Efficient Fine-Tuning Strategies for Federated Learning: Optimizing Model Performance Across Distributed Networks

Nicolas Bernier

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

November 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Nicolas Bernier**

Entitled: **Efficient Fine-Tuning Strategies for Federated Learning: Optimizing Model Performance Across Distributed Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Abdelhak Bentaleb

______________________ Examiner
Dr. Mirco Ravanelli

______________________ Supervisor
Dr. Eugene Belilovsky

Approved by ______________________
Joey Paquet, Chair
Department of Computer Science and Software Engineering

__________ 2024

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Efficient Fine-Tuning Strategies for Federated Learning: Optimizing Model Performance Across Distributed Networks

Nicolas Bernier

Federated Learning (FL) allows a global model to be trained collaboratively by a number of clients without sharing data. This setting is often characterized by resource-constrained clients connected over a low-bandwidth network. Hence, algorithms designed for the setting must account for important factors such as computer and memory requirements, robustness under changing data distributions and communication. Recent works, have started demonstrating the benefits of using pretrained models over random initialization on these considerations. We cover these recent advancements before introducing methods conceived along the same lines. We show that in the FL setting, fitting a classifier using the Neurest Class Means (NCM) can be done exactly. We demonstrate its efficiency and combine it with full fine-tuning to produce stronger performance. Then, we introduce an adapted zeroth-order method capable of bringing a model to convergence with a minimal per-round compute budget while reducing the memory burden for clients during training down to that of inference. This work presents several experiments demonstrating the effectiveness of the proposed methods and highlights the importance for additional work into the application pretrained models in the FL setting.

Acknowledgments

I would like to express my deepest gratitude to Dr. Eugene Belilovsky, my thesis advisor, for his constant guidance, invaluable feedback, and unwavering support throughout this research journey. Eugene has been an exceptional teacher and guide for these last two years. The value I have gotten from his patience and willingness to dedicate time to discuss ideas is immeasurable and a strong factor in me completing this program successfully. He has not only been a strong contributor to my academic growth, but his teachings have also had a strong impact on my professional life as well.

I also extend my gratitude to my collaborators at various stages of my Master's program, mainly Gwen Legate. Her collaboration and ability to take ownership of key aspects of our project significantly contributed to the success of this research. Her insights and dedication were invaluable, and I am deeply appreciative of her efforts.

Finally, I would like to dedicate this thesis to my family. Your constant support and belief in me continues to push me forward every single day. Your thoughts and opinions force me to reexamine my ideas and create new ones. You make me a better person in every aspect of my life and I am deeply grateful to you. This journey would be impossible without all five of you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Most of Machine Learning (ML) requires a centralized access to training data. However, its applications continue to grow to increasingly privacy-sensitive use cases where data is required to be collected closer to the end-user. In some of these common use cases, it may be intractable to share data amongst data producing clients for any number of reasons. An often cited example is that of hospitals each having a limited number of cases locally that may benefit from the collective knowledge of other institutions who are each unable to share their own cases for privacy and regulatory reasons (D. Li and Wang (2019)). This describes the cross-partition setting that tends to have a smaller number of stable clients each with their own sizeable dataset. It places itself in contrast to the cross-device setting characterized by a larger number of diverse and often unreliable set of clients with wide differences in local data distribution and volume. In this setting, these clients largely considered to be resource constrained devices like phones, sensors or other edge devices connected over a low-bandwidth and unstable network.

Federated Learning (FL) allows these clients to collaborate in training a global model while ensuring client data and model training remain on-device. The algorithms of interest to this thesis are orchestrated by a centralized server. They train the model for several rounds. During each round, the server selects a number of clients and sends them the global model weights. The selected clients train the model on their local data for a number of gradient steps and send the updated model weights back to the server. Lastly, the server aggregates the updated client models. Following the aggregation, the server begins the next round of updates by selecting a new set of clients. This setting presents a number of challenges often exacerbated by the heterogeneity of client data leading to performance degradation when the model is trained on popular baselines including FedAvg (McMahan, Moore, Ramage, Hampson, and y Arcas (2017)). Due to the nature of cross-device FL, it is desirable for algorithms to have smaller communication budgets, represented by the total amount of data sent between the server and its clients (in GB), to train a model to convergence. As state-of-the-art models have continued to grow in number of parameters as a consequence of the discovery of the Neural Scaling Laws (Kaplan et al. (2020)), communication budgets for common FL algorithms have faced upwards pressure. Intuitively, larger models also require more memory and compute when training, which, due to the hardware limitations of clients in the cross-device setting, introduces further points of consideration.

To minimize the amount of time required for training, transfer learning reuses a model developed for a given task on a different, but similar task instead of training another model from a randomly initialized set of weights. Transfer learning from pretrained models that have been trained on sufficiently abundant and diverse data is well known to produce strong results in tasks related to vision (Girshick, Donahue, Darrell, and Malik (2014); K. He, Girshick, and Dollar (2019)), natural language processing (NLP) (Radford et al. (2019)), and other domains. Indeed, pretraining combined with fine-tuning to specialize the model for a specific downstream task often leads to better generalization and faster model convergence in the centralized setting (Patel, Gopalan, Li, and Chellappa (2015); Weiss, Khoshgoftaar, and Wang (2016)). In the context of FL, faster convergence is exhibited as a smaller number of rounds required to train the model to convergence and, consequently, as a smaller amount of total communication cost. Although there has been a subset of studies in FL indicating that pretraining does improve performance of standard FL algorithms (Chen, Tu, Li, Shen, and Chao (2023); J. Nguyen et al. (2023)), FL literature has been largely focused on models trained from a random initialization (Karimireddy et al. (2020); T. Li et al. (2020a); McMahan et al. (2017)) and the impact of heterogeneity on algorithmic convergence.

To mitigate the compute requirements of transfer learning with large models, parameter-efficient fine-tuning (PEFT) methods freeze most of the model weights and only update a small fraction of total weights (Hu et al. (2021); X. L. Li and Liang (2021)). These methods do still require backprobagating the loss through the entire model. This important step ultimately results increased memory requirement for training that is up to an order of magnitude larger than that at inference. This presents an important constraint for FL clients that are often both compute and memory bound. It is important to note that clients only share updated model weights back to the server. Hence, PEFT reduces the communication cost per round. Outside of a few works (Babakniya et al. (2023)), the rate of model convergence when training on these algorithms in the FL setting remains largely under-explored.

1.2 Contributions

This thesis centers on the use of pretrained models within federated learning. We adapt our methods to optimize for important factors when considering an FL algorithm including the amount of data needing to be shared by clients per round (communication cost), total compute requirements and final model accuracy. We introduce methods adapting zeroth-order and first-order learning algorithms to the setting with the goal of improving on these measurements. The main contributions of this work are as follows:

- We provide empirical evidence for numerous downstream tasks in vision and text domains in the federated learning setting that using pretrained models ultimately results in better performance when the model is trained to convergence in both Chapter 3 and 4.
- We provide empirical evidence for numerous downstream tasks in vision and text domains in the federated learning setting that adapting PEFT methods ultimately leads to models lower communication cost, memory and compute requirements to train a global model to convergence in both chapter 3 and 4.
- In chapter 3, we propose a two-stage process consisting of an initial HeadTuning step via FedNCM or a linear probe, followed by full fine-tuning. Combining these approaches results in high accuracy and increased speed of convergence than using either approach on its own. For this work, we submitted to and were accepted at the NeurIPS 2023 main conference track under a paper titled **Guiding The Last Layer in Federated Learning with Pre-Trained Models**. This paper introduce FedNCM covered in this thesis.

- In chapter 4, we demonstrate the capability of zeroth-order methods to adapt to the distributed setting resulting in theoretically and empirically equivalent results for the method proposed when trained on iid or non-iid data across any number of clients.

- We present FedMeZO, an adaptation of MeZO, discussed in the Background section, that reduces communication cost per round and produces a strong baseline for future works. This is covered in chapter 4.

Personal Contributions: During my Master's program, I worked on meaningfully reducing the communication budget for FL algorithms by leveraging pretrained models.

Since starting my research, an increasing number of papers have been produced on the topic and accepted at top conferences including Guiding The Last Layer in Federated Learning with pre-trained models, which I collaborated on with others. This work focuses on tuning the last layer of the model using a Nearest Class Means Classifier. Gwen Legate and I worked closely to produce this paper with valuable contributions from Lucas Caccia, Edouard Oyallon and, my advisor, Eugene Belilovsky. For this work covered in chapter 3, I took the primary responsibility of building the code base starting from the original FLSim repository (J. Nguyen et al. (2023)) and running a portion of the main experiments for both Vision and NLP tasks. Beyond this, I contributed to a number of items.

- I implemented the majority of the code used in the research including the integration of the algorithms referred to as FedNCM, FedNCM+FT, LP and random methods for both vision and language tasks.

- I implemented the initial experimental framework used in the research and wrote the code for the scripts used to run experiments on SLURM-based clusters.

- I conducted the experiments and created the plots for CIFAR and NLP datasets.

- I was primarily responsible for the NLP experiments during the rebuttal and final draft production periods.

For the work covered in chapter 4, I was the primary contributor. I produced the entire code base starting from the original MeZO repository (Malladi et al. (2024)). I also took the responsibility of writing the paper and producing all the experiments.

Chapter 2

Background

2.1 Federated Learning

Federated learning aims to iteratively train a global model through local training of client models on-device and aggregating the result models. The most well known approach in FL is the FedAvg algorithm proposed by (McMahan et al. (2017)). FedAvg aggregates client models by averaging them out. More formally, distributed optimization occurs over K clients with each client $k \in \{1, ..., K\}$ having data $\mathbf{X}_k, \mathbf{Y}_k$ that contains n_k samples drawn from distribution D_k. The total number of samples across all clients can be defined as $n = \sum_{k=1}^{K} n_k$. The data $\mathbf{X}_k$ at each node may be drawn from different distributions and/or may be unbalanced with some clients possessing more training samples than others. The typical objective function for federated optimization is given in Eq. 4 (Konečnỳ, McMahan, Ramage, & Richtárik, 2016) and aims to find the minimizer of the loss over the sum of the client data:

$$\mathbf{w}^* \in \operatorname*{argmin}_{\mathbf{w}} \sum_{k=1}^{K} \frac{n_k}{n} \mathcal{L}(f(\mathbf{w}, \mathbf{X}_k)) \,. \tag{1}$$

In the random initialization setting, convergence of FedAvg and related algorithms has been widely studied for both independent and identically distributed (iid) client data (Stich (2019); Wang and Joshi (2018)) and non-iid settings (Fallah, Mokhtari, and Ozdaglar (2020); Karimireddy et al. (2020); T. Li et al. (2020a); Yu, Yang, and Zhu (2019)).

A commonly cited problem in the literature is the challenge of heterogeneous clients where a variety of algorithms have been developed to tackle this. Data heterogeneity, or non-iid data, is represented by the significantly varying data distributions across clients. A lot of the work in this area focuses on reducing the difference in client updates and limiting the importance of outliers (Hsu, Qi, and Brown (2019); Karimireddy et al. (2020); Legate, Caccia, and Belilovsky (2023); T. Li et al. (2020a)). System heterogeneity manifests itself with clients having varying physical limitations including compute, memory, network and storage capacity. Asynchronicity (X. Lian, Zhang, Zhang, and Liu (2018); Xie, Koyejo, and Gupta (2020); Zheng et al. (2020)) and resource-aware participation (Lyu, Xu, and Wang (2020); Nishio and Yonetani (2019)) are two areas of focus for their type of heterogeneity. We concern ourselves in this work with data heterogeneity.

In addition to heterogeneity, communication cost (in Gb) presents another important concern in distributed optimizations (McMahan et al. (2017)). With FL, this becomes especially relevant when discussing the cross-device setting where devices are connected over long distances through a low-bandwidth and unstable network. This budget is represented by the product of the amount of communication per round between clients and the server and the number of rounds it takes for the model to converge. Importantly, this presents two factors to improve upon. Some works have focused on compression of the updates themselves (Alistarh, Grubic, Li, Tomioka, and Vojnovic (2017); Du, Yang, and Huang (2020); Konečný et al. (2017); Lin, Han, Mao, Wang, and Dally (2020)). We instead choose to focus on improving convergence speed and reducing the number of parameters needing to be shared each round.

2.2 Privacy and Security in Federated Learning

One of the main motivations for FL as an approach to distributed learning is privacy and security. Indeed, FL enables federating training of a global model to a set of clients each training on their own local dataset which remains on-device at all times. FedAvg (McMahan et al. (2017)), the seminal paper in the field, runs local models through multiple epochs of the local datasets, which demonstratively makes stronger privacy guarantees. These often come through differential privacy designed to make strong privacy assurances by adding controlled noise to data or computations as to limit the impact of individual data points on output (Abadi et al. (2016); Dwork and Roth (2014)). More formally, differential privacy (DP) can be defined by Equation 2. (Dwork and Roth (2014)) where $\mathcal{M}$ is the algorithm of interest. D_1 and D_2 are two adjacent examples differing by a single unit in the dataset for which S is some subset of algorithm outputs. The privacy terms are defined as ϵ representing loss of privacy and δ representing some approximation (certainty) term of the algorithm's guarantee.

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta \tag{2}$$

Many FL algorithms have been developed to incorporate stronger differential privacy constraints (T. Li et al. (2020b); Noble, Bellet, and Dieuleveut (2023); Wei et al. (2019)). It is worth noting that common approaches aiming to improve model performance on non-iid data also tend to make improvements to the privacy guarantees. This is because they also often tend to focus on reducing the impact of individual examples. Beyond improving on DP guarantees, other works have focused on formalizing attacks on models in the distributed learning setting more broadly (Hitaj, Ateniese, and Perez-Cruz (2017); T. D. Nguyen et al. (2023)). We do not explicitly design our methods for differential privacy, but do make the claim that they are at least as private as the original FedAvg. We make this claim because our clients share an equal or smaller amount of information per round back to the server. Additionally, we use FedAvg as our primary aggregation mechanism in both methods proposed.

2.3 Transfer Learning

Transfer learning is widely used in many domains where data is scarce (Alyafeai, AlShaibani, and Ahmad (2020); Girshick et al. (2014); Yazdanpanah et al. (2022); Zhuang et al. (2020)). A number of approaches for transfer learning have been proposed including the most commonly used full model fine-tuning and last layer tuning (Kornblith, Shlens, and Le (2019)). Fine-tuning billions of parameters is a compute and memory intensive operation. In many cases, it is intractable to do on edge-devices, which are often the subject of FL studies. In the centralized setting, we have seen increased attention on efficient learning methods. For instance, feature selection reduces the computational cost by analyzing the activations of different model layers and selecting the most relevant ones based on the training dataset and objective (Evci, Dumoulin, Larochelle, and Mozer (2022)). The addition of trainable affine parameters, or smaller layers, at various points in the model also improves computational overhead to fine-tune a model to convergence, but can act as a form of regularization helping to limit forgetting (D. Lian, Daquan, Feng, and Wang (2022); Yazdanpanah et al. (2022)). Here, the rest of the weights remain frozen. Similarly, methods to attach adapters to transformers have also shown strong results (Houlsby et al. (2019); Hu et al. (2021)). Among these, Low-Rank Adaptation (LoRA) stands out for its ability to drastically reduce the number of trainable parameters in language models (Hu et al. (2021)). LoRA fixes the weight matrix, denoted W_0, in language models and redefines it as the sum $W_0 + B \cdot A$, where matrices B and A have a much smaller rank than W_0. The weights matrix is updated through training these much smaller matrices, which results in a lower computation cost compared to full fine-tuning. For language tasks where we use transformer models, our methods are compatible with LoRA.

Transfer learning and the effects of pretraining in FL have so far only been explored in limited capacity. There is evidence that initializing a model with pretrained weights consistently improves training accuracy and reduces the performance gap between homogeneous and heterogeneous client data distributions (J. Nguyen et al. (2023)). Follow-up work adapted parameter-efficient fine-tuning (PEFT) methods, more specifically LoRA, to the setting showing it can reduce performance degradation with non-iid clients (Babakniya et al. (2023)).

The primary constraint to using transfer learning in the FL setting is the inability to interact with client data to choose a source model trained on a similar dataset. Although mitigated by the recent surge in web-scale training enabling broader generalization for most use cases, it is still difficult to select the proper pretraining dataset given the often limited number of local examples that often differ widely between clients. Knowledge distillation in FL is a line of research presenting methods to learn from a broadly available public dataset as clients learn from their private data (Mora, Tenison, Bellavista, and Rish (2022); Zhu, Hong, and Zhou (2021)). Additionally, in the case where pretrained data is not readily available, producing synthetic data and training the global model centrally on this new data has been shown to be beneficial to FL model performance (Chen et al. (2023)). For much of the work covered in this thesis, we assume some level of overlap between the pretraining and target datasets.

2.4 Zeroth-Order Optimization

The application of gradient descent and other first-order optimizations in deep learning has been established for a long time (Rumelhart, Hinton, and Williams (1986)). During training, these methods generally backpropagate a global loss across the entire model. This requires storing intermediate activations and gradients from each layer in memory. In contrast, inference only requires the current layer's activations to be stored while they are used for computing the input to the next layer. Hence, most deep learning (DL) algorithms have much higher memory requirements during training than inference. This has important consequences for larger models on the maximum batch size and overall training times. Naturally, this memory overhead in training shows up in vision, language models and other domains (Chakrabarti and Moseley (2019); Gomez, Ren, Urtasun, and Grosse (2017); K. He, Zhang, Ren, and Sun (2015)).

Where first-order methods require explicit access to the gradients of the loss function across all model layers, zeroth-order methods seek to indirectly evaluate the loss at different points in the model during forward pass. Once the estimated gradient is calculated at a given point, activations can be discarded (S. Liu et al. (2018)). This represents the same amount of activations that needing to be retrained at inference time, thus making the memory requirements of training a model with zeroth-order algorithms equivalent to those at inference. However, these methods do require more forward passes and evaluations of the loss. This combined with their tendency to converge more slowly than first-order methods presents a computer efficiency and memory usage trade-off (S. Liu et al. (2020)).

More recently, MeZO (Malladi et al. (2024)) demonstrated the effectiveness of adapting zeroth-order stochastic gradient descent (ZO-SGD)(S. Liu et al. (2018)) with pretrained language models (LMs). The work showed that not starting from a random initialization helps zeroth-order methods significantly close the gap on their first-order equivalent. The method naturally eliminates the need for backpropagation through layer-wise gradient estimation. This gradient is estimated as the Simultaneous Perturbation Stochastic Approximation (SPSA) (S. Li, Xia, and Xu (2022)). Given some batch $\mathbf{X}$ and set of weights θ, SPSA can be described by Eq. 3 (Malladi et al. (2024)).

$$\hat{\nabla}\mathcal{L}(\theta;\mathcal{X}) = \frac{\mathcal{L}(\theta+\epsilon z;\mathcal{X}) - \mathcal{L}(\theta-\epsilon z;\mathcal{X})}{2\epsilon} \tag{3}$$

This definition requires at least two forward passes through the set of parameters where we set some perturbation ϵ in opposite directions. This perturbation is multiplied by some direction $\mathbf{z} \in \mathbb{R}$ with $\mathbf{z} \sim \mathcal{N}(0, I)$. z is sampled after every model update. In this work, we modify the sampling strategy to adapt it to the distributed setting.

2.5 Nearest Class Means Classifier

The use of the Nearest Class Means (NCM) algorithm in ML also has a long history. Each class is represented as a point in feature space defined by the mean feature vector of its training samples. New samples are classified by computing the distances between them and the class means and selecting the class whose mean is the nearest. In 1990, (Ratcliff (1990)) proposed to use NCM to mitigate catastrophic forgetting in continual learning and since then the use of NCM has been widely adopted and extended by continual learning researchers. This is due to its simplicity and minimal compute requirements to obtain a final classifier when a strong representation has already been learnt. Some of these methods include (Davari, Asadi, Mudur, Aljundi, and Belilovsky (2022); Z. Li and Hoiem (2017); Rebuffi, Kolesnikov, Sperl, and Lampert (2017)) who maintain a memory of exemplars used to compute an NCM classifier. Related to our work, recent literature in continual learning that have considered pretrained models were shown to ignore a simple NCM baseline (Janson, Zhang, Aljundi, and Elhoseiny (2022)) which can outperform many of the more complicated methods proposed.

Our focus on NCM here is based on a set of past observations commonly referred to as neural collapse (Mixon, Parshall, and Pi (2020); Papyan, Han, and Donoho (2020)). The phenomenon is an emergent property of neural network features after training. During training with examples of the same class, the model's logits begin cluster around a single point. As the model converges, these points grow further apart and equidistant from one another. NCM presents an important foundation for one of the methods proposed in Chapter 3.

2.6 Real-World Applications of Federated Learning

FL presents an array of opportunities for sectors with stringent privacy and regulatory requirements. The most commonly cited setting is that of healthcare where individual institutions having a relatively small number of examples may want to train a global model that stands to benefit from being trained on examples from other institutions (Darzidehkalani, Ghasemi-rad, and van Ooijen (2022); Dhade and Shirke (2023)). Internet of Things (IoT) is another domain often represented by studies in the cross-device setting with resource constrained clients (Bonawitz et al. (2019); Gill et al. (2024); D. C. Nguyen et al. (2021)). Larger institutions in finance and banking may also want to enable customers to benefit from each other's activities without have their highly confidential data leave their device or local branch. A few papers have looked at the applicability of FL in finance (T. Liu et al. (2023); Long, Tan, Jiang, and Zhang (2021)). Lastly, there has perhaps been the most work in the broader categories of NLP and computer vision (CV) in FL. Of the NLP use cases, search engines and next word predictions are an area of particular interest (Hard et al. (2019)). For CV, object detection and image classification are both tasks which have received their own attention (C. He et al. (2021); Y. Liu et al. (2020)). The methods covered in this work have applications across these domains including settings with comparatively low communication budgets where clients are connected over a wireless low-bandwidth network.

Chapter 3

Guiding the last layer with NCM

3.1 Method

3.1.1 Background and Notation

As we described in Chapter 2, distributed optimization in the FL setting occurs over K clients with each client $k \in \{1, ..., K\}$ having data $\mathbf{X}_k, \mathbf{Y}_k$ that contains n_k samples drawn from a distribution D_k. We concern ourselves with settings where $\mathbf{X}_k$ at each node is drawn from different distributions with some clients possessing more training samples than others. This setting proves more challenging for most FL algorithms who often suffer from performance degradation. We modify the definition for the typical objective function for federated optimization in Eq. 4 using the chain rule.

$$\mathbf{w}^*, \mathbf{v}^* \in \operatorname*{argmin}_{\mathbf{w},\mathbf{v}} \sum_{k=1}^{K} \frac{n_k}{n} \mathcal{L}(g(f(\mathbf{w}, \mathbf{X}_k), \mathbf{v}))\,. \tag{4}$$

Here we have split the model prediction into f, a base parameterized by $\mathbf{w}$ that produces representations, and g, a task head parameterized by $\mathbf{v}$. In this work we will focus on the case where the task head is a linear model, and the loss function, $\mathcal{L}$ represents a standard classification or regression loss. The $\mathbf{w}$ are derived from a pre-trained model and they can be optimized or held fixed.

One approach to obtain the task head while using a fixed $\mathbf{w}$ is to optimize only $\mathbf{v}$ in a federated manner over all the data. In the case that g is given as a linear model and we absorb the softmax into

Algorithm 1 FedNCM. K is the total number of clients, C is the number of classes in the training dataset, D_c is the total number of samples of class c

```
Require: $(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_K, \mathbf{Y}_K)$ - Local datasets, $w_{pt}$ - pre-trained model
    SERVER EXECUTES:
 1: for each client $k \in K$ in parallel do
 2:     $[m_c^k]_{c \in C} \leftarrow$ LocalClientStats($X_k, Y_k, \mathbf{w}_{pt}$)    ▷ Send to all clients, receive weighted class means
 3: end for
 4: for each class $c \in C$ do
 5:     $\mathbf{l}_c \leftarrow \frac{1}{D_c} \sum_{k=1}^{K} m_c^k$    ▷ $\mathbf{l}_c$ can be used in NCM classifier
 6: end for

    CLIENT SIDE:
 7: function LocalClientStats($\mathbf{X}, \mathbf{Y}, \mathbf{w}$)
 8:     for each class $c \in N$ do
 9:         Let $\mathbf{X}_c = \{x_i \in X, y_i = c\}$
10:         $m_c \leftarrow \sum_{x \in X_c} f_w(x)$
11:     end for
12:     return $[m_c]_{c \in C}$
13: end function
```

$\mathcal{L}$ this defaults to Linear Probing (LP)(J. Nguyen et al., 2023; Ren, Guo, Bae, & Sutherland, 2023).

3.1.2 FedNCM Algorithm

An alternative approach to derive an efficient g is through the use of NCM. We note that FedNCM, the federated version of NCM, can be derived exactly in a federated setting. In FedNCM, outlined in Algo. 1., the server only communicates pre-trained weights once to each of the clients and the clients only communicate once with the server to send back their weighted class means. The server can then use each client's class means to compute the NCM exactly and use them either to perform classification directly using the class centroids, or to initialize a linear task head for further fine-tuning. FedNCM allows an efficient classifier approximation for pre-trained models and addresses many critical concerns in the FL setting including:

(a) **communication and computation**: FedNCM is negligible in both compute and communication, it requires one communication from the server to each client and back. Used as an initialization for further FT, FedNCM speeds up convergence and reduces the communication and computation burden

(b) **client statistical heterogeneity**: Robust to typical non-iid distribution shifts (not the case for LP or FT). A FedNCM initialization also makes further FT more robust to heterogeneity.

Notably, FedNCM can be computed using secure aggregation methods. Furthermore, the lack of update to the base model parameters naturally improves differential privacy guarantees (Cattan, Choquette-Choo, Papernot, & Thakurta, 2022).

To use NCM as an initialization, consider the cross-entropy loss and $(g \circ f)(\mathbf{x}) = \mathbf{v}f(\mathbf{x};\mathbf{w}) + \mathbf{b}$. We can set the matrix v corresponding to the class c logit with the normalized class centroid $\mathbf{l}_c/\|\mathbf{l}_c\|$ and the bias term to 0. This allows us to initialize the task head with FedNCM and obtain further improvement through fine-tuning f.

3.1.3 Two-stage Approach for Transfer Learning in FL (HeadTune + FineTune)

This section was written by Edouard Oyallon as part of our collaboration for NeurIPS 2023.

FL algorithms are often unstable due to the mismatch in client objectives which can lead to large changes during local training causing significant deviations amongst the different client models. When using a pre-trained model which allows us a powerful initial representation, we argue that a two-stage procedure will improve training stability and converge more quickly. In the first stage (HeadTune) we perform HeadTuning where the parameters of g are updated *e.g.* by linear probing in federated fashion or by using FedNCM. As stated in Sec. 3.1.2, FedNCM is highly efficient, imposing a negligible cost in compute and communication with respect to any typical fine-tuning stage. In the second stage (FineTune), both f and the classifier initialized in stage one, are fine tuned together in a federated setting according to the FL objective function specified in Eq. 4. Taking the negligible cost of communication and compute provided by FedNCM into account, our two-stage approach can have a substantial advantage in convergence when compared to simply a fine-tuning stage (Chen et al., 2023; J. Nguyen et al., 2023).

We now give an intuitive interpretation of the advantages of our method using the framework of (Ren et al. (2023)). Assume that the k-th worker is initialized via $\mathbf{w}_0$, and trained locally with SGD for several steps until it reaches the parameter $\mathbf{w}_k$. Writing $\mathbf{w}^*$ the optimal parameter, via triangular inequality, we obtain the following inequality:

$$\mathbb{E}_{\mathbf{X}_k}[\|f(\mathbf{w}_k;\mathbf{X}_k)-f(\mathbf{w}^*;\mathbf{X}_k)\|] \leq \mathbb{E}_{\mathbf{X}_k}[\|f(\mathbf{w}_0;\mathbf{X}_k)-f(\mathbf{w}^*;\mathbf{X}_k)\|+\|f(\mathbf{w}_k;\mathbf{X}_k)-f(\mathbf{w}_0;\mathbf{X}_k)\|]\,. \tag{5}$$

In the neural tangent kernel (NTK) ((Jacot, Gabriel, & Hongler, 2018; Ren et al., 2023)) regime, for sufficiently small step size, (Ren et al. (2023)) showed that the second term depends on the approximation quality of the head g_0 at initialization, which is bounded (where σ is the sigmoid activation and $\{\mathbf{e}_i\}_i$ the canonical basis) for some $c > 0$, by:

$$\mathbb{E}_{\mathbf{X}_k}\|f(\mathbf{w}_k;\mathbf{X}_k)-f(\mathbf{w}_0;\mathbf{X}_k)\| \leq c\cdot\mathbb{E}_{(\mathbf{X}_k,\mathbf{Y}_k)}\|\mathbf{e}_{\mathbf{Y}_k}-g_{\mathbf{v}}(f(\mathbf{w}_0;\mathbf{X}_k))\|\,.$$

This suggests in particular that a good choice of linear head $\mathbf{v}$ will lead to a smaller right hand side term in Eq. 5, and thus reduce the distance to the optimum. Consequently, FedNCM or LP derived $\mathbf{v}$ (compared to a random $\mathbf{v}$) may be expected to lead to a more rapid convergence. Thanks to the initial consensus on the classifier, we may also expect less client drift to occur, at least in the first round of training, when $\mathbf{v}$ it initialized by HeadTuning, compared to a random initialization.

3.2 Experiments

In this section we will experimentally demonstrate the advantages of our proposed FedNCM and FedNCM+FT. Additionally, we show that simple LP tuning can at times be more stable and communication efficient than undertaking the full fine tuning considered almost exclusively in prior work on FL with pre-trained models.

Our primary experiments focus on standard image classification tasks. We also provide some NLP classification tasks in Sec. 3.2.2. We consider a setting similar to J. Nguyen et al. (2023) using the CIFAR 10 dataset (Krizhevsky, 2009) and expand our setting to include four additional standard computer vision datasets shown in Tab. 3.1. Following the method of Hsu et al. (2019), data is distributed between clients using a Dirichlet distribution parameterized by $\alpha = 0.1$ for our primary experiments. We set the number of clients to 100, train for 1 local epoch per round, and set client participation to 30% for CIFAR (as in J. Nguyen et al. (2023)). For all other datasets we use full client participation for simplicity.

Dataset	**Num. Classes**	**Num. Images**
CIFAR-10	10	50000
Flowers102	102	1020
Stanford Cars	196	8144
CUB	200	5994
EuroSAT-Sub	10	5000

Table 3.1: Summary of datasets used in our experiments.

Like J. Nguyen et al. (2023), we use SqueezeNet (Iandola et al., 2016), we also consider a ResNet18 (K. He, Zhang, Ren, & Sun, 2016) for experiments in Appendix A.5. When performing fine-tuning and evaluation for all datasets, we resize images to 224×224, the training input size of ImageNet. We run all experiments for three seeds using the FLSim library described in (J. Nguyen et al. (2023)).

Baseline methods We compare our methods to the following approaches as per J. Nguyen et al. (2023): (a) *Random*: the model is initialized at random with no use of pre-trained model or NCM initialization. This setting corresponds to the standard FL paradigm of (McMahan et al. (2017)). (b) *LP*: Given a pre-trained model, we freeze the base and train only the linear head using standard FL optimizer for training. (c) *FT*: A pre-trained model is used to initialize the global model weights and then a standard FL optimization algorithm is applied. (d) *LP and FT Oracles*: These are equivalent baselines trained in the centralized setting that provide an upper bound to the expected performance.

All of the above baseline methods as well as our FedNCM and FedNCM+FT can be combined with any core FL optimization algorithm such as FedAvg and FedAdam ((Reddi et al., 2020)). Our experiments, we focus on the high-performing FedAvg, FedProx and FedAdam which have been shown to do well in these settings in prior art (J. Nguyen et al., 2023).

Hyperpameters We follow the approach of (J. Nguyen et al. (2023); Reddi et al. (2020)) to select the learning rate for each method on the various datasets. For CIFAR-10 and SqueezeNet experiments we take the hyperparameters already derived in (J. Nguyen et al. (2023)). Additional details of selected hyperparameters are provided in Appendix A.2.

Communication and Computation Budget We evaluate the communication and computation costs of each proposed method. Costs are considered both in total and given a fixed budget for either communication or computation. For the communication costs, we assume that each model parameter that needs to be transmitted is transmitted via a 32-bit floating point number. This assumption allows us to compute the total expected communication between clients and server. It is important to emphasize that linear probing only requires that we send client updates for the classifier rather than the entire model as is the case in the other settings. Consequently, LP has much lower communication costs when compared to FT for any given number of rounds. Our proposed FedNCM is a one-round algorithm and therefore has even lower communication costs than any other algorithm considered.

For computation time we consider the total FLOPs executed on the clients. We assume for simplicity that the backward pass of a model is $2\times$ the forward pass. For example, in the case of LP (with data augmentation) each federated round leads to one forward communication on the base model, f, and one forward and one backward (equivalent to two forward passes) on the head, g. Similarly, for FedNCM the communication cost consists only one forward pass through the data.

3.2.1 Efficiency of Pure HeadTuning for FL

As discussed in Sec. 3.1.1 tuning the classifier head is at times as effective or more effective than updating the entire model in the context of transfer learning Evci et al. (2022). In prior work, this situation was briefly considered as a limited case in J. Nguyen et al. (2023, Appendix C.2) for CIFAR-10 and suggested that tuning just the linear head (LP) might be a weak approach in the

Dataset	Method	Accuracy (%)	Total Compute (×F)	Total Comm. (GB)
CIFAR-10	Random	67.8 ± 0.6	4.5×10^8	1803.71
	FT	85.4 ± 0.4	3.0×10^7	120.25
	FedNCM+FT	$\mathbf{87.2 \pm 0.2}$	3.0×10^7	120.25
	LP	82.5 ± 0.2	1.0×10^6	0.82
	FedNCM	64.8 ± 0.1	**1**	4.1×10^{-3}
CIFAR-10 ×32	Random (J. Nguyen et al. (2023))	34.2	1.5×10^8	601.24
	FT (J. Nguyen et al. (2023))	63.1	1.5×10^8	601.24
	FedNCM+FT	$\mathbf{67.9 \pm 0.4}$	7.5×10^7	300.62
	LP (J. Nguyen et al. (2023))	44.7	5.0×10^7	4.10
	FedNCM	40.02 ± 0.04	**1**	4.10×10^{-3}
FLOWERS-102	Random	33.2 ± 0.7	9.2×10^6	1916.76
	FT	64.5 ± 1.0	7.7×10^5	159.73
	FedNCM+FT	$\mathbf{74.9 \pm 0.2}$	7.7×10^5	159.73
	LP	74.1 ± 1.2	5.1×10^5	20.93
	FedNCM	71.8 ± 0.03	**1**	4.2×10^{-2}
CUB	Random	15.0 ± 0.7	5.4×10^7	2037.18
	FT	52.0 ± 0.9	1.8×10^7	679.06
	FedNCM+FT	$\mathbf{55.0 \pm 0.3}$	1.8×10^7	679.06
	LP	50.0 ± 0.3	9.0×10^6	122.88
	FedNCM	37.9 ± 0.2	**1**	8.2×10^{-2}
STANFORD CARS	Random	5.6 ± 0.8	8.6×10^7	2370.97
	FT	48.7 ± 2.0	2.4×10^7	677.42
	FedNCM+FT	$\mathbf{54.8 \pm 1.2}$	2.4×10^7	677.42
	LP	41.2 ± 0.5	2.0×10^7	200.70
	FedNCM	20.33 ± 0.04	**1**	8.0×10^{-2}
EUROSAT-SUB	Random	85.8 ± 2.7	5.3×10^7	2104.32
	FT	95.6 ± 1.0	1.5×10^7	601.24
	FedNCM+FT	$\mathbf{96.0 \pm 0.5}$	1.5×10^7	601.24
	LP	92.6 ± 0.4	5.0×10^6	4.10
	FedNCM	81.8 ± 0.6	**1**	$4..10 \times 10^{-3}$

Table 3.2: Accuracy, total computation and total communication costs of pure HeadTuning methods (below dashed lines) and their full training counterparts. CIFAR-10 ×32 indicates CIFAR-10 without re-sizing samples to 224 × 224, we include this setting so we can compare directly with values for FT, Random and LP reported by (J. Nguyen et al. (2023)). We observe pure HeadTuning approaches, FedNCM and LP can be powerful, especially under compute and communication constraints. The unit, F, used to measure communication is one forward pass of a single sample, details of the communication and computation calculations are provided in AppendixA.1.

heterogeneous setting. Here we first revisit this claim and expand the scope of these experiments to highlight where LP can be beneficial in terms of performance, communication costs, and compute time. Subsequently, we show another approach for approximating a good classifier, FedNCM, which can be competitive with orders of magnitude less computation and communication cost. We will demonstrate how to get the best of both HeadTuning and fine-tuning in the FL setting.

In (J. Nguyen et al. (2023)) the CIFAR-10 fine-tuning is done by feeding the 32×32 input image directly into a pre-trained ImageNet model. Since the architectures are adapted to the 224×224 size and trained at this scale originally, such an approach can lead to a very large distribution shift and may be sub-optimal for transfer learning. Thus we additionally compare to CIFAR-10 using the traditional approach of resizing the image to the source data (Evci et al., 2022; Kornblith et al., 2019).

Tab. 3.2 shows accuracy, compute, and communication cost results for Pure HeadTuning Methods (FedNCM and LP) as well as full tuning approaches including our FedNCM+FT. We note that in Tab. 3.2, CIFAR-10-32 $\times$ 32 refers to results published in J. Nguyen et al. (2023). We first point out the difference image input size has on the results and conclusion. Overall accuracy is much higher (highest is 86% vs 63%) and the gap between FT and LP is substantially smaller when using the model's native input size, it shows an absolute improvement of only 4.6% vs 18.4%. For both sizes of CIFAR-10 CUB, Stanford Cars and Eurosat. FedNCM without FT can substantially exceed random performance while maintaining a highly competitive compute and communication budget. For the Flowers102 dataset, FedNCM can already far exceed the difficult-to-train FT setting and furthermore, LP alone exceeds both FedNCM and FT. Our two-stage method of FedNCM+FT outperforms all other methods in terms of accuracy. In what follows we will show how FedNCM+FT also allows high efficiency given a specific, potentially limited compute and computational budget. When considering the results, we note that CIFAR-10 contains the same object categories as the original ImageNet dataset but the Flowers102 and CUB datasets, represent more realistic transfer learning tasks and under these conditions we observe the true effectiveness of HeadTuning methods such as FedNCM and LP.

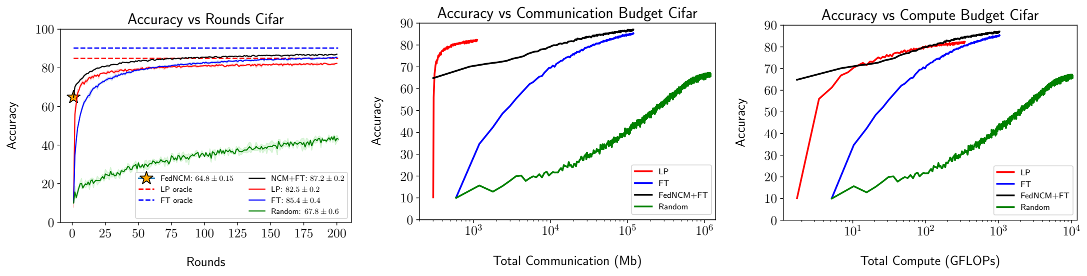

Figure 3.1: A comparison of the accuracy between models initialized with pre-trained weights and trained on CIFAR-10. The final result for both an LP and an FT oracle are shown and we remark that the NCM initialization allows the model to outperform the results of the LP oracle.

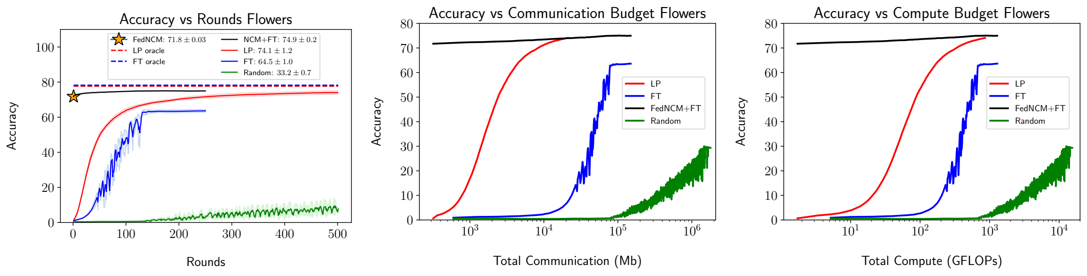

Figure 3.2: A comparison of the accuracy between models initialized with pre-trained weights and trained on Flowers. The final result for both an LP and an FT oracle are shown and we remark that the NCM initialization allows the model to outperform the results of the LP oracle.

3.2.2 FedNCM then FineTune

We now study in more detail the two-stage approach described in Sec. 3.1.3. Figures 3.1, 3.2, 3.3, 3.4 and 3.5 show the comparison of our baselines and FedNCM+FT with FedAvg. We show both accuracies versus rounds as well as accuracy given a communication and computation cost budget. Firstly, we observe that when going beyond CIFAR-10, LP can converge rather quickly and sometimes to the same accuracy as FT. This result shows the importance of HeadTuning and supports the use of LP in federated learning scenarios. Secondly, we can see the clear advantage of FedNCM+FT. After the stage one FedNCM initialization, it is able to achieve a strong starting accuracy and in stage two, it converges with a better accuracy than FT given the same computation budget. We note that FedNCM+FT for Cars and EuroSAT-Sub sees an initial performance decrease during the first few rounds of fine-tuning before rebounding to reach the best performance. This phenomenon is related to the chosen learning rate and further study is required to determine if this drop behaviour can be eliminated, potentially by using an initial warm-up period at a lower learning

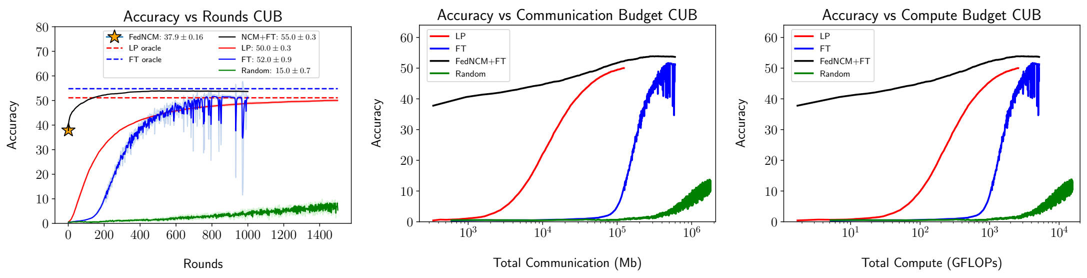

Figure 3.3: A comparison of the accuracy between models initialized with pre-trained weights and trained on CUB. The final result for both an LP and an FT oracle are shown and we remark that the NCM initialization allows the model to outperform the results of the LP oracle.

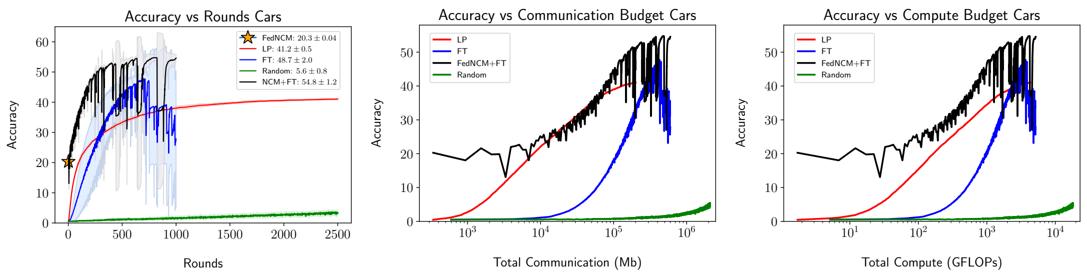

Figure 3.4: A comparison of the accuracy between models initialized with pre-trained weights and trained on Cars. The final result for both an LP and an FT oracle are shown and we remark that the NCM initialization allows the model to outperform the results of the LP oracle.

rate.

FedNCM+FT converges rapidly, allowing it to be highly efficient under most communication budgets compared to other methods. The second column of Tab. 3.2 shows that as we impose increasingly severe limitations on communication budgets, the performance gap between FedNCM+FT and all other methods widens significantly in favour of FedNCM+FT. Indeed for all the datasets FedNCM+FT is always optimal early on. For three of the datasets (Flowers, CUB, Cars) it exceeds LP over any communication budget. For CIFAR-10 and Eurosat LP can overtake it after the early stage, however, FedNCM+FT remains competitive and ultimately reaches higher performance. Similar trends are observed for computation time.We note overall as compared to FT the performance improvement can be drastic when considering the trade-off of accuracy as a function of communication and compute available. We also remark that the variance of LP and FedNCM+FT is lower across runs than the FT and Random counterparts.

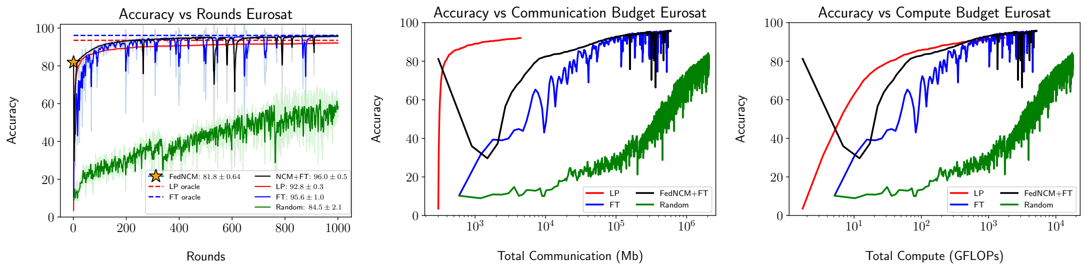

Figure 3.5: A comparison of the accuracy between models initialized with pre-trained weights and trained on EuroSAT. The final result for both an LP and an FT oracle are shown and we remark that the NCM initialization allows the model to outperform the results of the LP oracle.

We note that the Random baseline, typically requires longer training than others to reach the convergence criteria, thus for the purpose of our visualization we do not show the fully converged random baseline, which always requires many more communication rounds than the other approaches; however, the full curves are included in Appendix A.6.

NLP Experiments

We now explore the effectiveness of our approach on NLP tasks. For these experiments, we train a DistillBert model (Sanh, Debut, Chaumond, & Wolf, 2019) on the AG News dataset (Zhang, Zhao, & LeCun, 2015). Fig. 3.6 shows the comparison of LP, FT and FedNCM+FT where we vary the number of examples per class (15, 90, 1470). We observe the strong performance of FedNCM as a stand alone method, reinforcing previous observations around the importance of HeadTuning. As models get larger, the importance of the final layer

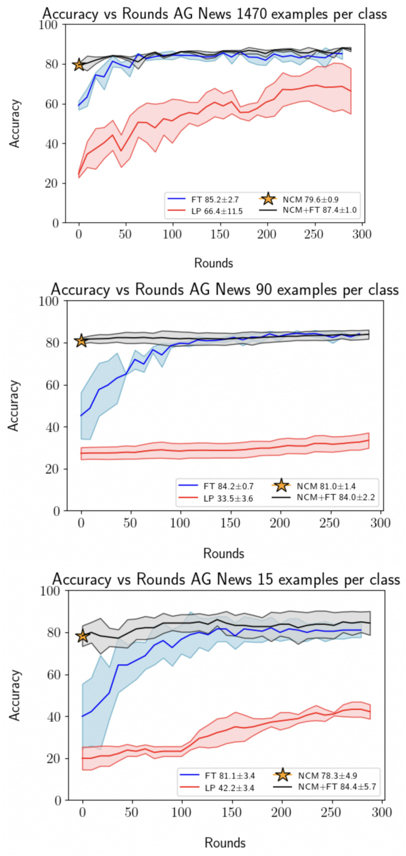

Figure 3.6: A comparison of the accuracy between models initialized with pre-trained trained using LP, FT and FedNCM+FT. We remark that the FedNCM alone obtains strong performance.

on the overall size of the model diminishes resulting in larger gaps in communication cost between one LP or FedNCM round and an FT round. Additionally, we find that FedNCM's accuracy is robust to sample size decreases and FedNCM+FT provides substantial convergence, and by extension communication cost benefits. Indeed it is well known that centroid based classifiers perform robustly in the case of few samples (Snell, Swersky, & Zemel, 2017). We provide additional confirmation of this in Appendix A.4 with experiments using small subsets of Cifar-10.

3.2.3 Analysis and Ablations

We now focus on demonstrating other advantages of our two-stage method, with a focus on using FedNCM as the method of choice for stage one (FedNCM+FT). In particular, we investigate robustness to larger number of clients, insensitivity to hyperparameters and compatibility with multiple FL algorithms and architectures. In Appendix A.3 we ablate our two-stage method by comparing FedNCM, LP and a three-layer MPL as methods for stage one.

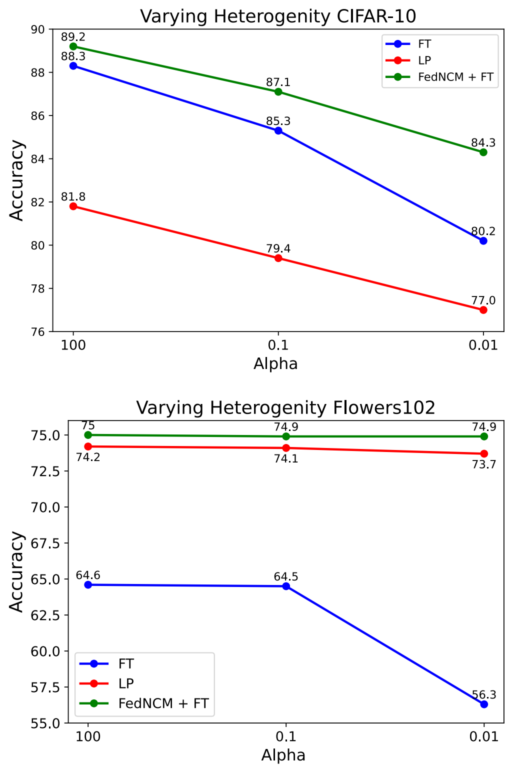

Figure 3.7: Varying the heterogeneity (Dirichlet-α) for CIFAR-10 and Flowers102 datasets.

Choice of FL Algorithm

So far we have focused on comparisons using FedAvg as the baseline algorithm; however, since our method can be widely applied in FL, we further analyze FedNCM+FT using the FedAdam optimizer and the FedProx algorithm. Tab. 3.3 summarizes the results of using each method with 1.) FedNCM+FT and 2.) only FT, for the Cifar-10 and Flowers datasets.

Dataset	Algorithm	FedNCM	FedNCM + FT	FT
CIFAR-10	FedAvg	64.8 ± 0.1	87.2 ± 0.2	85.4 ± 0.4
	FedProx	64.8 ± 0.1	88.1 ± 0.1	87.8 ± 0.08
	FedADAM	64.8 ± 0.1	$\mathbf{89.4 \pm 1.1}$	88.2 ± 0.2
Flowers102	FedAvg	71.8 ± 0.03	74.9 ± 0.2	64.5 ± 1.0
	FedProx	71.8 ± 0.03	75.2 ± 0.1	65.5 ± 1.5
	FedADAM	71.8 ± 0.03	$\mathbf{76.7 \pm 0.2}$	66.6 ± 1.0

Table 3.3: Model performance with different methods for a variety of FL algorithms for FedAvg, FedADAM and FedProx. FedNCM+FT outperforms in all cases.

We observe that improved FL optimizers can complement the two-stage FedNCM+FT which systematically exceeds the performance obtained when only fine-tuning. Regardless of the federated algorithm used, FedNCM alone continues to exceed the performance of FT on Flowers102. Our results suggest that choice of the FL optimization algorithm (J. Nguyen et al., 2023) is not always the most critical consideration for optimal performance when using pre-trained models.

Hyperparameter Tuning FL algorithms are known to be challenging for hyperparameter selection (Reddi et al., 2020) and this can affect their practical application. We first note that FedNCM does not have any hyperparameters which already provides a large advantage. In Fig. 3.9, we observe the final performance for a grid search over a range of server and client learning rates for FedAdam using both FT and FedNCM+FT. We observe that FedNCM+FT not only has higher performance but it is also more stable over the entire hyperparameter grid on Flowers dataset, and outperforms for all settings on CIFAR-10.

Heterogeneity J. Nguyen et al. (2023) points out that starting from a pre-trained model can reduce the effect of system heterogeneity. This is evaluated by comparing a specific Dirichlet distribution ($\alpha = 0.1$) used to partition data into a non-iid partitioning. Although the effect of heterogeneity is reduced we observe that in highly heterogeneous settings we still see substantial degradation in FT as shown in Fig. 3.7. Here we consider for CIFAR-10 the nearly iid $\alpha = 100$, $\alpha = 0.1$ as considered in J. Nguyen et al. (2023), and a very heterogeneous $\alpha = 0.01$. Firstly, we observe that FedNCM+FT can provide benefits in the iid setting.

As heterogeneity degrades the naive FT setting sees a large absolute and relative drop in performance. On the other hand, FedNCM+FT as well as LP are able to degrade more gracefully.

Varying the Local Epoch The number of local epochs can drastically affect FL aglorithms, typically a larger amount of local computation between rounds is desired to minimize communication. However, this can often come at a cost of degraded performance. We observe in Fig. 3.8 as in J. Nguyen et al. (2023) that FT can be relatively robust in some cases (CIFAR-10) to increasing local epochs. However, we also observe for some datasets that it can degrade, while LP and FedNCM+FT are less likely to degrade. Overall FedNCM+FT continues to outperform for larger local epochs.

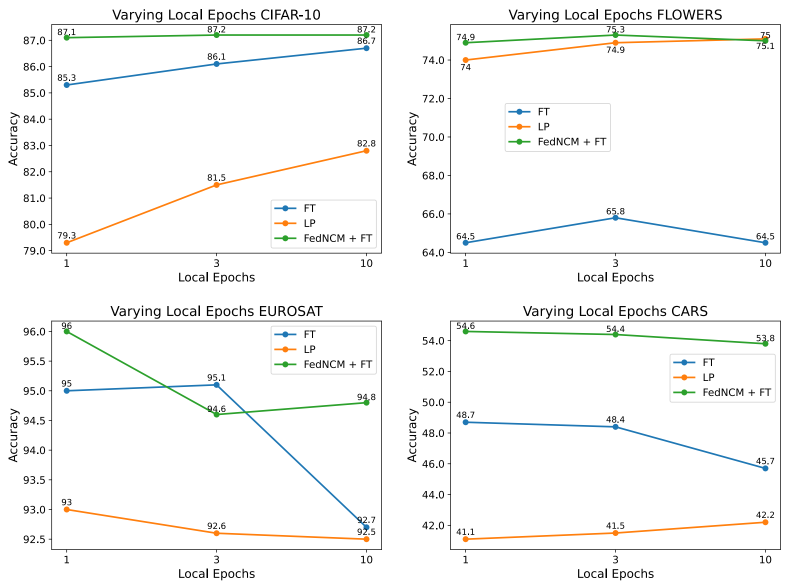

Figure 3.8: We vary the number of local epochs. FedNCM+FT always outperforms FT and nearly always LP in this challenging setting.

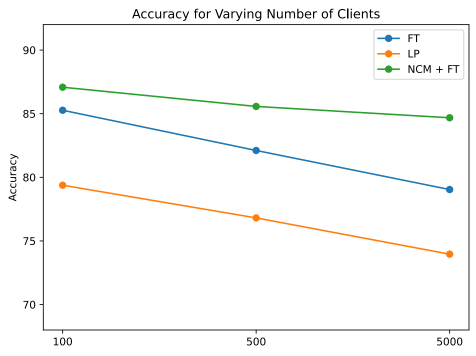

Figure 3.10: We increase the number of clients on CIFAR-10. FedNCM+FT degrades more gracefully than FT and LP.

Increasing clients Tab. 3.10 shows that as we increase the number of clients we observe that the degradation of FedNCM+FT is less severe than both LP and FT, suggesting it is stable under a large number of workers being averaged. As discussed in Sec. 3.1.3 it is expected in the same round that a representation would shift less from a starting point, and therefore since the starting point is the same for all clients, we expect the client drift within a round to be less given a fixed update budget. Hence, we expect to observe the same pattern

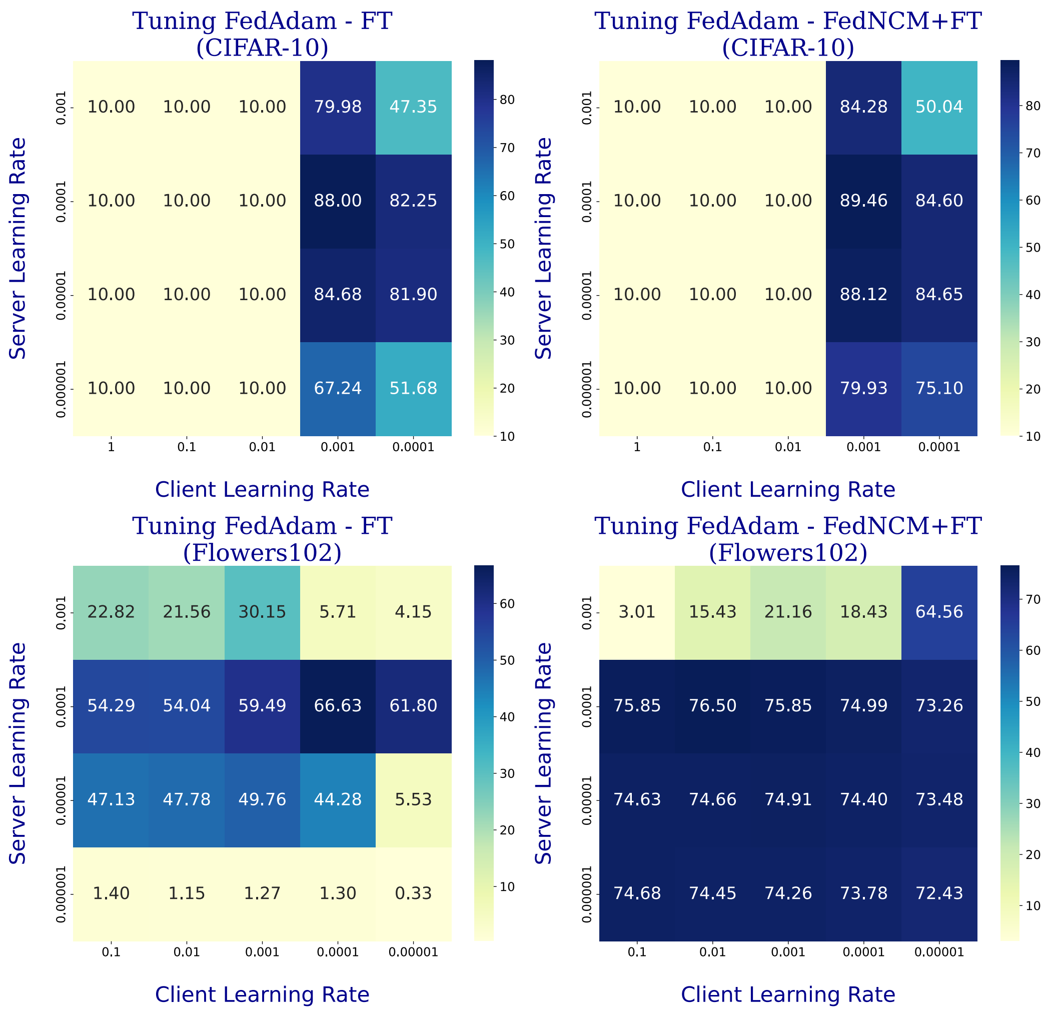

Figure 3.9: Hyperparameter grids for FedAdam for CIFAR-10 FT, FedNCMFT (left) and Flowers (right). We observe CIFAR-10 FedNCM-FT tends to do better or equal for all hyperparameters compared to FT. For Flowers it is much easier to tune, achieving strong values over a wide range, a noticeable advantage in FL

of robustness for FedNCM+FT under an increasing number of clients when training on tasks.

3.3 Conclusion

We have highlighted the importance of the last layers in FL from pre-trained models. We used this observation to then derive two highly efficient methods FedNCM and FedNCM+FT whose advantages in terms of performance, communication, computation, and robustness to heterogeneity were demonstrated. A limitation of our work is that it focus on image data and models, as this the primary set of data and models studied in prior work particularly in the context of transfer learning.

Chapter 4

Adapting zeroth-order learning to optimize communication costs

4.1 Method

In this section we describe FedMeZO a zeroth-order learning algorithm adapted to the FL setting. The algorithm follows the initial setup described in Chapter 2 where each client $k \in \{1, ..., K\}$ possesses data $\mathbf{X}_k$ consisting of n_k samples drawn from distribution D_k. The total number of samples across all clients is denoted as $n = \sum_{k=1}^{K} n_k$. The data $\mathbf{X}_k$ at each client may stem from different distributions and differ in quantity between individual clients. However, we demonstrate both theoretically and empirically that FedMeZO yields equivalent results regardless of the number of clients and whether data is iid or non-iid.

FedMeZO has a per-round communication cost of just a few bytes making it optimal in settings where total communication budget is extremely low. This is an important characteristic of the cross-device setting where clients are connected over unstable and low-bandwidth networks. Indeed, the clients and server limit themselves to exchanging a couple of scalar values per round. We describe this exchange in detail in Section 4.1.1. Importantly, our algorithm requires us to sample a consistent direction Z across clients enabling them to reconstruct the global model on-device once the average projected gradient is shared. This is not only key to optimizing client-server communication and preventing the exchange of any model weights, but it also ensures FedMeZO produces results equivalent to its counterpart in the centralized setting Malladi et al. (2024) given full client participation and an equivalent step size.

4.1.1 FedMeZO

As is the case for most FL algorithms, FedMeZO trains a global model iteratively by performing local updates on a set of chosen clients and aggregating the updated models. For most algorithms, local updates measure model gradients by performing forward passes on the client's local data used to generate a global loss signal, which is backpropagated through the model. However, FedMeZO performs client updates using an adaption of MeZO (Malladi et al. (2024)). As described in Chapter 2, MeZO estimates a layer-wise loss by conducting two forward passes with some perturbation in opposite directions. It uses this signal to measure the gradient direction used to update the weights for the current model layer. With FedMeZO, we return the gradient direction to the server rather than update the client model directly. This direction is a scalar. We average projected gradients across all selected clients and use it to update models as we would in the centralized setting. Global model update occurs on-device across all clients. We describe FedMeZO more formally in Algorithms 2 (server orchestration) and 3 (local training). When measuring the amount of communication per round, we importantly notice that we do not share any weights beyond the initial pretrained model at the start of training. Rather, a round involves the communication of three numbers (1 byte each) between the server and each of its clients. These integers are shared at different times during a given round and are represented in chronological order by the seed, the individual client's estimated gradient and the average estimated gradient across clients.

In FedMezo, the server's only job is to select a random seed s and compute the average of all projected client gradients. The server does not receive or update model weights as this is done by each client on-device. Intuitively, the information shared each round represents a set of scalars numbered in proportion to the number of round participants. This procedure addresses the following critical concerns in the FL setting:

(1) **communication**: FedMeZO requires the server and clients to communicate a relatively small

Algorithm 2 FedMeZO: Server (main) orchestrates training across clients.

```
Require: Number of rounds $R$, number of clients $K$, model parameters $w_0$, seed $s$, number of samples $num\_z$.
 1: procedure SERVER EXECUTES:
 2:     Send initial model $w_0$ to all clients
 3:     for $i = 1$ to $R$ do
 4:         Pick seed $s$
 5:         for each client $k \in K$ do
 6:             $proj\_grads\_k \leftarrow$ ClientFedMeZOProj$(s, num\_z)$    ▷ Compute projected gradients for client $k$
 7:         end for
 8:         $proj\_grad\_avg \leftarrow$ Average$(proj\_grads\_k)$ for $k \in K$    ▷ Average gradients across all clients
 9:         for each client $k \in K$ do
10:             ClientFedMeZOUpdate$(s, num\_z, proj\_grad\_avg)$
11:         end for
12:     end for
13: end procedure
```

Algorithm 3 FedMeZO: FedMeZOProj and FedMeZOUpdate run the local forward passes and reconstruct the global model using the average projected gradient.

```
Require: Perturbation scale $\epsilon$, model parameters $w\_k$, local learning rate $lr$, seed $s$, number of
    samples $num\_z$, size of model layer $D$.
    CLIENT EXECUTES:
1:  function CLIENTFEDMEZOPROJ($\mathbf{s}, \mathbf{num\_z}$)
2:      SetSeed($s$)
3:      $Z = \text{random}(num\_z, D)$          ▷ The same $Z$ is consistently generated across clients.
4:      for $p = 1$ to $num\_z$ do
5:          $X_k = \text{LoadNextBatch}()$
6:          $z_p = Z[p, :]$                     ▷ Use the $p$-th slice of random matrix
7:          $proj\_grad\_J[p] = \text{MeZO}(X_k, \text{current model on client } w_k, \epsilon, z_p)$
8:      end for
9:      return $proj\_grad$                     ▷ Return the projected gradient
10: end function

11: function CLIENTFEDMEZOUPDATE($\mathbf{s}, \mathbf{num\_z}, \mathbf{proj\_grad\_avg}$)
12:     SetSeed($s$)
13:     $Z = \text{random}(num\_z, D)$          ▷ The same $Z$ is consistently generated across clients.
14:     $grad\_est = 0$                         ▷ Initialize gradient estimate
15:     for $p = 1$ to $num\_z$ do
16:         $grad\_est + = proj\_grad\_avg[p] * Z[p, :]$
17:     end for
18:     $grad\_est = grad\_est/num\_z$           ▷ Estimate the client gradient direction
19:     $w_{k+1} = w_k - lr * grad\_est$        ▷ Update the model parameters locally
20: end function
```

number of scalars rather than all or portions of the entire model. This reduces the communication cost for model convergence. It also holds the per-round communication budget steady regardless of model size. As models get larger, the gap in communication budget for FedMeZO and other PEFT or full fine-tuning methods grows.

(2) **client memory**: As described in the original paper, MeZO requires 5x less memory than traditional backpropagation. The algorithm is also complementary to other PEFT methods like LoRA (Hu et al. (2021)).

(3) **privacy**: Given the low dimensionality of the information shared by clients, FedMeZO reduces the risk of information leakage. The method never requires clients to directly share model parameters or their gradients with the server. Notably, information is also never shared directly between clients who are the only ones to hold updated model parameters. Returning the average of the projected gradients to clients rather than a set of individual client projected gradients ensures the solution is at least as private as widely accepted aggregation methods like FedAvg (McMahan et al. (2017)).

4.1.2 Setting the seed

In this section, we reinforce the importance and manner in which we set the random seed across clients for local training. Broadly, seeds are used by random number generators to create reproducible outputs. Hence, FedMeZO requires a random number generator capable of maintaining context across all clients, so that the server and its clients K generate the same set of random directions Z for a given round as shown in Fig. 4.1.

This is critical to ensure the model update is equivalent regardless of the number of clients and whether the data is iid or non-iid across clients. Empirically, we observe that allowing clients to independently generate Z produces a gap in performance when training on non-iid data. For a given training step, the same Z is used during both gradient projection and client update. FedMeZO's gradient projection step is equivalent to running MeZO (Malladi et al. (2024)) for multiple batches and accumulating the projected gradient for a number of batches equal to the number of clients. Consequently, our current implementation of FedMeZO requires us to limit local training to a single

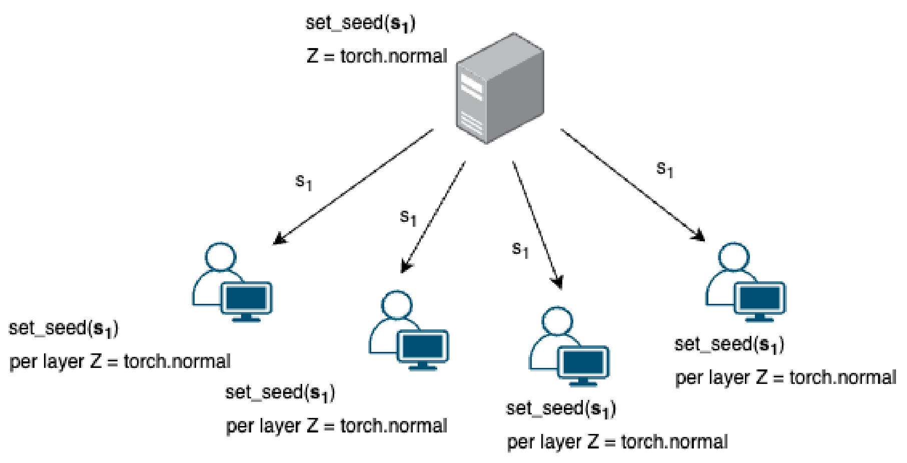

Figure 4.1: For each round, a new seed s is set by the server and shared with every client. During local gradient projection, clients randomly generate the same value for Z.

step per client per local round. Then, the client model update step enables each clients to reconstruct the global model locally using the average projected gradient shared by the server. When scaling the step size to the X_k, this update is again equivalent to that of the model update in the centralized setting. In Fig. 4.2, we emphasize the importance of properly setting the seed by highlighting the gap in accuracy between FedMeZO and a naive implementation of MeZO (Malladi et al. (2024)) in the distributed setting where the seed is set independently for each client and local training occurs for a number of steps. FedMeZO outperforms DistMeZO Naive across all tasks.

4.2 Experiments

In this section, we detail the main components of our experimentation and demonstrate the ability of FedMeZO to perform well on iid and non-iid data. Our primary experiments focus on natural language processing tasks and more specifically sentence classification.

4.2.1 Experimental Details

All experiments are carried out using a pretrained RoBERTa-large (Y. Liu et al. (2019)) based on implementations in (Gao, Fisch, and Chen (2021); Malladi, Wettig, Yu, Chen, and Arora (2023)). We perform some tuning of the learning rate and perturbation scale. Both have similar effects on

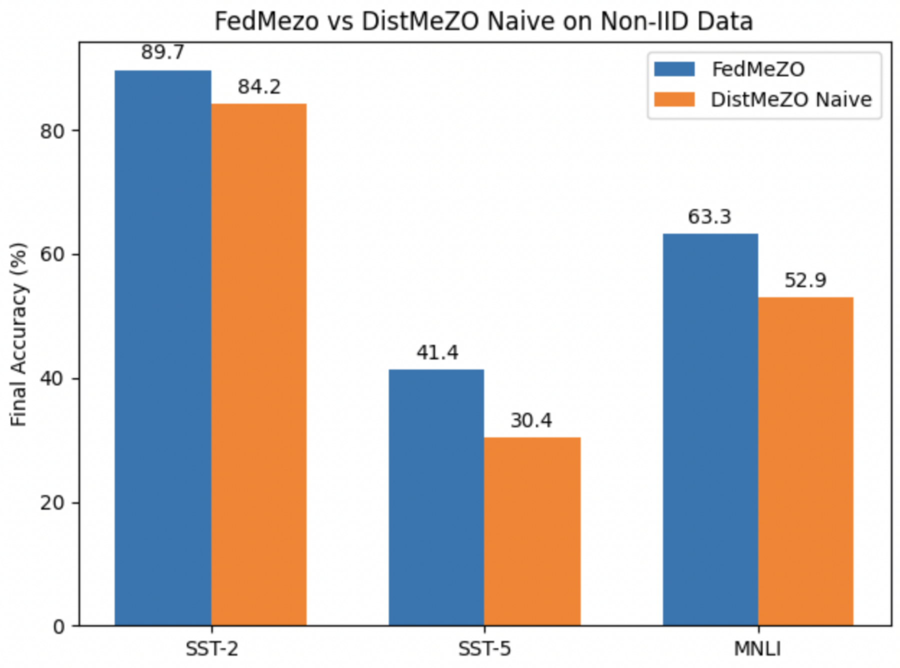

Figure 4.2: A comparison of the final accuracy between models trained using FedMeZO and a naive implementation of MeZO in the distributed setting (DistMeZO Naive). To converge, DistMeZO Naive requires a larger amount of local steps per round. As described, FedMeZO results in better model performance across the set of tasks.

the step size, so we choose to hold the perturbation scale steady at $1e-3$ across our experiments after some testing and choose to tune the learning rate.

Baseline methods We compare our method to SLoRA (Babakniya et al. (2023)), which is designed to improve the performance of tuning using LoRA in the FL setting. As reflected in Table 4.1, this method has a considerably larger per-round communication cost. However, we choose this method as our baseline given its ability to demonstrate strong performance on NLP tasks using pretrained models in the FL setting. We also chose it given its compatibility with LoRA and other PEFT methods, which is

Dataset	**Num. Classes**	**Num. Rows (in thousands)**
SST-2	2	67.3
SST-5	5	8.54
MNLI	3	392.7

Table 4.2: Summary of datasets used in our experiments.

Dataset	**Method**	**# Clients**	**Accuracy (%)**	**Total Comm. (GB)**
SST-2	FedMeZO IID	10	89.7	7.863×10^{-5} (2621 rounds)
	FedMeZO NON-IID	10	89.7	7.863×10^{-5} (2621 rounds)
	SLoRA IID	10	93.5	69.254
	SLoRA NON-IID	10	83.1	69.254
MNLI	FedMeZO IID	10	63.3	5.115×10^{-5} (1705 rounds)
	FedMeZO NON-IID	10	63.3	5.115×10^{-5} (1705 rounds)
	SLoRA IID	10	82.3	69.254
	SLoRA NON-IID	10	78.1	69.254
SST-5	FedMeZO IID	10	41.4	2.907×10^{-5} (969 rounds)
	FedMeZO NON-IID	10	41.4	2.907×10^{-5} (969 rounds)
	SLoRA IID	10	52.0	69.254
	SLoRA NON-IID	10	41.8	69.254

Table 4.1: Accuracy and total communication cost of FedMeZO and SLoRA (above dashed line). We collect results for the methods on Stanford Sentiment Treebank (SST-2, SST-5) and the Multi-Genre Natural Language Inference (MNLI). We exclude the initial model sharing step between the server and clients from the total communication given its outsides impact ($> 99.99\%$) on the total communication when training with FedMezo.

also a property of our method. We run experiments for MeZO, but do not include them out explicitly. As mentioned in Section 4.1.1, it is trivial to derive the equivalence between MeZO in the centralized setting with gradient accumulation for multiple batches and FedMeZO assuming full client participation. Hence, MeZO and FedMeZO result in the same final accuracy when adjusting for step size. For SLoRA, we conduct our experiments with 50 total global rounds and 10% density during model priming (step 1). We train the model using step 1 for 20% of the total training rounds. This setup aligns with a subset of the main results displayed in Babakniya et al. (2023). They report the importance of ensuring a sufficient number of training rounds for step 1.

Federated Learning We carry out our experiments in the iid and non-iid setting where data is distributed to clients using a Dirichlet distribution parameterized by $\alpha = 100$ and $\alpha = 0.1$ respectively (Hsu et al. (2019)). Our experiments are conducted with 1 local step per round with full client participation across the 10 clients. The random seed is set by the server and shared with clients. Its value is unique to every round.

Datasets We measure the effectiveness of our approach on sentence classification and textual entailment tasks. SST-2 (Socher et al. (2013)) is a binary classification task where the goal is to predict whether a given movie review expresses positive or negative sentiment. SST-5 has more granularity (5 classes), but is similarly composed by a set of reviews. Lastly, the Multi-Genre Natural Language Inference (MNLI) (Williams, Nangia, and Bowman (2018)) dataset evaluates the model's ability to perform natural language inference across various genres, testing its understanding of textual entailment across a number of contexts. The datasets are described in Table 4.2.

Communication Budget We measure the communication cost of proposed methods in Gb. We derive the per-round cost and extrapolate based on the number of rounds to train the model to convergence. We assume models are set at 32-bit floating point precision for model parameters and 8-bit numbers for the random seeds and gradient directions. This allows us to calculate the total communication between the server and clients.

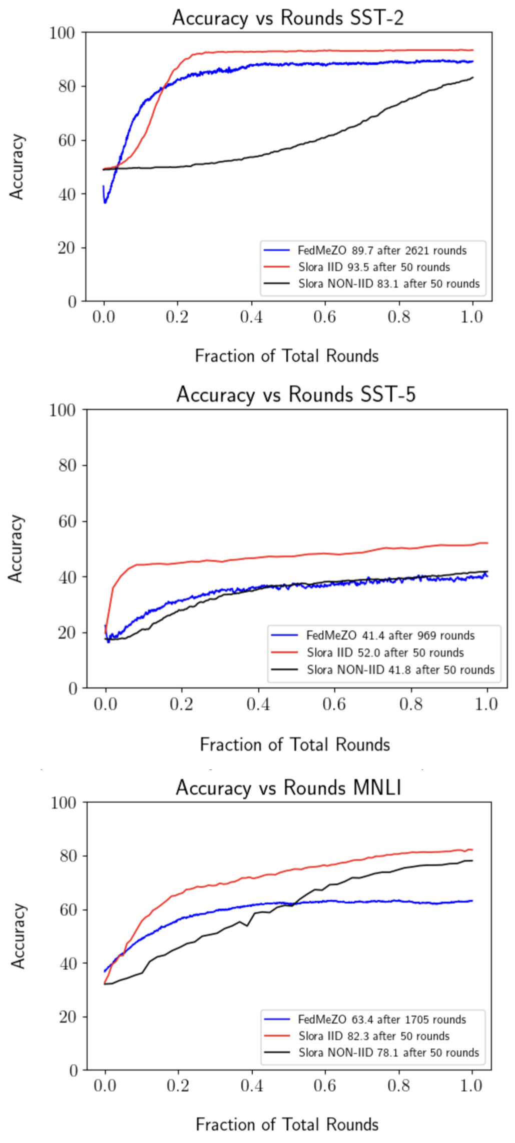

Figure 4.3: A comparison of the accuracy between models trained using FedMeZO, SLoRA on iid data and SLoRA on non-iid data. As described, FedMeZO converges the same regardless of number of clients or data distribution.

For FedMeZO, the per-round communication cost is directly proportional to the number of forward passes used to estimate the projected gradient denoted in Algorithms 2 (server orchestration) and 3 (local training) by num_z and the number of participating clients K. We set num_z to 2 for all of our experiments as to be consistent with the method described in Malladi et al. (2024). This represents two random directions for 1 step per local client update. To reconstruct the global model locally, clients are each sent $proj\_grad\_avg$ and the seed used to generate Z. Equation 6 describes MeZO's total communication cost $Cost$ after sending the initial model to clients.

$$\text{Cost} = R \cdot (K + 2 \cdot K \cdot \text{num\_z}) \cdot 1 \times 10^{-9} \text{ in GB} \tag{6}$$

For SLoRA, the communication cost can be easily derived from the the density used for step 1 (model priming) and the percentage of total rounds covered by step 1. The remaining rounds are covered by step 2 (LoRA). As noted, we use RoBERTa-large in all of our experiments, which is composed of 355M parameters with LoRA set the dimension of the low-rank matrices at 16. Eq. 7 describes SLoRA's total communication cost $Cost$.

$$\text{Cost} = R_{\text{step\_1}} \cdot K \cdot \text{density} \cdot \text{model\_size\_in\_GB} + R_{\text{step\_2}} \cdot K \cdot \text{LoRA\_size in GB} \tag{7}$$

When comparing the communication cost of both algorithms, we notice they are naturally a factor of the number of total rounds R. However, the term $density * \text{model\_size\_in\_GB}$ is the primary driver across both algorithms considering:

$$\text{model\_size\_in\_GB} \gg (K + 2 \cdot K \cdot \text{num\_z}) \text{ bytes.}$$

Hence, the large gap in communication budget between FedMeZO and SLoRA can be explained by the former leveraging a form of compressed update information rather than sharing its raw model weights. This method of aggregation also shelters FedMeZO's communication cost from rising dramatically due to training substantially larger models.

4.2.2 Results

Although FedMeZO can be applied to a number of NLP tasks, our initial experiments focus on text classification. As demonstrated in Table 4.1, FedMeZO reaches convergence with minimal communication cost. Across all datasets in Figure 4.3, FedMeZO initially converges faster than SLoRA in the non-iid setting, thus demonstrating that it can potentially outperform SLoRA in settings where the communication budget represents a small fraction of the total model size. For SLoRA, our results are consistent with previous findings stressing the importance of proper model priming. Indeed, we find that model convergence at the end of step 1 is a strong indicator of final model performance. As expected, SLoRA does suffer from some performance degradation in the non-iid setting. The gap in performance caused by non-iid data is most pronounced for our simplest task, SST-2. We do not experiment with pathological non-iid data where SLoRA seems to suffer from its sharpest performance drop Babakniya et al. (2023).

The results for FedMeZO are equivalent regardless of the data being iid or non-iid. Empirically, we find it is critical that the local batch size is set to the size of the local dataset. This ensures the model performs a forward pass using every example in the dataset once per round across all settings. It also improves stability and reduces variance of projected local gradients regardless of the number of clients, thus ensuring a more accurate estimation of the projected gradient across all clients. Reducing the batch size prevents the model from converging. Assuming full client participation, a complete distribution of the original dataset to clients and a tuned step size via the local learning rate and perturbation scale, FedMeZO converges at the same rate as MeZO regardless of the number of clients. As mentioned in Section 4.1.1, FedMeZO is able to accommodate for models up to x5 larger as measured by the number of total parameters compared to SLoRA. We leave this comparison up to future works.

4.3 Conclusion

We highlighted the resource constraints presented by the FL setting and explored the ability to learn in a distributed setting without directly sending updated model parameters over the network.

Our method, FedMeZO, inherits properties of MeZO in the centralized setting, which reduces memory constraints for model fine-tuning down to those required for model inference. We further adapt it to minimize the total communication budget. Future directions for this work will not only include more experiments, but also an adaptation of FedMeZO incorporating some model priming and local learning based on a layer-wise objective.

Chapter 5

Conclusions and Future Work

5.1 Contributions

In this work, we focused on the use of pretrained models to increase convergence speed in the federated learning setting. We introduced methods for first-order and zeroth-order learning using pretrained models in the language and vision domains. We demonstrated our methods' ability to reduce communication budget, compute and memory requirements for clients.

Last layer tuning: With this work, we confirmed previous observations made by J. Nguyen et al. (2023) demonstrating the effectiveness of pretrained models in FL where the objective is to produce a global model. Through a number of experiments, we highlighted the importance of tuning the last layer or classifier. We continued by exploring the application of Nearest Means Classifiers and introduced FedNCM and LP as two forms of head tuning. We demonstrated the effectiveness of combining FedNCM with full fine-tuning to improve final model accuracy and convergence speed.

Zeroth-order optimization: With this work, we demonstrated the ability of zeroth-order methods using pretrained models to translate to the FL setting. We adapted the algorithm described by Malladi et al. (2024) to the setting reducing the communication cost per round to a factor of 3 scalars per client. We emphasized the method's lowered memory requirements and showed that it yields equivalent results regardless of client data distribution. This positioned our method as a viable option in the cross-device where clients are often hardware-constrained and connected over unreliable networks.

5.2 Future Work

Improving convergence: We established the benefits of pretrained over randomly initialized models in the FL setting to reduce client memory and compute as well as increase robustness with non-iid data. During our training, we notice some success tuning clients for very large number of local rounds as compared to global rounds. It may be relevant to expand on this observation to formalize the relationship between these two hyperparameters on their effects on accuracy. Although we demonstrate strong performance under tight communication budgets, gaps also remain between FedMeZO and other state-of-the-art FL methods like SLoRA applying PEFT to the setting for larger budgets. Algorithms could be developed improving on the model's learning objective. Improving on the model's local objective would maintain memory requirements while perhaps ultimately reducing the need for a second forward pass and improve final model accuracy.

Moving beyond classification: The bulk of the experiments in this thesis cover classification tasks across language and vision. With sequence completion in text being the primary driver for the broader application of Large Language Models (LLMs), it could be beneficial to adapt the methods described in this work to this task and others such as sequence labeling, summary creation or question and answering. Beyond language and vision, the methods are applicable to a broader set of domains.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In *Proceedings of the 2016 acm sigsac conference on computer and communications security.* ACM. Retrieved from http://dx.doi.org/10.1145/2976749.2978318 doi: 10.1145/2976749.2978318

Alistarh, D., Grubic, D., Li, J., Tomioka, R., & Vojnovic, M. (2017). *Qsgd: Communication-efficient sgd via gradient quantization and encoding.* Retrieved from https://arxiv.org/abs/1610.02132

Alyafeai, Z., AlShaibani, M. S., & Ahmad, I. (2020). A survey on transfer learning in natural language processing. *arXiv preprint arXiv:2007.04239*.

Babakniya, S., Elkordy, A. R., Ezzeldin, Y. H., Liu, Q., Song, K.-B., El-Khamy, M., & Avestimehr, S. (2023). *Slora: Federated parameter efficient fine-tuning of language models.*

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., . . . Roselander, J. (2019). *Towards federated learning at scale: System design.* Retrieved from https://arxiv.org/abs/1902.01046

Cattan, Y., Choquette-Choo, C. A., Papernot, N., & Thakurta, A. (2022). Fine-tuning with differential privacy necessitates an additional hyperparameter search. *arXiv preprint arXiv:2210.02156*.

Chakrabarti, A., & Moseley, B. (2019). *Backprop with approximate activations for memory-efficient network training.* Retrieved from https://arxiv.org/abs/1901.07988

Chen, H.-Y., Tu, C.-H., Li, Z., Shen, H.-W., & Chao, W.-L. (2023). On the importance and applicability of pre-training for federated learning.

Darzidehkalani, E., Ghasemi-rad, M., & van Ooijen, P. (2022). Federated learning in medical imaging: Part i: Toward multicentral health care ecosystems. *Journal of the American College of Radiology*, *19*(8), 969-974. Retrieved from https://www.sciencedirect.com/science/article/pii/S1546144022002800 doi: https://doi.org/10.1016/j.jacr.2022.03.015

Davari, M., Asadi, N., Mudur, S., Aljundi, R., & Belilovsky, E. (2022). Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 16712–16721).

Dhade, P., & Shirke, P. (2023). Federated learning for healthcare: A comprehensive review. *Engineering Proceedings*, *59*(1). Retrieved from https://www.mdpi.com/2673-4591/59/1/230 doi: 10.3390/engproc2023059230

Du, Y., Yang, S., & Huang, K. (2020). High-dimensional stochastic gradient quantization for communication-efficient edge learning. *IEEE Transactions on Signal Processing*, *68*, 2128–2142. Retrieved from http://dx.doi.org/10.1109/TSP.2020.2983166 doi: 10.1109/tsp.2020.2983166

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, *9*, 211-407. Retrieved from https://api.semanticscholar.org/CorpusID:207178262

Evci, U., Dumoulin, V., Larochelle, H., & Mozer, M. C. (2022). Head2toe: Utilizing intermediate representations for better transfer learning. In *International conference on machine learning* (pp. 6009–6033).

Fallah, A., Mokhtari, A., & Ozdaglar, A. (2020). Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 3557–3568). Curran Associates, Inc.

Gao, T., Fisch, A., & Chen, D. (2021). *Making pre-trained language models better few-shot learners.*

Gill, S. S., Golec, M., Hu, J., Xu, M., Du, J., Wu, H., . . . Uhlig, S. (2024). *Edge ai: A taxonomy, systematic review and future directions.* Retrieved from https://arxiv.org/

abs/2407.04053

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Gomez, A. N., Ren, M., Urtasun, R., & Grosse, R. B. (2017). *The reversible residual network: Backpropagation without storing activations.* Retrieved from https://arxiv.org/abs/1707.04585

Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., … Ramage, D. (2019). *Federated learning for mobile keyboard prediction.* Retrieved from https://arxiv.org/abs/1811.03604

He, C., Shah, A. D., Tang, Z., Sivashunmugam, D. F. N., Bhogaraju, K., Shimpi, M., … Avestimehr, S. (2021). *Fedcv: A federated learning framework for diverse computer vision tasks.* Retrieved from https://arxiv.org/abs/2111.11066

He, K., Girshick, R., & Dollar, P. (2019, October). Rethinking imagenet pre-training. In *Proceedings of the ieee/cvf international conference on computer vision (iccv).*

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.* Retrieved from https://arxiv.org/abs/1512.03385

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hitaj, B., Ateniese, G., & Perez-Cruz, F. (2017). *Deep models under the gan: Information leakage from collaborative deep learning.* Retrieved from https://arxiv.org/abs/1702.07464

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., … Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International conference on machine learning* (pp. 2790–2799).

Hsu, T.-M. H., Qi, H., & Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335.*

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., … Chen, W. (2021). *Lora: Low-rank* *adaptation of large language models.*

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size.

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc.

Janson, P., Zhang, W., Aljundi, R., & Elhoseiny, M. (2022). A simple baseline that questions the use of pretrained-models in continual learning. *arXiv preprint arXiv:2210.04428*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., Child, R., ... Amodei, D. (2020). *Scaling laws for neural language models.* Retrieved from https://arxiv.org/abs/2001.08361 (arXiv:2001.08361 [cs.LG])

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning* (pp. 5132–5143).

Konečnỳ, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2017). *Federated learning: Strategies for improving communication efficiency.* Retrieved from https://arxiv.org/abs/1610.05492

Kornblith, S., Shlens, J., & Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 2661–2671).

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. , 32–33. Retrieved from https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Legate, G., Caccia, L., & Belilovsky, E. (2023). Re-weighted softmax cross-entropy to control forgetting in federated learning. In *Proceedings of the 2nd conference on lifelong learning agents.*

Li, D., & Wang, J. (2019). *Fedmd: Heterogenous federated learning via model distillation.* Retrieved from https://arxiv.org/abs/1910.03581

Li, S., Xia, Y., & Xu, Z. (2022). *Simultaneous perturbation stochastic approximation: towards one-measurement per iteration.* Retrieved from https://arxiv.org/abs/2203.03075

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020a). Federated optimization in heterogeneous networks.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020b). *Federated optimization in heterogeneous networks.* Retrieved from https://arxiv.org/abs/1812.06127

Li, X. L., & Liang, P. (2021). *Prefix-tuning: Optimizing continuous prompts for generation.*

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, *40*(12), 2935–2947.

Lian, D., Daquan, Z., Feng, J., & Wang, X. (2022). Scaling & shifting your features: A new baseline for efficient model tuning. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in neural information processing systems.* Retrieved from https://openreview.net/forum?id=XtyeppctGgc

Lian, X., Zhang, W., Zhang, C., & Liu, J. (2018). *Asynchronous decentralized parallel stochastic gradient descent.* Retrieved from https://arxiv.org/abs/1710.06952

Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, W. J. (2020). *Deep gradient compression: Reducing the communication bandwidth for distributed training.* Retrieved from https://arxiv.org/abs/1712.01887

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero, A., & Varshney, P. K. (2020). *A primer on zeroth-order optimization in signal processing and machine learning.* Retrieved from https://arxiv.org/abs/2006.06224

Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., & Amini, L. (2018). Zeroth-order stochastic variance reduction for nonconvex optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/

2018/file/ba9a56ce0a9bfa26e8ed9e10b2cc8f46-Paper.pdf

Liu, T., Wang, Z., He, H., Shi, W., Lin, L., Shi, W., . . . Li, C. (2023). *Efficient and secure federated learning for financial applications.* Retrieved from https://arxiv.org/abs/2303.08355

Liu, Y., Huang, A., Luo, Y., Huang, H., Liu, Y., Chen, Y., . . . Yang, Q. (2020). *Fedvision: An online visual object detection platform powered by federated learning.* Retrieved from https://arxiv.org/abs/2001.06202

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). *Roberta: A robustly optimized bert pretraining approach.*

Long, G., Tan, Y., Jiang, J., & Zhang, C. (2021). *Federated learning for open banking.* Retrieved from https://arxiv.org/abs/2108.10749

Lyu, L., Xu, X., & Wang, Q. (2020). *Collaborative fairness in federated learning.* Retrieved from https://arxiv.org/abs/2008.12161

Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., & Arora, S. (2024). *Fine-tuning language models with just forward passes.*

Malladi, S., Wettig, A., Yu, D., Chen, D., & Arora, S. (2023). *A kernel-based view of language model fine-tuning.*

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282).

Mixon, D. G., Parshall, H., & Pi, J. (2020). *Neural collapse with unconstrained features.* Retrieved from https://arxiv.org/abs/2011.11619

Mora, A., Tenison, I., Bellavista, P., & Rish, I. (2022). *Knowledge distillation for federated learning: a practical guide.* Retrieved from https://arxiv.org/abs/2211.04742

Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Vincent Poor, H. (2021). Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys amp; Tutorials*, *23*(3), 1622–1658. Retrieved from http://dx.doi.org/10.1109/COMST.2021.3075439 doi: 10.1109/comst.2021.3075439

Nguyen, J., Wang, J., Malik, K., Sanjabi, M., & Rabbat, M. (2023). Where to begin? on the impact of pre-training and initialization in federated learning.

Nguyen, T. D., Nguyen, T., Nguyen, P. L., Pham, H. H., Doan, K., & Wong, K.-S. (2023). *Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions.* Retrieved from https://arxiv.org/abs/2303.02213

Nishio, T., & Yonetani, R. (2019, May). Client selection for federated learning with heterogeneous resources in mobile edge. In *Icc 2019 - 2019 ieee international conference on communications (icc).* IEEE. Retrieved from http://dx.doi.org/10.1109/ICC.2019.8761315 doi: 10.1109/icc.2019.8761315

Noble, M., Bellet, A., & Dieuleveut, A. (2023). *Differentially private federated learning on heterogeneous data.* Retrieved from https://arxiv.org/abs/2111.09278

Papyan, V., Han, X. Y., & Donoho, D. L. (2020, September). Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, *117*(40), 24652–24663. Retrieved from http://dx.doi.org/10.1073/pnas.2015509117 doi: 10.1073/pnas.2015509117

Patel, V. M., Gopalan, R., Li, R., & Chellappa, R. (2015). Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, *32*(3), 53-69. doi: 10.1109/MSP.2014.2347059

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners..

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, *97 2*, 285-308.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2001–2010).

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., … McMahan, H. B. (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Ren, Y., Guo, S., Bae, W., & Sutherland, D. J. (2023). How to prepare your task head for finetuning. In *The eleventh international conference on learning representations.* Retrieved from https://openreview.net/forum?id=gVOXZproe-e

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536. Retrieved from https://api.semanticscholar.org/CorpusID:205001834

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS*.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on empirical methods in natural language processing*. Retrieved from https://api.semanticscholar.org/CorpusID:990233

Stich, S. U. (2019). Local sgd converges fast and communicates little. In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019.*

Wang, J., & Joshi, G. (2018). Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farhad, F., ... Poor, H. V. (2019). *Federated learning with differential privacy: Algorithms and performance analysis.* Retrieved from https://arxiv.org/abs/1911.00222

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 1–40.

Williams, A., Nangia, N., & Bowman, S. R. (2018). *A broad-coverage challenge corpus for sentence understanding through inference.* Retrieved from https://arxiv.org/abs/1704.05426

Xie, C., Koyejo, S., & Gupta, I. (2020). *Asynchronous federated optimization.* Retrieved from https://arxiv.org/abs/1903.03934

Yazdanpanah, M., Rahman, A. A., Chaudhary, M., Desrosiers, C., Havaei, M., Belilovsky, E., & Kahou, S. E. (2022, June). Revisiting learnable affines for batch norm in few-shot transfer learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 9109-9118).

Yu, H., Yang, S., & Zhu, S. (2019). Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 5693–5700).

Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Nips.*

Zheng, S., Meng, Q., Wang, T., Chen, W., Yu, N., Ma, Z.-M., & Liu, T.-Y. (2020). *Asynchronous stochastic gradient descent with delay compensation.* Retrieved from https://arxiv.org/abs/1609.08326

Zhu, Z., Hong, J., & Zhou, J. (2021). *Data-free knowledge distillation for heterogeneous federated learning.* Retrieved from https://arxiv.org/abs/2105.10056

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., . . . He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Appendix A

Appendix

Appendix

A.1 Calculation of Communication and Computation Values in Tab. 3.2

A.1.1 Communication

For HeadTuning methods such as LP and FedNCM, communication is calculated according to equation 8 where LL_{in} and LL_{out} are the input and output dimensions of the linear layer, R is the number of federated rounds, K is the total number of clients and F is the fraction $F \in (0, 1]$ of clients participating in each federated round:

$$comm_{lp} = 2 * (LL_{out} * LL_{in}) * K * R * F * (32\ bit) \tag{8}$$

For methods training the complete model, communication is calculated according to equation 9 where P is the number of parameters present in the base model (excluding the linear layer) and LL_{in}, LL_{out}, R, K and F have the same meanings as above:

$$comm_{ft} = 2((LL_{out} * LL_{in}) + P) * K * R * F * (32\ bit) \tag{9}$$

A.1.2 Computation

For computation, let F be one forward pass of a single sample, Let S be the subset of clients, K, selected to participate in each federated round, R. We define one backward pass as $2F$ therefore each sample contributes $3F$ per round in a full fine-tuning situation. For HeadTuning a complete backward pass is not necessary since we only update the linear layer weights and each sample contributes F per round. Eq. 10 and Eq. 11 formalize our computation calculation method in units of F, where E is the number of local epochs performed by each client and N_s is the sum of the number of samples at each selected client, S.

$$comp_{ft} = 3 * R * E * N_s \tag{10}$$

$$comp_{ft} = R * E * N_s \tag{11}$$

A.2 Hyperparameter Settings

For CIFAR-10, CUB, Stanford Cars and Eurosat datasets the learning rates for the FedAvg algorithm were tuned via a grid search over learning rates $\{0.1, 0.07, 0.05, 0.03, 0.01, 0.007, 0.005, 0.003, 0.001\}$. For Flowers102, based on preliminary analysis we used lower learning rates were tuned over learning rates
$\{0.01, 0.007, 0.005, 0.003, 0.001, 0.0007, 0.0005, 0.0003, 0.0001\}$. Tab. A.1 summarizes the number of rounds conducted for each dataset and method combination.

Prior work on federated learning with pre-trained models has indicated that for FedADAM lower global learning rates and higher client learning rates were more effective. As a result for CIFAR-10 and Flowers the client learning rate was tuned over $\{1, 0.1, 0.01, 0.001, 0.0001\}$ and the server learning rate was tuned over $\{0.001, 0.0001, 0.00001, 0.000001\}$, each combination of server and client learning rates were tried.

Dataset	FT+FedNCM	FT	LP	Random
CIFAR-10	200	200	200	3000
CIFAR-10 32 × 32J. NGUYEN ET AL. (2023)	1000	500	1000	1000
FLOWERS-102	250	250	500	3000
STANFORD CARS	1000	1000	2500	3500
CUB	1000	1000	1500	3000
EUROSAT-SUB	1000	1000	1000	3500

Table A.1: rounds conducted for each dataset and method combination. For CIFAR-10 32 × 32 experiments were conducted in J. Nguyen et al. (2023) with the exception of FedNCM+FT (our method)

A.3 Additional HeadTuning Analysis

A.3.1 MLP HeadTuning

We compare HeadTuning methods by replacing the linear layer in the LP method with a three layer MLP. Tab. A.2 shows the HeadTuning results obtained using LP and MLP. We observe no advantage to using an MLP instead on a linear layer since most of our results are comparable or in some cases even inferior to the LP case.

A.3.2 LP and FT

In this section we explore the use of LP as the HeadTuning method in our two-stage algorithm. Tab. A.3 indicates performance of our two stage HeadTuning + fine-tuning method using five rounds of LP, ten rounds of LP and FedNCM as the HeadTuning methods. The bottom row in each dataset category where stage 1 Methods is denoted as n/a, is the result of only fine-tuning, *i.e.* not using our two stage method, which shows that HeadTuning prior to fine-tuning is always at least as effective as FT alone and, depending on the HeadTuning method selected, much more effective.

Dataset	HeadTuning Method	Accuracy
CIFAR-10	LP	82.5 ± 0.2
	MLP	84.1 ± 0.2
FLOWERS102	LP	74.1 ± 1.2
	MLP	72.5 ± 0.3
CUB	LP	50.0 ± 0.3
	MLP	50.3 ± 0.3
CARS	LP	41.2 ± 0.5
	MLP	41.0 ± 0.4
EUROSAT-SUB	LP	92.6 ± 0.4
	MLP	91.7 ± 0.8

Table A.2: LP and MLP HeadTuning results prior to the fine-tuning stage.

A.4 Experiments Using Small Subsets of Data

We conduct experiments using only a small subset of Cifar-10 with nine samples of each class, *i.e.* 90 samples in total. We distribute these samples between five clients using a Dirichlet distribution with $\alpha = 0.1$ so each client will have very few samples and some will be entirely missing samples from many classes. Tab. A.4 shows a $\geq 10\%$ improvement in accuracy between FT and FedNCM, which we attribute to overfitting of the FT model and the challenge of heterogeneity that FedNCM is not susceptible to. We also observe that LP does quite a bit better than FT indicating the benefit of HeadTuning methods in this setting.

A.5 ResNet Experiments

We perform experiments for LP, FT, FedNCM and FedNCM+FT using the ResNet18 model using the FedAvg algorithm and the CIFAR-10 and Flowers102 datasets. Results are summarized in Tab. A.5 where we observe that FedNCM performance is better by almost 13% compared to squeezenet, while FT performance is degraded compared to squeezenet. We hypothesis this is due to the challenges of deeper networks in heterogenous federated learning. For the Flowers102 dataset, FedNCM and FedNCM+FT produce the best results by far. Additionally, for flowers FedNCM

Dataset	stage 1 Method	Accuracy
CIFAR-10	LP (5 ROUNDS)	85.9 ± 0.4
	LP (10 ROUNDS)	84.5 ± 0.17
	FEDNCM	$\mathbf{87.2 \pm 0.2}$
	N/A	85.4 ± 0.4
FLOWERS102	LP (5 ROUNDS)	68.6 ± 1.3
	LP (10 ROUNDS)	68.3 ± 0.6
	FEDNCM	$\mathbf{74.9 \pm 0.2}$
	N/A	64.5 ± 0.1

Table A.3: Outcomes using LP as the HeadTuning method in our two stage training algorithm, FedNCM results from Tab. 3.2 are included for comparison.

LP	FT	FedNCM	FedNCM + FT
53.8 ± 0.3	45.2 ± 2.4	55.6 ± 0.8	56.7 ± 0.7

Table A.4: ResNet18 model performance for FedAvg. As with Squeezenet, FedNCM+FT continues to outperforms in all cases.

outperformed all other methods. The variance between runs using ResNet18 is much higher than was observed for SqueezeNet, FedNCM appears to help stabilize the results since it provides the most consistency by for both datasets.

A.6 Extended Accuracy Comparison Figures

Fig. A.1 is the extended version of Fig. 1 in the main body of the paper. In the paper we truncate the number of round displayed for the random setting since random requires many more rounds to converge than the other methods. Fig. A.1 shows these same figures with the entirety of the training rounds displayed for the random setting.

Dataset	FedNCM	FedNCM + FT	FT+Pretrain	LP+Pretrain
CIFAR-10	77.74 ± 0.05	79.05 ± 1.31	77.87 ± 4.07	74.73 ± 3.03
FLOWERS102	74.13 ± 0.31	74.1 ± 0.26	34.41 ± 10.16	25.35 ± 2.59

Table A.5: ResNet18 model performance for FedAvg. As with Squeezenet, FedNCM+FT continues to outperforms in all cases.

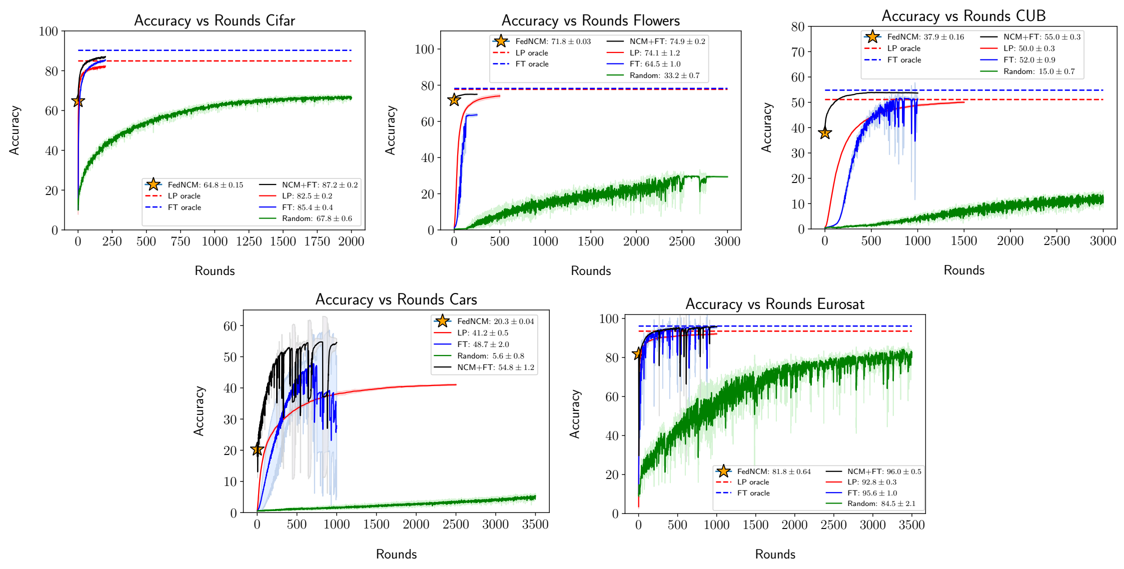

Figure A.1: The full training of Random baseline corresponding to Fig. 1 in the paper is shown. We observer Random is always very far from the other baseliens and converges slowly.

A.7 Compute

We use a combination of NVIDIA A100-SXM4-40GB, NVIDIA RTX A4500, Tesla V100-SXM2-32GB and Tesla P100-PCIE-12GB GPUs for a total of 1.1 GPU years . In addition to the experiments reported in the paper, this includes preliminary experiments and hyperparameter searches.