefore, there is an urgent need to explore alternative solutions that can mitigate the challenges of existing FL paradigm and make it more feasible to edge users.

We revisits the question of how FL mines the distributed data in iterative training rounds, and exploit the emerging foundation model (FM) to optimize the FL training. FM refers to large neural model that trained on large scale data and has strong adaptation capability for various downstream tasks. We let federated participants cooperatively learn prompts instead of models to unleash the power of FM in a distributed way, whereby both the local training and global aggregation can be significantly accelerated. Our paper aims to provide a new perspective by rethinking if FMs can be applied to FL as a new paradigm of training.

We investigate the behavior of the nascent model in a standard FL setting using popular off-the-shelf FMs, *e.g.*, CLIP, and methods for FM adaptation. We propose PromptFL, a novel federated framework that leverages the benefit of large vision-language model. PromptFL trains the prompt instead of the whole model parameters, which can simultaneously exploit the insufficient local data and reduce the aggregation overhead. PromptFL ships an off-the-shelf public CLIP to users and apply continuous prompts (*a.k.a.* soft prompts) for FM adaptation, which requires very few data samples from edge users. The framework is technically very simple but effective, and such insight sheds light on the development of the federated society with more possibilities. The focus of our investigation is whether it meets the key principles:

- **Feasibility.** What are the system costs? We examine the feasibility of PromptFL on modern hardware, focusing conservatively on personal cell phones. We demonstrate the feasibility of the system in terms of overhead in communication, training, and inference dimensions.
- **Performance.** Are PromptFL competitive with FL? FL does not baseline against any such approach, so we implement a proof-of-concept in the framework, spanning a range of popular image classification tasks. We observe PromptFL competitive with strong FL baselines.
- **Privacy.** Is PromptFL privacy-preserving? We show that PromptFL keeps data on each device private, aiming to learn global prompts updated only by

communicating gradients rather than the data itself, and thus not less private than FL.

4.2 Preliminaries

4.2.1 Foundation Model

AI is going through a paradigm shift with the rise of models (*e.g.*, BERT, GPT-3, CLIP, DALL-E·2 [25, 9, 105, 108]) trained on broad data using self-supervision at scale that can be adapted to a wide range of downstream tasks. Researchers call these models foundation models (FMs) to emphasize their key core. From a technical standpoint, FMs are not new. However, the sheer size and scope of FMs over the past few years has expanded our imagination of what is possible. FMs are scientifically interesting for their impressive performance and capabilities, but what makes them critical to research is that they are rapidly being integrated into real-world deployments of AI systems, with profound implications for users.

CLIP Contrastive Language-Image Pre-Training (CLIP) is a neural network trained on hundreds of millions of (image, caption) pairs [105]. CLIP encodes images and captions separately as vectors, enabling users with visual modality samples to retrieve, score, or classify samples from textual modalities. Models are often very fragile and only know very specific things you trained them to do. CLIP extends the knowledge of classification models to a wider range of things by leveraging semantic information in text. Standard classification models completely discard the semantic meaning of class labels and simply enumerate numeric classes behind the scenes; CLIP works by understanding the meaning of the classes. ALIGN is another CLIP-like vision-language pre-training [59].

Image Classification with CLIP CLIP pre-trains an image encoder and a text encoder to predict which images are paired with which texts. We can use this behavior to convert the CLIP to an image classifier. We may convert all [class] to captions such as "picture of [class]" and predict the caption class CLIP estimates the best pairing with the given image. In many previous works, this has involved prompt template engineering, in which human engineers or algorithms search for the best template for each class [35, 77, 118, 146].

4.2.2 Federated Learning

Recent neural models require large amounts of training data [26], and users typically hold limited-scale labeled data. To address the challenge of lack of sufficient data for individual users, federated learning of data across multiple privacy spheres (*i.e.*, users) has become a popular framework.

The term *federated learning* was introduced by [87]. In a centralized setting, the federated server initially sends global model parameters to each client. After training with local data, the participants are only required to share gradients for model updates. Then the server aggregates the gradients and transmits the updated model back to each client. More specifically, federated learning is a machine learning setting where a set of n clients (*e.g.,* mobile devices) collaboratively train a model under the orchestration of a federated server (*e.g.*, service provider), while the training data of clients is stored locally and not exchanged [60]. The federated server orchestrates the collaborative training process, by repeating the following steps until training is converged:

Client Selection Given the unstable client availability, for the round t of federated learning, the federated server samples a small subset of m clients meeting eligibility requirements out of all n clients to participate in the learning.

Local Training Upon notification of being selected at the round t, each selected client downloads the current parameters θ of global model and a training program from the federated server. Each selected client locally computes an update to the global model on its local training data by executing the training program. More specifically, the gradients updated at one client (denoted as G), are computed by $\frac{\partial \ell(X,y,\theta)}{\partial \theta}$, where X, y denote the batches of training data and corresponding labels, and $\ell(\cdot)$ refers to the loss function.

The gradients G in typical federated learning settings are the minimum that must be shared to the server, corresponding to FedSGD method. In FedAVG [87], models are consecutively updated on more batches of local data, which can be several epochs of training, and then shared. We note that a common way is to share the updated model $\theta + G$, but this practically amounts to sharing G since all participants know θ.

Global Aggregation Upon having received local updates from m clients, the federated server aggregates these updates and update its global model, and initiates next round learning. In addition to the federated learning framework that relies on the centralized server node, there are also some federated learning implementations based on the decentralized framework [112, 68, 55]. This means that the aggregation of gradients does not necessarily occur in a fixed federation server, but may also occur in some clients.

4.3 Prompt-Based Federated Learning

We hypothesize that an off-the-shelf public CLIP-like model is shipped to the user device. The CLIP-like model is a powerful image classifier that utilizes linguistic knowledge to classify images. In other words, CLIP already knows a lot about the content of images. But to unleash the power of CLIP in FL, we need to take advantage of something called prompt engineering that was mentioned in the preliminaries.

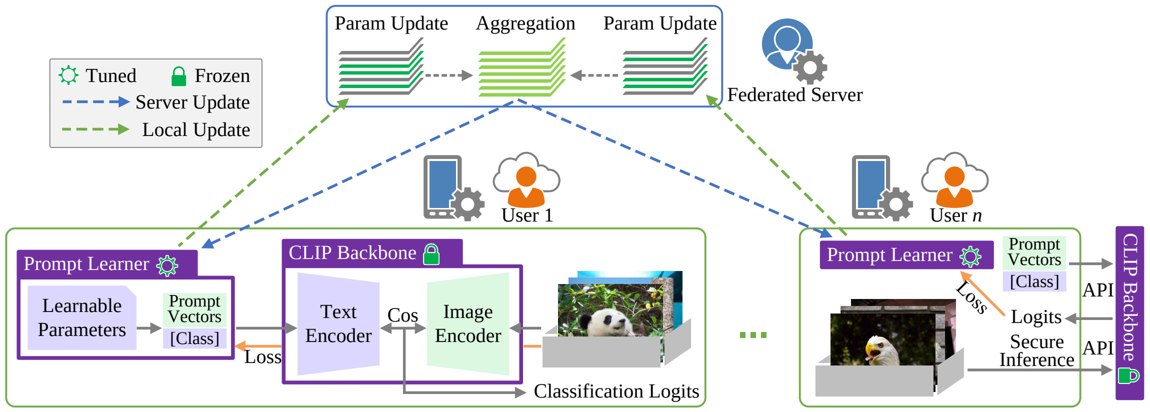

Figure 4.1: Framework and workflow of PROMPTFL. Each client includes a prompt learner (with only a small amount of trainable parameters) and an out-of-the-box CLIP (with backbone frozen). The federated server aggregates only the parameter updates of prompt learners over multiple users, and transmit the updated parameters back to each user.

4.3.1 Prompt Engineering

The prompting function $f_{\text{prompt}}(\cdot)$ is applied to modify the class label $\mathbf{y}$ into a prompt $\mathbf{y}' = f_{\text{prompt}}(\mathbf{y})$. The most natural form of implementing a prompting function is to manually create an intuitive template based on human introspection. For example, as referred in [9] we may manually craft prefix prompts to handle an image classification task by using templates like "picture of [class]" or "a photo of a [class]". Based on that, many approaches have been proposed to automate the template design process.

Specifically, the automated prompting can be further separated into discrete prompts (*a.k.a.* hard prompts), where the prompt is an actual text string, and continuous prompts (*a.k.a.* soft prompts), where the prompt is performed directly in the embedding space of the model [81]. Discrete prompts constraint that the embeddings of template words be the embeddings of natural language words [115, 37]. Thus, discrete prompting is a clear way to visualize what "word" are learned for the vectors [24].

Our paper adopts continuous prompts instead of discrete prompts in FL for the reason that (1) Our purpose of prompt construction is to find a way to enable FL to efficiently perform the image classification tasks, not for human interpretation, there is no need to limit prompts to human-interpretable natural language. (2) The templates have their own parameters that can be tuned based on training data from the user, which is a natural compatibility connecting FL and prompting. More related topics of continuous prompts can refer to [76, 69, 130, 46, 150].

4.3.2 Framework to Learn Prompts in FL

The framework of PromptFL is presented in Figure 4.1. Each FL client consists of a prompt learner and an out-of-the-box CLIP model. PromptFL introduces only a small amount of trainable parameters in the prompt learner while keeping the CLIP backbone frozen. In other words, during local training, only the parameters of the prompt learner are updated while the whole CLIP model turns off gradients in both the image and the text encoder. The federated server is designed to aggregate only the parameter updates of prompt learners over multiple users, and transmit the updated parameters back to each user. Thus, PromptFL evolves the goal of FL from model training to prompt learner training.

The CLIP backbone comprises two encoders, one for images and the other for texts. The image encoder will map high-dimensional images into a low-dimensional embedding space. The network of the image encoder can take the form of a CNN such as ResNet50 [50] or Vision Transformer [27]. The text encoder will generate text representations from input. The network of the text encoder is a Transformer [132].

Prompt Learner Given a pre-trained CLIP backbone, the input to the text encoder is designed in the form of [prompt vectors][class]. Inspired by the simple and straightforward prompt design in [150], we introduce a set of p continuous embeddings of dimension d in the [prompt vectors]. d is same as the dimension of

Table 4.1: System cost comparison. Assumes 32 local training batch size, 1 local training epoch, 100 total communication rounds for FL. Assumes 196 input sequence length, full precision for PromptFL and FL.

Frameworks / Dimensions	PromptFL (150M parameter model)	Federated Learning (100M parameter model)	Modern Mobile Phone Hardware [30]
Communication	600 MB File Download 1.4 Minutes	40 GB File Download + 40 GB File Upload Totally 9 Hours	54 Mbps Downstream RateLimit 12 Mbps Upstream RateLimit [93]
Training	Much Less	4 TFLOPs	1.5 TFLOPs, 8 GB RAM
Inference	60 GFLOPs	40 GFLOPs	1.5 TFLOPs, 8 GB RAM
Storage	600 MB on Disk	400 MB on Disk	1 TB on Disk

word embeddings in the text encoder, thus 512 by default. p is a hyperparameter specifying the number of embeddings. In a word, [prompt vectors] are p learnable d-dimensional vectors.

Given a batch of image-text pairs, CLIP will maximize the cosine similarity for matched pairs while minimize the cosine similarity for all other unmatched pairs. Since CLIP is pre-trained to predict whether an image matches a textual description, it can compute the classification loss and logits by aligning the two embedding spaces for images and texts (*i.e.*, [prompt vectors][class]) respectively. Formally, let $g(\cdot)$ and $h(\cdot)$ be the feature extraction function of the image and text encoder. Let $w_i = h(\mathbf{P}, \mathbf{K}_i)$ be the weight vector generated by the text encoder, where $i \in [1, k]$. k denotes the number of classes and each $(\mathbf{P}, \mathbf{K}_i)$ is derived from the prompt in the form of [prompt vectors][class]$_i$, where [class]$_i$ is replaced by the word embedding vector of specific class label name. Let $\cos[\cdot|\cdot]$ denote the cosine similarity used by CLIP. By forwarding a $(\mathbf{P}, \mathbf{K}_i)$ and an image $\mathbf{x}$, the classification prediction probability and logits are computed as

$$\mathrm{p}(\mathbf{y} = i|\mathbf{x}) = \frac{\exp\left(cos[g(\mathbf{x})|h(\mathbf{P}, \mathbf{K}_i)]\right)}{\sum_{j=1}^{k} \exp\left(cos[g(\mathbf{x})|h(\mathbf{P}, \mathbf{K}_j)]\right)}, \tag{4.1}$$

where $\mathbf{P}$ is the only part that is updated in local back propagation and aggregated in the federated server.

Prompting are particularly useful in the FL case, as using prompts to push the model in the correct direction is particularly effective. This feature enables prompting to converge quickly in FL, requires less data per user, and is less affected by adverse factors in the process, *e.g.*, non-IID and unbalanced data, limited communication bandwidth, and unreliable and limited device availability. In this paper, the prompt learner employed in PromptFL though simple and straightforward as a bridge to our core idea is easy to follow. We also envision that more complex and effective bridges would be there to replace the role and should be a valuable direction.

4.3.3 System Feasibility

We examine the feasibility of PromptFL on modern hardware, focusing conservatively on personal cell phones. We notice that users can access GPUs from their mobile phones. Enterprise users have more abundant resources. Without loss of generality, we take a 100M parameter model for FL and 150M parameter CLIP backbone for image similarity-search of PromptFL. The prompt learner introduces only a small number of parameters, that can be ignored. We assume that the FL configures 32 local training batch size, 1 local training epoch, and 100 total communication rounds, which suggested in [104]. We also assume that both FL and PromptFL configure 196 input sequence length and the full precision. The system cost comparison is summarized in Table 4.1 along the following dimensions:

Communication The average download speed within the globe for mobile internet was 54 Mbps, and the average upload speed for mobile internet was 12 Mbps that reported by 2021 [93]. PromptFL requires locally downloading while FL requires communicating the model repeatedly between users and the federated server. Thus, the communication cost in terms of file transfer volume is that it takes only 1.4 minutes to transfer 600MB for PromptFL, and 9 hours for FL to transfer 40GB.

Training and Inference FL requires FLOPs computed by (2$\times$3$\times$model parameters$\times$local training epoch$\times$local training batch size$\times$input sequence length) for training, while

Table 4.2: **Performance of PromptFL against existing FL framework with iid data distribution**. The table report the accuracy, F-1 score and learnable parameters according to the corresponding backbone and method under the iid data distribution. We report the best score of each group with respect to method and model and annotate in bold. Compared with finetuning and training from the scratch, PromptFL only update 0.01% ∼ 0.1% parameters, however, still outperforms other methods in most cases. Despite encountering suboptimal cases, our method still approaches the optimal performance with small gap.

(a) Caltech101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	32.4	10.5	100
	Vit	32.5	12.9	100
Finetuning	Rn50	90.0	84.7	100
	Vit	93.1	89.1	100
PromptFL	Rn50	**90.2**	**86.1**	**0.1**
	Vit	**94.7**	**91.8**	**0.01**

(b) Flowers102

Method	Model	Acc	F-1	Para
From Scratch	Rn50	33.2	25.7	100
	Vit	38.0	32.5	100
Finetuning	Rn50	**92.6**	**91.6**	100
	Vit	**91.9**	**90.7**	100
PromptFL	Rn50	88.2	87.6	**0.1**
	Vit	90.5	90.1	**0.01**

(c) OxfordPets

Method	Model	Acc	F-1	Para
From Scratch	Rn50	10.3	7.6	100
	Vit	8.7	8.3	100
Finetuning	Rn50	**90.4**	**90.1**	100
	Vit	92.1	91.9	100
PromptFL	Rn50	88.5	88.5	**0.1**
	Vit	**92.9**	**92.8**	**0.01**

(d) Food101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	21.1	19.8	100
	Vit	21.0	19.9	100
Finetuning	Rn50	69.3	69.1	100
	Vit	76.7	76.9	100
PromptFL	Rn50	**78.0**	**77.9**	**0.1**
	Vit	**85.8**	**85.7**	**0.01**

(e) DTD

Method	Model	Acc	F-1	Para
From Scratch	Rn50	10.4	7.4	100
	Vit	12.1	9.9	100
Finetuning	Rn50	**67.8**	**68.2**	100
	Vit	**70.6**	**70.0**	100
PromptFL	Rn50	55.0	53.9	**0.1**
	Vit	58.6	57.8	**0.01**

(f) UCF101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	19.7	16.9	100
	Vit	19.6	18.7	100
Finetuning	Rn50	**74.1**	**72.9**	100
	Vit	**80.5**	**79.6**	100
PromptFL	Rn50	66.4	64.0	**0.1**
	Vit	75.6	74.4	**0.01**

(g) Sun397

Method	Model	Acc	F-1	Para
From Scratch	Rn50	10.0	7.9	100
	Vit	9.8	8.2	100
Finetuning	Rn50	57.6	57.2	100
	Vit	64.1	63.8	100
PromptFL	Rn50	**66.4**	**65.9**	**0.1**
	Vit	**70.9**	**70.1**	**0.01**

(h) Average

Method	Model	Acc	F-1	Para
From Scratch	Rn50	19.6	13.7	100
	Vit	20.2	15.8	100
Finetuning	Rn50	**77.4**	**76.2**	100
	Vit	81.3	80.3	100
PromptFL	Rn50	76.1	74.8	**0.1**
	Vit	**81.3**	**80.4**	**0.01**

Table 4.3: **Performance of PromptFL against existing FL framework with non-iid data distribution**. The table report the accuracy and F-1 score according to the corresponding backbone and method under the non-iid data distribution. We report the best score of each group with respect to method and model and annotate in bold. Other than the iid scenario in Tab. 4.2, our method surpasses the alternatives method by a significant margin across all datasets under the non-iid settings, with only updating 0.01% ∼ 0.1% parameters. By contrast, finetuning and training from scratch are not able to address shifted class distribution problem caused by non-iid setting.

(a) Caltech101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	11.9	2.9	100
	Vit	12.1	3.5	100
Finetuning	Rn50	29.8	12.2	100
	Vit	29.9	12.2	0.1
PromptFL	Rn50	**88.7**	**84.0**	**0.1**
	Vit	**94.1**	**90.5**	**0.01**

(b) Flowers102

Method	Model	Acc	F-1	Para
From Scratch	Rn50	16.4	6.1	100
	Vit	18.4	7.9	100
Finetuning	Rn50	24.4	10.7	100
	Vit	24.5	11.2	100
PromptFL	Rn50	**66.3**	**60.1**	**0.1**
	Vit	**74.8**	**69.1**	**0.01**

(c) OxfordPets

Method	Model	Acc	F-1	Para
From Scratch	Rn50	8.3	3.8	100
	Vit	6.8	3.4	100
Finetuning	Rn50	24.8	11.3	100
	Vit	25.3	11.9	100
PromptFL	Rn50	**87.0**	**86.9**	**0.1**
	Vit	**89.5**	**88.5**	**0.01**

(d) Food101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	14.1	7.1	100
	Vit	11.9	5.9	100
Finetuning	Rn50	22.9	10.2	100
	Vit	23.8	10.7	100
PromptFL	Rn50	**78.1**	**78.0**	**0.1**
	Vit	**85.9**	**85.8**	**0.01**

(e) DTD

Method	Model	Acc	F-1	Para
From Scratch	Rn50	7.4	2.9	100
	Vit	8.5	3.2	100
Finetuning	Rn50	42.4	37.3	100
	Vit	36.3	31.2	100
PromptFL	Rn50	**44.4**	**42.3**	**0.1**
	Vit	**47.5**	**45.4**	**0.01**

(f) UCF101

Method	Model	Acc	F-1	Para
From Scratch	Rn50	7.4	2.9	100
	Vit	8.5	3.2	100
Finetuning	Rn50	42.4	37.3	100
	Vit	36.3	31.2	100
PromptFL	Rn50	**44.4**	**42.3**	**0.1**
	Vit	**47.5**	**45.4**	**0.01**

(g) Sun397

Method	Model	Acc	F-1	Para
From Scratch	Rn50	6.5	2.7	100
	Vit	5.8	2.4	100
Finetuning	Rn50	23.5	15.0	100
	Vit	22.1	12.2	100
PromptFL	Rn50	**61.1**	**59.9**	**0.1**
	Vit	**66.9**	**65.5**	**0.01**

(h) Average

Method	Model	Acc	F-1	Para
From Scratch	Rn50	10.6	4.3	100
	Vit	10.5	4.4	100
Finetuning	Rn50	30.0	18.9	100
	Vit	28.3	17.0	100
PromptFL	Rn50	**70.0**	**67.4**	**0.1**
	Vit	**75.6**	**73.3**	**0.01**

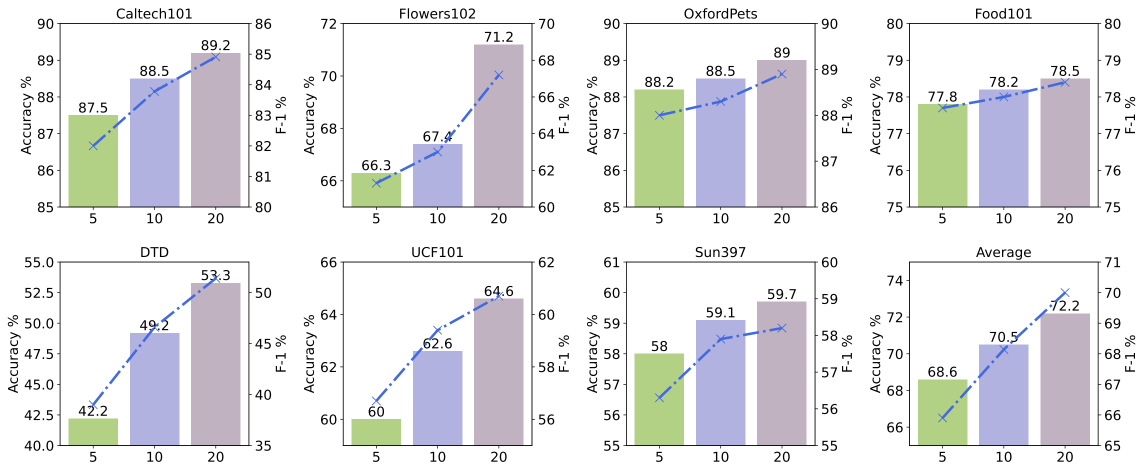

Figure 4.2: **Performance of PromptFL with different class distribution.** We evaluate the performance on seven datasets and record the average performance. X-axis represents the number of classes on each client. Bars represent accuracy and lines indicate F-1 score. We place random fixed number of classes on each client and range the number from 5 to10 to 20. As the number of classes on client gets larger, performance on both the accuracy and the F1 value improve. Furthermore, as the number of classes becomes sufficient, the improvement speed gets slower.

the training FLOPs of PromptFL is much smaller and negligible compared to FL. For both PromptFL and FL, inference requires FLOPs computed by (2×model parameters×input sequence length), in the setting where the key and value vectors for attention computation are cached. Compared to the acceptable computational and storage costs, the RAM on the modern cell phones is a key bottleneck. We believe that this bottleneck will no longer be a problem in the near future as the techniques evolve: (1) Out-of-the-box offloading inference [106]. (2) Trends for more RAM [98] and tiny CLIPs [119]. (3) Inference with quantization methods [38].

Compatibility Existing CLIP-based backbone in our method focuses on the classification task. However, apart from image classification, many different vision tasks are compatible with PromptFL, such as object detection [40], video understanding [140] and visual question answering [113] by changing different backbone models. This means that the system cost of PromptFL is shared by many tasks.

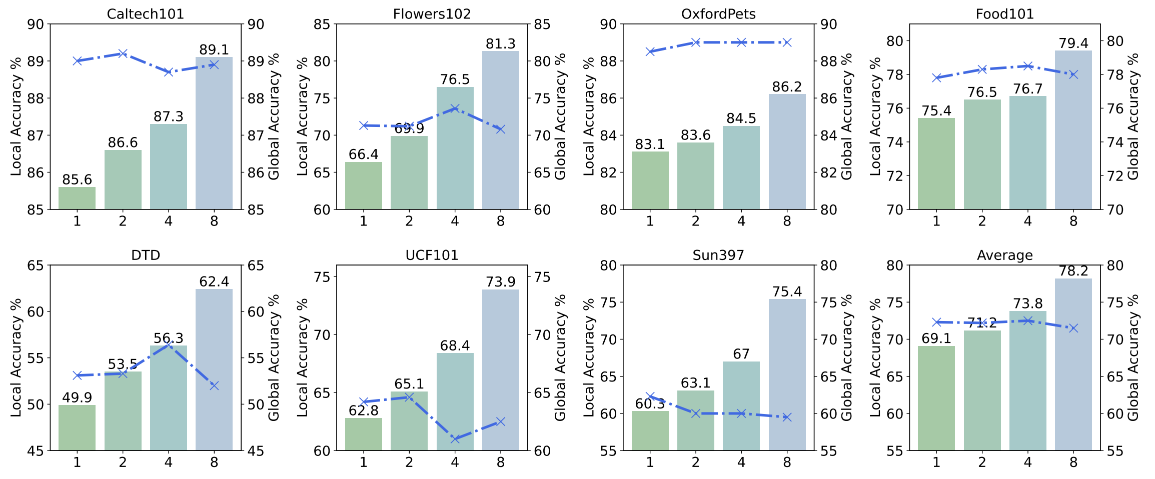

Figure 4.3: **Performance of PromptFL with different shots.** We deploy the experiments on seven datasets and record the average one. X-axis represents the number of shots for each class, ranging from 1 to 2 to 4 to 8. Bars represent the local accuracy and lines represent the global accuracy, which implies the personalization and generalization ability respectively. As the number of shots increasing, the local performance improves. However, global performance is not affected much by the variation of number of shots, as we can observe that the global performance remains stable as shots increase.

The prompt learner incurs these costs per personal task specific user subset requires. PromptFL is thus competitive in terms of economics.

4.3.4 Privacy Concerns

As we have outlined in the framework, PromptFL achieves to train prompts in concert with the federated server. Each participant user only needs to upload its local parameter update of the prompt learner rather than the raw data of images. Such a method avoids leakage of raw images, thereby better adapting to the privacy-preserving settings of the FL. On the other hand, the parameters of prompt learner only describes the correlation between classes and textual prompts, and do not directly contain any visual feature embeddings. Also, the parameters of prompt learner are static (*i.e.*, input-agnostic) across the training data. This is useful when faced with a server that wants to recover the raw data from an update [155].

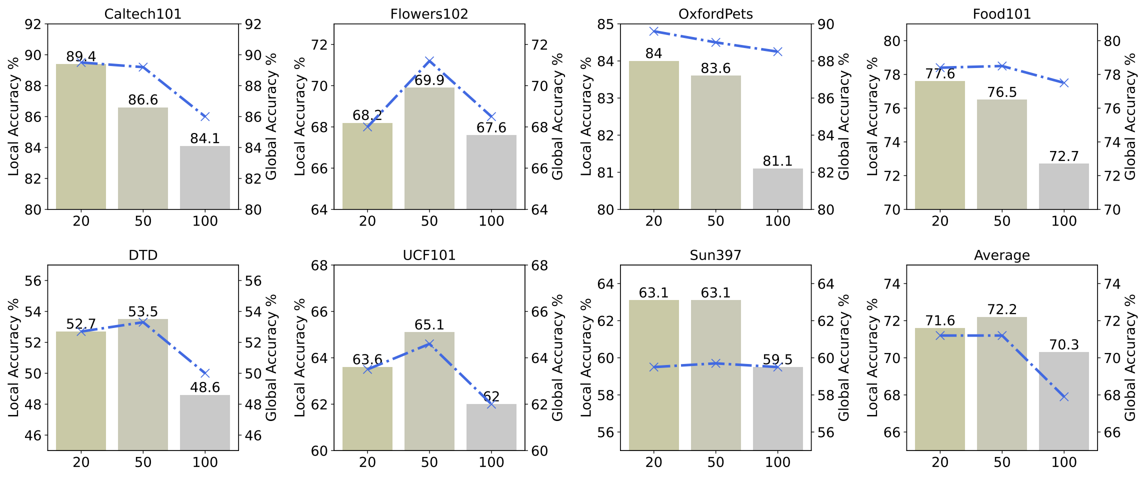

Figure 4.4: **Performance of PromptFL with different clients.** X-axis represents the number of clients during the training, ranging from 20 to 50 to 100. For each setting, we set the same participation rate of $r = 10\%$. The overall performance does not obey the strict increasing or decreasing trend as the number of clients changes. We observe that the personalization ability may bed affected when the number of clients gets larger, since the clients which do not engage in the training increase. Also, too few clients may lead to insufficient diverse of the training classes, thus lead to under representative of generalization ability.

Inference APIs While pre-trained CLIPs are available for download at the time of writing this paper, high-performance models in these domains are often costly to train. For example, the CLIP model trained on 400 million labeled images. The training process took 30 days across 592 V100 GPUs [105]. This would have cost million dollars to train on AWS on-demand instances. The value of these models and their exposure over publicly-accessible APIs make us rethink the framework of PromptFL. As illustrated in Figure 4.1, we hypothesize that the model APIs typically return low-dimensional outputs like confidence scores or logits, so information leakage is significantly reduced [29]. In such a case, the prompt learner can still be trained normally, because the CLIP backbone is kept frozen during the training process. The difference is that users need to make queries to the model APIs with their private images. Some lightweight secure inference techniques like [83] can be used in the framework to protect privacy.

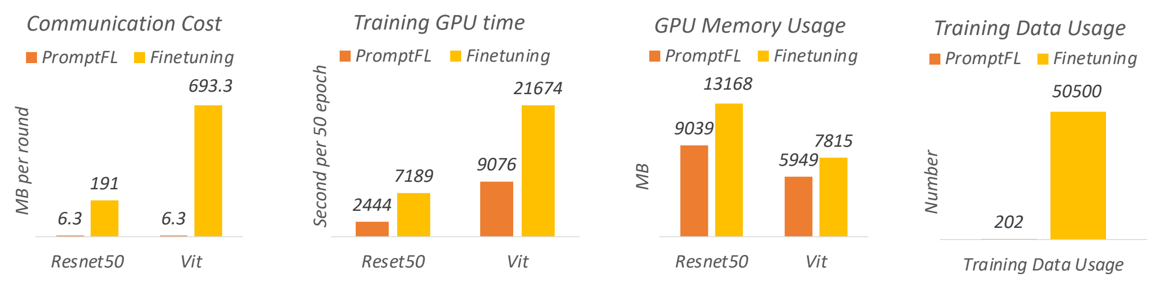

Figure 4.5: **Comparison of computation and communication cost of PromptFL and Finetuning FL.** We measure the communication cost by the size of uploaded data per round, and observe that finetuning FL takes up to 110 times of cost more than PromptFL. Furthermore, finetuning and training from scratch take 2 to 3 times of round more than PromptFL for training, which exacerbate the communication expenses. We also utilize GPU memory usage, training GPU time and training data usage to evaluate the computational cost. Training GPU time is calculated by the time of training 50 epoch and training data usage is reported by training food101, which we can observe that finetuning require 250× more than PromptFL. We can see that PromptFL surpasses the existing framework in the entire aspects of communication and computation efficiency.

4.4 Experiments

Our experiments aim to answer the following research questions that are important for the practical deployment of FL methods, while also contributing to our understanding of the PromptFL paradigm.

- Is PromptFL able to train a competitive performance in FL as compared to which have been the de-facto method on image classification tasks?
- Is PromptFL capable of handling heterogeneous data distributions (*a.k.a.* non-IID settings) across clients?
- Is PromptFL competitive with the de-facto method in terms of computational communication overhead?
- What is the difference between PromptFL and the fine-tuning of visual pre-trained models in FL?

- What practical tips help the service provider and participants deploy PromptFL in FL?

4.4.1 Experimental Setup

Datasets We select a representative collection of recognition datasets used in CLIP as our benchmarks. **General Objects:** Caltech101 [33] for general object detection. **Fine-grained Categories:** Flowers102 [92], OxfordPets [96] and Food101 [7] for fine-grained classification from diversified categories. **Action Recognition:** UCF101 [120]. **Texture Classification:** DTD [19]. **Scene Recognition:** Sun397 [139] for scene recognition.

Baselines As compared to our proposed PromptFL, we choose current representative framework in FL, FedAVG, by updating and averaging the model weights collaboratively among server and clients. We compare both training from the scratch and fine-tuning with pretrained models as our baseline method. We select the most prevailing models, Vit_b16 and Retnet50, as our backbone in both our image encoder of PromptFL and the corresponding backbone in the baseline method.

Fine-tuning *vs.* Prompting How does the prompting differ from the existing adaptation method in FL? Currently in vision, the standard adaptation method is fine-tuning. Therefore we consider fine-tuning as the de-facto way of adapting visual pre-trained models in FL. Fine-tuning is highly flexible in its usage: it can adapt the pre-trained models to new input domains or new tasks with different output semantics. Yet it also requires some level of access to the pre-trained models: often entire parameters. Unlike fine-tuning, prompting adapts the inputs to a pre-trained model by modifying the model's inputs. This opens up unique applications: the input-space adaptation puts control in the hands of the FL user; FL users only need to find the prompts, they don't need to control the pre-trained model itself while training and testing. In this way, FL users can provide adapted images and prompts to an online API that can only operate on their inputs. On the other hand, fine-tuning is typically conditioned on inputs. Its update also directly contains some embeddings of visual feature information. In contrast, the prompts we explore in this paper are input-agnostic across the training data. So the prompting can prevent leaking of user's private information from FL update to a certain extent.

CLIP PromptFL For CLIP, an image-language model, PromptFL organizes users to collaboratively learn prompts as the CLIP's output transformation function. Given a frozen pre-trained CLIP $\mathcal{F}$ and a task dataset $\mathbf{D}\{(\mathbf{x}_m, \mathbf{y}_m)\}$ across clients, the target of PromptFL is to learn a single, static, task-specific prompting f_{prompt} on class space parameterized by [prompt vectors]. Image classes are represented by labels (*e.g.*, 'panda') which are then prompted (*i.e.*, '[prompt vectors][panda]') to specify the context of the user's task. We follow CLIP's protocol and compute the cosine similarity of the embeddings for each class, normalized to a probability distribution via softmax. The class with the highest probability is selected as the model output. The prompting is added to the class space to form a prompted output $\mathbf{y}+v_f$. During training, PromptFL will maximize the likelihood of the correct label $\mathbf{y}$,

$$\max_{f_{\text{prompt}}} \mathrm{p}_{\mathcal{F};f_{\text{prompt}}}(\mathbf{y} + v_f|\mathbf{x}), \tag{4.2}$$

while the gradient updates are applied only to the [prompt vectors] v_f and the CLIP parameters $\mathcal{F}$ remain frozen. During validation, the optimized prompt is added to all test-time classes, $\mathbf{D}_{\text{test}}\{(\mathbf{x}_m, \mathbf{y}_m + v_f)\}$, which will be then processed through the frozen $\mathcal{F}$.

Training Details To validate the effectiveness of our method, we compare the performance of PromptFL with existing framework by 1) training the collaborative model from the scratch and 2) fine-tuning the full model with pretrained weights. We evaluate the performance on seven representative datasets used in CLIP across various categories like general objects, fine-grained classification, action recognition, texture classification and scene recognition. We report the performance with two representative and influential backbone, Resnet50 (38.3M parameters) and Vit_b16 (86.6M parameters). All experiments are conducted with Pytorch on GeForce RTX 3090 GPU. Training is performed with SGD with 0.001 learning rate.

For the evaluation metrics, we select three aspects to assess the performance of each method, 1) representative Top-1 accuracy on the test set, 2) F1 score to measure the weighted and unified average of precision and recall, which is more useful especially on unbalanced class distribution, 3) as well as the computational and communication cost reported in Fig. 4.5. We presuming that higher result on accuracy and F-1 score as well as lower result on computation latency will lead to better a framework, detailed comparison results show the superior if PromptFL in Tab. 4.2 and Tab. 4.3.

Main Results Tab. 4.2 and Tab. 4.3 measures the overall performance of PromptFL against existing framework from the perspective of two data distribution settings. 1) For the iid setting, each client shares the same classes, and the shots for each class on client is identical. We can see that from 4.2, PromptFL obtains superior results with similar or better accuracy and F1 value, but with only 0.01% $\sim$ 0.1% learnable parameters with the iid setting. Specially, for Vit-b16 served backbone, PromptFL surpasses the alternatives over the average across benchmarks with only 1 % 100 of the learnable parameters compared to the others. While for the Resnet50, although PromptFL does not achieve the optimal performance one some datasets, the gap is negligible. 2) For the extreme non-iid setting, each client owns the independent and non-overlapping classes. We can observe that from Tab. 4.3 that PromptFL achieves competitive performance on both accuracy and efficiency, and outbeats the existing framework comprehensively across all benchmarks by a large margin under the with the non-iid setting. Superior outcome on both settings manifest the advantage of our proposed PromptFL. What's more, the outstanding generalization ability exhibited in PromptFL under the non-iid scenarios further validate the effectiveness of our method. We further analyze the ability with the non-iid setting in the following sections. On the contrary, existing framework shows miserable stability when encountering shifted class distribution other than unified mode by observing the Tab. 4.3.

Data Distribution Analysis After obtaining the decent performance in both extreme iid and non-iid settings, we hope to further testify the stability of PromptFL and figure out the impact of different data distribution on clients to the performance of PromptFL. Inspired by the previous personalized work, we consider the pathological non-iid setting in our experiments. Here we set $n = 50$ clients with $r = 10\%$ participation. To observe the intermediate status, we select a fixed number p of classes on each client, ranging from 5 to 10 to 20, which means that random number of p classes appears on each client. Fig. 4.2 reports the general test accuracy and F1 with corresponding distribution. From the result, we observe that as the number of classes on each client increases, the performance of both test accuracy and the corresponding F1 value improves. The observation implies that the lack of certain classes may empire the the overall performance, which is in consistent with our training logistic. We also notice that as the data for the whole training system reaches sufficient status, the performance becomes stable, as we can see that the gap decreases as the number of classes large enough.

Impact of number of shots Following the few-shot evaluation setting adopted in CLIP, we further explore the effect of number of shots within PromptFL. We select fixed number of shots on each client from 1, 2, 4, 8 during the training process and validate the performance with corresponding test sets. We not only observe the generalization ability of PromptFL, but also place significant emphasis on the personalization aspect. We record the two indicators as global accuracy and local accuracy. From the result in Fig. 4.3, we observe that as the number of training examples per class increases, personalized local performance of PromptFL enhanced. However, general global performance remains stable as the number of shots fluctuates. The observation implies that the number of training data only influence the personalization performance of the local model, while has little impact over the generalization ability.

Comparison with different clients Further, to explore the possible impact caused by different clients, we further study the performance of PromptFL with different clients from 20 to 50 to 100, with the non-iid data distribution and $r = 10\%$ participation for each mode. We set the fixed shots for different mode, here we set $s = 2$. From the result in Fig. 4.4, we observe that for different number of clients, performance with relatively large clients will be harmed. This phenomenon is more server in the local personalized data performance other than the general global performance. The reason is that as the number of clients becomes larger, the number of clients which may not be chosen during the training increases. Thus, the local performance is more likely to be affected. On the other hand, for some tasks such as fine-grained categories, action recognition and texture classification, limited clients implies shortage of available data sources which may restrict the diversity of of training data. Thus, the general performance in some cases reaches unsatisfactory with fewer clients. However, the number of clients will not influence the performance trend caused by different data distribution or number of shots as shown in Fig. 4.3 and Fig. 4.2.

Computation and Communication Cost Analysis We also analyse the efficiency of PromptFL with regard to the computation and communication cost during training. We measure the communication cost by the size of uploaded data per round, and the total round to be transmitted. For the computation cost, we calculate the GPU memory utilization and training GPU time for given steps. Fig.

4.5 shows the comparison between existing finetuning framework and our proposed PromptFL. We observe that PromptFL can save at most 110 times communication cost per round compared to existing prevailing method, let alone that PromptFL takes half of rounds to reach convergence, which makes a wider disparity in communication cost between them. As for the computation cost, we report the comparison of GPU time as in the same given steps, where PromptFL remains outperform existing framework around 3 times. Further more, there is huge advantage that PromptFL consumes far less GPU memory during training, which can alleviate the system burden in practical.

4.5 Chapter Summary

Overall, there are many unknowns about PromptFL and this paper sets out to investigate its feasibility. In summary: (1) We demonstrate the system feasibility of PromptFL on modern hardware, in terms of overhead in communication, training, and inference. (2) We show that PromptFL keeps data on each device private, aiming to learn global prompts updated only by communicating gradients rather than the data itself, and thus not less private than FL. (3) We implement a proof-of-concept in the framework, spanning a range of popular image classification tasks. We find PromptFL to be competitive with strong FL baselines.

Chapter 5

*p*FedPrompt: Learning Personalized Prompt for Vision-Language Models in Federated Learning

5.1 Introduction

User modeling has been widely employed in Federated Learning (FL) by collaboratively capturing the latent characteristics of users from their behaviors with the exchange of locally obtained parameters [138, 60]. Meanwhile, such cooperative ecosystem has been applied in various scenarios to realize benefits, including recommendation [143], medicine [101], and finance [84].

However, with the significant increase in user data and model capacity, the communication and computational overhead generated by the FL co-modeling process will become increasingly unbearable for users [104]. Even worse, when the model is large, achieving the model's performance inherently requires the user to expose copious amounts of private data to the system [26]. This private information can be recovered from the exchanged parameters or intermediate results, raising potential privacy risks [90, 89, 155].

Fortunately, as pretrained vision-language models like CLIP [105] show great potential in learning representations, a recently proposed method called Contextual Optimization (*CoOp*) [152] introduces the concept of training prompt for adapting pretrained vision-language models. Based on the lightweight nature of this adaptation, researchers [44] have shifted the paradigm of *CoOp* to FL to overcome the problems outlined above. Their core idea is to use *CoOp* at each client to convert context words in prompt into a set of learnable vectors, and to optimize prompt via standard FL algorithm. According to [81, 152], activating the pre-trained knowledge via training prompt is both data- and parameter-efficient, thereby greatly benefiting FL over existing frameworks in terms of computation, communication, and privacy.

Although using prompt in FL to activate the pre-trained knowledge is a promising direction, a major challenge for deploying such approaches in FL is the heterogeneity of users. In this paper, we show that current prompt training is essential to model the user consensus. When the learned consensus is applied to the user's task, the significant gap between them will reduce the effectiveness of user modeling [149, 73, 75]. Research over the past few years has applied personalized FL (pFL) approaches to customizing models for heterogeneous users. These model-based pFL methods can be categorized into four types: local fine-tuning [17, 124, 135], parameter decomposition [4, 20, 10], regularization [74, 47, 48, 123], and clustering [57, 148]. We investigate a range of vanilla methods by directly applying ideas of personalized methods in the paradigm of pre-trained models and prompt. Such vanilla methods easily inherit advances of pFL, yet are unable to capture the multimodality of vision-language models, thereby leading to insufficient personalization and performance.

As the first attempt to learn personalized prompt in FL, we propose p`FedPrompt`, which takes advantage of the multimodality of vision-language models through two components. Specifically, the Global User Consensus (GUC) component allows full exploration in the word embedding space by globally optimizing continuous vectors, which facilitates the learning of general user consensus. The Local Feature Attention (LFA) component leverages a local personalized attention module by interacting with the spatial visual features in the visual space to dynamically lookup user-relevant features, which adapts the consensus knowledge encoded in GUC by feature retrieval. By incorporating the knowledge retrieved from GUC and LFA, the learned prompt turns out to be personalized according to users' features, so that the user achieves improved accuracy in practical FL classification tasks. Since FL does not exist baseline against any such personalized approach, we implement p`FedPrompt` and other pFL methods in the framework as baselines. Extensive experiments spanning a range of popular image classification tasks are conducted under the FL setting. We find that p`FedPrompt` beats baselines with competitive and robust performance. To summarize, the main contributions of this paper are four-fold:

- We find that current prompt training in FL is essentially to model the user consensus and lacks adaptation to user characteristics. We thus propose the problem of learning personalized prompt in FL (see figure 5.1).
- We survey existing model-based approaches in pFL and adapt them into the prompt training manner. We find that these existing personalized techniques cannot capture the multimodality of vision-language models, thereby leading to insufficient personalization and performance.
- To unleash the multimodality, we present p`FedPrompt`, which learns user consensus in linguistic space and adapts to user features on each client in visual space respectively. By incorporating the knowledge retrieved from multimodality, the challenge of user statistical heterogeneity is addressed.
- We evaluate p`FedPrompt` against the existing personalized techniques on widely-adopted datasets. Extensive experiments and ablation studies demonstrate the

superiority of our methods.

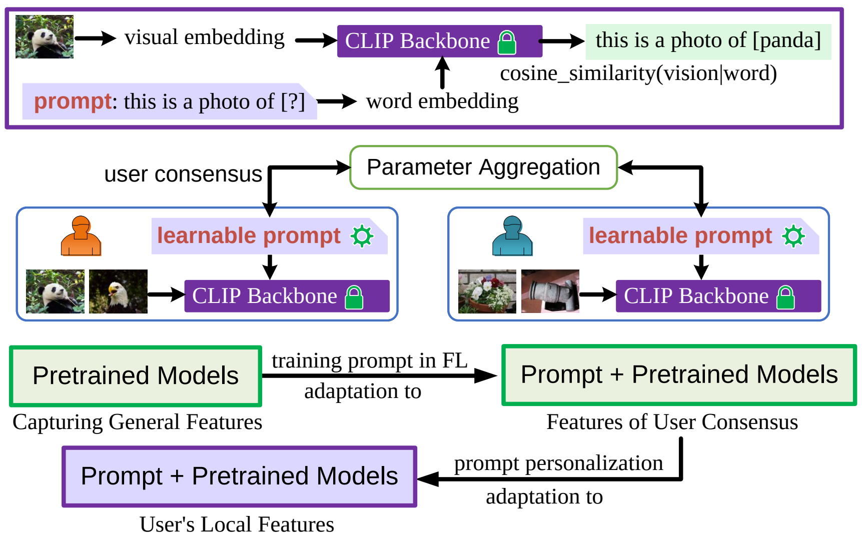

Figure 5.1: Stages of using pre-trained models with prompt in federated learning: (1) pre-trained vision-language models contain general knowledge that is transferable across a wide range of user modeling; (2) prior work activates the knowledge of pre-trained models by training prompt in the word embedding space so as to model user consensus; (3) our work aims to personalize prompt and further adapt the user consensus to the user's local features.

5.2 Preliminaries

5.2.1 User Heterogeneity

The fundamental challenge in addressing user heterogeneity is the presence of non-IID data [60], so we begin by investigating this issue and highlight potential mitigations. While the meaning of IID is generally clear, data can be non-IID in many ways [73]. The most common sources of non-IID data are due to each user corresponding to a particular device, web service, geographic location, and/or time window. For example, users in different regions may have very different disease distributions. There may be more skin cancer patients in southern hemisphere countries than in the northern hemisphere due to the ozone hole. Thus, the label distribution varies

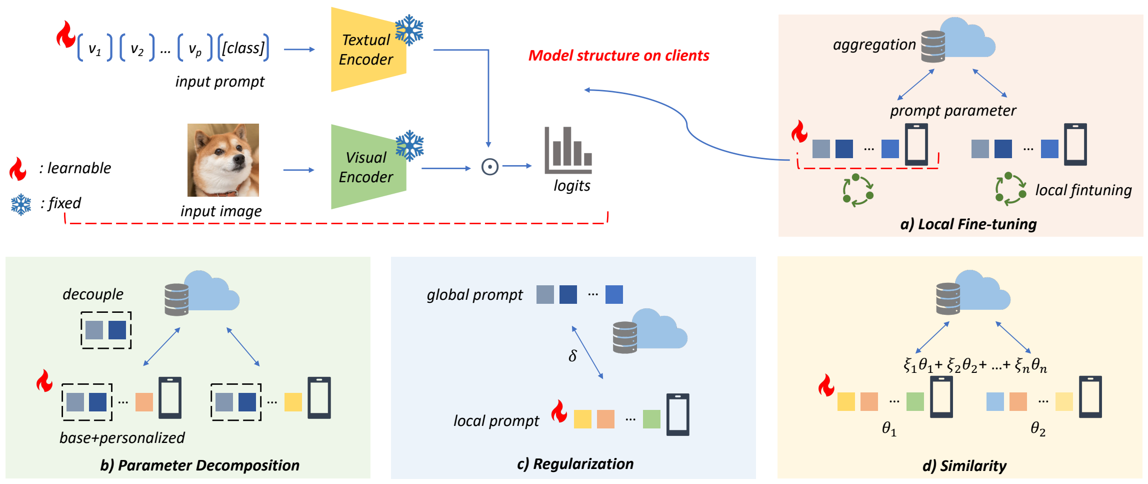

Figure 5.2: **Illustration of baseline methods for personalized prompt for vision-language models in federated learning.** The left above part shows the detailed structure of models on clients, which contains textual and visual encoders which are frozen and prompt which is learnable. To simplify the illustration of client model on for each method, we only utilize the learnable prompt to represent. Four personalized prompt learning techniques are introduced: a) local fine-tuning of prompt performed after obtaining global prompt, b) base vectors are aggregated while personalized vectors update locally, c) regularization is performed between global prompt and local prompt, d) clients relationship is leveraged for better personalization.

from user to user. Another example is that users have different writing styles. In this case, the feature distributions of the users are different. In this paper, we consider differences in the data or feature distribution on each user. According to previous research [54, 62, 75], non-IID data settings reduce the effectiveness of user modeling in FL.

5.2.2 Personalized Federated Learning

Federated Learning. The term *federated learning* was introduced by [87]. In a centralized setting, the federated server initially sends global model parameters to each user. Then the server aggregates the user's parameters and transmits the updated model back to each user. In addition to centralized federated learning, there are also some implementations of federated learning based on decentralized frameworks, where the aggregation of parameters occurs in some users [112, 68, 55]. The utilization of stochastic gradient descent (SGD) [21, 8] in FL makes it prone to face statistical challenges, since IID sampling of the training data is important to ensure the unbiased estimate of the full gradient [107]. In practice, it is unrealistic to assume that each user's local data is always IID.

Personalized FL. In recent years, personalized federated learning has received increasing attention due to its potential in handling user statistical heterogeneity. The core idea of model-based pFL is to produce customized model structures or parameters for different users. Existing model-based pFL methods can be categorized into two types according to the number of global models applied in the server, i.e., single global model, and multiple global models. Single global model type is a close variant of conventional FL algorithms like FedAVG [87], that combines global model optimization process with additional local model customization, and consists of three different kinds of approaches: local fine-tuning [17, 124, 135], parameter decomposition [4, 20, 10], and regularization [74, 47, 48, 123]. These pFL methods apply a single global model and thus limit the customized level of the local model on the user side. Some researchers recommend training multiple global models on the server, where users are clustered into several groups according to their similarity and different global models are trained for each group [57, 148, 126]. We will discuss more details in section 5.3.1 how to directly apply existing pFL methods to prompt training.

5.2.3 Prompted Vision-Language Models

Vision-Language Models. The most trendy vision-language models like CLIP [105] and ALIGN [59] are neural networks pretrained on hundreds of millions of image and caption pairs. These models encode images and captions separately as vectors, enabling users with visual modality samples to retrieve, score, or classify samples from textual modalities. In other words, these models extend the knowledge of classification models to a wider range of things by leveraging semantic information in text.

Prompt Training. The pretrained vision-language models like CLIP consist of an image encoder and a text encoder to predict the pairing relationship between images and texts. Therefore, these models can be converted to an image classifier. As shown in figure 5.1, the users may convert all [class] to prompt such as “this is a photo of [class]” and predict the caption class the model estimates the best pairing with the given image. Previous research has involved prompt engineering [35, 77, 118, 146], in which human engineers or algorithms search for the best template for the classes. Prior work [44] proposes a federated prompt engineering framework and optimizes the prompt collaboratively via standard FL algorithm. The federated server aggregates only the parameter updates of the prompt across users, and keeps the CLIP backbone frozen locally. Therefore, using prompt training in FL incurs significant less computation and communication overhead than conventional FL. Nevertheless, the current prompt training in FL is essentially to train the user consensus (see figure 5.1). Different from previous research, our work aims to personalize prompt and further adapt the user consensus to the user's local features.

5.2.4 Attention Mechanism

Attention was first presented in the [5] and later emerged from [132] as an important component in the transformer architecture to decouple the long-range dependency of sequences, in the field of neural language processing. Nowadays, attention mechanism has developed vigorously in the field of computer vision by adaptively weighting features according to the importance of the input, and has shown its advantage in deep feature representation for visual tasks [42]. Other than the above training-based attention mechanisms which aims to select the important channels [56], branches [15] or spatial regions [11] inside the neural network, we inspire by [41] and propose a non-parametric attention module to capture the global data context for the local adaptation.

5.3 Prompt Personalization

In this section, we first investigate the under-explored methods of how to apply existing advances of pFL (as referred to in section 5.2.2) to prompt training in a straightforward manner. Unfortunately, these vanilla methods cannot capture the multimodality of vision-language models, thereby leading to insufficient personalization. We then present p`FedPrompt`, which can unleash and incorporate the knowledge retrieved from the multimodality.

5.3.1 pFL – Straightforward But Insufficient

Local Fine-tuning. *"FL training + local adaptation"* is usually regarded as a simple yet effective personalization paradigm by the FL community [124, 60, 86]. After obtaining a collaboratively trained global model, each client adapts their local model through additional training with local datasets. Recently, the significance and effectiveness of this two-step paradigm have been brought up and emphasized by [17].

Similar in our case, when learning on heterogeneous data, all the clients train collaboratively by aggregating only the parameters of prompts but freezing the corresponding textual and visual backbone. After reaching a global user consensus prompt, each client fine-tunes the global prompt with its own few-shot data and obtains a personalized prompt. Personalized prompts are utilized with previously frozen backbones for further inference.

Parameter Decomposition. Parameter Decomposition is an architecture-based approach which aims to address the personalization problem by decoupling the personalized parameters from the global ones. [4] believes that deep learning model can be divided into two parts, *"base layers"* and *"personalization layers"*. Base layers are uploaded to join the formation of global model, while the personalized layers are kept locally by each client. [20] shares the same idea with different training procedures.

Inspired from [4, 20], here we intend to achieve personalization by viewing the learnable vector as *base + personalization vectors* and intend to decouple the personalized one from the base one. We presume that the former vectors act on common effects and intend to lead to a general performance, while the latter vectors which next to the class token emphasize on the specific performance related to the labels. Thus, during each iteration, we only transmit and aggregate the parameters of base vectors to the server and leave the personalized vectors on the local.

Regularization. Regularization is always employed in controlling the model expression ability during the training process [64]. In federated learning, regularization-based techniques are alleviated to address the client shift problem due to data heterogeneity by controlling the relationship between clients and global model. [74] introduced a proximal term on the local objective function to effectively limit the capability of local updates by restricting them to the current local model.

In our context, we aim to maintain the general instructive ability of prompt, but also allow them to approach the performance of their own local data distribution. Thus, we apply the method of [74] on learnable prompt by restricting the update of local prompt to not deviate too much from the current global prompt.

Similarity. Other than the above methods, similarity-based approaches are commonly used by leveraging the relationship and data distribution between clients. [57] propose a similarity-based mechanism to enforce that FL clients with similar data distributions collaborate intensely with each other, while clients with different data distributions have less impact on each other. Specifically, in each iteration, each client will maintain a cloud model which is the linear combination of the other clients, after obtaining the new model each client will perform local training with its private data.

Here we follow the idea of weight combination in prompt learning. Specifically, each client obtains a personalized prompt as a linear combination of the other local prompts, $u_c = \xi_{c,1}\theta_1 + \cdots + \xi_{c,m}\theta_m$ where $\sum_{m \in C} \xi_{c,m} = 1$. C is the set of local prompts, ξ is the coefficient that should be applied on θ, and θ is the weight parameter of other local prompts. For each round we obtain the above new prompt for each client and then update them locally, we perform this two-step interactively.

Limitations. The above methods are novel personalized prompt attempts for vision-language models when encountering data heterogeneity. However, problems may be encountered when transferring the setting from traditional model architecture to learnable prompt. First, ***parameters are few***. Compared with the backbone model, the amount of the learnable parameters is very small, which lets us think if the above personalized techniques suit well for models with small parameters? Second, ***shots are few***. Few-shot learning is employed in prompt learning instead of the traditional large amount of data, which might incur poor effect to the techniques that are data-driven. Third, ***two modalities exist***. Other than the single modality which only trains for images, vision-language models leverage the alignment of both textual and visual modality to enhance the performance of visual tasks with zero-shot or few-shot applications. However, the existing approaches only focus on the update of the prompt. i.e., the input of the text encoder, which raises a question that if it is enough to only adapt the single modality. Based on the above thinking, we employ several experiments in Sec. 5.4 and propose the following *p*`FedPrompt` approach.

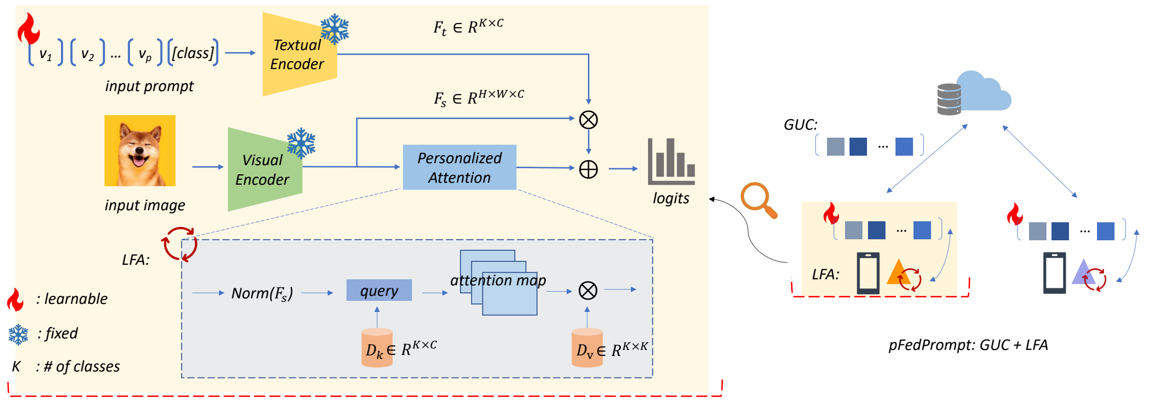

Figure 5.3: **Illustration of *p*FedPrompt of personalized prompt for vision-language models in federated learning.**The right part shows the workflow of *p*FedPrompt and the left part shows the detailed topography and pipeline of local model on the client. During the training process, only learnable prompt on each client is uploaded to capture the global user consensus. After obtaining the global prompt, textual encoder on each client is leveraged to generate the common textual features. On the other hand, each client maintains a non-parametric personalized attention module respectively, and combines with the visual encoder to generate the local personalized spacial visual features additionally. In this way, GUC and LFA work together to achieve superior performance for all clients under the heterogeneity setting.

5.3.2 *p*FedPrompt – Unleashing Multimodality

Motivation. As we survey the existing approaches on vision-language models in federated learning, several attempts have been explored to capture user consensus through prompt training [44, 137]. However, user characteristics, especially heterogeneous data distribution with real-world scenarios have been neglected so far. What's more, we observe that all the attempts, including the personalized techniques above, are conducted with prompts according to the existing paradigm, leaving both textual and visual encoder fixed.

Although achieving adequate performance by collaborative prompt training during the learning process, we notice that visual feature has not been leveraged for adaptation at all. This finding leads us to presume that there is still a room for improvement since the semantic gap incurred by local dataset is much larger between visual features than the one between text features.

Design. To capture the general features for all clients as well as adapting to the local personalization on each client, we present p`FedPrompt`, which contains two parts, Global User Consensus (GUC) and Local Feature Attention (LFA). GUC is captured through the textual space by global optimization of the learnable prompt. After obtaining the GUC, local personalization is realized on each client with the help of LFA through parameter-free attention to capture the additional personalized features and merge them to the final logits. The pipeline of p`FedPrompt` is shown in fig 5.3, which unleashes the modality in vision-language models. We will introduce GUC and LFA in detail as follows:

Global User Consensus (GUC). Each client is given a pre-trained vision-language model, with a fixed textual and visual encoder. Instead of the hand-craft prompt in[105], we introduce a set of p continuous vectors with d-dimension to form a learnable prompt that can be optimized through training. Here we use $p = 16$ and $d = 512$ as the word embedding in the text encoder.

Let $g(\cdot)$ and $h(\cdot)$ be the feature extraction function of the image and text encoder, k denotes the number of classes and each $\mathbf{P}_i$ is derived from the prompt in the form of $[v_1][v_2]...[v_p][\text{class}]_i$, where $[\text{class}]_i$ is replaced by the word embedding vector of the corresponding i_{th} class label name. By forwarding the image-text pairs, each vision-language model will maximize the cosine similarity of the correct pairs and minimize the remaining incorrect pairs. The prediction probability on each client is computed as follows:

$$\mathrm{p}(\mathbf{y} = i|\mathbf{x}) = \frac{\exp\left(cos[g(\mathbf{x})|h(\mathbf{P}_i)]\right)}{\sum_{j=1}^{k}\exp\left(cos[g(\mathbf{x})|h(\mathbf{P}_j)]\right)}, \tag{5.1}$$

where P_i is the only part that can be updated during training. Each client optimizes local prompt for iterations between rounds.

After obtaining the latest updated prompt on each communication round, selected prompts will be uploaded to the server for aggregation. At each communication round $t + 1$, $\mathbf{C}_k$ is the set of selected clients in joining in this round, and k is the client index. The aggregated prompt $\mathbf{P}_{t+1}$ for each round can be expressed as:

$$\mathbf{P}_{t+1} = \frac{1}{n_k} \sum_{k \in \mathbf{C}_k} \mathbf{P}_{t+1}^{k}. \tag{5.2}$$

Global prompt after aggregation will be downloaded to each client from the server. After several rounds between server and clients, global user consensus is captured collaboratively.

Local Feature Attention (LFA). After achieving global user consensus through the textual part, we leverage the visual counterpart to adapt personalization on each client, which precisely makes use of the modality advantages of vision-language model. For each input image, we obtain the intermediate spatial visual feature $F_s \in R^{H \times W \times C}$ extracted by visual encoder, and leverage the visual feature F_s to interact with a non-parametric attention module for the additional personalized logits.

We propose an external non-parametric attention module named *local personalization attention*, which computes the attention between the input visual feature F_s and an external memory unit M. M contains two parts, M_k and M_v, e.g., the key-value pairs, as our prior knowledge. To directly compensate the semantic gap in visual feature, we regard M as a memory of the local few-shots training data. For M_k, we first reshape the intermediate spatial visual feature F_s from $R^{H \times W \times C}$ into a 1D vector sequence $R^{HW \times C}$, and then use the normalized features as our keys in M_k. And as for M_v, we use the corresponding ground-truth label L_{train} after one-hot operation as our values in M_v. Given K class and N shots images per class, the dimension of M_k should be $NK \times C$. To maintain a stable and negligible overhead, we disentangle the buffer size with shots number and reshape the dimension of M_k

to $K \times C$ on each client.

$$\mathrm{M}_k = Norm(Reshape(\mathrm{F}_{s(train)})), \tag{5.3}$$

$$\mathrm{M}_v = OneHot(\mathrm{L}_{train}). \tag{5.4}$$

After obtaining the external memory unit M, we calculate the pair-wise affinity between the input visual feature F_s and M_k to get the attention map A. To be concise, A is the additional attention map inferred from the affinity between the prior local knowledge and the current input features, which can be obtained through a query function. Here we use cosine similarity as our query function. Afterwards, the personalized feature can be generated as AM_v.

$$\mathrm{A} = cos(\mathrm{F}_s, \mathrm{M}_k), \tag{5.5}$$

$$\mathrm{F}_{personalized} = \mathrm{AM}_v. \tag{5.6}$$

Thus, the final logits can be expressed as the original logits obtained by the interaction between textual features $F_t \in R^{K \times C}$ and visual features $F_s \in R^{H \times W \times C}$, with the additional personalized features generated by the additional non-parametric attention:

$$\text{logits} = \mathrm{F}_s\mathrm{F}_t^t \cdot exp(t) + \alpha \cdot \mathrm{F}_{personalized}, \tag{5.7}$$

where α represents the weight for the additional personalized logits. If the local data generates a large semantic gap between the local and global prompt, the value should be large, otherwise, the value should be small. Specifically, in echo with the above consideration, the final logits is also composed of two parts: 1) the original logits represent the global user consensus (GUC) captured by the participation of collaborative trained prompt, and 2) the additional personalized logits realized by local feature attention (LFA) with the adaption of local data on each client.

5.4 Experiments

In this section, we numerically evaluate our proposed method in the scenarios of heterogeneous data distribution and conduct comprehensive experiments.

5.4.1 Experimental Setup

Datasets. We select 6 representative image classification datasets used in CLIP [105] as our benchmark, which consists of various classification tasks. *General objects:* Caltech101 [33]. *Fine-grained Categories:* Flower102 [92], OxfordPets [96], Food101 [7]. *Action Recognition:* UCF101 [120]. *Texture Classification:* DTD [19].

Models. As for the local vision-language model, we use the same architecture with CLIP [105], which consists of an image encoder and a text encoder for feature extraction respectively. We use ResNet50 [50] as the backbone for image encoder and transformer [132] as the textual encoder. To quick-adjust and exploit the capacity of the pre-trained vision-language model, we follow the predecessor [152] to keep the prompt learnable and the two encoder freeze instead the complete zero-shot inference in CLIP.

Baselines. Since personalized techniques for the vision-language model is under-explored when encountering the heterogeneous scenarios [44]. We absorb the concept in traditional pFL techniques and adapt them to the scenarios of vision-language model as our baselines. We compare our method with the existing PromptFL and five adapted baseline methods, e.g., Local, PromptFL+Finetuning [17], PromptPer [4], PromptProx [74] and PromptAMP [57], as introduced in Sec 5.3.1.

Heterogeneity Simulation. Combined with previous works setting and practical situations of few-shot learning in our scenarios, we consider two pathological Non-IID settings in our experiments. In the pathological Non-IID setting, each client will be assigned only a specified number of labels, e.g. 5 random labels as shown in Tab. 5.2. Practical Non-IID setting with specific data distribution among clients is also a common setting in traditional personalized federated learning, however, since few-shots are employed here, this setting is not applicable in this scenario. Concerning different clients number n, participation rate r and Non-IID data distribution, we simulate the following two settings: 1) $n = 10$ clients with $r = 100\%$ participation, each client shares a completely disjoint random class with each other. 2) $n = 100$ clients with $r = 10\%$ participation, and $S = 5$ random classes are assigned to each client, thus repetition will appear in each class among clients.

Implementation Details. We implement all the methods in Pytorch and all experiments are conducted on GeForce RTX 3090 GPU. We use SGD optimizer with 0.001 learning rate with all methods except FedProx, which uses the corresponding modified optimizer with the stated best hyper-parameters reported in the preceding works. We set a local epoch $E = 5$ for both cases, while for the global communication round, we perform a $R = 10$ for $n = 10$ clients case and $R = 30$ for $n = 100$ clients case. For the fine-tuning baseline, we conduct an additional adaption epoch $AE = 10$ for $n = 10$ clients case and $AE = 30$ for $n = 100$ clients case.

For the setting of soft learnable prompt, we introduce a set of p continuous embeddings of dimension d in consist of the [prompt vectors]. d is the same as the dimension of word embeddings in the text encoder, i.e., 512 by default. p is a hyperparameter specifying the number of embeddings. Here we use $p = 16$ vectors as the best case shown in [152, 44]. We also follow the precedent to place the class token in the end of the of the prompt.

Table 5.1: **Performance of *p*FedPrompt against adapted baselines on the pathological Non-IID Setting 1**: The table reports the average test accuracy according to six diversified datasets. Six baselines are selected for comparison. Among them, PromptFL [44] is the novel paradigm for FL with vision-language model and the other four of them are adapted from the latest *p*FL researches. Here we use the extreme Non-IID setting, where 10 clients are simulated here with $r = 100\%$ participation rate and non-overlapping class on each client, which means that each class only appears once among clients. The best score of each group appears in bold. Compared with the adapted baseline methods, *p*FedPrompt outperforms other methods across datasets.

Dataset	Caltech101	Flowers102	Pets101	Food101	DTD	UCF101
(Setting)	(10 clients, non-overlapping)					
Local Training	87.37 ± 0.44	70.14± 0.76	83.21± 1.30	70.43±2.42	44.23±0.63	62.53± 0.09
PromptFL [44]	89.70 ± 1.99	72.80±1.14	90.79± 0.61	77.31±1.64	54.11± 0.22	67.87±0.74
PromptFL+FT [17]	89.70 ± 0.25	72.31± 0.91	91.23±0.50	77.16± 1.56	53.74±1.36	66.36±0.65
PromptPer [4]	86.72 ± 1.45	72.11± 1.35	89.50±1.62	71.29± 1.87	50.23±0.82	65.81±1.42
PromptProx [74]	89.41 ± 0.55	66.40± 0.29	89.24± 0.41	76.24± 1.94	44.26±1.11	63.27± 1.20
PromptAMP [57]	87.31± 1.60	69.10± 0.13	80.21±0.44	74.48±1.71	47.16±0.92	62.37±0.81
pFedPrompt (Ours)	**96.54 ± 1.31**	**86.46± 0.15**	**91.84± 0.41**	**92.26± 1.34**	**77.14±0.09**	**86.22±1.02**

Table 5.2: **Performance of *p*FedPrompt against adapted baselines on the pathological Non-IID Setting 2**: The table report the average test accuracy corresponding datasets and methods as stated in Tab. 5.1. Each baseline method is recorded with their optimal performance. 100 clients are simulated here and $r = 10\%$ of clients are selected to participate in each round. 5 random classes are selected on each client, which means that same classes may encounter overlapping on different clients. The best score of each group appears in bold. Compared with the adapted baseline methods, pFedPrompt not only reaches supreme performance on the extreme case with 10 clients setting in Tab. 5.1, but also outperforms other methods with more general case.

Dataset	Caltech101	Flowers102	Pets101	Food101	DTD	UCF101
(Setting)	(100 clients, 5 random class)					
Local Training	85.50± 0.32	72.8± 0.59	85.50 ±0.63	77.51±0.29	55.06±0.38	66.80±0.74
PromptFL [44]	82.92± 0.43	69.08±0.74	84.49±1.06	73.35±1.11	52.49±1.59	66.56±0.22
PromptFL+FT [17]	84.45± 0.29	71.04± 0.57	85.49± 0.49	74.61 ±0.82	56.20±0.51	68.40±0.21
PromptPer [4]	82.19±0.61	69.52± 0.23	83.66±0.85	73.72±0.84	53.34±1.44	66.78±0.53
PromptProx [74]	85.52± 0.42	67.63±1.10	85.76±0.80	73.36±2.12	46.23±0.18	62.31±0.11
PromptAMP [57]	88.30± 0.71	75.01±0.91	**87.50±0.51**	77.70±0.36	57.30±0.46	69.80±0.51
pFedPrompt (Ours)	**92.24 ± 0.31**	**85.72±0.18**	87.31±0.32	**90.11±0.60**	**73.44±0.96**	**85.97±0.42**

5.4.2 Performance Evaluation

Comparison with state-of-the-art. To show the effectiveness of our method, we compare our proposed *p*`FedPrompt` with corresponding adapted state-of-the-arts methods across six representative datasets. And as stated in the above section, both the selection of the baseline methods and datasets aims to guarantee the generosity and comprehensiveness of our evaluation. Due to the newly emergence of prompt training in FL, personalized problems and techniques in this scenario have not been considered and developed yet, which causes the lack of corresponding baselines for comparison. To make up for this deficiency, we utilize various state-of-the-art personalized techniques and adapt them in the form of prompt learning as our baseline. To enhance the persuasiveness of our proposed method, we select approaches from diversified categories for adaptation, e.g., Local Adaptation, Parameter Decoupling, Regularization-based and Similarity-based approaches. As for the datasets, we cover several categories including general objects, fine-grained objects, action recognition, and texture classification.

We report the performance of *p*`FedPrompt` against baselines for the two heterogeneous setting in Tab. 5.1 and 5.2. All the baselines are performed under their optimized setting. In almost all cases, *p*`FedPrompt` strongly outperforms the alternatives. Comparing the two tables, Tab. 5.2 shares more classes between clients since that classes are randomly shared, while classes are fully independent between clients in Tab. 5.1. As a result, the performance gap is wider in Tab. 5.1 than that of Tab. 5.2 as local data distribution become more extreme. When more classes are shared between clients, individual client possesses a greater possibility to benefit with each other. However, even though dissatisfaction appears in other approaches when heterogeneity increases, our proposed *p*`FedPrompt` remains robust performance across datasets. Within the Table, we observe that, for the other personalized approaches, the performance deteriorates strongly as datasets type change from general objects to fine-grained objects or other specific tasks. While for our method, the degradation process is comparatively slow, which on the other hand verifies the effectiveness and robustness of p`FedPrompt`.

Analysis of Number of Shots. Under the setting of few-shots learning, we also want to find out how is the number of shots in training data will affect the overall performance. To analyze the effect of the number of shots, we vary the shots in [1, 2, 4, 8, 16]. In Fig. 5.4, we report six datasets and with each dataset, we record the performance of the five different shots setting. The horizontal axis shows the shots number and the vertical axis shows the average test accuracy. Heterogeneity simulation 2 is employed here.

We observe that in most cases, p`FedPrompt` already achieves promising performance when the number of training data is small, even within one shot. However, alternative approaches appear poor performance when shots are small. And as the number of shots increases, the corresponding performance will enhance gradually, but still can not beat the top-performance of p`FedPrompt`. Such phenomenon exactly verifies our concerns above that current personalization approaches only applies with the large amount training data. To be concise, p`FedPrompt` remains robustness against the variation of number of shots in comparing with the alternative methods, which exactly in accordance with the our purpose to propose a unique personalized technique which suits our particular few-shot scenarios.

Effects of Hyper-Parameter α. Apart from the above superiority of p`FedPrompt`, we also want to find out to what extent does the final performance benefit from the local feature attention. As mentioned above, α serves as the indicator to control the balance of the general user consensus(GUC) and local personalized feature(LFA).

Larger value of α denotes to integrate more knowledge from local personalized feature and vice versa.

Within each number of shots, we vary the value of α from 0.0 to 5.0, and select the best α value that can produce the best performance. Tab. 5.3 reports the best α with the according number of shots, we can observe that as the number of shots increase, the demand for additional local feature information gradually decrease, which means that global prompt tuning can capture more user characteristics when given more data. And when the number of shots is small, the overall performance may benefit more from the local feature attention, which from the side proves that the local personalized attention module plays an important part in adjusting the personalization within the few-shot learning behavior.

Effects of Buffer Size. To guarantee the additional memory overhead is limited and negligible, we resize the dimension of buffer size of M_k from $NK \times C$ to $K \times C$, which means that regardless of how many shots are utilized in training, the memory for the attention still remains for the same dimension of K, e.g, number of classes. To achieve this purpose, we aggregate and average the spatial visual feature of local training data to maintain that each class only exists one aggregated features as M_k. Tab. 5.3 shows the performance before and after reshape with different training shots of 1, 2, 4, 8 and 16. We can find out that even after reshape, the performance of *p*`FedPrompt` still outperforms other methods and the accuracy drop is negligible(around 1%) compared with the original one.

Computational Efficiency Analysis. We further analyse the computational overhead to ensure the efficiency of our method. We observe the required training epochs against the achieved test accuracy and compare it with the same local adaptation category method `PromptFL+FT`. However, unlike PromptFL+FT which needs extra

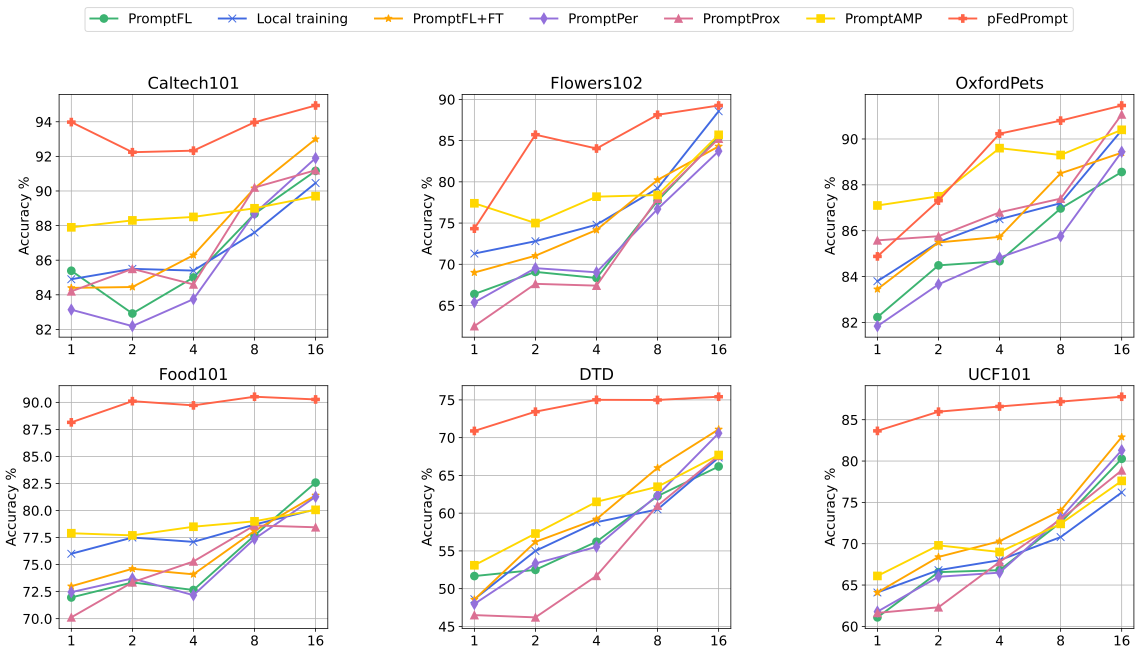

Figure 5.4: **Peformance of different personalized approaches over six datasets.** Average Local Test Accuracy is reported with different methods and number of shots. In each subplot, horizontal axis represents number of shots and vertical axis represents the corresponding test accuracy. We range the number of shots on each client from 1, 2, 4, 8 to 16. We observe that pFedPrompt strongly outperforms alternatives across datasets, as shown in the red line. Furthermore, when the number of shots decreases, the gap widens between pFedPrompt and other methods. Compared with the alternatives, pFedPrompt remains robust and outstanding performance against the variation of number of shots.

local training for the adaptation, pFedPrompt adapts instantly during inference, which do not incur additional training overhead but still achieve extraordinary performance, as shown in Tab 5.4.

5.5 Chapter Summary

Large pre-trained vision-language models have shown great potential in federated learning. However, challenges when encountering real-world problems like data heterogeneity have not been well-addressed. To solve the problem, we first explore the existing personalized technique and adapt them to the prompt training manner.

Table 5.3: **Ablation results on effect of hyper-parameter α and buffer size**. The best α for corresponding shots are reported. As shots decrease, model takes more advantage of the local personalized features for personalization. Compared with the performance before reshape of the buffer size, the decrease of average test accuracy after reshape is negligible.

♯ SHOTS	1	2	4	8	16
BEST α VALUE	3.77	2.55	2.06	1.08	0.59
ORIGINAL ACCURACY	93.98	92.24	92.33	93.97	94.94
RESHAPE ACCURACY	93.98	91.32	91.09	93.01	94.12

Table 5.4: **Efficiency Analysis for pFedPrompt**. Training overhead with the corresponding accuracy of pFedPrompt against the two baselines are reported. The adaptation overhead is negligible while gain is considerable.

Method	Global (round)	Local (epoch)	Accuracy	Gain
PromptFL	30	0	82.92	-
PromptFL+FT	30	30	84.45	+ 1.53
pFedPrompt	30	0	92.24	+ 9.32

As few-shot learning and relatively limited learnable parameters are employed here, available methods achieve inadequate performance in this particular scenario. To bridge the gap, our research proposes *p*`FedPrompt`, which uniquely addresses the above challenge by leveraging both the textual and visual modality at the same time. *p*`FedPrompt` learns the global user consensus in the linguistic space with the collaboratively updating of prompt and adapts to user features in visual space with the local personalized attention module. Since the personalized attention module is non-parametric, the additional overhead is negligible. By incorporating the knowledge retrieved from multimodality, the challenge of user statistical heterogeneity is addressed. Our research provides a novel insight and direction in addressing the personalization problem in this scenario, which made an important step forward to the development and application of pre-trained vision-language models in federated learning.

Chapter 6

Explore and Cure: Unveiling Sample Effectiveness with Context-Aware Federated Prompt Tuning

6.1 Introduction

In recent years, Federated Learning (FL) has experienced rapid growth as a decentralized machine learning framework that keeps user privacy intact by not sharing raw data with a remote server [87]. FL serves as an ecosystem for parties to derive benefits and has been applied in a range of scenarios including finance [84], medicine [101], and recommendation [143].

With the increase in user data and model capacity, the communication and computational overhead necessary for collaborative processes in FL will become increasingly burdensome for users [104]. Furthermore, when the model is large, achieving its desired performance typically involves disclosing a vast amount of private data to the system [26]. This private information can be extracted from the parameters or intermediate results exchanged, which poses a potential risk to user privacy

[89, 90, 155].

The promising performance of pre-trained vision-language models like CLIP [105] has spurred interest in learning representations. A new method, Contextual Optimization (*CoOp*) [152], introduces the training prompt to adapt pre-trained vision-language models. *CoOp*'s lightweight nature has shifted the paradigm for researchers to explore its use in FL [44, 43, 122]. They have demonstrated its considerable advantages over existing frameworks in terms of computation, communication, and privacy. The core concept behind this approach is to apply *CoOp* at each client to transform context words in prompts into learnable vectors and optimize the prompts using standard FL algorithms.

Previous studies, while promising, have overlooked a fundamental issue: *how does the prompt vector affect the behavior of pre-trained neural networks?* Our study delves into this issue and proposes that prompt training helps the prompt vector locate a previously learned task, rather than acquiring new knowledge. We support our argument by noting that prompt training is a data-efficient method, with performance saturation as the number of shots increases. Our experiments also demonstrate that federated prompting is a data-efficient but data-sensitive paradigm, making it crucial to carefully select participating data. The previous underestimation of the results of precisely steering neural networks is a key finding of our study.

Inspired by these findings, we propose a new framework, coined Context-aware Federated Prompt Tuning (CaFPT), which facilitates the retrieval process by conditioning on the examples capable of activating the most pertinent knowledge inside the pre-trained models. Our work focuses on using trained prompts to precisely locate the domain of previously learned tasks by utilizing a local data selection strategy based on *informative vectors*. These vectors identify the most informative direction in the weight space, enabling precise control over pre-trained neural network behavior and improving performance on the client task. We conducted experiments that demonstrate the ability to update and combine these informative vectors through arithmetic operations such as FedAVG, resulting in behavior that is precisely steered for multiple client tasks in vision-language models. To create informative prompt vectors, CaFPT samples from candidate dataset for each client based on their $\mathcal{V}$-*usable* information [142], a variational extension of Shannon's information theory. Such indicator measures how much guiding information that prompt can leverage in retrieving the knowledge within the pre-trained model. The resulting informative vectors offer promising robustness, making them a simple yet effective way to enhance the performance of federated prompting. Extensive experiments have shown the improvement and robustness our method can provide, leading to stable and strong performance across various benchmarks.

In this study, we contribute to the field by:

- We revisit the local pre-trained vision-language models (VLMs) as knowledge bases and explore the training mechanism behind prompt tuning. We argue that instead of acquiring new knowledge, prompt tuning retrieves previous learned knowledge inside the models, targeting on the information most relevant to the present task. This underscores the importance of the retrieval trajectory during the searching process. (§6.3)

- We propose that prompting is a data-efficient but data-sensitive paradigm, and classify federated prompt tuning into two categories, *context-unaware prompt tuning*, i.e, *PromptFL* and *context-aware prompt tuning*, i.e, CaFPT. Throughout the tuning process, samples act as the contextual foundation upon which the prompt can be conditioned, which determines the quality of retrieval during the process. (§6.4)

- Recognizing the crucial role of samples involving in the tuning process, we propose a context aware framework based on $\mathcal{V}$-information. Our experiments demonstrate that this strategy identifies the most informative direction in the weight space of the local task and extends its effect to multiple clients' tasks through operations like FedAVG, making it a simple yet effective method to improve the performance of federated prompting. (§6.4)

- We evaluate our method by conducting extensive experiments on various datasets following the standard federated setting of CLIP and *CoOp*. These experiments cover a range of visual classification tasks, such as generic objects, scenes, actions, fine-grained categories, and more. The significant improvement in results obtained from these experiments demonstrates that our method is effective in learning robust and comprehensive prompts in FL. (§6.5)

6.2 Preliminaries

6.2.1 Vision-Language Pre-trained Models

Recent years have witnessed rapid advancements in transformer technology [132], resulting in the pre-eminent role of pre-trained models in natural language processing (NLP) and computer vision (CV). Vision-language pre-trained models, which combine both modalities in large-scale models, have received considerable attention. Their pre-training methods can be categorized based on their objectives, which include reconstruction [72, 63, 52, 28], contrastive matching [105, 59], or a combination of both [70, 61]. The extensive pre-training of vision-language models on image-text corpora has led to the acquisition of universal cross-modal representations, which have translated to superior performance on downstream tasks. An illustration of this is CLIP [105], which uses 400 million image-text pairs for contrastive matching and has demonstrated remarkable results in visual recognition tasks [36, 152, 151, 13, 43], making it the primary focus of this paper. In addition to recognition, these models display immense potential in other downstream applications, including dense prediction [109], image generation [97, 91, 108], and action understanding [127].

6.2.2 Prompt Training

A pre-trained vision-language model like CLIP comprises of an image encoder and a text encoder to predict image-text pairs (i.e. Image-Text Retrieval). As such, CLIP can be adapted to function as an image classifier. The prompting function $f_{\text{prompt}}(\cdot)$ is used to alter the class captions (or labels) Y into a prompt $Y' = f_{\text{prompt}}(Y)$. Previous research has centered on designing prompts by utilizing the expertise of human engineers or using algorithms to identify the most suitable template [9, 146, 81, 118]. Our work is most closely related to methods of continuous prompting, which optimize continuous vectors in the word embedding space [77, 69, 130, 46]. Contextual Optimization (*CoOp*), a recently proposed continuous prompting method [152], is a straightforward yet efficient technique specifically designed for adapting CLIP for visual recognition tasks. Researchers [44, 43, 122] have also moved the *CoOp* paradigm to FL, revealing its superior performance in terms of computation, communication, and privacy compared to existing frameworks.

6.2.3 Federated Learning

Federated Learning, a term first introduced in [87], is a machine learning technique where a group of n clients, such as mobile devices, work together to train a model under the direction of a federated server, such as a service provider. The clients' training data remains on their local devices and is not shared [60]. The federated server coordinates the collaborative training process by repeatedly performing the following steps until convergence is achieved:

- **Client Selection.** Due to the unpredictability of client availability, for round t of federated learning, the federated server selects a small group of m clients

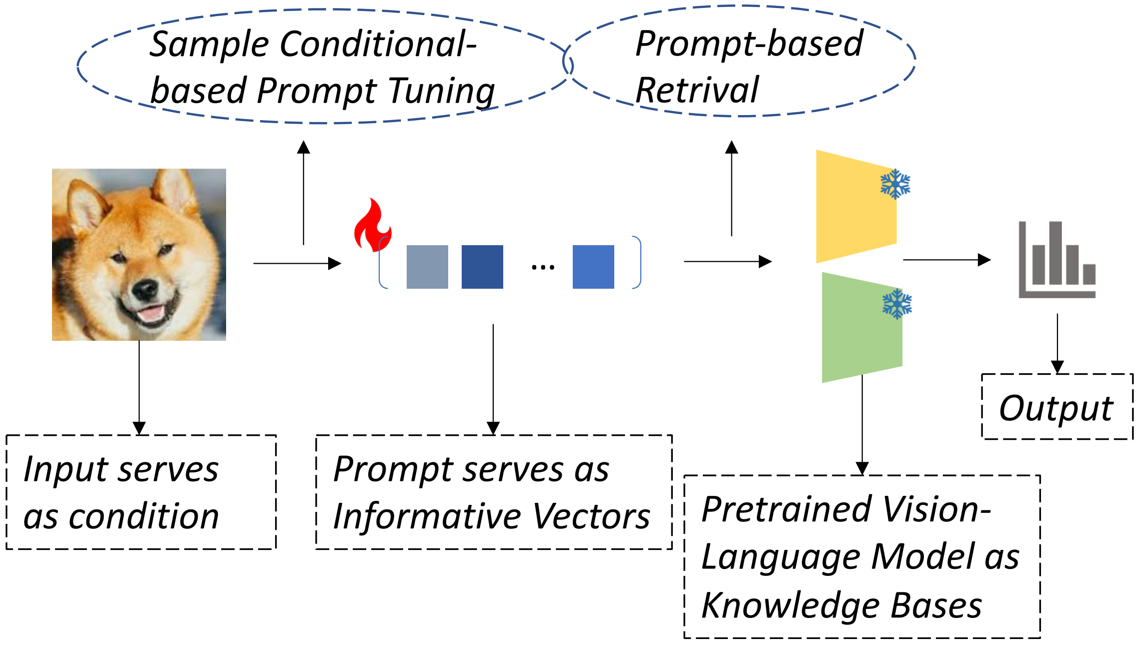
Sample Conditional-based Prompt Tuning
Prompt-based Retrival
...
Output
Input serves as condition
Prompt serves as Informative Vectors
Pretrained Vision-Language Model as Knowledge Bases

Figure 6.1: Explore the paradigm of prompt tuning mechanism on pre-trained vision-language models (VLMs). Prompt serves as the informative vectors to instruct the query direction for the knowledge retrieval. Samples serve as condition that contribute to the formation of informative vectors.

out of the total n clients that meet certain qualifications to participate in the learning process.

- **Local Training.** When notified of their selection for round t, each chosen client retrieves the current parameters $\mathcal{V}$ of the global model and a training program from the federated server. They then compute updates to the global model on their local training data by running the training program. Specifically, each client calculates the update θ by computing $\frac{\partial \ell(X,Y,\mathcal{V})}{\partial \mathcal{V}}$, where X, Y represent the batches of training data and corresponding labels, and $\ell(\cdot)$ refers to the loss function. FedAVG method [87] updates the model consecutively using multiple batches of local data, which can be several rounds of training and then shared. A common practice is to share the updated model $\mathcal{V} + \theta$, but this is equivalent to sharing the update θ only, since all the clients are aware

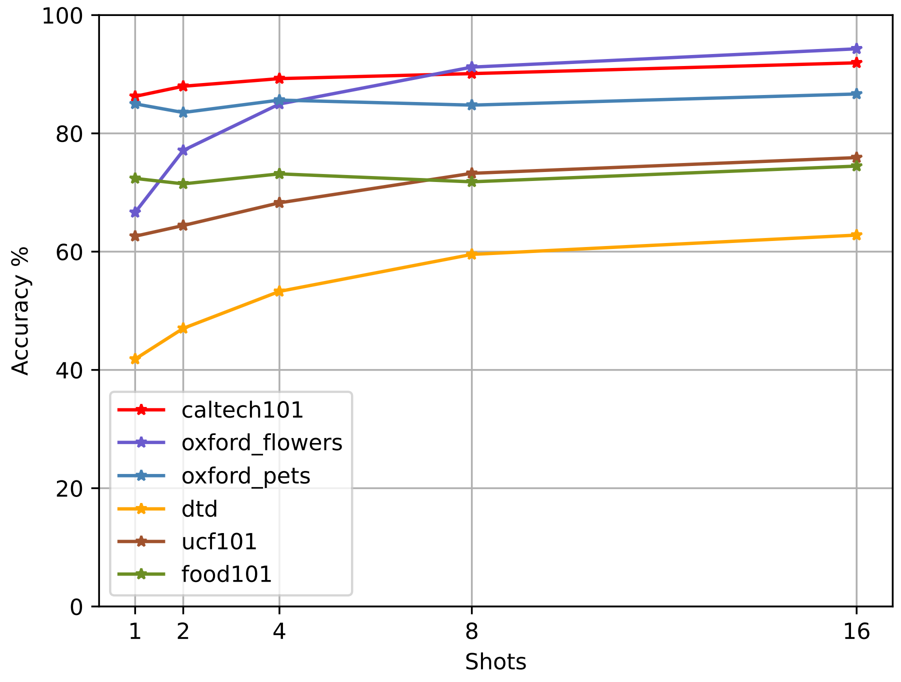

Figure 6.2: **Performance of prompt tuning with increasing shots.** We show the performance of prompt tuning as shots increasing across six different datasets. 1) The overall performance enhanced as the number of shots increasing regardless of the margin. 2) However, as the shots become larger, the improvement of the performance saturates. 3) Most datasets do not change much during the process.

of $\mathcal{V}$.

- **Global Aggregation.** After receiving local updates from m clients, the federated server combines these updates and updates its global model, and then begins the next round of learning.

How to adapt pre-trained vision-language models to the field of FL and improve downstream task performance becomes an emerging practice. Researchers [44, 43, 122] have adapted the *CoOp* paradigm to FL and optimized the prompts using the standard FL algorithm. The goal of federated prompt training is to learn a specific prompting function f_{prompt} on the word space, using the parameterized [prompt vectors], given a pre-trained CLIP $\mathcal{V}$ and a task dataset $\mathbf{D}(X, Y)$ distributed

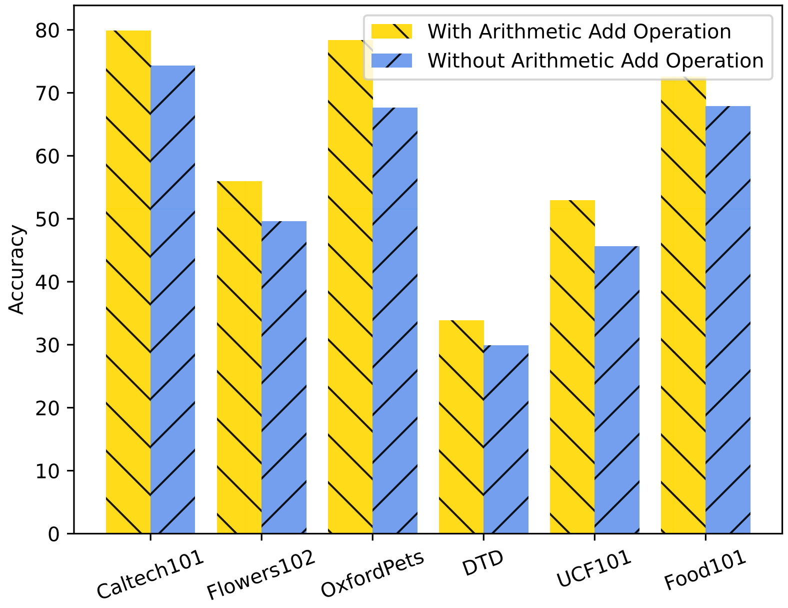

Figure 6.3: **Performance comparison between prompt vectors with or without arithmetic operation.** Yellow bar shows the test accuracy of prompt learning model when vectors incorporate with each other, while the blue one learns with individual tasks locally. We can observe that by engaging with the prompt vectors from other tasks, performance of their own enhanced.

across clients. Image classes, represented by labels such as "panda", are prompted by [prompt vectors][panda]. During training, FL aims to maximize the likelihood of the correct labels Y,

$$\max_{f_{\text{prompt}}} \mathrm{P}_{\mathcal{V};f_{\text{prompt}}}(Y+\Phi|X), \tag{6.1}$$

while the parameter updates are only applied to the prompt vectors Φ, and the CLIP's parameters $\mathcal{V}$ stay frozen. During validation, the optimized prompt is added to all test-time classes, $\mathbf{D}_{\text{test}}(X, Y+\Phi)$, which will be then passed through the frozen $\mathcal{V}$. While federated prompt training has shown impressive performance on downstream tasks, previous research has neglected to address a critical issue: *how do prompt vectors affect the behavior of pre-trained neural networks?* We seek to further investigate this problem.

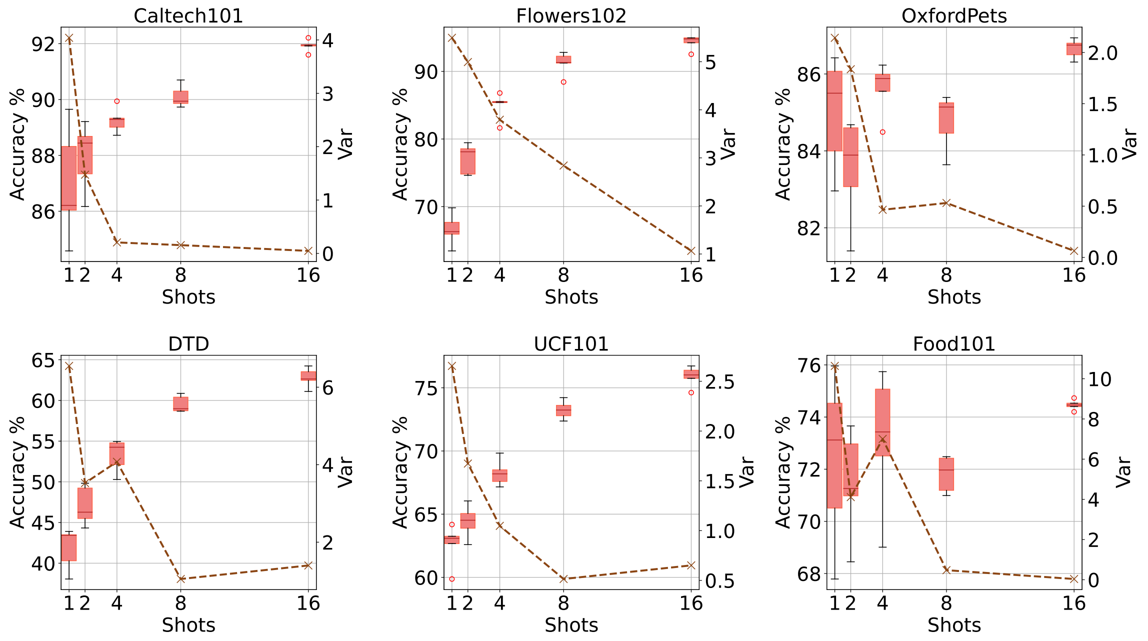

Figure 6.4: Distribution summary of model performance with random sampling. We display the performance distribution for groups of random sampled data. The orange box shows the values spread and the blue dash line shows the trend of group gap over shots. Model performance fluctuates when shots are small and gradually reaches stable when shots get larger.

6.3 How the Prompting Works

To understand how prompt works, we employed extensive experiments with various shots and datasets to investigate the operation principle of prompt during the learning and inference process. We achieve the following three observations from the experiments and come up with the thought-provoking inspiration: Other than traditional downstream tasks where training data aims to learn new knowledge, *prompt works as informative vectors to activate the existing knowledge inside the pretrained models.*

6.3.1 Observation 1: Performance Saturated in Prompt Learning

To examine how shot numbers affect prompt learning performance, we conduct experiments with various shots from 1, 2, 4, 8, 16, 32 to 64, and observe the performance variation as shot increase, as shown in Fig. 6.2. We observe the experiments on six representative datasets with various tasks as our benchmark, Caltech101 [33], Flowers102 [92], OxfordPets [96], DTD [19], UCF101 [120] and Food101 [7].

As shown in Fig. 6.2, we can observe that all six dataset performance enhanced as the labelled samples increase. However, improvement saturated as the shots become larger, even with almost no progress in the end. Furthermore, most of the experimented datasets show negligible enhancement benefit from the increased training samples from the start with regard to the test accuracy, given the tendency line is nearly parallel with the x-axis.

6.3.2 Observation 2: Random Examples Result in Fluctuated Performance

According to the existing paradigm of prompt learning techniques, random sampling of training data is performed before actual training. Such behavior results in a determinant effect of the quality of selected samples to the final performance, with the following two reasons. 1) Not all training data are created equal. Some of the samples contain more representative features than the others, which can provide more positive information towards the correct classification. Furthermore, noisy samples or wrongly-labelled samples may provide misleading information and deteriorate the overall performance. 2) Prompt learning leverages few samples in training other than the whole training datasets like traditional machine learning, which aggravate the dependency between the final result with the quality of selected samples. In other words, the choice of samples has significant impact on the model performance.

To validate the above assumption, we experiment random samples for 1, 2, 4, 8, and 16 shots, and choose five random groups for each shot, as shown in Figure 6.4. We record the test accuracy distribution summary via box plot and variance within groups. We observe that the performance with fewer shots shows relatively unstable results, e.g., test accuracy of 1 shot prompt tuning express turbulent and large variance. However, as the shot get larger, the turbulence as well as the variance narrows down. The red box represents the distribution summary, and the dash line indicates the variance. We can observe that the box range shrinks as the number of shots grows larger, so as the variance line decreases, which validates the fluctuated performance of random samples.

6.3.3 Observation 3: Arithmetic Operations Benefit Prompt Learning

To explore the impact of prompt vectors, we investigate the interactions between prompt vectors with different tasks. We set five parties with each learns prompt vectors for different tasks and data. To find out whether different tasks may benefit each other. We compare the performance of two scenarios. 1) all five parities learn their own prompt vectors with local data and use the trained prompt vectors to instruct their own tasks. 2) prompt vectors incorporate others information by implementing arithmetic add operation with other prompt vectors for training and then use the trained prompt vectors to instruct their own tasks.

We record and compare the experiment results of the two scenarios above across six datasets, as shown in Fig. 6.3. We observe that the performance with arithmetic add operation employed is better than the one without it across all settings. Such observation result shows that the arithmetic operation like add operation manages to enhance the generalization ability of prompt vectors thereby improve the overall performance.

6.3.4 Inspiration: Prompt Leverages Knowledge Inside Models Instead of Augmenting Them

From the above three observations, we can reach the following conclusions. First, unlike traditional model where performance improvement is largely driven by the training set size, performance of prompt learning model does not improve much as the number of shots increasing. From this point, We infer that pre-trained vision-language model serves as knowledge base and prompt vectors serve as key or query to instruct the retrieval of relative knowledge inside the pre-trained model. Second, training samples have great impact over the formation of prompt vectors, hence influence the instruction of retrieval of specific model knowledge, especially in the fewer shots cases. Thus, it is indispensable to quantify individual differences between data points and select the most informative data samples that can help to instruct the formation of prompt vectors. Third, we observe that the arithmetic add operation is beneficial across different tasks. Therefore, it comes to us that is it possible to apply the scenario of federated learning since the combining and updating of prompt vectors with add operation can be realized through FedAVG with multiple clients. We show the relationship between samples and prompt in Figure 6.1.

6.4 Methodology

The overview of our approach is shown in Fig. 6.5. First, we talk about federated prompting. Then, we present the ideology and technical details of our method. We categorize them into two categories, the context-unaware one like PromptFL and context-aware one like our method. Our method, can serve as a plug-in component to enhance the performance and generalization ability of existing federated prompting methods.

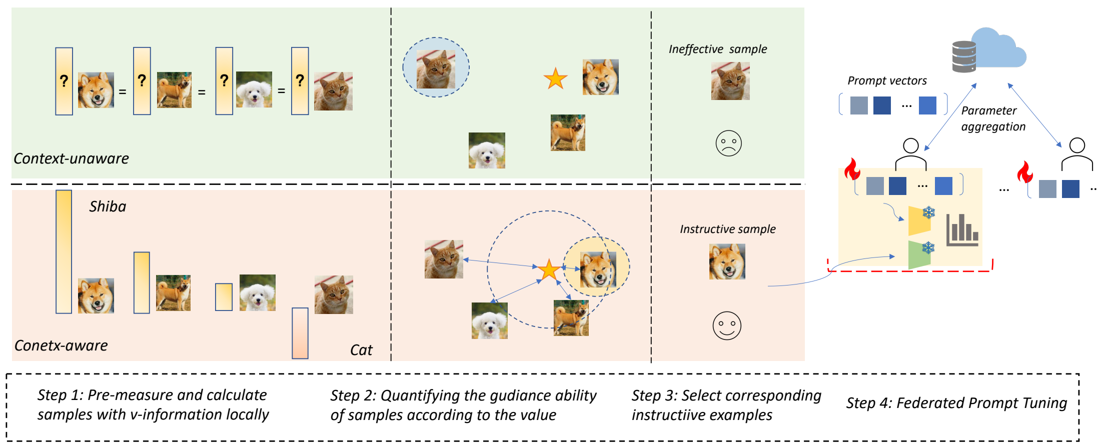
Context-unaware
Ineffective sample
Prompt vectors
Parameter aggregation
Shiba
Conetx-aware
Cat
Instructive sample
Step 1: Pre-measure and calculate samples with v-information locally
Step 2: Quantifying the gudiance ability of samples according to the value
Step 3: Select corresponding instructiive examples
Step 4: Federated Prompt Tuning

Figure 6.5: **Comparison of context-unaware and context-aware framework.** Existing context-unaware approach employ a context unaware mechanism and directly applies random sampling in the visual space to select the random samples for prompt tuning. However, such behavior treats all the samples equal and may lead to fluctuated performance. Our method, instead, measures the retrieval ability of prompts condition on each sample with $\mathcal{V}$-information on each client. After that, we deliberately select the top-k representative samples with awareness that boost the formation of informative vectors to the right direction for further federated prompting.

6.4.1 Federated Prompting

Recently, researches have witnessed the development of vision-language foundation models (e.g. CLIP [105], ALIGN [70]) and have made efforts to adapt the models to a more widely applied down-stream tasks (e.g. CoOp [152], CoCoop [151]). Furthermore, migration from centralized to decentralized manner like federated learning has also been explored in recent studies [44].

Here we re-describe federated prompting in our study. Each client possesses a pre-trained CLIP-based vision-language model, which consists of a visual encoder $g(\cdot)$ and a textual encoder $h(\cdot)$. Both the two encoders are frozen, i.e, the parameters concerning to them can not be changed during the learning process. Consequently, prompt vectors $\mathbf{P}_i$ can be learnt and updated during the training, with the form of $[v_1][v_2]...[v_p][\text{class}]_i$, where $[\text{class}]_i$ is replaced by the word embedding vector of the corresponding i_{th} class label name. Given a batch of data x, the similarity between corresponding image-text pairs is maximized. Let $cos[\cdot|\cdot]$ be the similarity we used, and $g(\cdot)$ and $h(\cdot)$ be the feature extraction function of the image and text encoder, we have the prediction probability computed as:

$$\mathrm{p}(\mathbf{y} = i|\mathbf{x}) = \frac{\exp\left(cos[g(\mathbf{x})|h(\mathbf{P}_i)]\right)}{\sum_{j=1}^{k} \exp\left(cos[g(\mathbf{x})|h(\mathbf{P}_j)]\right)}, \tag{6.2}$$

During the federated prompting, $\mathbf{P}_i$ is the only part that can be transmitted for aggregation in the communication round. After epochs of local training on clients, sampled clients transmit their prompt vectors to server for aggregation. The aggregated vectors are then leveraged to update the parameters on clients for the next round. Assuming that we are at the communication round $t + 1$, and C_k represents the selected clients set. Thus the updated prompt vectors at this point can be described as:

$$\mathbf{P}_{t+1} = \frac{1}{n_k} \sum_{k \in \mathbf{C}_k} \mathbf{P}_t^k. \tag{6.3}$$

6.4.2 Ensure Your Data for Federated Prompting

Federated prompting is a data-efficient method which can leverage the knowledge inside the pre-trained model with only a few labeled samples in the federated manner. However, as we discussed above, the existing federated prompting still adopts random sampling for the selection of labelled training samples, which may incur poor performance since data on client is insufficient and not equal. Hence, we argue that good examples for federated prompting may bring better performance.

What Makes Good Examples. To clarify the definition of good examples, we propose some potential criterion here. The limited labeled samples help to train prompt vectors, where the prompt vectors are then utilized to retrieve the relative information inside the model. Thus, it is reasonable to select the samples x 1) contain more information about the corresponding label y, and 2) that can be easier utilized by the given model V. We achieve the above goal by utilizing $\mathcal{V}$-information from [142].

Definitions of $\mathcal{V}$-information. As defined in [142]: Consider inputs X and labels Y. Let $\mathcal{V}$ be a predictive model family, here $\mathcal{V}$ represents the vision-language backbone and P be the prompt vectors. The predictive $\mathcal{V}$-*entropy* and the *conditional* $\mathcal{V}$-*entropy* are defined as:

$$H_{\mathcal{V}}(Y) = \inf_{f \in V} \mathbb{E}[-log_2 f[\varnothing, P](Y)]. \tag{6.4}$$

$$H_{\mathcal{V}}(Y|X) = \inf_{f \in V} \mathbb{E}[-log_2 f[X, P](Y)]. \tag{6.5}$$

where $f \in V$ and $f[\varnothing](Y)$ and $f[X](Y)$ represent the probability measure on Y given side information X or without side information $\varnothing$, using model from V. Further, similar with Shannon mutual information, $\mathcal{V}$-information can be defined as:

$$I_{\mathcal{V}}(X \to Y) = H_{\mathcal{V}}(Y) - H_{\mathcal{V}}(Y|X). \tag{6.6}$$

Measuring Good Examples with $\mathcal{V}$-information. The above theorem reflects how much information can be extracted from input samples X about labels Y, under the constraint of model V. However, to select good examples, we need to obtain the information from individual instance. Thus, by extending definition from [31], the *pointwise* $\mathcal{V}$-*information* for individual instance can be defined as:

$$PVI(x \to y) = -log_2 g[\varnothing, p](y) + log_2 g[x, p'](y). \tag{6.7}$$

where $g \in \mathcal{V}$, $p \in P$ s.t. $\mathbb{E}[-log_2 f[\varnothing, P](Y)] = H_{\mathcal{V}}(Y)$ and $p' \in P$ s.t. $-\mathbb{E}[log_2 g[x, p'](y)] = H_{\mathcal{V}}(Y|X)$.

Since $\mathcal{V}$ here represents the frozen visual and textual encoder, p' and p be the prompt vectors after the model trained with or without the input respectively with a few epochs. Examples with higher scores will be selected as good examples for training on each client. The amount of examples selected depends on the number of shots is set in the experiment setting.

As shown in Fig. 6.5, examples are selected according to the *v-information* between individual examples and the corresponding labels. We select the examples with the higher scores, i.e., containing more *v-information*. Other than random sampling, our method select good examples used for training. After that, selected examples from all clients are leveraged to train the prompt vectors collaboratively with each other. We validate the effectiveness of our proposed approach in sec. 6.5.

6.5 Evaluation

In this section we conduct comprehensive experiments to validate our approach under the federated scenarios with heterogeneous data distributed on each client. We aim to answer the following research questions by conducting the evaluation part.

- **RQ1**: The instability of the existing federated context-unaware prompt tuning and the necessity of addressing the issue.
- **RQ2**: How effective is CaFPT contributing on the performance of federated prompt tuning?
- **RQ3**: How effective if CaFPT contributing on the robustness of federated prompt tuning?

6.5.1 Dataset

We select six representative image classification datasets used in CLIP as our benchmark, as following [44, 152, 151], which consists of various classification tasks. *Gen-*

eral objects: Caltech101 [33]. *Fine-grained Categories:* Flower102 [92], OxfordPets [96], Food101 [7]. *Action Recognition:* UCF101 [120]. *Texture Classification:* DTD [19].

6.5.2 Baselines

We investigate the most representative context-unaware mechanism in current federated prompting category, *PromptFL*[44]. During the process, visual and textual encoder from the pretrained CLIP models are fixed, while the only learnable part is the prompt vectors. *PromptFL* employs the few-shot training protocol that randomly select the training samples and update the prompt vectors on them. Both visual and textual encoder are kept locally, only the prompt vectors are aggregated and updated by server in each communication iteration. Such mechanism is data-sensitive and largely depends on the data sampled, which leads to the turbulence of performance. CaFPT on the other hand addresses the above challenge and offers stability and robustness in federated prompt tuning.

6.5.3 Implementation Details

We conduct all the experiments on Ubuntu 20.04 with Pytorch on GeForce RTX 3090. The training is performed using SGD optimizer with 0.001 learning rate. To fit for the practical situation in federate learning, and address the data heterogeneity problem of it, we consider Non-IID setting in our experiments. We simulate $n = 50$ clients with $r = 10\%$ participation, each client is assigned with $s = 5$ random classes. We measure the instance level $\mathcal{V} - information$ on each client before training with a few epochs. And then leverage the selected examples for federated prompting. We set a local epoch $E = 5$. And for the global communication round, we set $R = 20$. For the prompt vectors setting, we use $p = 16$ vectors for the CoOp-based setting, as the best case shown in [152, 44]. We use ResNet-50 here as the backbone of the

Table 6.1: **Robustness Comparison between Context-unaware and aware Prompt Tuning.** We record the performance and robustness indicator of CaFPT and PromptFL. Δ shows the discrepancy between the two paradigm in terms of random and the worst case respectively. 'Gap' represents the intra-group gap, which indicates the discrepency between the worst case and the random case. Both 'Gap' and 'Var' indicates the turbulence of the method, *the lower the better.* As we can observe, CaFPT outperforms the existing PromptFL by high performance and strong stability.

(a) Caltech101

	Acc	Δ	Gap	Var
PFL-min	82.60	–	–	–
PFL	85.20	–	2.60	4.34
CaFPT-min	**86.25**	3.65	–	–
CaFPT	**87.51**	2.32	**1.26**	**0.99**

(b) Flowers102

	Acc	Δ	Gap	Var
PFL-min	60.05	–	–	–
PFL	62.67	–	2.62	4.72
CaFPT-min	**63.18**	3.13	–	–
CaFPT	**65.41**	2.74	**2.24**	**2.89**

(c) OxfordPets

	Acc	Δ	Gap	Var
PFL-min	80.65	–	–	–
PFL	84.91	–	4.26	10.47
CaFPT-min	**87.45**	6.80	–	–
CaFPT	**87.85**	2.95	**0.40**	**0.18**

(d) DTD

	Acc	Δ	Gap	Var
PFL-min	29.13	–	–	–
PFL	41.05	–	11.93	50.16
CaFPT-min	**43.28**	14.15	–	–
CaFPT	**46.09**	5.04	**2.82**	**5.93**

(e) UCF101

	Acc	Δ	Gap	Var
PFL-min	55.65	–	–	–
PFL	58.30	–	2.65	5.01
CaPT-min	**60.28**	4.63	–	–
CaPT	**61.64**	3.34	**1.36**	**1.48**

(f) Food101

	Acc	Δ	Gap	Var
PFL-min	67.20	–	–	–
PFL	73.72	–	6.52	22.19
CaFPT-min	**76.43**	9.23	–	–
CaFPT	**77.17**	3.45	**0.74**	**0.56**

visual encoder. We also choose to place the class token in the end of the of the prompt as before.

6.5.4 Overall Result

We summarize the overall performance of CaFPT and *PromptFL* in Table 6.1 and Figure 6.6. We record the test accuracy of CaFPT and *PromptFL* and the corresponding worst case we encounter.

Analysis on Federated Prompting (RQ1) Existing researches training with the framework of *PromptFL* follows a context-unaware tuning by randomly sampling the training examples and tuning the soft prompt on each client, and upload the vectors for aggregation. However, the specialty of few shots samples and few parameters of the learnable prompt introduce unfairness of such evaluation metrics when employing in prompt tuning, resulting in unstable outcomes. To validate the aforementioned assumption, we record the test accuracy of *PromptFL* as well as its worst performance, recorded as 'PromptFL' and 'PromptFL-min' respectively. As anticipated, *PromptFL* can obtain the relatively poor performance that is notably distant from the reported one. In Figure 6.6, the yellow filled area shows the gap between the worst case against the usually reported one, i.e, the gap between 'PromptFL-min' and 'PromptFL', which is large enough to result in turbulence. From Table 6.1, we can observe that for certain datasets, i.e, DTD, the performance of prompt tuning on certain samples is too poor for the prediction on the downstream task. Thus, we investigate the possibility in addressing the instability.

Effectiveness of CaFPT (RQ2) We leverage CaFPT to address the aforementioned issue. As shown in Table 6.1, we can observe that CaFPT outperforms *PromptFL* in both average and the worst scenarios by a large margin. For example, in datasets DTD, CaFPT obtains large performance gains over *PromptFL* by 12.2% in the random scenario and 48.6% in the worst scenario. We further record the performance as the shots increase in Figure 6.6, where CaFPT shows in red while *PromptFL* shows in orange. As the shots become larger, the performance of CaFPT enhance, surpassing *PromptFL* in nearly all cases.

Robustness of CaFPT (RQ3) Besides the outstanding overall performance, CaPT provides robustness and stability. The pink and yellow area in Figure 6.6 represents the range of test accuracy for each shots of CaFPT and *PromptFL* respectively. We can observe that the area of yellow region is larger than the pink one to a large extent. In most dataset, the pink area is nearly shrink into a line along with the CaFPT red line, which shows the advantage of robustness in CaFPT against *PromptFL*. Furthermore, we investigate the turbulence indicators comparison between the two. For each method, we randomly samples five times and record the 'Gap' and 'Var' of each benchmark in Table 6.1. 'Gap' represents the intra-group gap, which indicates the discrepency between the worst case and the averaged one. 'Var' stands for variance. Both 'Gap' and 'Var' indicates the turbulence of the method, the lower the better. In conclusion, CaFPT not only offers better performance but also provides robustness and stability.

6.5.5 Analysis with Interpretations

We investigate whether the informative vectors reflect the effectiveness of samples and validate the correction of the selection. We leverage the BPE representation used in CLIP for tokenization, and find the nearest meaning words for interpretation based on the Euclidean distance. In Food101, we observe **'harvest'** (1.2273), **'fresh'** (0.8280) **'eating'** (0.7234) in the top-3 observation of each tokens with our method, which is meaningful and steers the right direction for extracting the related knowledge. *CoOp* on the other hand, obtains meaningless or task-unrelated subwords like, **'declined'** (0.9331), **'cheeks'** (0.8678), **'secretary'** (0.6472). Thus, the interpretation of prompt further validate the effectiveness of our method.

6.6 Related Work

Data Pruning. The field most closely related to this paper is data pruning. Recent studies [129, 34, 18, 99, 88] have established the feasibility of this approach by proposing various metrics to rank training examples based on their difficulty or significance, from easy or redundant examples to difficult or essential ones, and by

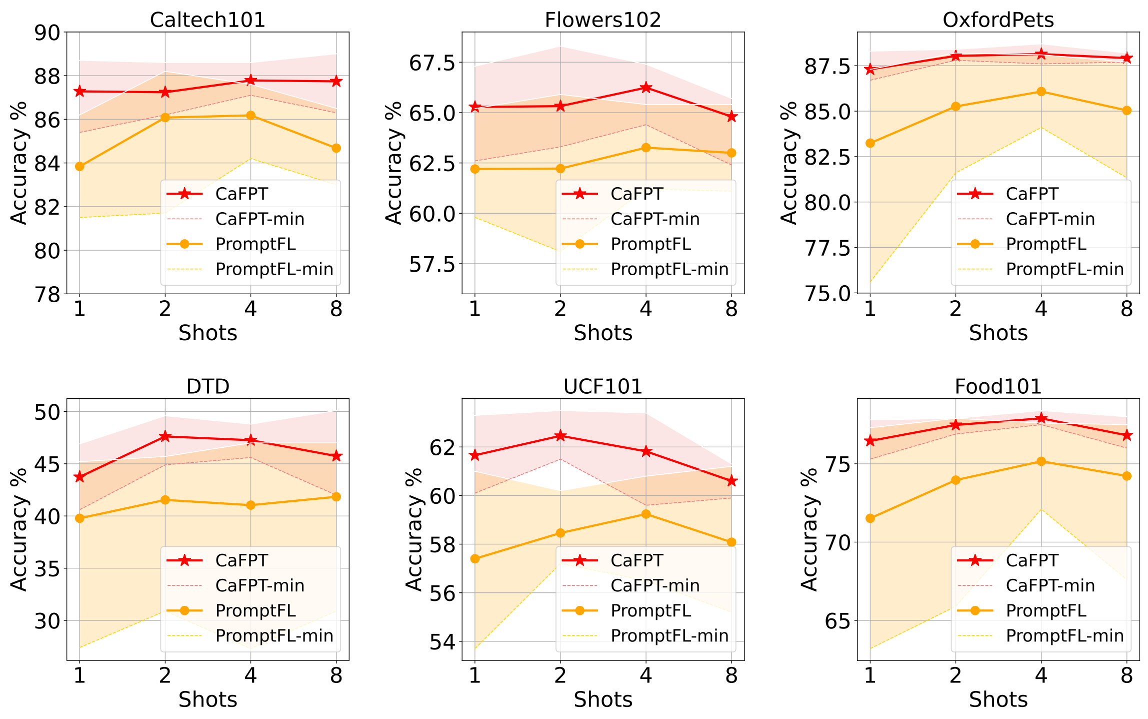

Figure 6.6: **Performance Comparison between Context-unaware and aware Federated Prompt Tuning.** We record the performance of *PromptFL* and CaFPT, and observe the worst case we sampled during the evaluation. We randomly sample five times for each shot setting and display the performance distribution. The red line and orange line represents CaFPT and *PromptFL* respectively. The pink area shows the accuracy that CaFPT can achieves randomly, while the yellow area indicates for *PromptFL*. The gold dash line stands for the worst case that *PromptFL* obtains during the evaluation. We can conclude that performance fluctuates largely in *PromptFL*. However, CaFPT outbeats its context-unaware counterpart from both performance and stability.

pruning datasets by keeping a portion of the most challenging examples. However, these studies leave questions unresolved: Under what conditions and why is successful data pruning possible? Intuitively, problematic examples may also be those that are difficult in the sense that learning them would require the algorithm to contradict other training examples or to increase its complexity. Acquiring these difficult examples may result in the algorithm's inability to generalize well or to overfit. The new discovery in [121] is that keeping easy examples rather than hard ones is more effective when data is scarce.

$\mathcal{V}$-information. An information-theoretic framework called $\mathcal{V}$-*usable* information [142] has been adopted by researchers, which quantifies the degree of information contained in a representation that is accessible to a model family $\mathcal{V}$. It can be quantified within a framework called *predictive* $\mathcal{V}$-information, which generalizes Shannon information to determine the amount of information that can be gleaned from X about Y when limited to functions $\mathcal{V}$, denoted as $I_{\mathcal{V}}(X \rightarrow Y)$. The higher the $I_{\mathcal{V}}(X \rightarrow Y)$, the simpler the dataset is for $\mathcal{V}$. If $\mathcal{V}$ is the set of all functions - that is, under unrestricted computation - $\mathcal{V}$-information reduces to Shannon information. More recently, there has been a growing adoption of $\mathcal{V}$-information in NLP. It has been utilized to investigate the context features that Transformers actually employ [95], as well as to filter out information for interpretability techniques based on probing [102, 51]. It has also been applied for evaluating the complexity of NLP datasets [31].

Our study discovers that a significant portion of the enhancement originates from a specific elimination of the data utilized in prompt training. The trimmed dataset would offer a more robust guidance signal for training informative vectors to precisely identify the previously acquired task in the pre-trained neural networks.

6.7 Chapter Summary

Is it possible for a training example to have a negative impact on federated prompt training? Based on the habitual random sampling in existing federated prompt tuning mechanism, none of the researches consider this issue. However, experiments show that the unawareness of samples involved in the federated prompt tuning leads to significant poor and unstable performance. In other words, the relevance of the examples is crucial. We investigate a context-aware method, CaFPT, to deliberately select samples that are beneficial for prompting to achieve superior robustness.

This technique is intended as a pre-training step to acquire improved data for further training. The method involves generating informative prompt vectors from client's dataset, each of which is affected differently by individual examples. The opinions of multiple clients regarding which examples are informative are mediated by a $\mathcal{V}$-information mechanism. Extensive experiments demonstrates the significant improvement and robustness that CaPT offers. Moreover, our research makes an important step in taking the input level into consideration for enhancing pre-trained vision-language models in federated environments.

Chapter 7

Conclusions and Suggestions for Future Research

This chapter summarizes the original contributions of the entire thesis in Section 7.1 and proposes possible future directions of our research in Section 7.2.

7.1 Conclusions

Tremendous edge devices have emerged in the past decade and has changed and benefit people's daily life and societal sectors. To better make use of the isolated personal data, collaborative edge learning enables the possibility to learn a neural network jointly with the effort of multiple edge devices within a long range with privacy and security. Federated learning and split learning emerges as the mainstream frameworks in supporting current collaborative edge learning system. Although made great impact, each paradigm has it own limitations, especially in the heterogeneous environment. In this thesis, we focus on the efficiency and personalization of collaborative edge learning in the heterogeneous environment.

We propose four frameworks in improving the existing collaborative edge learning from the perspective of communication efficiency, computation efficiency, resource heterogeneity and data heterogeneity.

First, we analyze the challenges of existing split learning when encountering edge devices with different memory and storage capabilities. To alleviate the idleness of training parities during the training process and improve the training efficiency, we propose *Tree Learning*, a system-algorithm co-design framework for collaborative training acceleration. Such framework adopts a tree aggregation scheme and dynamically allocates different layers to heterogeneous clients according to their different local capacities. To assign the appropriate layers for heterogeneous clients, we propose a partition algorithm for our framework. We also performs a global level parallelism scheme to enable the minimum synchronization overhead among participants.

Second, we rethink the iterative training procedure of existing federated learning, and provide a new perspective by rethinking if foundation models can be applied to FL as a new paradigm of training. We propose PromptFL, a framework that replaces existing federated model training with prompt training, i.e., FL clients train prompts instead of a model. During the local training, backbones on each client are frozen and the only trainable part is the prompt vectors. During the communication between clients and server, local client transmit the weight of learnt prompt vectors other than the whole model, which can simultaneously exploit the insufficient local data and reduce the aggregation overhead.

Third, current frameworks of federated prompting focus on modeling user consensus, leaving the personalization of prompt under-explored. Although using prompt in FL to activate the pre-trained knowledge shows tremendous benefits, a major challenge for deploying such approaches in FL is the heterogeneity of users. To alleviate the influence caused by statistical heterogeneity, we carefully examine the-state-of-arts pFL techniques and adapt them to the federated prompting scenarios. However, none of these methods leverage the natural advantages of multimodality of vision-language models. We propose p`FedPrompt`, which learns user consensus in linguistic space and adapts to user features on each client in visual space respectively.

By incorporating the knowledge retrieved from multimodality, the challenge of user statistical heterogeneity is addressed.

Forth, current federated prompting framework leverages random selection of the examples to guide the decision direction of the pretrained models. However, our experiments discover the data sensitivity in federated prompting, which larges influence the model performance. Local data on the other hand is not equal, given to the distributed origin of it. To recognize the importance of being discerning in selecting the data for participation in the process, we propose some potential criterion here to decide what makes good examples. We propose an information-based strategy to select representative data samples to prompt while filter out the noisy one.

7.2 Future Directions

We close this thesis with the discussion of future direction in which the current research can be advanced.

First, in terms of the *tree Lesrning*, two direction can be further developed. In this thesis, we alleviate a more stable method to achieve the best split points during the training process. Thus, one direction is to further develop a more dynamic and efficient way to reschedule the split point for each client during the training. Furthermore, although our method achieves outstanding performance on extensive cnn-based neural network, generalization to other types of neural networks like rnn can also be explored.

Second, the development of federated prompting is still in its infancy stage. Our thesis considers the general prompt during the training to obtain the user consensus. In the next step, first, we consider to decompose the prompt into two part, i.e, the general prompt and the specific prompt. The general prompt control the common knowledge while the specific prompt guide the additional knowledge to different domains, which can be effective in the scenarios of out of domain problems. Next, we consider to add more semantic and context information to the prompt. In this thesis, we leverage soft prompt while do not give much detailed information of the classes. To improve the interpertability of prompt, we will leverage the help of the large language models and give more side information to describe the class. With the help of semantic information, federated prompting can be more interpretable with human's understanding and enhanced performance. Further, although the ensemble of all the prompts with context information may improve the model performance, however, the communication and computation efficiency have been scarified, which the most important factors during the collaborative edge learning. Further, not all the prompts are useful, most information can be represented with only small fraction of the prompts. Thus, we will explore the pruning of prompts during the federated prompting.

References

[1] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[2] Ali Abedi and Shehroz S Khan. Fedsl: Federated split learning on distributed sequential data in recurrent neural networks. *arXiv preprint arXiv:2011.03180*, 2020.

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30:1709–1720, 2017.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[7] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019.

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[12] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. Communication-efficient federated learning with adaptive parameter freezing. In *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 1–11, 2021.

[13] Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. Plot: Prompt learning with optimal transport for vision-language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[14] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vafl: A method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

[15] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11030–11039, 2020.

[16] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Biao Chen, and Zhiqi Shen. Focus: Dealing with label quality disparity in federated learning. *arXiv preprint arXiv:2001.11359*, 2020.

[17] G Cheng, K Chadha, and J Duchi. Fine-tuning in federated learning: A simple but tough-to-beat baseline. *arXiv*, 2021.

[18] Kashyap Chitta, José M Álvarez, Elmar Haussmann, and Clément Farabet. Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[19] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[20] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pages 2089–2099. PMLR, 2021.

[21] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[22] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE conference on computer vision and pattern recognition,*, pages 248–255, 2009.

[24] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[26] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[27] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[28] Zi-Yi Dou, Yichong Xu, Zhe Gan, Jianfeng Wang, Shuohang Wang, Lijuan Wang, Chenguang Zhu, Pengchuan Zhang, Lu Yuan, Nanyun Peng, et al. An empirical study of training end-to-end vision-and-language transformers. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18166–18176, 2022.

[29] Adam Dziedzic, Nikita Dhawan, Muhammad Ahmad Kaleem, Jonas Guan, and Nicolas Papernot. On the difficulty of defending self-supervised learning against model extraction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[30] Andrew E. Freedman. Apple a15 bionic powers iphone 13 and ipad mini, 2021.

[31] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR, 2022.

[32] Cong Fang and Zhouchen Lin. Parallel asynchronous stochastic variance reduction for nonconvex optimization. In *Proc AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[33] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2004.

[34] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[35] Andreas Fürst, Elisabeth Rumetshofer, Viet Tran, Hubert Ramsauer, Fei Tang, Johannes Lehner, David Kreil, Michael Kopp, Günter Klambauer, Angela Bitto-Nemling, et al. Cloob: Modern hopfield networks with infoloob outperform clip. *arXiv preprint arXiv:2110.11316*, 2021.

[36] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

[37] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[38] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[39] Lin Gu, Weiying Zhang, Zhongkui Wang, and Deze Zeng. Service management and energy scheduling toward low-carbon edge computing. *IEEE Transactions on Sustainable Computing*, 2022.

[40] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[41] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[42] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, 8(3):331–368, 2022.

[43] Tao Guo, Song Guo, and Junxiao Wang. pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference*, 2023.

[44] Tao Guo, Song Guo, Junxiao Wang, and Wenchao Xu. Promptfl: Let federated participants cooperatively learn prompts instead of models–federated learning in age of foundation model. *arXiv preprint arXiv:2208.11625*, 2022.

[45] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[46] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. Warp: Word-level adversarial reprogramming. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[47] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 33:2304–2315, 2020.

[48] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[49] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[51] John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher D Manning. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639, 2021.

[52] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1643–1653, 2021.

[53] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[54] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[55] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[56] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[57] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *AAAI*, pages 7865–7873, 2021.

[58] Zhouyuan Huo, Bin Gu, Heng Huang, et al. Decoupled parallel backpropagation with convergence guarantee. In *International Conference on Machine Learning*, pages 2098–2106. PMLR, 2018.

[59] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[60] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[61] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[62] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[63] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

[64] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[65] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[66] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[68] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Proceedings of the NeurIPS Workshop on Bayesian Deep Learning*, 2018.

[69] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[70] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705, 2021.

[71] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020.

[72] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[73] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[74] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[75] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019.

[76] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[77] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[78] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017.

[79] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[80] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[81] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

[82] Yang Liu, Xiong Zhang, and Libin Wang. Asymmetrical vertical federated learning. *arXiv preprint arXiv:2004.07427*, 2020.

[83] Zhijian Liu, Zhanghao Wu, Chuang Gan, Ligeng Zhu, and Song Han. Datamix: Efficient privacy-preserving edge-cloud inference. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[84] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated learning*, pages 240–254. Springer, 2020.

[85] WANG Luping, WANG Wei, and LI Bo. Cmfl: Mitigating communication overhead for federated learning. In *proc. IEEE (ICDCS)*, pages 954–964, 2019.

[86] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[87] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics,*, pages 1273–1282. proc., 2017.

[88] Kristof Meding, Luca M Schulze Buschoff, Robert Geirhos, and Felix A Wichmann. Trivial or impossible—dichotomous data difficulty masks model differences (on imagenet and beyond). In *International Conference on Learning Representations*, 2022.

[89] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[90] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against

centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[91] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022.

[92] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, 2008.

[93] S. O'Dea. Average global mobile and fixed broadband download & upload speed worldwide, 2021.

[94] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

[95] Joe O'Connor and Jacob Andreas. What context features can transformer language models use? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 851–864, 2021.

[96] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[97] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[98] Blake Patterson. Blake's ios device specifications grid, 2022.

[99] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

[100] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[101] Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: a systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, 21(2):1–31, 2021.

[102] Tiago Pimentel and Ryan Cotterell. A bayesian framework for information-theoretic probing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2869–2887, 2021.

[103] Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

[104] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[105] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[106] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (HPCA)*, 2021.

[107] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1571–1578, 2012.

[108] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[109] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18082–18091, 2022.

[110] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020.

[111] Google Researches. Federated learning: Collaborative machine learning without centralized training data, 2017. Last accessed April 6, 2017.

[112] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[113] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[114] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[115] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[116] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[117] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

[118] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[119] Vinay Sisodia. Distillation of clip model and other experiments, 2021.

[120] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[121] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*, 2022.

[122] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. Cross-domain federated adaptive prompt tuning for clip. *arXiv preprint arXiv:2211.07864*, 2022.

[123] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[124] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[125] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022.

[126] Xueyang Tang, Song Guo, and Jingcai Guo. Personalized federated learning with contextualized generalization. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2241–2247, 2022.

[127] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision (ECCV)*, 2022.

[128] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8485–8493, 2022.

[129] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

[130] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models.

Advances in Neural Information Processing Systems (NeurIPS), 34:200–212, 2021.

[131] Vasileios Tsouvalas, Aaqib Saeed, Tanir Ozcelebi, and Nirvana Meratnia. Federated learning with noisy labels. *arXiv preprint arXiv:2208.09378*, 2022.

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[133] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[134] Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*, 2018.

[135] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[136] Qiyun WAnG. Design and evaluation of a collaborative learning environment. *Computers & Education*, 53(4):1138–1146, 2009.

[137] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Privacy-preserving, efficient, and effective machine learning. 2022.

[138] Jinze Wu, Qi Liu, Zhenya Huang, Yuting Ning, Hao Wang, Enhong Chen, Jinfeng Yi, and Bowen Zhou. Hierarchical personalized federated learning for user modeling. In *Proceedings of the Web Conference 2021*, pages 957–968, 2021.

[139] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[140] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[141] Jingyi Xu, Zihan Chen, Tony QS Quek, and Kai Fong Ernest Chong. Fedcorr: Multi-stage federated learning for label noise correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10184–10193, 2022.

[142] Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. *arXiv preprint arXiv:2002.10689*, 2020.

[143] Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. Federated recommendation systems. In *Federated Learning*, pages 225–239. Springer, 2020.

[144] Miao Yang, Hua Qian, Ximin Wang, Yong Zhou, and Hongbin Zhu. Client selection for federated learning with label noise. *IEEE Transactions on Vehicular Technology*, 71(2):2193–2197, 2021.

[145] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[146] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[147] Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712, 2016.

[148] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2020.

[149] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[150] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *arXiv preprint arXiv:2109.01134*, 2021.

[151] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825, 2022.

[152] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[153] Qihua Zhou, Song Guo, LIU Yi, Jie Zhang, Jiewei Zhang, GUO Tao, XU Zhenda, and Zhihao Qu. Hierarchical channel-spatial encoding for communication-efficient collaborative learning. In *Advances in Neural Information Processing Systems*.

[154] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[155] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.