FedFisher: Leveraging Fisher Information for One-Shot Federated Learning

Divyansh Jhunjhunwala
Carnegie Mellon University

Shiqiang Wang
IBM Research

Gauri Joshi
Carnegie Mellon University

Abstract

Standard federated learning (FL) algorithms typically require multiple rounds of communication between the server and the clients, which has several drawbacks, including requiring constant network connectivity, repeated investment of computational resources, and susceptibility to privacy attacks. One-Shot FL is a new paradigm that aims to address this challenge by enabling the server to train a global model in a single round of communication. In this work, we present `FedFisher`, a novel algorithm for one-shot FL that makes use of Fisher information matrices computed on local client models, motivated by a Bayesian perspective of FL. First, we theoretically analyze `FedFisher` for two-layer over-parameterized ReLU neural networks and show that the error of our one-shot `FedFisher` global model becomes vanishingly small as the width of the neural networks and amount of local training at clients increases. Next, we propose practical variants of `FedFisher` using the diagonal Fisher and K-FAC approximation for the full Fisher and highlight their communication and compute efficiency for FL. Finally, we conduct extensive experiments on various datasets, which show that these variants of `FedFisher` consistently improve over competing baselines.

1 Introduction

Data collection and storage are becoming increasingly decentralized, both due to the proliferation of smart devices and privacy concerns stemming from transferring and storing data at a centralized location. Federated Learning (FL) is a framework designed to learn the parameters $\boldsymbol{W} \in \mathbb{R}^d$ of a model $f(\boldsymbol{W}, \cdot)$ on decentralized data distributed across a network of clients under the supervision of a central server (Kairouz et al., 2021; Li et al., 2020a; Yang et al., 2019). Formally, we formulate FL as the following distributed optimization problem:

$$\min_{\boldsymbol{W} \in \mathbb{R}^d} \left\{ L(\boldsymbol{W}) := \frac{1}{M} \sum_{i=1}^{M} L_i(\boldsymbol{W}) \right\} \text{ where}$$
$$L_i(\boldsymbol{W}) = \frac{1}{n} \sum_{(\boldsymbol{x}_{ij}, \boldsymbol{y}_{ij}) \in \mathcal{D}_i} \ell(f(\boldsymbol{W}, \boldsymbol{x}_{ij}), \boldsymbol{y}_{ij}). \quad (1)$$

Here, M is the number of clients, $\mathcal{D}_i$ is the i-th client's local dataset consisting of input-label pairs $\{(\boldsymbol{x}_{ij}, \boldsymbol{y}_{ij})\}_{j=1}^{n}$ where $\boldsymbol{x} \in \mathbb{R}^p$ is the input and $\boldsymbol{y} \in \mathbb{R}^C$ is the label, and $n = |\mathcal{D}_i|$ is the dataset size. For simplicity, we consider the case where clients have equal amounts of data; our algorithm and analysis can easily be extended to the case where client objectives are unequally weighted based on their local dataset sizes. The loss function $\ell(\cdot, \cdot)$ penalizes the difference between the prediction of the model $f(\boldsymbol{W}, \boldsymbol{x})$ and the true label $\boldsymbol{y}$. We use $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{M}$ to denote the collection of data across all clients and $N = Mn$ to denote the total number of data samples across clients.

The fundamental challenge with learning in federated settings is data heterogeneity across clients (Karimireddy et al., 2020; Li et al., 2020a). In particular, while clients can independently train a model that fits their local data, it is unclear how the server can combine these local models to get a global model that works for all the clients. To tackle this problem, standard FL algorithms such as `FedAvg` (McMahan et al., 2017) and `FedProx` (Li et al., 2020b) resort to an iterative approach requiring multiple rounds of communication between clients and the server in order for the local models to reach a consensus. However, this multi-round approach has several drawbacks. First, clients need to frequently connect with the server to send and receive updates. Second, a client needs to invest computational resources repeatedly to update the global model every

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

time it participates in training. Third, it increases susceptibility to attacks such as data and model poisoning since attackers can continuously modify and update their attacks based on the global model they receive from the server in each round.

To overcome these drawbacks, a recent line of work has focused on the paradigm of *one-shot FL* which aims to learn the parameters of the global model in a *single* round of communication between clients and the server. Existing work for one-shot FL can be broadly split into two categories: (i) knowledge distillation methods and (ii) neuron matching methods. Knowledge distillation methods treat the collection of client models as an ensemble and propose to distill the knowledge from this ensemble into a single global model. To perform the distillation step, some works assume that the server has access to an auxiliary public dataset (Gong et al., 2021; Li et al., 2021; Lin et al., 2020), which clearly simplifies the problem. Another set of works proposes training generative models such as GANs (Zhang et al., 2022; Zhu et al., 2021) or variational autoencoders (Heinbaugh et al., 2023) to artificially generate data using local models of the clients. This again raises privacy questions, since a sufficiently powerful generative model can simply reconstruct client data (Hitaj et al., 2017). Moreover, both the data generation and distillation steps impose a significant computation cost on the server and require careful hyperparameter tuning (Kurach et al., 2019), which is itself a challenge to implement in one-shot FL.

Neuron matching methods are based on the observation that neural networks (NNs) are permutation invariant, i.e., it is possible to create NNs that differ only in the ordering of weights while having the same output (Ainsworth et al., 2023; Akash et al., 2022; Entezari et al., 2022; Jordan et al., 2023; Liu et al., 2022; Singh and Jaggi, 2020). Based on this observation, these works propose to first align the weights of the client models according to a common ordering (called matching) and then average the aligned client models. Although this idea has been shown to work well when combining simple models like feedforward NNs, the performance drops considerably for more complex models such as CNNs (Wang et al., 2020a). Another line of work, termed as *model fusion* has looked at fusing the capabilities of multiple existing models into a single model (Choshen et al., 2022; Ilharco et al., 2023; Jin et al., 2023; Matena and Raffel, 2022; Yadav et al., 2023). While not explicitly designed for one-shot FL, some of these techniques, such as Jin et al. (2023); Matena and Raffel (2022); Yadav et al. (2023) can be considered as an improvement over vanilla averaging. However, we note that these methods focus mostly on fusing pre-trained models and do not consider the effect of data heterogeneity used in training the models. We defer a more extensive discussion of such methods and other related work to Appendix A. Finally, we note that none of these existing works provide any theoretical guarantees for their proposed methods.

Thus, motivated by the limitations of multi-round FL and current approaches for one-shot FL, we ask the following questions: *Can we devise a one-shot FL method that is simultaneously communication and computation efficient (at both the server and clients), privacy-preserving, and has good practical performance? Furthermore, can we provide theoretical guarantees for such a method?*

Our Contributions. In this work, we take a step towards providing an affirmative answer to both of the questions formulated above. To do so, we use the idea that Equation (1) can alternatively be reformulated as a *posterior inference* problem, specifically finding the mode of a global posterior over the model parameters (Al-Shedivat et al., 2021; Guo et al., 2023). While Al-Shedivat et al. (2021) and Guo et al. (2023) use this idea to propose a multi-round algorithm, our contribution lies in showing how this reformulation can yield a novel one-shot FL algorithm, which we term as `FedFisher`. Some highlights of our contribution are as follows.

- We propose `FedFisher` and show how the problem of finding the mode of a global posterior can be solved in a one-shot manner, using the local models at clients and some approximation of the Fisher information matrices computed at these local models (Section 2).
- We theoretically analyze `FedFisher` for the case of over-parameterized two-layer neural networks. In particular, we show that when we utilize the full Fisher information in `FedFisher`, the error of our one-shot global model becomes vanishingly small as the width of the neural networks and the amount of local training at the clients increase (Section 3).
- We propose practical variants of `FedFisher` using the diagonal and K-FAC approximation, which we term as `FedFisher(Diag)` and `FedFisher(K-FAC)`, respectively, and highlight the communication and compute efficiency of these variants along with their compatibility with secure aggregation (Section 4).
- We evaluate `FedFisher(Diag)` and `FedFisher(K-FAC)` on a range of one-shot FL tasks using deep neural networks and show that they give a consistent $5-10\%$ improvement in global model accuracy compared to competing

one-shot baselines involving knowledge distillation, neuron matching, or model fusion (Section 5).

2 Proposed Algorithm: FedFisher

To begin our discussion, we first state the following standard assumption on the loss function $\ell(\cdot,\cdot)$, which is true for most common loss functions such as the squared loss and the cross-entropy loss.

Assumption 1. *Given $\boldsymbol{z} = f(\boldsymbol{W},\boldsymbol{x})$, we assume that $\ell(\boldsymbol{z},\boldsymbol{y})$ is proportional to the negative log likelihood of $\boldsymbol{y}$ under some exponential-family probabilistic model, i.e., $\ell(\boldsymbol{z},\boldsymbol{y}) \propto -\log \mathbb{P}(\boldsymbol{y}|\boldsymbol{z})$ where $\mathbb{P}(\boldsymbol{y}|\boldsymbol{z}) = h(\boldsymbol{y})\exp(\boldsymbol{z}^\top T(\boldsymbol{y}) - A(\boldsymbol{z}))$ and $h(\boldsymbol{y}), T(\boldsymbol{y}), A(\boldsymbol{z})$ are some real-valued functions.*

Let us define the likelihood for a data point $(\boldsymbol{x},\boldsymbol{y})$ for a given $\boldsymbol{W}$ as $\mathbb{P}((\boldsymbol{x},\boldsymbol{y})|\boldsymbol{W}) = \mathbb{P}(\boldsymbol{x})\mathbb{P}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{W}) \propto q(\boldsymbol{x})\exp(-\ell(f(\boldsymbol{W},\boldsymbol{x}_{ij}),\boldsymbol{y}))$, where $q(\cdot)$ is some prior on $\boldsymbol{x}$, independent of $\boldsymbol{W}$. Given this definition, we can adopt a Bayesian viewpoint and try to find the *maximum a posteriori probability* (MAP) estimate, i.e., find $\boldsymbol{W}$ where the *posterior* likelihood $\mathbb{P}(\boldsymbol{W}|\mathcal{D}) \propto \mathbb{P}(\mathcal{D}|\boldsymbol{W})\mathbb{P}(\boldsymbol{W})$ is maximized with $\mathbb{P}(\boldsymbol{W})$ being some prior belief over $\boldsymbol{W}$. Our motivation to do so comes from the following proposition.

Proposition 1. *(Global Posterior Decomposition (Al-Shedivat et al., 2021)) Under the flat prior $\mathbb{P}(\boldsymbol{W}) \propto 1$, the global posterior decomposes into a product of local posteriors, i.e., $\mathbb{P}(\boldsymbol{W}|\mathcal{D}) \propto \prod_{i=1}^{M} \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)$. Furthermore, the modes of the global posterior coincide with the optima of the FL objective in Equation (1), i.e, $\arg\max_{\boldsymbol{W}\in\mathbb{R}^d} \mathbb{P}(\boldsymbol{W}|\mathcal{D}) = \arg\min_{\boldsymbol{W}\in\mathbb{R}^d} L(\boldsymbol{W})$.*

Proposition 1 tells us that as long as clients compute and send their local posteriors $\mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)$ to the server, no further server-client communication is needed to find the global MAP estimate or equivalently a minimizer to our FL objective, giving us a one-shot inference procedure. However, doing so is challenging since $\mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)$ typically does not have an analytical expression. To get a tractable solution, we propose to use some approximate inference techniques, as discussed below.

Mode of Local Posterior. To apply the approximate inference techniques detailed below, clients first need to compute $\tilde{\boldsymbol{W}}_i \approx \arg\max_{\boldsymbol{W}} \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)$, an estimate of the mode of their local posterior under the flat prior. Note that this corresponds to a minimizer of $L_i(\boldsymbol{W})$ under Assumption 1 and therefore $\tilde{\boldsymbol{W}}_i$ can be obtained using standard gradient-based optimizers.

Laplace Approximation for Local Posterior. Now using a second order Taylor expansion around $\tilde{\boldsymbol{W}}_i$, we can get the following approximation for the log-posterior at the i-th client:

$$\log \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i) \approx \log \mathbb{P}(\tilde{\boldsymbol{W}}_i|\mathcal{D}_i) - \frac{n}{2}(\boldsymbol{W}-\tilde{\boldsymbol{W}}_i)^\top \boldsymbol{H}_i (\boldsymbol{W}-\tilde{\boldsymbol{W}}_i), \quad (2)$$

where we additionally use $\nabla \log \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)|_{\boldsymbol{W}=\tilde{\boldsymbol{W}}_i} \approx 0$. Here $\boldsymbol{H}_i = -\frac{1}{n}\nabla^2 \log \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)|_{\boldsymbol{W}=\tilde{\boldsymbol{W}}_i}$ is the Hessian of the negative log-posterior at client i computed at $\tilde{\boldsymbol{W}}_i$. Thus, we see that the local posterior of a client is now parameterized by $\tilde{\boldsymbol{W}}_i$ and $\boldsymbol{H}_i$.

Approximating Hessian with Fisher. The Fisher information matrix $\boldsymbol{F}_i$ (for brevity, we refer to it as "the Fisher" in the rest of the paper) of the local model $\tilde{\boldsymbol{W}}_i$ at client i is defined as follows:

$$\boldsymbol{F}_i = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{y}}\left[\nabla \log \mathbb{P}(\boldsymbol{y}|\boldsymbol{x}_{ij},\boldsymbol{W}) \nabla \log \mathbb{P}(\boldsymbol{y}|\boldsymbol{x}_{ij},\boldsymbol{W})^\top\right] \quad (3)$$

computed at $\boldsymbol{W} = \tilde{\boldsymbol{W}}_i$. Now, under Assumption 1 and assuming $\tilde{\boldsymbol{W}}_i$ fits the data at client i perfectly, i.e., $f(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij}) = \boldsymbol{y}_{ij}, \forall j \in [n]$, it can be shown that the Hessian at $\tilde{\boldsymbol{W}}_i$ corresponds exactly to the Fisher, i.e., $\boldsymbol{H}_i = \boldsymbol{F}_i$ (Martens, 2020; Singh and Alistarh, 2020). The latter condition usually holds true for modern over-parameterized deep learning models when trained for sufficient epochs. Motivated by this observation, we can further approximate $\boldsymbol{H}_i$ with $\boldsymbol{F}_i$. This approximation is useful since, unlike the Hessian, the Fisher is guaranteed to be positive semi-definite, a condition required for tractable inference at the global server. However, computing (and communicating) the full Fisher would require $\mathcal{O}(d^2)$ bits, which is infeasible when the models are neural networks with d in the order of millions. In practice, clients can replace the full Fisher $\boldsymbol{F}_i$ with another computationally tractable approximation $\tilde{\boldsymbol{F}}_i$ such as the diagonal Fisher or K-FAC (Grosse and Martens, 2016), which preserves the positive semi-definiteness of $\boldsymbol{F}_i$ (see discussion in Section 4). Thus we have

$$\boldsymbol{H}_i \approx \boldsymbol{F}_i \approx \tilde{\boldsymbol{F}}_i. \quad (4)$$

Computing Mode of Global Posterior. Now assuming clients compute and send back $\tilde{\boldsymbol{W}}_i$ and $\tilde{\boldsymbol{F}}_i$ to the server, we can use Proposition 1, Equation (2) and Equation (4) to approximate the logarithm of the global posterior as

$$\log \mathbb{P}(\boldsymbol{W}|\mathcal{D}) \approx \sum_{i=1}^{M} \log \mathbb{P}(\tilde{\boldsymbol{W}}_i|\mathcal{D}_i) - \frac{n}{2}\sum_{i=1}^{M}(\boldsymbol{W}-\tilde{\boldsymbol{W}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{W}-\tilde{\boldsymbol{W}}_i). \quad (5)$$

Algorithm 1 `FedFisher`

Input: initial $\boldsymbol{W}_0$, no. of iterations K, T, client and server step sizes η and η_S respectively
Global server does:
  Communicate $\boldsymbol{W}_0$ to all clients;
Clients $i \in [M]$ **in parallel do:**
  Set $\boldsymbol{W}_i^{(0)} \leftarrow \boldsymbol{W}_0$;
  For $k = 0, \ldots, K-1$ **iterations:**
    $\boldsymbol{W}_i^{(t+1)} \leftarrow \boldsymbol{W}_i^{(t)} - \eta \nabla L_i(\boldsymbol{W}_i^{(t)})$;
  Set $\tilde{\boldsymbol{W}}_i \leftarrow \boldsymbol{W}_i^{(K)}$;
  Compute $\tilde{\boldsymbol{F}}_i$; // Approximation to true Fisher
  Communicate $\tilde{\boldsymbol{W}}_i$ and $\tilde{\boldsymbol{F}}_i$ to the server;
Global server does:
  Set $\boldsymbol{W}^{(0)} = \sum_{i=1}^M \tilde{\boldsymbol{W}}_i / M$;
  For $t = 0, \ldots, T-1$ **iterations:**
    $\boldsymbol{W}^{(t+1)} \leftarrow \boldsymbol{W}^{(t)} - \eta_S \sum_{i=1}^M \left(\tilde{\boldsymbol{F}}_i \boldsymbol{W}^{(t)} - \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i \right)$;

With this approximation, finding a mode of the global posterior can be written as the following optimization problem:

$$\min_{\boldsymbol{W} \in \mathbb{R}^d} \sum_{i=1}^M (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i). \tag{6}$$

Since each $\tilde{\boldsymbol{F}}_i$ is positive semi-definite, a global minimizer of Equation (6) can be found by simply setting the derivative of the objective to zero. Doing so, we have the following proposition.

Proposition 2. *Any $\boldsymbol{W}$ that satisfies $(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i)\boldsymbol{W} = \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i$ is a minimizer of the objective $\sum_{i=1}^M (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)$.*

For over-parameterized models, i.e, $d \gg N$, the rank of $\sum_{i=1}^M \tilde{\boldsymbol{F}}_i$ will be smaller than d and therefore the system of equations $(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i)\boldsymbol{W} = \sum_{i=1}^M \boldsymbol{F}_i \tilde{\boldsymbol{W}}_i$ will not have a unique solution. To resolve this, we propose to use the solution that minimizes the sum of distances from the local models $\tilde{\boldsymbol{W}}_i$ of each client, i.e., minimizing $\sum_{i=1}^M \left\| \boldsymbol{W} - \tilde{\boldsymbol{W}}_i \right\|_2^2$. Such a constraint ensures fairness of the `FedFisher` objective towards all clients and prevents the solution from drifting too far away from the local models of each client. We express this mathematically as follows.

`FedFisher` objective:

$$\min_{\boldsymbol{W} \in \mathbb{R}^d} \left\{ \tilde{L}(\boldsymbol{W}) = \sum_{i=1}^M \left\| \boldsymbol{W} - \tilde{\boldsymbol{W}}_i \right\|_2^2, \right. \\ \left. \text{such that } \left(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i \right) \boldsymbol{W} = \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i \right\}. \tag{7}$$

We refer to the minimizer $\boldsymbol{W}^*$ of Equation (7) as the `FedFisher` global model in the rest of our discussion. We now show that $\boldsymbol{W}^*$ can be easily computed using gradient descent (GD) with proper initialization and learning rate conditions via the following lemma.

Lemma 1. *Let $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots$ be the iterates generated by running the following gradient descent (GD) procedure: $\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta_S \sum_{i=1}^M \left(\tilde{\boldsymbol{F}}_i \boldsymbol{W}^{(t)} - \tilde{\boldsymbol{F}}_i \boldsymbol{W}_i \right)$ with $\boldsymbol{W}^{(0)} = \sum_{i=1}^M \tilde{\boldsymbol{W}}_i / M$ and $\eta_S \leq 1/\lambda_{\max}$ where $\lambda_{\max}$ is the maximum eigenvalue of $\sum_{i=1}^M \tilde{\boldsymbol{F}}_i$. Then, $\lim_{T \to \infty} \boldsymbol{W}^{(T)} = \boldsymbol{W}^*$.*

Proofs for all lemmas and theorems in this paper can be found in Appendix B. This concludes our discussion of the proposed algorithm `FedFisher`. An algorithm outline for `FedFisher` can be found in Algorithm 1, with practical implementation considerations discussed in Section 4. Next, we analyze the performance of `FedFisher` for over-parameterized two-layer networks.

3 Theoretical Analysis for Two-layer Over-parameterized Neural Network

In this section, we characterize the performance of the `FedFisher` global model $\boldsymbol{W}^*$, on the true global objective $L(\boldsymbol{W})$ when $f(\boldsymbol{W}, \cdot)$ is a two-layer over-parameterized neural network.

Sources of Error for `FedFisher`. We see that the mismatch between `FedFisher` objective $\tilde{L}(\boldsymbol{W})$ in Equation (7) and $L(\boldsymbol{W})$ can be characterized into the following sources of error: **(i)** the suboptimality of $\tilde{\boldsymbol{W}}_i$, i.e, $\nabla L_i(\tilde{\boldsymbol{W}}_i) \neq \boldsymbol{0}$, **(ii)** the Laplace approximation, i.e., neglecting higher order terms in the Taylor expansion in Equation (2), and **(iii)** the error introduced by approximating the true Fisher $\boldsymbol{F}_i$ with $\tilde{\boldsymbol{F}}_i$ in Equation (4). The error introduced by **(iii)** is orthogonal to **(i)** and **(ii)** and depends on the specific approximation used. Moreover, commonly used approximations, such as the diagonal and K-FAC, are primarily empirically motivated and have limited theoretical guarantees. Therefore, to simplify our analysis, we assume that clients send their full Fisher and focus on bounding the error due to the first two sources.

Novelty of Analysis. While the error due to **(i)** can be thought of as a function of the number of local training steps K, it is less clear how the error due to **(ii)** can be bounded. In this analysis, we show that for over-parameterized two-layer ReLU NNs, the error due to **(ii)** can be controlled using the *width* of the network. Intuitively, the quality of the approximation in Equation (2) depends on the distance between the weights of the `FedFisher` global model $\boldsymbol{W}^*$ and clients' local models $\tilde{\boldsymbol{W}}_i$; the larger the distance the greater

will be the effect of the higher-order terms and hence worse the approximation. Our contribution lies in showing that for a sufficiently wide model, this distance decreases as $\mathcal{O}(1/m)$ where m is the width of the model. Doing so requires a careful analysis of the trajectory of iterates generated by the optimization procedure in Lemma 1 and novel bounds on the smallest and largest eigenvalues of the aggregated Fisher $\sum_{i=1}^{M} \boldsymbol{F}_i/M$ (see proofs in Appendix B). We believe that our analysis can also be extended to deeper NNs as future work.

We begin by modeling a two-layer ReLU NN as follows:

$$f(\boldsymbol{W}, \boldsymbol{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\boldsymbol{x}^\top \boldsymbol{w}_r). \tag{8}$$

Here, m is the number of hidden nodes in the first layer, $\{\boldsymbol{w}_r\}_{r=1}^{M}$ are the weights of the first layer, $\{a_r\}_{r=1}^{m}$ are the weights of the second layer, and $\sigma(x) = \max\{x, 0\}$ is the ReLU function. We use $\boldsymbol{W} = \text{vec}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_m)$ to parameterize the weights of the first layer. Similar to Du et al. (2019), we consider a_r's to be fixed beforehand (initialized to be $+1$ or -1 uniformly at random) and only consider the case where $\boldsymbol{w}_r$'s are trained. We also consider the squared loss function $\ell(z, y) = \frac{1}{2}(y - z)^2$.

Recall that $\mathcal{D}$ is the collection of data across all the clients. Define $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^{N} = \{(\boldsymbol{x}_{11}, y_{11}), (\boldsymbol{x}_{12}, y_{12}), \ldots, (\boldsymbol{x}_{Mn}, y_{Mn})\}$ to be an ordering of the data in $\mathcal{D}$. We state some definitions and assumptions that will be needed in our analysis.

Definition 1. *(Minimum eigenvalue of Gram Matrix (Du et al., 2019)) Define the matrix* $\boldsymbol{H}^\infty \in \mathbb{R}^{N \times N}$ *as* $\boldsymbol{H}^\infty_{k,l} = \mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \left[\boldsymbol{x}_k^\top \boldsymbol{x}_l \mathbb{I}\left\{ \boldsymbol{w}^\top \boldsymbol{x}_k \geq 0 \right\} \mathbb{I}\left\{ \boldsymbol{w}^\top \boldsymbol{x}_l \geq 0 \right\} \right]$ *and* $\lambda_0 = \lambda_{\min}(\boldsymbol{H}^\infty)$.

Assumption 2. *(Data normalization) For any* $(\boldsymbol{x}, y) \in \mathcal{D}$, *we have* $\|\boldsymbol{x}\|_2 = 1$ *and* $|y| \leq C$, *where* C *is some positive constant.*

Assumption 3. *(Data uniqueness) For any* $(\boldsymbol{x}, y), (\boldsymbol{x}', y') \in \mathcal{D}$, *we have* $\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \geq \phi$, *where* ϕ *is some strictly positive constant.*

Assumption 4. *(Full Fisher) Clients compute and send their full Fisher, i.e.,* $\tilde{\boldsymbol{F}}_i = \boldsymbol{F}_i$.

Note on Definition and Assumptions. The rate of convergence of GD for two-layer ReLU neural networks depends closely on $\boldsymbol{H}^\infty$ and λ_0 as shown in Du et al. (2019). Assumption 2 and Assumption 3 are standard in NN optimization literature (Allen-Zhu et al., 2019a,b; Du et al., 2019; Zou and Gu, 2019) and are needed to ensure bounded loss at initialization and $\lambda_0 > 0$, respectively. Assumption 4 is specific to our setting, as discussed at the start of this section.

Theorem 1. *Under Assumptions 2, 3, 4, for* $m \geq \text{poly}(N, \lambda_0^{-1}, \delta^{-1}, \kappa^{-1})$, *and i.i.d Gaussian initialization weights of* $\boldsymbol{W}_0$ *as* $\boldsymbol{w}_{0,r} \sim \mathcal{N}(\boldsymbol{0}, \kappa)$, *and initializing the second layer weights* $a_r = \{-1, 1\}$ *with probability* $1/2$ *for all* $r \in [m]$, *for step sizes* $\eta = \mathcal{O}(\lambda_0/N^2)$, $\eta_S = \mathcal{O}(\lambda_0/N^2)$ *and for a given failure probability* $\delta \in (0, 1)$, *the following is true with probability* $1 - \delta$ *over the random initialization:*

$$L(\boldsymbol{W}^*) \leq \underbrace{\mathcal{O}\left((1 - \eta\lambda_0/2)^K \frac{N}{\delta} \right)}_{\text{local optimization error}} + \underbrace{\mathcal{O}\left((2 - (1 - \eta\lambda_0/2)^K) \frac{\text{poly}(N, \kappa^{-1}, \lambda_0^{-1}, \delta^{-1})}{m} \right)}_{\text{Laplace approximation error}}. \tag{9}$$

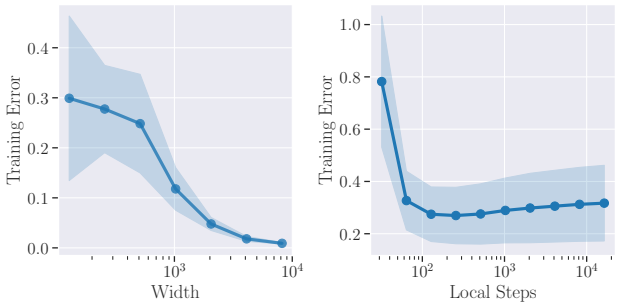
Corollary 1. *Under the conditions of Theorem 1, if we set* $m = \mathcal{O}\left(\frac{\text{poly}(N, \kappa^{-1}, \lambda_0^{-1}, \delta^{-1})}{\epsilon} \right)$ *and* $K = \mathcal{O}\left(\frac{1}{\eta\lambda_0} \log(N/\delta\epsilon) \right)$, *then for failure probability* $\delta \in (0, 1)$ *and target error* $\epsilon \in (0, 1)$, *the global loss* $L(\boldsymbol{W}^*) \leq \epsilon$ *with probability* $1 - \delta$.

Takeaways. Theorem 1 shows that for sufficiently wide networks, $L(\boldsymbol{W}^*)$ can be decomposed into two distinct sources of error. The first term is the local optimization error, which measures how well $\tilde{\boldsymbol{W}}_i$ fits the data at the i-th client, and the second term corresponds to the error introduced by the Laplace approximation in Equation (2). While the overall error always decreases as the width m increases, there is a trade-off in the number of local optimization steps K. In particular, a larger K reduces the local optimization error but increases the Laplace error as each local model $\tilde{\boldsymbol{W}}_i$ drifts further away from $\boldsymbol{W}^*$. Corollary 1 shows that by setting m and K large enough, the total training error can be made vanishingly small in just one round.

Effect of Data Heterogeneity. Note that our result does not make any explicit assumptions on the heterogeneity of data across clients such as bounded dissimilarity (Karimireddy et al., 2020; Wang et al., 2020b). We find that the notion of heterogeneity that implicitly appears in our analysis is the distance between the weights of any two local models, i.e, $\|\tilde{\boldsymbol{w}}_{i,r} - \tilde{\boldsymbol{w}}_{j,r}\|_2$ for any $i \neq j$. For two-layer neural networks this quantity can be bounded as follows: $\|\tilde{\boldsymbol{w}}_{i,r} - \tilde{\boldsymbol{w}}_{j,r}\|_2 \leq \|\tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\|_2 + \|\tilde{\boldsymbol{w}}_{j,r} - \tilde{\boldsymbol{w}}_{0,r}\|_2 \leq \mathcal{O}(1/\sqrt{m})$ (see first part of Lemma 6 in Appendix B), where $\boldsymbol{w}_{0,r}$ are the weights of the common initialization point across clients. This bound is agnostic to the correlation of data across clients and shows that $\|\tilde{\boldsymbol{w}}_{i,r} - \tilde{\boldsymbol{w}}_{j,r}\|_2$ always decreases as the width of the model increases. Improving this bound by incorporating the effects of data correlation can be an interesting direction for future work.

Empirical Validation. We conduct experiments on a synthetic dataset with $M = 2$ clients, $p = 2$ dimensions,

and $n = 100$ samples to verify our theoretical findings. This setting allows us to ensure that the conditions in Theorem 1, such as computing the full Fisher, are computationally tractable. The synthetic data is generated following a similar procedure as Li et al. (2020b). Further details of the experimental setup can be found in Appendix D. In Figure 1a we fix the number of local steps $K = 2048$ and vary the width of the model. As predicted by our theory, we see that the training error for `FedFisher` monotonically decreases as we increase the width of the model. In Figure 1b we fix the width of the model to $m = 512$ and vary the number of local steps. Here we see a trade-off in the effect of local steps – the error initially decreases as clients do more local steps and then rises again, as predicted by our theory.



(a) (b)

Figure 1: Empirical validation of Theorem 1 in a synthetic setting. For a fixed number of local steps (Figure 1a), the error for `FedFisher` decreases as the width of the model increases. For a fixed width (Figure 1b), the error first decreases and then increases as local steps increases.

Generalization Bounds. Motivated by Arora et al. (2019), we can also provide generalization bounds on the performance of the `FedFisher` global model by bounding $\|\boldsymbol{W}^* - \boldsymbol{W}_0\|_2$, which is the *total distance* between the weights of $\boldsymbol{W}^*$ and the initialization point $\boldsymbol{W}_0$. We provide these bounds and subsequent discussion in Appendix B.

4 A Practical Implementation of FedFisher

Two of the most popular approximations for the Fisher are the diagonal Fisher and the Kronecker Factored Approximate Curvature (K-FAC). The diagonal Fisher simply approximates the full Fisher with its diagonal elements and has been used in previous work for applications such as network pruning (LeCun et al., 1989) and transfer learning (Kirkpatrick et al., 2017). The K-FAC introduced by Grosse and Martens (2016); Martens and Grosse (2015) uses two approximations: (i) it assumes that the Fisher is block diagonal where each block corresponds to the Fisher of a particular layer, and (ii) it approximates the Fisher of each block as the Kronecker product of two smaller matrices, i.e, $\tilde{\boldsymbol{F}} \approx \boldsymbol{A} \otimes \boldsymbol{B}$. Thus, storing (and communicating) the K-FAC only takes $\mathcal{O}\left(\dim(\boldsymbol{A}) + \dim(\boldsymbol{B})\right)$ bits instead of $\mathcal{O}\left(\dim(\boldsymbol{A}) \cdot \dim(\boldsymbol{B})\right)$ bits. Using these approximations as a substitute for the true Fisher in Equation (4), we get two practical variants of `FedFisher`, which we term as `FedFisher(Diag)` and `FedFisher(K-FAC)`, respectively. We highlight the computation and communication efficiency of these variations along with their compatibility with secure aggregation below.

Computation Efficiency. To compute their diagonal Fisher or K-FAC, clients need to perform an additional forward-backward pass over the data, i.e., an additional epoch of training, plus some small overhead cost. This is a reasonable cost for FL setups, since most of the computation cost goes into computing $\tilde{\boldsymbol{W}}_i$, which requires multiple local epochs. This is demonstrated in Table 1, where we see that `FedFisher(Diag)` and `FedFisher(K-FAC)` add less than 14.5% of the total computational cost of `FedAvg` at clients for our FL setups described in Section 5.

Table 1: Computation time in seconds averaged across $M = 5$ clients for FL setup described in Section 5.

Method	FashionMNIST		CIFAR10	
`FedAvg`	14.5	–	20.9	–
`FedFisher(K-FAC)`	16.0	+10.3%	23.1	+10.5%
`FedFisher(Diag)`	16.6	+14.4%	23.7	+13.4%

Communication Efficiency. The communication cost of `FedFisher` depends on the number of parameters in the approximate Fisher. For the diagonal Fisher this is exactly d, while for the K-FAC it is model specific, but usually less than $10d$. We find that the total communication cost of both `FedFisher(Diag)` and `FedFisher(K-FAC)` (including local models) can be easily reduced to match that of `FedAvg` using a mix of standard compression techniques such as quantization and low-rank decomposition without significantly affecting the accuracy of the final one-shot model. We discuss more details about this compression in Appendix C. For all our experiments in Section 5, we ensure that the communication cost of `FedFisher(Diag)` and `FedFisher(K-FAC)` matches that of `FedAvg`.

Compatibility with Secure Aggregation. In theory, the server optimization in `FedFisher` just needs the aggregate quantities $\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i$, $\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i$ and $\sum_{i=1}^{M} \tilde{\boldsymbol{W}}_i$ (see Algorithm 1). For the diagonal Fisher, these quantities can be computed using secure aggregation (Bonawitz et al., 2016; Kadhe et al., 2020), thus

Table 2: Test accuracy results on different datasets by keeping number of client $M = 5$ fixed and varying heterogeneity parameter α (smaller α is more heterogeneous). `FedFisher` variants consistently outperform other baselines.

Dataset	α	FedAvg	OTFusion	PFNM	RegMean	DENSE	Fisher Merge	FedFisher (Diag)	FedFisher (K-FAC)
FashionMNIST	0.2	59.11±3.82	58.40±4.95	63.50±3.07	71.09±2.19	72.69±1.99	61.81±3.98	65.44±3.04	**76.28**±2.56
	0.1	41.72±4.54	43.77±2.03	59.27±3.19	56.98±4.08	50.11±4.99	54.71±4.96	55.04±4.15	**68.36**±3.44
	0.05	36.02±2.77	37.53±1.20	45.69±5.32	50.40±2.69	46.62±3.21	43.03±5.51	45.92±6.13	**53.29**±4.17
SVHN	0.2	63.39±2.27	68.05±0.98	69.11±1.35	74.04±1.20	**77.73**±2.07	65.42±3.72	68.68±3.42	75.92±1.16
	0.1	39.42±2.52	52.34±0.18	53.19±4.18	64.16±3.51	56.31±2.17	64.06±2.86	64.20±3.08	**69.09**±2.81
	0.05	27.03±0.71	37.24±0.88	45.62±3.55	55.83±4.44	49.16±1.63	49.32±2.83	51.48±2.91	**57.41**±3.79
CIFAR10	0.2	41.77±0.79	40.35±1.96	45.75±0.58	43.41±1.56	44.79±1.04	39.90±3.21	44.04±2.10	**51.67**±1.03
	0.1	36.43±2.51	40.45±0.96	39.43±1.80	36.65±1.29	38.65±2.67	36.53±2.74	40.04±1.35	**47.01**±1.87
	0.05	29.75±1.32	30.62±1.20	30.58±3.65	32.84±1.39	32.65±2.35	30.89±1.11	31.08±1.52	**40.02**±3.64
CINIC10	0.2	37.15±0.87	36.88±1.44	39.94±0.60	36.19±1.07	35.64±1.22	37.01±4.07	41.88±2.53	**43.45**±0.54
	0.1	32.43±1.57	34.92±1.68	35.86±1.65	31.54±0.67	30.89±1.44	32.91±2.44	37.68±2.21	**39.16**±1.19
	0.05	27.58±0.76	26.39±0.35	29.04±1.26	28.14±0.79	28.13±1.10	29.77±1.05	31.72±1.48	**32.90**±2.60

preventing the server from accessing individual $\tilde{\boldsymbol{W}}_i$'s and $\tilde{\boldsymbol{F}}_i$'s and improving privacy. For K-FAC, however, while the server does not need access to individual $\tilde{\boldsymbol{W}}_i$'s, it does need access to individual $\tilde{\boldsymbol{F}}_i$'s. This is because there does not exist a summation operation over K-FAC matrices that preserves the Kronecker factorization property. In other words, to compute matrix vector products of the form $\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \boldsymbol{x}$, the server needs to store each $\tilde{\boldsymbol{F}}_i$ and individually compute and sum up $\tilde{\boldsymbol{F}}_i \boldsymbol{x}$. However, we argue that having access to $\tilde{\boldsymbol{F}}_i$ can be more private than having access to local models $\tilde{\boldsymbol{W}}_i$, where the latter is needed for knowledge distillation and neuron matching baselines. We discuss this in more detail, including an experiment on measuring privacy using model inversion attacks (Zhang et al., 2020), in Appendix E.

5 Experiments

5.1 Setup

We evaluate the performance of `FedFisher` in comparison to state-of-the-art (SOTA) neuron matching, knowledge distillation, and model fusion baselines across a range of image recognition tasks in a FL setting. The datasets that we use are (i) FashionMNIST (Xiao et al., 2017), (ii) SVHN (Netzer et al., 2011), (iii) CIFAR10 (Krizhevsky et al., 2009) and (iv) CINIC10 (Darlow et al., 2018). Our code is available at `https://github.com/Divyansh03/FedFisher`.

Baselines. We compare `FedFisher` with 6 other one-shot baselines. `FedAvg` is the de facto baseline in all our experiments. The other baselines that we compare with are (i) `PFNM` (Wang et al., 2020a; Yurochkin et al., 2019), (ii) `OTFusion` (Singh and Jaggi, 2020), (iii) `DENSE` (Zhang et al., 2022), (iv) `FisherMerge` (Matena and Raffel, 2022) and (v) `RegMean` (Jin et al., 2023). `PFNM` and `OTFusion` are popular neuron matching methods that first permute the weights of local models and then average them, rather than directly averaging the weights. `DENSE` is a SOTA method for one-shot FL based on data-free knowledge distillation that uses GANs to artificially generate data for distillation at the server. `FisherMerge` and `RegMean` are SOTA model fusion baselines. We note that, similar to `FedFisher(Diag)`, `FisherMerge` also proposes using the diagonal Fisher when merging models. However, our application of the diagonal Fisher differs in how we compute the Fisher averaged model (Lemma 1) and additional regularization (Equation (7)). We defer a more detailed discussion on the difference between the algorithms to Appendix A. We avoid comparing with baselines that need auxiliary data or maintain an ensemble of local models (Diao et al., 2023; Guha et al., 2019) at the server to ensure fairness of comparison. We also avoid comparing with algorithms that are inherently multi-round in nature such as `FedProx` (Li et al., 2020b), `SCAFFOLD` (Karimireddy et al., 2020) and adapative variants of `FedAvg` such as `FedAdam` (Reddi et al., 2021) since their performance would be similar to `FedAvg` for one round.

Models and Other Details. For FashionMNIST we use the LeNet architecture (LeCun et al., 1998); for other datasets we use a CNN model proposed in Wang et al. (2020a). To simulate data heterogeneity among the client datasets, we split our original image dataset into M partitions using a Dirichlet sampling procedure with parameter α (Hsu et al., 2019; Reddi et al., 2021), with a lower value of α implying a more heterogeneous split. The local optimization procedure is the same

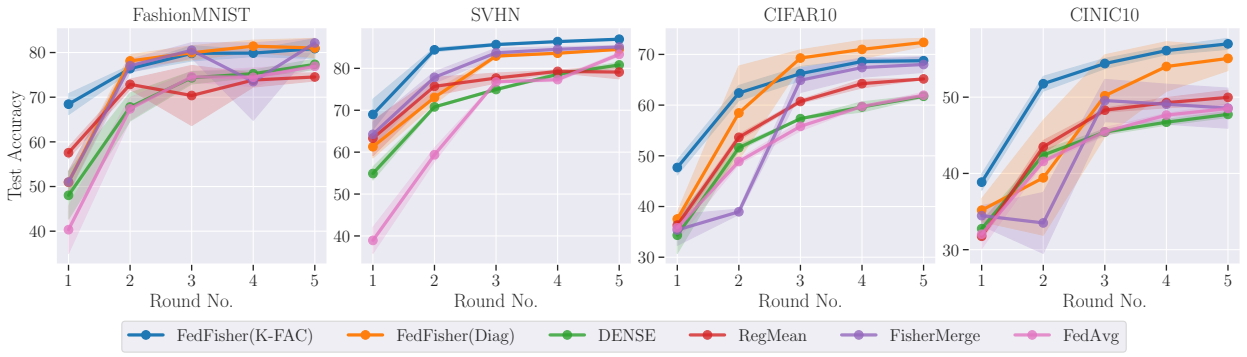


Figure 2: Results of performing 5 rounds of local training and aggregation across different datasets for $\alpha = 0.1$ and $M = 5$. `FedFisher` variants offer additional utility in multi-round settings and continue to improve over baselines.

Table 3: Results on one-shot aggregation using a pre-trained ResNet-18 model with $\alpha = 0.1$ and $M = 5$. `FedFisher` variants show a significant improvement in performance compared to baselines when using a pre-trained model.

Dataset	`FedAvg`	`OTFusion`	`RegMean`	`DENSE`	`Fisher Merge`	`FedFisher (Diag)`	`FedFisher (K-FAC)`
CIFAR10	56.89±1.31	56.89±1.33	60.98±1.84	57.18±1.11	78.76±2.30	79.06±0.82	**80.42**±1.13
CINIC10	50.05±0.70	50.04±0.69	50.53±0.69	50.83±0.32	66.89±1.87	**69.33**±3.44	68.04±1.64
GTSRB	40.63±3.30	40.40±3.35	46.05±2.52	44.87±1.13	61.27±2.80	62.80±4.43	**69.29**±1.72
CIFAR100	30.54±0.46	30.43±0.36	32.10±0.58	33.29±0.48	43.27±1.06	**49.00**±1.88	48.73±0.58

across all algorithms. In particular, clients perform $E = 30$ epochs of local training using the SGD optimizer with local learning rate $\eta = 0.01$, batch size 64 and momentum factor 0.9. To compute the Fisher diagonal and Fisher K-FAC we use the `nngeometry` package (George, 2021). Further details, including how we tuned hyperparameters, can be found in Appendix D.

5.2 Results

`FedFisher` outperforms baselines across varying heterogeneity. As a first step, we seek to understand the impact of data heterogeneity on the performance of `FedFisher` and other one-shot baselines. To do so, we fix the number of clients $M = 5$ and vary α in the range $\{0.05, 0.1, 0.2\}$ which can be considered moderate to high data heterogeneity. Table 2 summarizes the results obtained by the algorithms across the various datasets. We see that `FedFisher` variants, especially `FedFisher(K-FAC)`, consistently outperform baselines and give almost $10 - 20\%$ improvement over vanilla averaging in most cases. This highlights the effectiveness of `FedFisher` as a one-shot algorithm that can tackle data heterogeneity in FL settings while being computation and communication efficient.

`FedFisher` outperforms baselines across varying number of clients. In Appendix D, we provide experimental results in which we fix heterogeneity $\alpha = 0.3$ and vary the number of clients as $M = \{10, 20, 30\}$. Our results show that `FedFisher(K-FAC)` continues to outperform baselines across different M with up to 10% improvement in some cases like CIFAR10.

Extension to Few-Shot Settings. A natural question to consider is if we can gain additional utility from `FedFisher` by extending it to the *few shot* setting, i.e., using a few more rounds of local training and aggregation. Figure 2 presents the results of performing 5 rounds of aggregation with different algorithms with $\alpha = 0.1$ and $M = 5$. We omit comparison with `OTFusion` here because we did not find the performance to be competitive with `FedAvg` after the second round. We also omit comparison with `PFNM` as it modifies the global model architecture, leading to increased client computation and communication in every round. We see that while the use of multiple rounds improves the performance of all algorithms, `FedFisher` variants continue to show the greatest improvement, especially for CINIC10 which can be considered the hardest dataset in our experiments. This highlights the additional utility offered by `FedFisher` in few-shot settings.

Using Pre-trained Models. As motivated by recent literature in FL (Chen et al., 2022; Nguyen et al., 2022; Tan et al., 2022), in many cases the server might have access to a model pre-trained on a large public dataset. We consider a setting where the server has a ResNet-18 (He et al., 2016) pre-trained on Tiny-ImageNet (Le and Yang, 2015) and wants to fine-tune

this model. The fine-tuning datasets that we consider are CIFAR10, CINIC10, GTSRB Stallkamp et al. (2011) and CIFAR100. In all cases we split the fine-tuning dataset across $M = 5$ clients with $\alpha = 0.1$ heterogeneity. We focus on full fine-tuning, i.e., clients update all weights in the model, for $E = 30$ local epochs for CIFAR100 and $E = 10$ local epochs for the rest. We also use a smaller step size of $\eta = 0.001$ in this setting. We omit comparison with `PFNM` since it does not support ResNet-like architectures. Table 3 summarizes the results of one-shot aggregation in this setting. We see that algorithms that use Fisher information, including `FisherMerge`, improve on `FedAvg` and other baselines by almost 20%, with our methods achieving the highest accuracy. We attribute this large improvement to the reduced distance between weights of the local models when starting from a pre-trained model, which in turn reduces the approximation error in Fisher averaging as discussed in Section 3.

6 Conclusion

In this work, we propose `FedFisher`, a novel algorithm for one-shot FL motivated by a Bayesian perspective of FL. We theoretically analyze `FedFisher` for two-layer over-parameterized neural networks and propose its practical versions `FedFisher(Diag)` and `FedFisher(K-FAC)` that outperform current state-of-the-art one-shot methods while being computation and communication efficient. As future work, we would like to extend the analysis of `FedFisher` for deeper neural networks and investigate the use of differential privacy to improve the privacy guarantees of the practical versions of `FedFisher`.

Acknowledgments

This work was supported in part by NSF grants CCF 2045694, CNS-2112471, CPS-2111751, SHF-2107024, ONR N00014-23-1-2149 and CMU Dean's and Barakat fellowships.

References

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

Aditya Kumar Akash, Sixu Li, and Nicolás García Trillos. Wasserstein barycenter-based model fusion and linear mode connectivity of neural networks. *arXiv preprint arXiv:2210.06671*, 2022.

Maruan Al-Shedivat, Jennifer Gillenwater, Eric P. Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *9th International Conference on Learning Representations*, 2021.

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in Neural Information Processing Systems*, 32, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019b.

Aleksandar Armacki, Dragana Bajovic, Dusan Jakovetic, and Soummya Kar. One-shot federated learning for model clustering and learning in heterogeneous environments. *arXiv preprint arXiv:2209.10866*, 2022.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. CINIC-10 is not Imagenet or CIFAR-10. *arXiv preprint arXiv:1810.03505*, 2018.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning, 2020.

Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*, pages 2611–2620. PMLR, 2021.

Yiqun Diao, Qinbin Li, and Bingsheng He. Towards addressing label skews in one-shot federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. Gradient descent provably optimizes over-

parameterized neural networks. In *7th International Conference on Learning Representations*, 2019.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *The Tenth International Conference on Learning Representations*, 2022.

Marie Garin, Theodoros Evgeniou, and Nicolas Vayatis. Weighting schemes for one-shot federated learning. 2022.

Thomas George. NNGeometry: Easy and Fast Fisher Information Matrices and Neural Tangent Kernels in PyTorch, February 2021. URL `https://doi.org/10.5281/zenodo.4532597`.

Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15076–15086, 2021.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Han Guo, Philip Greengard, Hongyi Wang, Andrew Gelman, Yoon Kim, and Eric P. Xing. Federated learning as variational inference: A scalable expectation propagation approach. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Clare Elizabeth Heinbaugh, Emilio Luz-Ricca, and Huajie Shao. Data-free one-shot federated learning under very high statistical heterogeneity. In *The Eleventh International Conference on Learning Representations*, 2023.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. In *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with NeurIPS 2019 (FL-NeurIPS'19)*, December 2019.

Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning analysis. In *International Conference on Machine Learning*, pages 4423–4434. PMLR, 2021.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: renormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, 2023.

Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

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

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

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

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). `http://www.cs.toronto.edu/~kriz/cifar.html`, 2009.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR, 2019.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, May 2021.

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

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization for heterogeneous networks. In *Proceedings of the 3rd MLSys Conference*, January 2020b.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *International Conference on Machine Learning*, pages 13857–13869. PMLR, 2022.

James Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417. PMLR, 2015.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agøura y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *International Conference on Artificial Intelligenece and Statistics (AISTATS)*, April 2017. URL `https://arxiv.org/abs/1602.05629`.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? exploring the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2206.15387*, 2022.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. *arXiv preprint arXiv:2006.05148*, 2020.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Bingqing Song, Prashant Khanduri, Xinwei Zhang, Jinfeng Yi, and Mingyi Hong. Fedavg converges to zero training loss linearly: The power of overparameterized multi-layer neural networks.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and C. Igel. The german traffic sign recognition benchmark: A multi-class classification competition. *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, 2011. URL `https://api.semanticscholar.org/CorpusID:15926837`.

Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in Neural Information Processing Systems*, 35:19332–19344, 2022.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *8th International Conference on Learning Representations*, 2020a.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33:7611–7623, 2020b.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *https://arxiv.org/abs/1708.07747*, aug 2017.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.

Yaodong Yu, Alexander Wei, Sai Praneeth Karimireddy, Yi Ma, and Michael Jordan. Tct: Convexifying federated learning using bootstrapped neural tangent kernels. *Advances in Neural Information Processing Systems*, 35:30882–30897, 2022.

Kai Yue, Richeng Jin, Ryan Pilgrim, Chau-Wai Wong, Dror Baron, and Huaiyu Dai. Neural tangent kernel empowered federated learning. In *International Conference on Machine Learning*, pages 25783–25803. PMLR, 2022.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 35: 21414–21428, 2022.

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 253–261, 2020.

Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pages 12878–12889. PMLR, 2021.

Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable]
   Yes. Please see Section 2 and Section 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable]
   Yes. Please see Section 3 and Section 4.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]
   Yes. Please see supplemental.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable]
   Yes. Please see Section 3.
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable]
   Yes. Please see Appendix B.
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable]
   Yes. Please see Section 3.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]
   Yes. Please see Section 5 and Appendix D in supplemental.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable]
   Yes. Please see Section 5 and Appendix D in supplemental.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times).
   Yes. Please see Appendix D in supplemental.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable]
Yes. Please see Appendix D.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable]
   Yes.

   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable]
   Not Applicable.

   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable]
   Not Applicable.

   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable]
   Not Applicable.

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable]
   Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable]
   Not Applicable.

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable]
   Not Applicable.

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable]
   Not Applicable.

Appendix for "FedFisher: Leveraging Fisher Information for One-Shot Federated Learning"

A Additional Related Work

One-Shot FL. We review here some additional work on one-shot FL apart from the Knowledge Distillation and Neuron Matching baselines discussed in our work. Initial works such as Guha et al. (2019) propose to just use the ensemble of client models at the server. However, this approach increases the storage and computation cost by a factor of M where M is the number of clients. The work of Diao et al. (2023) discusses how we can improve the prediction of this ensemble when the label distribution across client data is highly skewed. Another line of work proposes that clients send some distilled form of their data to the server (Shin et al., 2020; Zhou et al., 2020). However, the privacy guarantees of such methods is unclear. The work of Garin et al. (2022) proposes techniques to optimize the weights of given to the local models when aggregating them at the server to improve one-shot performance. However, their analysis is limited to simple linear models and does not consider combing neural networks. We also note the existence of works that propose to perform clustering in a one-shot manner in FL setting (Armacki et al., 2022; Dennis et al., 2021); these approaches our orthogonal to our problem of finding a global minimizer in a one-shot manner.

Convergence of overparameterized NNs in FL. The works of Huang et al. (2021), Deng et al. (2020); Song et al. study the convergence of `FedAvg` for overparameterized neural networks. We note that these works are primarily concerned with convergence and do not propose any new algorithms as such compared to our work. We also note the existence of related works (Yu et al., 2022; Yue et al., 2022) that proposes to use NTK style Jacobian features to speed up FL training; however these works usually require multiple training rounds.

Comparison with (Matena and Raffel, 2022). We note that, similar to `FedFisher(Diag)`, `FisherMerge` (Matena and Raffel, 2022) also proposes using the diagonal Fisher when merging models. However, there are several key differences which we would like to mention. Firstly our problem is more well-defined in the sense that we have an explicit objective in Equation (1) that we are trying to solve with Fisher merging. This theoretical formulation is what leads us to propose the additional regularization in `FedFisher` in Equation (7) (and also `FedFisher(Diag)` subsequently) and guarantees for two-layer networks. Matena and Raffel (2022) on the other hand does not provide any theoretical guarantees. Also note that the procedure to compute the Fisher averaged model is different (gradient descent in our work versus diagonal Fisher inversion in Matena and Raffel (2022)). Finally, on a practical side, Matena and Raffel (2022) do not consider the effect of data heterogeneity in their experiments and focus on fusing two models whereas we explicitly account for data heterogeneity in our model training and aggregate up to thirty local models in our experiments (see results in Figure 3).

B Proofs

B.1 Proof of Proposition 1.

Proposition 1. *(Global Posterior Decomposition (Al-Shedivat et al., 2021)) Under the flat prior $\mathbb{P}(\boldsymbol{W}) \propto 1$, the global posterior decomposes into a product of local posteriors, i.e., $\mathbb{P}(\boldsymbol{W}|\mathcal{D}) \propto \prod_{i=1}^{M} \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i)$. Furthermore, modes of the global posterior coincide with the optima of the FL objective in Equation (1), i.e, $\arg\max_{\boldsymbol{W}\in\mathbb{R}^d} \mathbb{P}(\boldsymbol{W}|\mathcal{D}) = \arg\min_{\boldsymbol{W}\in\mathbb{R}^d} L(\boldsymbol{W})$.*

Proof.

We have,

$$\begin{aligned}
\mathbb{P}(\boldsymbol{W}|\mathcal{D}) &\propto \mathbb{P}(\mathcal{D}|\boldsymbol{W}) && (\because \mathbb{P}(\boldsymbol{W}) \propto 1) \\
&= \prod_{i=1}^{M} \mathbb{P}(\mathcal{D}_i|\boldsymbol{W}) && (\mathcal{D}_i \text{ are i.i.d generated}) \\
&\propto \prod_{i=1}^{M} \mathbb{P}(\boldsymbol{W}|\mathcal{D}_i). && (10)
\end{aligned}$$

Also,

$$\begin{aligned}
\arg\max_{\boldsymbol{W}\in\mathbb{R}^d} \mathbb{P}(\boldsymbol{W}|\mathcal{D}) &= \arg\max_{\boldsymbol{W}\in\mathbb{R}^d} \log \mathbb{P}(\mathcal{D}|\boldsymbol{W}) \\
&= \arg\max_{\boldsymbol{W}\in\mathbb{R}^d} \sum_{i=1}^{M}\sum_{j=1}^{n} (\log q(\boldsymbol{x}_{ij}) - \ell(f(\boldsymbol{W}, \boldsymbol{x}_{ij}), \boldsymbol{y}_{ij})) && (\textit{Assumption } 2) \\
&= \arg\max_{\boldsymbol{W}\in\mathbb{R}^d} - \sum_{i=1}^{M}\sum_{j=1}^{n} \ell(f(\boldsymbol{W}, \boldsymbol{x}_{ij}), \boldsymbol{y}_{ij}) \\
&= \arg\min_{\boldsymbol{W}\in\mathbb{R}^d} L(\boldsymbol{W}). && (11)
\end{aligned}$$

□

B.2 Proof of Proposition 2.

Proposition 2. *Any $\boldsymbol{W}$ which satisfies $\left(\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i\right) \boldsymbol{W} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i$ is a minimizer of the objective $\sum_{i=1}^{M} (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)$.*

Proof.

Define $G(\boldsymbol{W}) = \sum_{i=1}^{M} (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{W} - \tilde{\boldsymbol{W}}_i)$. First note that

$$\nabla^2 G(\boldsymbol{W}) = 2\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \succcurlyeq 0 \qquad (\tilde{\boldsymbol{F}}_i \text{ is positive semi-definite and symmetric } \forall i \in [M]). \tag{12}$$

This implies that $G(\boldsymbol{W})$ is a convex function. Therefore any $\boldsymbol{W}$ which satisfies $\nabla G(\boldsymbol{W}) = 0$ is a minimizer of $G(\boldsymbol{W})$. We have,

$$\nabla G(\boldsymbol{W}) = 2\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i (\boldsymbol{W} - \boldsymbol{W}_i). \tag{13}$$

Setting $\nabla G(\boldsymbol{W}) = 0$ we get $\left(\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i\right) \boldsymbol{W} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i$. □

B.3 Proof of Lemma 1.

Lemma 1. *Let $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots$ be the iterates generated by running the following gradient descent (GD) procedure: $\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta_S \sum_{i=1}^{M} \left(\tilde{\boldsymbol{F}}_i \boldsymbol{W}^{(t)} - \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i \right)$ with $\boldsymbol{W}^{(0)} = \sum_{i=1}^{M} \tilde{\boldsymbol{W}}_i / M$ and $\eta_S \leq 1/\lambda_{\max}$ where $\lambda_{\max}$ is the maximum eigenvalue of $\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i$. Then, $\lim_{T \to \infty} \boldsymbol{W}^{(T)} = \boldsymbol{W}^*$.*

Proof.

Recall,

$$\boldsymbol{W}^* = \underset{\boldsymbol{W} \in \mathbb{R}^d}{\arg\min} \left\{ \tilde{L}(\boldsymbol{W}) = \sum_{i=1}^{M} \left\| \boldsymbol{W} - \tilde{\boldsymbol{W}}_i \right\|_2^2 \text{ such that } \left(\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \right) \boldsymbol{W} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i \right\}. \tag{14}$$

Let $\boldsymbol{F} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i$, $\boldsymbol{b} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{W}}_i$ and $\bar{\boldsymbol{W}} = \sum_{i=1}^{M} \tilde{\boldsymbol{W}}_i / M$. Also note that $\sum_{i=1}^{M} \left\| \boldsymbol{W} - \tilde{\boldsymbol{W}}_i \right\|_2 = M \left\| \boldsymbol{W} - \bar{\boldsymbol{W}} \right\|_2^2 + \sum_{i=1}^{M} \left\| \bar{\boldsymbol{W}} - \tilde{\boldsymbol{W}}_i \right\|_2^2$. Therefore, the expression for $\boldsymbol{W}^*$ can be simplified as,

$$\boldsymbol{W}^* = \underset{\boldsymbol{W} \in \mathbb{R}^d}{\arg\min} \left\{ \tilde{L}(\boldsymbol{W}) = \left\| \boldsymbol{W} - \bar{\boldsymbol{W}} \right\|_2^2 \text{ such that } \boldsymbol{F}\boldsymbol{W} = \boldsymbol{b} \right\}. \tag{15}$$

Since $\boldsymbol{F}$ is symmetric and PSD (see Equation (12)), we have by the spectral decomposition of $\boldsymbol{F}$,

$$\boldsymbol{F} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^\top \\ \boldsymbol{V}_2^\top \end{bmatrix} = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top. \tag{16}$$

Here $\boldsymbol{V} \in \mathbb{R}^{(d \times d)}$ is an orthogonal matrix consisting of the eigenvectors of $\boldsymbol{F}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix consisting of all the eigenvalues of $\boldsymbol{F}$. $\boldsymbol{\Sigma}_1$ is a diagonal matrix consisting of only the positive eigenvalues. $\boldsymbol{V}_1$ consists of the eigenvectors corresponding to the positive eigenvalues while $\boldsymbol{V}_2$ consists of the eigenvectors corresponding to the zero eigenvalues. Also note that we have $\boldsymbol{V}_1^\top \boldsymbol{V}_2 = \boldsymbol{0}$.

We first observe that for any $\boldsymbol{W}_1^*$ and $\boldsymbol{W}_2^*$ such that $\boldsymbol{F}\boldsymbol{W}_1^* = \boldsymbol{F}\boldsymbol{W}_2^* = \boldsymbol{b}$ we have,

$$\begin{aligned} \boldsymbol{F}\boldsymbol{W}_1^* = \boldsymbol{F}\boldsymbol{W}_2^* &\iff \boldsymbol{V}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_1^* = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_2^* \\ &\iff \left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{V}_1^\top \right) \boldsymbol{V}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_1^* = \left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{V}_1^\top \right) \boldsymbol{V}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_2^* \\ &\iff \boldsymbol{V}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_1^* = \boldsymbol{V}_1 \boldsymbol{V}_1^\top \boldsymbol{W}_2^*. \end{aligned} \tag{17}$$

Next we observe,

$$\boldsymbol{V}_2 \boldsymbol{V}_2^\top \boldsymbol{W}^* = \boldsymbol{V}_2 \boldsymbol{V}_2^\top \bar{\boldsymbol{W}}. \tag{18}$$

The proof for this follows via simple contradiction argument as follows. Suppose $\boldsymbol{V}_2 \boldsymbol{V}_2^\top \boldsymbol{W}^* \neq \boldsymbol{V}_2 \boldsymbol{V}_2^\top \bar{\boldsymbol{W}}$. Let $\boldsymbol{W}^\diamond$ be a vector such that $\boldsymbol{F}\boldsymbol{W}^\diamond = \boldsymbol{b}$ and $\boldsymbol{V}_2 \boldsymbol{V}_2^\top \boldsymbol{W}^\diamond = \boldsymbol{V}_2 \boldsymbol{V}_2^\top \bar{\boldsymbol{W}}$. Then,

$$\begin{aligned} \left\| \bar{\boldsymbol{W}} - \boldsymbol{W}^* \right\|_2^2 &= \left\| \boldsymbol{V}_2 \boldsymbol{V}_2^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^* \right) + \boldsymbol{V}_1 \boldsymbol{V}_1^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^* \right) \right\|_2^2 && \left(\boldsymbol{V}_2 \boldsymbol{V}_2^\top + \boldsymbol{V}_1 \boldsymbol{V}_1^\top = \boldsymbol{I} \right) \\ &= \left\| \boldsymbol{V}_2 \boldsymbol{V}_2^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^* \right) + \boldsymbol{V}_1 \boldsymbol{V}_1^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^\diamond \right) \right\|_2^2 && \text{(from Equation (17))} \\ &= \left\| \boldsymbol{V}_2 \boldsymbol{V}_2^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^* \right) \right\|_2^2 + \left\| \boldsymbol{V}_1 \boldsymbol{V}_1^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^\diamond \right) \right\|_2^2 && \left(\boldsymbol{V}_1^\top \boldsymbol{V}_2 = \boldsymbol{0} \right) \\ &= \left\| \boldsymbol{V}_2 \boldsymbol{V}_2^\top \left(\bar{\boldsymbol{W}} - \boldsymbol{W}^* \right) \right\|_2^2 + \left\| \bar{\boldsymbol{W}} - \boldsymbol{W}^\diamond \right\|_2^2 && \left(\boldsymbol{V}_2 \boldsymbol{V}_2^\top (\bar{\boldsymbol{W}} - \boldsymbol{W}^\diamond) = 0 \right) \\ &> \left\| \bar{\boldsymbol{W}} - \boldsymbol{W}^\diamond \right\|_2^2 && \left(\boldsymbol{V}_2 \boldsymbol{V}_2^\top \boldsymbol{W}^* \neq \boldsymbol{V}_2 \boldsymbol{V}_2^\top \bar{\boldsymbol{W}} \right). \end{aligned} \tag{19}$$

leading to a contradiction.

According to the GD step in Lemma 1 we have,

$$\begin{aligned}\boldsymbol{W}^{(t+1)} &= \boldsymbol{W}^{(t)} - \eta_S(\boldsymbol{F}\boldsymbol{W}^{(t)} - \boldsymbol{b}) \\ &= (\boldsymbol{I} - \eta_S\boldsymbol{F})\boldsymbol{W}^{(t)} + \eta_S\boldsymbol{b}. \end{aligned} \tag{20}$$

Therefore,

$$\begin{aligned}\boldsymbol{W}^{(T)} &= (\boldsymbol{I} - \eta_S\boldsymbol{F})^T\boldsymbol{W}^{(0)} + \eta_S\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{F})^t\boldsymbol{b} \\ &= (\boldsymbol{I} - \eta_S\boldsymbol{F})^T\boldsymbol{W}^{(0)} + \eta_S\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{F})^t\boldsymbol{F}\boldsymbol{W}^* \\ &= \boldsymbol{V}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma})^T\boldsymbol{V}^\top\boldsymbol{W}^{(0)} + \eta_S\sum_{t=0}^{T-1}\left(\boldsymbol{V}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma})^t\boldsymbol{V}^\top\right)\boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top\boldsymbol{W}^* \\ &= (\boldsymbol{V}_1(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{V}_2^\top)\boldsymbol{W}^{(0)} + \eta_S\left(\boldsymbol{V}_1\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{V}_2^\top\right)\boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top\boldsymbol{W}^* \\ &= (\boldsymbol{V}_1(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T\boldsymbol{V}_1^\top + \boldsymbol{V}_2\boldsymbol{V}_2^\top)\boldsymbol{W}^{(0)} + \eta_S\left(\boldsymbol{V}_1\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t\boldsymbol{V}_1^\top\right)\boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top\boldsymbol{W}^*. \end{aligned} \tag{21}$$

In the limit $T \to \infty$ and with $\eta_S \leq 1/\lambda_{\max}(\boldsymbol{\Sigma}_1)$, we have,

$$\lim_{T\to\infty}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T = \boldsymbol{0} \text{ and } \lim_{T\to\infty}\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t = \frac{1}{\eta_S}\boldsymbol{\Sigma}_1^{-1}. \tag{22}$$

Thus,

$$\begin{aligned}\lim_{T\to\infty}\boldsymbol{W}^{(T)} &= \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{W}^{(0)} + \boldsymbol{V}_1\boldsymbol{V}_1^\top\boldsymbol{W}^* \\ &= \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{W}} + \boldsymbol{V}_1\boldsymbol{V}_1^\top\boldsymbol{W}^* \\ &= \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{W}^* + \boldsymbol{V}_1\boldsymbol{V}_1^\top\boldsymbol{W}^* \qquad \text{(from Equation (18))} \\ &= \boldsymbol{W}^*. \end{aligned} \tag{23}$$

This completes the proof. □

B.4 Proof of Theorem 1

In this subsection, we provide the proof for Theorem 1 in Section 3 of our work. To do so, we first introduce some additional notation and basic results.

B.4.1 Additional Notation and Basic Results

Recall the two-layer ReLU NN is modeled as follows,

$$f(\boldsymbol{W}, \boldsymbol{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \boldsymbol{x}^\top \boldsymbol{w}_r \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_r \geq 0\right\}. \tag{24}$$

We can write the output of the neural network alternatively as,

$$\begin{aligned} f(\boldsymbol{W}, \boldsymbol{x}) &= \sum_{r=1}^{m} \left(\frac{1}{\sqrt{m}} a_r \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_r \geq 0\right\} \boldsymbol{x} \right)^\top \boldsymbol{w}_r \\ &= \phi(\boldsymbol{W}, \boldsymbol{x})^\top \boldsymbol{W}, \end{aligned} \tag{25}$$

where

$$\phi(\boldsymbol{W}, \boldsymbol{x}) = \frac{1}{\sqrt{m}} \begin{bmatrix} a_1 \boldsymbol{x} \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_1 \geq 0\right\} \\ a_2 \boldsymbol{x} \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_2 \geq 0\right\} \\ \vdots \\ a_m \boldsymbol{x} \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_m \geq 0\right\}] \end{bmatrix} \in \mathbb{R}^{mp}. \tag{26}$$

Initialization. Recall $\boldsymbol{W}_0$ is the common initialization point for all the local models before they perform local optimization, i.e., $\boldsymbol{W}_i^{(0)} = \boldsymbol{W}_0$. Also recall $\boldsymbol{W}^{(0)} = \bar{\boldsymbol{W}} = \sum_{i=1}^{M} \tilde{\boldsymbol{W}}_i / M$ is the initialization point for the server optimization.

Definition 2. *(Matrices $\boldsymbol{A}_0$, $\tilde{\boldsymbol{A}}$, $\boldsymbol{H}_0$, $\tilde{\boldsymbol{H}}$) Matrices $\boldsymbol{A}_0$ and $\tilde{\boldsymbol{A}}$ are defined as follows:*

$$\boldsymbol{A}_0 = \begin{bmatrix} \phi(\boldsymbol{W}_0, \boldsymbol{x}_{11})^\top \\ \phi(\boldsymbol{W}_0, \boldsymbol{x}_{12})^\top \\ \vdots \\ \phi(\boldsymbol{W}_0, \boldsymbol{x}_{Mn})^\top \end{bmatrix} \in \mathbb{R}^{(N \times mp)} \tag{27}$$

and,

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \phi(\tilde{\boldsymbol{W}}_1, \boldsymbol{x}_{11})^\top \\ \vdots \\ \phi(\tilde{\boldsymbol{W}}_M, \boldsymbol{x}_{Mn})^\top \end{bmatrix} \in \mathbb{R}^{(N \times mp)}. \tag{28}$$

Furthermore we define $\boldsymbol{H}_0 = \boldsymbol{A}_0 \boldsymbol{A}_0^\top$, $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{A}}^\top \in \mathbb{R}^{N \times N}$.

Definition 3. *(Vector of true labels $\boldsymbol{y}$ and vector of predicted outputs $\tilde{\boldsymbol{y}}$)*

The vector of true labels $\boldsymbol{y}$ is defined as follows,

$$\boldsymbol{y} = \begin{bmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{Mn} \end{bmatrix} \in \mathbb{R}^N. \tag{29}$$

Given local models $\tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2, \ldots, \tilde{\boldsymbol{W}}_M$, the vector of predicted outputs $\tilde{\boldsymbol{y}}$ is defined as follows,

$$\tilde{\boldsymbol{y}} = \begin{bmatrix} f(\tilde{\boldsymbol{W}}_1, \boldsymbol{x}_{11}) \\ f(\tilde{\boldsymbol{W}}_1, \boldsymbol{x}_{12}) \\ \vdots \\ f(\tilde{\boldsymbol{W}}_M, \boldsymbol{x}_{Mn}) \end{bmatrix} \in \mathbb{R}^N. \tag{30}$$

Definition 4. *(Proxy output and vector of proxy outputs at iteration t)*

Given local models $\tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2, \ldots, \tilde{\boldsymbol{W}}_M$, the proxy output $\tilde{f}(\boldsymbol{W}, \boldsymbol{x}_{ij})$ for any $\boldsymbol{W} \in \mathbb{R}^d$ and for any $\boldsymbol{x}_{ij}$ where $i \in [M], j \in [n]$ is defined as,

$$\tilde{f}(\boldsymbol{W}, \boldsymbol{x}_{ij}) = \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})^\top \boldsymbol{W}. \tag{31}$$

Let $\boldsymbol{W}^{(0)}, \boldsymbol{W}^{(1)}, \ldots$ be the sequence of iterates generated by the global optimization process in FedFisher (Algorithm 1 Lines 12-14). We define $\tilde{\boldsymbol{f}}(t)$ as follows:

$$\tilde{\boldsymbol{f}}(t) = \begin{bmatrix} \tilde{f}(\boldsymbol{W}^{(t)}, \boldsymbol{x}_{11}) \\ \tilde{f}(\boldsymbol{W}^{(t)}, \boldsymbol{x}_{12}) \\ \vdots \\ \tilde{f}(\boldsymbol{W}^{(t)}, \boldsymbol{x}_{Mn}) \end{bmatrix} \in \mathbb{R}^N. \tag{32}$$

Claim 1. *(Bounded gradient) For any $\boldsymbol{W} \in \mathbb{R}^d$ and $\|\boldsymbol{x}\|_2 \leq 1$ we have $\|\nabla_{\boldsymbol{W}} f(\boldsymbol{W}, \boldsymbol{x})\|_2^2 \leq 1$.*

Proof.

Firstly observe that,

$$\nabla_{\boldsymbol{W}} f(\boldsymbol{W}, \boldsymbol{x}) = \phi(\boldsymbol{W}, \boldsymbol{x}). \tag{33}$$

We have,

$$\begin{aligned} \|\phi(\boldsymbol{W}, \boldsymbol{x})\|_2^2 &= \frac{1}{m} \sum_{r=1}^{m} \left\| a_r \boldsymbol{x} \mathbb{I}\left\{\boldsymbol{x}^\top \boldsymbol{w}_r \geq 0\right\} \right\|_2^2 \\ &\leq \frac{1}{m} \sum_{r=1}^{m} \|a_r \boldsymbol{x}\|_2^2 \\ &\leq 1 \qquad\qquad (a_r\text{'s are } \pm 1,\ \|\boldsymbol{x}\|_2 \leq 1). \end{aligned} \tag{34}$$

Claim 2. *(Fisher closed-form expression) Under the squared loss $\ell(z, y) = \frac{1}{2}(y - z)^2$, the Fisher at client i simplifies to,*

$$\boldsymbol{F}_i = \frac{1}{n} \sum_{j=1}^{n} \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij}) \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})^\top. \tag{35}$$

Proof.

Under the squared loss, we have $\mathbb{P}(y|\boldsymbol{x}_{ij}, \boldsymbol{W}) \propto \exp(-(y - f(\boldsymbol{W}, \boldsymbol{x}_{ij}))^2/2)$. Thus,

$$\begin{aligned} \nabla_{\boldsymbol{W}} \log \mathbb{P}(y|\boldsymbol{x}_{ij}, \boldsymbol{W}) &= (f(\boldsymbol{W}, \boldsymbol{x}_{ij}) - y) \nabla_{\boldsymbol{W}} f(\boldsymbol{W}, \boldsymbol{x}_{ij}) \\ &= (f(\boldsymbol{W}, \boldsymbol{x}_{ij}) - y) \phi(\boldsymbol{W}, \boldsymbol{x}_{ij}) \qquad \text{(using Equation (33))}. \end{aligned} \tag{36}$$

Thus,

$$\begin{aligned}
\boldsymbol{F}_i &= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_y\left[\nabla_{\boldsymbol{W}}\log\mathbb{P}(y|\boldsymbol{x}_{ij},\boldsymbol{W})\nabla_{\boldsymbol{W}}\log\mathbb{P}(y|\boldsymbol{x}_{ij},\boldsymbol{W})^\top\right]_{\boldsymbol{W}=\tilde{\boldsymbol{W}}_i} \\
&= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_y\left[(y-f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^2\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\right] \\
&= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_y\left[(y-f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^2\right]\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top \\
&\overset{(a)}{=} \frac{1}{n}\sum_{j=1}^{n}\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top
\end{aligned} \tag{37}$$

where (a) uses $\mathbb{P}(y|f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij}))\propto\exp(-(y-f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij}))^2/2)$ implying $y|f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\sim\mathcal{N}(f(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij}),1)$, following Assumption 1.

Claim 3. *Define the matrix* $\boldsymbol{H}_i^\infty\in\mathbb{R}^{n\times n}$ *as* $(\boldsymbol{H}_i^\infty)_{k,l}=\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\boldsymbol{x}_{ik}^\top\boldsymbol{x}_{il}\mathbb{I}\left\{\boldsymbol{w}^\top\boldsymbol{x}_{ik}\geq 0\right\}\mathbb{I}\left\{\boldsymbol{w}^\top\boldsymbol{x}_{il}\geq 0\right\}\right]$. *Then* $\lambda_{\min}(\boldsymbol{H}_i^\infty)\geq\lambda_{\min}(\boldsymbol{H}^\infty)=\lambda_0$.

Proof. Suppose $\lambda_{\min}(\boldsymbol{H}_i^\infty)=\lambda_i<\lambda_{\min}(\boldsymbol{H}^\infty)$. Let $\boldsymbol{x}$ be such that $\boldsymbol{x}^\top\boldsymbol{H}_i^\infty\boldsymbol{x}=\lambda_i$. Define $\tilde{\boldsymbol{x}}\in\mathbb{R}^{N\times N}$ to be a vector such that $\tilde{\boldsymbol{x}}_{(i-1)\times n+1:i\times n}=\boldsymbol{x}$ and zero everywhere else. Then $\tilde{\boldsymbol{x}}^\top\boldsymbol{H}^\infty\tilde{\boldsymbol{x}}=\boldsymbol{x}^\top\boldsymbol{H}_i^\infty\boldsymbol{x}=\lambda_i<\lambda_{\min}(\boldsymbol{H}^\infty)$ leading to a contradiction. Therefore $\lambda_{\min}(\boldsymbol{H}_i^\infty)\geq\lambda_{\min}(\boldsymbol{H}^\infty)=\lambda_0$.

Claim 4. *Let* $\boldsymbol{X}_1,\boldsymbol{X}_2\ldots,\boldsymbol{X}_M\in\mathbb{R}^{n\times p}$. *Define,*

$$\boldsymbol{X}=\begin{bmatrix}\boldsymbol{X}_1\\ \boldsymbol{X}_2\\ \vdots\\ \boldsymbol{X}_M\end{bmatrix}\in\mathbb{R}^{N\times p}$$

Also define

$$\boldsymbol{D}_{\boldsymbol{X}}=\begin{bmatrix}\boldsymbol{X}_1\boldsymbol{X}_1^\top & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{X}_2\boldsymbol{X}_2^\top & \cdots & \boldsymbol{0}\\ \vdots & & & \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{X}_M\boldsymbol{X}_M^\top\end{bmatrix}\in\mathbb{R}^{N\times N} \tag{38}$$

Then $M\boldsymbol{D}_{\boldsymbol{X}}\succcurlyeq\boldsymbol{X}\boldsymbol{X}^\top$.

Proof. Let $\boldsymbol{a}=\text{vec}(\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_M)\in\mathbb{R}^N$ where each $\boldsymbol{a}_i\in\mathbb{R}^n$. Then,

$$\begin{aligned}
\boldsymbol{a}^\top(M\boldsymbol{D}_{\boldsymbol{X}}-\boldsymbol{X}\boldsymbol{X}^\top)\boldsymbol{a} &= M\sum_{i=1}^{M}\boldsymbol{a}_i^\top\boldsymbol{X}_i\boldsymbol{X}_i^\top\boldsymbol{a}_i-\sum_{i=1}^{M}\sum_{j=1}^{M}\boldsymbol{a}_i^\top\boldsymbol{X}_i\boldsymbol{X}_j^\top\boldsymbol{a}_j \\
&= (M-1)\sum_{i=1}^{M}\boldsymbol{a}_i\boldsymbol{X}_i\boldsymbol{X}_i^\top\boldsymbol{a}_i-\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\boldsymbol{a}_i^\top\boldsymbol{X}_i\boldsymbol{X}_j^\top\boldsymbol{a}_j \\
&= \sum_{i=1}^{M}\sum_{j=i+1}^{M}\boldsymbol{a}_i^\top\boldsymbol{X}_i\boldsymbol{X}_i^\top\boldsymbol{a}_i+\boldsymbol{a}_j^\top\boldsymbol{X}_j\boldsymbol{X}_j^\top\boldsymbol{a}_j-\boldsymbol{a}_i^\top\boldsymbol{X}_i\boldsymbol{X}_j^\top\boldsymbol{a}_j-\boldsymbol{a}_j^\top\boldsymbol{X}_j\boldsymbol{X}_i^\top\boldsymbol{a}_i \\
&= \sum_{i=1}^{M}\sum_{j=i+1}^{M}\left\|\boldsymbol{X}_i^\top\boldsymbol{a}_i-\boldsymbol{X}_j^\top\boldsymbol{a}_j\right\|_2^2 \\
&\geq 0
\end{aligned}$$

Thus $M\boldsymbol{D}_{\boldsymbol{X}}\succcurlyeq\boldsymbol{X}^\top\boldsymbol{X}$.

Claim 5.

$$M \begin{bmatrix} \boldsymbol{H}_1^\infty & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{H}_2^\infty & \cdots & \mathbf{0} \\ \vdots & & & \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{H}_M^\infty \end{bmatrix} \succcurlyeq \boldsymbol{H}^\infty \tag{39}$$

where $\boldsymbol{H}_i^\infty$ is defined in Claim 3 and $\boldsymbol{H}^\infty$ is defined in Definition 1.

Proof. This follows from an application of Claim 4.

B.4.2 Key Lemmas

Before moving to the theorem proof, we first state some key lemmas and their proofs. Note that the probabilities of all events in this proof are over the random initialization of $\boldsymbol{W}_0$.

Lemma 2 is used to bound the local optimization at clients. Note that since the local optimization at each client is independent of all other clients, we can consider them to be M instances of centralized optimization starting from the same initialization. Thus we can use existing results in the centralized setting to give these bounds.

Lemma 2. *(Theorem 3.1 and Lemma C.1 in Arora et al. (2019)) If we set $m = \Omega\left(\frac{N^6}{\lambda_0^4 \kappa^2 \delta^3}\right)$ and $\eta = \mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$, then with probability at least $1-\delta$ over the random initialization of $\boldsymbol{W}_0$, for all clients $i \in [M]$ simultaneously, we have*

1. $\sum_{j=1}^n (y_{ij} - f(\boldsymbol{W}_0, \boldsymbol{x}_{ij}))^2 = \mathcal{O}\left(\frac{N}{\delta}\right)$
2. $\sum_{j=1}^n (y_{ij} - f(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij}))^2 \leq (1 - \eta\lambda_0/2)^K \mathcal{O}\left(\frac{N}{\delta}\right)$
3. $\|\tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\|_2 \leq \frac{4\sqrt{N}(1-(1-\eta\lambda_0/4)^K)}{\sqrt{m}\lambda_0} \sqrt{\sum_{j=1}^n (y_{ij} - f(\boldsymbol{W}_0, \boldsymbol{x}_{ij}))^2} = R_0 \; \forall r \in [m]$.

where $\tilde{\boldsymbol{w}}_{i,r}$ denotes the r-th weight in the local model of the i-th client and $\boldsymbol{w}_{0,r}$ denotes the r-th weight of the randomly initialized model $\boldsymbol{W}_0$.

Firstly note that the local optimization at client i depends on $\boldsymbol{H}_i^\infty$. Here we are using the result in Claim 3 which shows that $\lambda_{\min}(\boldsymbol{H}_i^\infty) \geq \lambda_{\min}(\boldsymbol{H}^\infty) = \lambda_0$. Also note that using Arora et al. (2019) we can only guarantee that the above event holds at a single client with probability $1-\delta$. In order for this event to hold simultaneously with probability $1-\delta$ for all clients, we use the union bound and set the failure probability at each client to be $\delta' = \delta/M$. This leads the bound to have a dependence on N instead of n as in the single client case.

Also note that substituting the bound in part (1) in part (3), we have the following upper bound on R_0:

$$\begin{aligned} R_0 &= \mathcal{O}\left(\frac{N(1-(1-\eta\lambda_0/4)^K)}{\sqrt{m\delta}\lambda_0}\right) \\ &= \mathcal{O}\left(\frac{N}{\sqrt{m\delta}\lambda_0}\right). \end{aligned} \tag{40}$$

Lemma 3 and Lemma 4 are standard in two-layer NN optimization and are adopted as is.

Lemma 3. *(Lemma 3.2 in Du et al. (2019)) For a given R, define the following event:*

$$A_{ijr}(R) = \{\exists \boldsymbol{w} : \|\boldsymbol{w} - \boldsymbol{w}_{0,r}\|_2 \leq R, \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w} \geq 0\right\}\}. \tag{41}$$

We have $\Pr(A_{ijr}(R)) \leq \frac{2R}{\sqrt{2\pi\kappa}}$ where the randomness is over the initialization of $\boldsymbol{w}_{0,r}$.

Lemma 4. *(Lemma 3.1 in Du et al. (2019)). If $m = \Omega\left(\frac{N^2}{\lambda_0^2} \log \frac{N}{\delta}\right)$, we have with probability $1-\delta$, $\|\boldsymbol{H}_0 - \boldsymbol{H}^\infty\|_2 \leq \frac{\lambda_0}{4}$ and $\lambda_{\min}(\boldsymbol{H}_0) \geq \frac{3\lambda_0}{4}$.*

Lemma 5, Lemma 6 and Lemma 7 are our contribution and form the basis of our proof.

Lemma 5. *If* $m = \Omega\left(\frac{N^6}{\lambda_0^4 \kappa^2 \delta^3}\right)$, *with probability* $1-\delta$, $\lambda_{\min}(\tilde{\boldsymbol{H}}) \geq \lambda_0/2$ *and* $\lambda_{\max}(\tilde{\boldsymbol{H}}) \leq N$

Proof.

The (k,l)-th entry of $\tilde{\boldsymbol{H}} \in \mathbb{R}^{N\times N}$ is given by,

$$\tilde{\boldsymbol{H}}_{kl} = \frac{1}{m}\boldsymbol{x}_{k'k''}^\top \boldsymbol{x}_{l'l''} \sum_{r=1}^{m} \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \tilde{\boldsymbol{w}}_{k',r} \geq 0\right\} \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \tilde{\boldsymbol{w}}_{l',r} \geq 0\right\} \tag{42}$$

where $k' = \lceil k/n \rceil, k'' = k - k' + 1, l' = \lceil l/n \rceil, l'' = l - l' + 1$.

We have,

$$\begin{aligned}
&\mathbb{E}\left[|\tilde{\boldsymbol{H}}_{kl} - (\boldsymbol{H}_0)_{kl}|\right] \\
&= \mathbb{E}\left[\frac{1}{m}\left|\boldsymbol{x}_{k'k''}^\top \boldsymbol{x}_{l'l''}\right| \sum_{r=1}^{m} \mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \tilde{\boldsymbol{w}}_{k',r} \geq 0\right\} \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \tilde{\boldsymbol{w}}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \boldsymbol{w}_{0,r} \geq 0\right\} \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \\
&\overset{(a)}{\leq} \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m} \mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \tilde{\boldsymbol{w}}_{k',r} \geq 0\right\} \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \tilde{\boldsymbol{w}}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \boldsymbol{w}_{0,r} \geq 0\right\} \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \\
&\leq \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m} \mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \tilde{\boldsymbol{w}}_{k',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\} + \mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \tilde{\boldsymbol{w}}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \\
&\overset{(b)}{\leq} \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m} \mathbb{I}\left\{A_{k'k''r}(R_0)\right\} + \mathbb{I}\left\{A_{l'l''r}(R_0)\right\}\right] \\
&= \frac{1}{m}\sum_{r=1}^{m} \Pr(A_{k'k''r}(R_0)) + \Pr(A_{l'l''r}(R_0)) \\
&\overset{(c)}{\leq} \frac{4R_0}{\sqrt{2\pi}\kappa},
\end{aligned} \tag{43}$$

where (a) uses $\left|\boldsymbol{x}_{k'k''}^\top \boldsymbol{x}_{l'l''}\right| \leq \|\boldsymbol{x}_{k'k''}\|_2 \|\boldsymbol{x}_{l'l''}\|_2 \leq 1$ (Assumption 2), (b) follows from the definition of A_{ijr} in Lemma 3 and definition of R_0 in Lemma 2, (c) uses the result in Lemma 3.

Thus we have,

$$\mathbb{E}\left[\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_0\|_F\right] \leq \mathbb{E}\left[\sum_{k,l} |\tilde{\boldsymbol{H}}_{kl} - (\boldsymbol{H}_0)_{kl}|\right] \leq \frac{4N^2 R_0}{\sqrt{2\pi}\kappa}. \tag{44}$$

By Markov's inequality, with probability $1-\delta$, we have

$$\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_0\|_F \leq \frac{\mathbb{E}\left[\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_0\|_F\right]}{\delta} \leq \frac{4N^2 R_0}{\sqrt{2\pi}\delta\kappa}. \tag{45}$$

Thus,

$$\left\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_0\right\|_2 \leq \|\tilde{\boldsymbol{H}} - \boldsymbol{H}_0\|_F \leq \frac{4N^2 R_0}{\sqrt{2\pi}\delta\kappa}. \tag{46}$$

This implies,

$$\lambda_{\min}(\tilde{\boldsymbol{H}}) \geq \lambda_{\min}(\boldsymbol{H}_0) - \frac{4N^2 R_0}{\sqrt{2\pi}\delta\kappa} \geq \frac{\lambda_0}{2} \tag{47}$$

using Lemma 4 and substituting $m = \Omega\left(\frac{N^6}{\lambda_0^4 \delta^3 \kappa^2}\right)$ in the upper bound on R_0 in **??**.

We also have,

$$\begin{aligned}
\left\|\tilde{\boldsymbol{H}}\right\|_2 &= \left\|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top\right\|_2 \\
&= \left\|\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}}\right\|_2 \\
&= \left\|\sum_{i=1}^{M}\sum_{j=1}^{n}\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\right\|_2 \\
&\le \sum_{i=1}^{M}\sum_{j=1}^{n}\left\|\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\right\|_2 \\
&\le \sum_{i=1}^{M}\sum_{j=1}^{n}\left\|\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})\right\|_2^2 \\
&\le N \qquad \text{(Claim 1).}
\end{aligned} \tag{48}$$

Therefore $\lambda_{\max}(\tilde{\boldsymbol{H}}) = \left\|\tilde{\boldsymbol{H}}\right\|_2 \le N$. □

Lemma 6. *If we set* $m = \Omega\left(\frac{N^6}{\lambda_0^4\kappa^2\delta^3}\right)$ *and* $\eta = \mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$*, then with probability at least* $1-\delta$ *over the random initialization of* $\boldsymbol{W}_0$*, we have*

1. $\left\|\tilde{\boldsymbol{f}}(0)-\tilde{\boldsymbol{y}}\right\|_2^2 = \mathcal{O}\left(\frac{N^3}{\delta\lambda_0^2}\right)$
2. $\left\|\tilde{\boldsymbol{f}}(t)-\tilde{\boldsymbol{y}}\right\|_2^2 \le (1-\eta_S\lambda_0/2n)^t\left\|\tilde{\boldsymbol{f}}(0)-\tilde{\boldsymbol{y}}\right\|_2^2$
3. $\left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 \le \frac{4\sqrt{N}}{\sqrt{m}\lambda_0}\left\|\tilde{\boldsymbol{f}}(0)-\tilde{\boldsymbol{y}}\right\|_2 = R_1$.

Proof.

Part (1).

We have,

$$\begin{aligned}
(\tilde{f}(\boldsymbol{W}^{(0)},\boldsymbol{x}_{ij})-\tilde{y}_{ij})^2 &= (\phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\boldsymbol{W}^{(0)} - \phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\tilde{\boldsymbol{W}}_i)^2 \\
&= \left(\frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\ge 0\right\}\boldsymbol{x}_{ij}^\top\left(\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right)\right)^2 \\
&\le \sum_{r=1}^{m}\left(a_r\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\ge 0\right\}\boldsymbol{x}_{ij}^\top\left(\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right)\right)^2 \qquad \text{(Jensen's inequality)} \\
&\overset{(a)}{\le} \sum_{r=1}^{m}\left\|\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right\|_2^2 \\
&\le \frac{1}{M}\sum_{r=1}^{m}\sum_{l=1}^{M}\|\tilde{\boldsymbol{w}}_{l,r}-\tilde{\boldsymbol{w}}_{i,r}\|_2^2 \qquad \text{(Jensen's inequality)} \\
&\le \frac{2}{M}\sum_{r=1}^{m}\sum_{l=1}^{M}\|\tilde{\boldsymbol{w}}_{l,r}-\boldsymbol{w}_{0,r}\|_2^2 + \|\tilde{\boldsymbol{w}}_{i,r}-\boldsymbol{w}_{0,r}\|_2^2 \\
&\le 4mR_0^2 \qquad \text{(Lemma 2 Part 3)} \\
&= \mathcal{O}\left(\frac{N^2}{\lambda_0^2\delta}\right) \qquad \text{(??)}
\end{aligned} \tag{49}$$

where (a) uses Cauchy-Schwartz and $\left\|a_r \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\} \boldsymbol{x}_{ij}\right\|_2 \leq 1$.

Thus, $\left\|\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}\right\|_2^2 = \sum_{i=1}^M \sum_{j=1}^n (\tilde{f}(\boldsymbol{W}^{(0)}, \boldsymbol{x}_{ij}) - \tilde{y}_{ij})^2 = \mathcal{O}\left(\frac{N^3}{\lambda_0^2 \delta}\right)$.

Part (2).

The GD step that the central server performs for `FedFisher` can be written as,

$$\begin{aligned}
\boldsymbol{W}^{(t+1)} &= \boldsymbol{W}^{(t)} - \eta_S \sum_{i=1}^M \boldsymbol{F}_i \left(\boldsymbol{W}^{(t)} - \tilde{\boldsymbol{W}}_i\right) \\
&= \boldsymbol{W}^{(t)} - \tilde{\eta_S} \sum_{i=1}^M \sum_{j=1}^n \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij}) \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})^\top \left(\boldsymbol{W}^{(t)} - \tilde{\boldsymbol{W}}_i\right) && \text{(Claim 2, } \tilde{\eta_S} = \eta_S/n) \\
&= \boldsymbol{W}^{(t)} - \tilde{\eta_S} \sum_{i=1}^M \sum_{j=1}^n \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij}) \left(\tilde{f}(\boldsymbol{W}^{(t)}, \boldsymbol{x}_{ij}) - \tilde{y}_{ij}\right) && \text{(Definition 4, Definition 3)} \\
&= \boldsymbol{W}^{(t)} - \tilde{\eta_S} \tilde{\boldsymbol{A}}^\top (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}) && \text{(Definition 2).} \qquad (50)
\end{aligned}$$

We have,

$$\begin{aligned}
\tilde{\boldsymbol{f}}(t+1) - \tilde{\boldsymbol{f}}(t) &= \tilde{\boldsymbol{A}}(\boldsymbol{W}^{(t+1)} - \boldsymbol{W}^{(t)}) \\
&= -\tilde{\eta_S} \tilde{\boldsymbol{H}} (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}) && \text{(Definition 2, Equation (50)).} \qquad (51)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\left\|\tilde{\boldsymbol{f}}(t+1) - \tilde{\boldsymbol{y}}\right\|_2^2 &= \left\|\tilde{\boldsymbol{f}}(t+1) - \tilde{\boldsymbol{f}}(t) + \tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}\right\|_2^2 \\
&= \left\|\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}\right\|_2^2 - 2\tilde{\eta_S} (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}) \tilde{\boldsymbol{H}} (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}) && \text{(Equation (51))} \\
&\quad + \tilde{\eta_S}^2 \left\|\tilde{\boldsymbol{H}} (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}})\right\|_2^2 \\
&\leq (1 - \tilde{\eta_S} \lambda_0 + \tilde{\eta_S}^2 N^2) \left\|\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}\right\|_2^2 && \text{(Lemma 5)} \\
&\leq \left(1 - \frac{\eta_S \lambda_0}{2n}\right) \left\|\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}}\right\|_2^2 && \left(\tilde{\eta_S} \leq \frac{\lambda_0}{2N^2}\right). \qquad (52)
\end{aligned}$$

Part (3).

From Equation (50) we have,

$$\begin{aligned}
\left\|\boldsymbol{w}_r^{(t+1)} - \boldsymbol{w}_r^{(t)}\right\|_2 &\overset{(a)}{=} \left\|\tilde{\eta_S} \tilde{\boldsymbol{A}}_{(r-1)p+1:rp}^\top (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(t))\right\|_2 \\
&\leq \tilde{\eta_S} \left\|\tilde{\boldsymbol{A}}_{(r-1)p+1:rp}^\top\right\|_2 \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(t)\right\|_2 && \text{(Cauchy Schwartz)} \\
&\leq \tilde{\eta_S} \left\|\tilde{\boldsymbol{A}}_{(r-1)p+1:rp}^\top\right\|_F \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(t)\right\|_2 \\
&\leq \frac{\tilde{\eta_S}\sqrt{N}}{\sqrt{m}} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(t)\right\|_2 && \text{(Definition 2, } \|\boldsymbol{x}_{ij}\|_2 \leq 1) \qquad (53)
\end{aligned}$$

where in (a) we use the notation $\boldsymbol{A}_{x:y}$ to denote the submatrix of $\boldsymbol{A}$ having row numbers x to y.

Therefore,

$$\begin{aligned}
\left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 &\leq \sum_{t=0}^{\infty} \left\|\boldsymbol{w}_r^{(t+1)} - \boldsymbol{w}_r^{(t)}\right\|_2 \\
&\leq \frac{\tilde{\eta}_S\sqrt{N}}{\sqrt{m}} \sum_{t=0}^{\infty} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(t)\right\|_2 && \text{(using Equation (53))} \\
&\leq \frac{\tilde{\eta}_S\sqrt{N}}{\sqrt{m}} \sum_{t=0}^{\infty} \left(1 - \frac{\tilde{\eta}_S\lambda_0}{4}\right)^t \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(0)\right\|_2 && \text{(using Equation (52))} \\
&= \frac{4\sqrt{N}}{\sqrt{m}\lambda_0} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(0)\right\|_2 . && (54)
\end{aligned}$$

□

Lemma 7. *Let* $S_{ij} = \{r \in [m] : \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} = \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\}\}$ *and* $S_{ij}^\perp = [m] \backslash S_{ij}$*. With probability* $1-\delta$ *over the initialization, we have* $\sum_{i=1}^M \sum_{j=1}^n |S_{ij}^\perp|^2 = \mathcal{O}\left(\frac{m^2N^2(R_0^2+R_1^2)}{\delta^2\kappa^2}\right)$ *where* R_0 *is defined in Lemma 2 and* R_1 *is defined in Lemma 6.*

Proof.

We have,

$$\begin{aligned}
\mathbb{E}\left[|S_{ij}^\perp|\right] &= \sum_{r=1}^m \Pr(\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\}) \\
&\leq \sum_{r=1}^m \Pr\left(\left\{\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\} \cup \left\{\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right) \\
&\overset{(a)}{\leq} \sum_{r=1}^m \Pr\left(\left\{\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right) + \Pr\left(\left\{\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right) \\
&\overset{(b)}{\leq} \sum_{r=1}^m \Pr(A_{ijr}(R_0+R_1)) + \Pr(A_{ijr}(R_0)) \\
&\overset{(c)}{\leq} \frac{4m(R_0+R_1)}{\sqrt{2\pi}\kappa} && (55)
\end{aligned}$$

where (a) uses union bound, (b) uses Lemma 3 and $\|\boldsymbol{w}_r^* - \boldsymbol{w}_{0,r}\|_2 \leq \left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 + \left\|\boldsymbol{w}_r^{(0)} - \boldsymbol{w}_{0,r}\right\|_2 = \left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 + \left\|\frac{1}{M}\sum_{i=1}^M \tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\right\|_2 \leq \left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 + \frac{1}{M}\sum_{i=1}^M \|\tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\|_2 \leq R_1 + R_0$ and $\|\tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\|_2 \leq R_0$, (c) again uses Lemma 3.

Thus, using Markov's inequality, with probability at least $1-\delta$, we have for any $i \in [M]$ and $j \in [n]$,

$$|S_{ij}^\perp| = \mathcal{O}\left(\frac{m(R_0+R_1)}{\delta\kappa}\right). \tag{56}$$

Setting the failure probability as δ/N in Equation (56), we have with probability $1-\delta$, for all $i \in [M]$ and $j \in [n]$ simultaneously,

$$\forall i \in [M]; \forall j \in [n] : |S_{ij}^\perp| = \mathcal{O}\left(\frac{mN(R_0+R_1)}{\delta\kappa}\right). \tag{57}$$

This implies,

$$\sum_{i=1}^M \sum_{j=1}^n |S_{ij}^\perp|^2 = \mathcal{O}\left(\frac{m^2N^2(R_0^2+R_1^2)}{\delta^2\kappa^2}\right). \tag{58}$$

□

B.4.3 Proof of Theorem 1

We first state the full theorem statement with the exact dependence of m on $(N, \lambda_0^{-1}, \delta^{-1}, \kappa^{-1})$.

Theorem 1. *Under Assumptions 2, 3, 4, for $m = \Omega\left(\frac{N^9}{\lambda_0^8 \delta^4 \kappa^2}\right)$, and i.i.d Gaussian initialization weights of $\boldsymbol{W}_0$ as $\boldsymbol{w}_{0,r} \sim \mathcal{N}(\boldsymbol{0}, \kappa)$, and initializing the second layer weights $a_r = \{-1, 1\}$ with probability $1/2$ for all $r \in [m]$, for step sizes $\eta = \mathcal{O}(\lambda_0/N^2)$, $\eta_S = \mathcal{O}(\lambda_0/N^2)$ and for a given failure probability $\delta \in (0, 1)$, the following is true with probability $1 - \delta$ over the random initialization:*

$$L(\boldsymbol{W}^*) \leq \underbrace{\mathcal{O}\left((1 - \eta\lambda_0/2)^K \frac{N}{\delta}\right)}_{\text{local optimization error}} + \underbrace{\mathcal{O}\left((2 - (1 - \eta\lambda_0/2)^K) \frac{N^9}{\lambda_0^8 \delta^4 m}\right)}_{\text{Laplace approximation error}}. \tag{59}$$

Proof.

Note that the conditions on m, η and η_S in Lemma 2, Lemma 4, Lemma 5 and Lemma 6 are satisfied by setting $m = \Omega\left(\frac{N^9}{\lambda_0^8 \delta^4 \kappa^2}\right)$, $\eta = \mathcal{O}(\lambda_0/N^2)$, $\eta_S = \mathcal{O}(\lambda_0/N^2)$ and hence we can now apply these lemma results for our proof. Furthermore, we can scale down the failure probability in these lemmas by a constant factor to ensure that all the results in the lemmas hold simultaneously with high probability via union bound.

First observe that setting $t \to \infty$ in Lemma 6 part (2), we have,

$$\tilde{f}(\boldsymbol{W}^*, \boldsymbol{x}_{ij}) = \tilde{y}_{ij} \tag{60}$$

Now,

$$\begin{aligned}
L(\boldsymbol{W}^*) &= \frac{1}{N} \sum_{i=1}^{M} \sum_{j=1}^{n} (f(\boldsymbol{W}^*, \boldsymbol{x}_{ij}) - y_{ij})^2 \\
&= \frac{1}{N} \sum_{i=1}^{M} \sum_{j=1}^{n} \left((\phi(\boldsymbol{W}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{W}^* - \tilde{y}_{ij} + \tilde{y}_{ij} - y_{ij}\right)^2 && \text{(Equation (25))} \\
&\leq \underbrace{\frac{2}{N} \sum_{i=1}^{M} \sum_{j=1}^{n} \left(\phi(\boldsymbol{W}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{W}^* - \tilde{y}_{ij}\right)^2}_{T_1} + \underbrace{\frac{2}{N} \sum_{i=1}^{M} \sum_{j=1}^{n} (\tilde{y}_{ij} - y_{ij})^2}_{T_2}.
\end{aligned} \tag{61}$$

T_2 measures how well the local models fit their local data and can be bounded as

$$T_2 = \mathcal{O}\left((1 - \eta\lambda_0/2)^K \frac{N}{\delta}\right) \tag{62}$$

using the result from Lemma 2.

We now bound T_1 as follows:

$$
\begin{aligned}
T_1 &= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{W}^*,\boldsymbol{x}_{ij})^\top\boldsymbol{W}^* - \tilde{y}_{ij}\right)^2 \\
&= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{W}^*,\boldsymbol{x}_{ij})^\top\boldsymbol{W}^* - \tilde{f}(\boldsymbol{W}^*,\boldsymbol{x}_{ij})\right)^2 && \text{(Equation (60))} \\
&= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{W}^*,\boldsymbol{x}_{ij})^\top\boldsymbol{W}^* - \phi(\tilde{\boldsymbol{W}}_i,\boldsymbol{x}_{ij})^\top\boldsymbol{W}^*\right)^2 && \text{(Definition 4)} \\
&= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*\left(\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*\geq 0\right\} - \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\geq 0\right\}\right)\right)^2 \\
&\overset{(a)}{=} \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\frac{1}{\sqrt{m}}\sum_{r\in S_{ij}^\perp}a_r\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*\left(\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*\geq 0\right\} - \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\geq 0\right\}\right)\right)^2 \\
&\leq \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|}{m}\sum_{r\in S_{ij}^\perp}\left(a_r\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*\right)^2 && \text{(Jensen's inequality)} \\
&\overset{(b)}{\leq} \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|}{m}\sum_{r\in S_{ij}^\perp}\left(\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^* - \boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\right)^2 \\
&\leq \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|^2}{m}\max_{r\in[m]}\|\boldsymbol{w}_r^* - \tilde{\boldsymbol{w}}_{i,r}\|_2^2 \\
&\overset{(c)}{=} \mathcal{O}\left(\frac{m^2N(R_0^4+R_1^4)}{\delta^2\kappa^2 m}\right) \\
&\overset{(d)}{=} \mathcal{O}\left(\frac{N^9}{\lambda_0^8\delta^4\kappa^2 m}(2-(1-\eta\lambda_0/2)^K)\right) && (63)
\end{aligned}
$$

where (a) follows from the definition of $S_{ij}^\perp$ in Lemma 7. (b) uses the observation that since $r\in S_{ij}^\perp$ we have $\text{sign}(\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*)\neq\text{sign}(\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r})$ which implies $|\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^*|\leq|\boldsymbol{x}_{ij}^\top\boldsymbol{w}_r^* - \boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}|$. For (c) we use Lemma 7 to bound $\sum_{i=1}^M\sum_{j=1}^n|S_{ij}|^2$ as $\mathcal{O}\left(\frac{m^2N^2(R_0^2+R_1^2)}{\delta^2\kappa^2}\right)$ and $\|\boldsymbol{w}_r^* - \tilde{\boldsymbol{w}}_{i,r}\|_2^2\leq 2\left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2^2 + 2\left\|\boldsymbol{w}_r^{(0)} - \tilde{\boldsymbol{w}}_{i,r}\right\|_2^2\leq 2R_0^2+2R_1^2$. For (d) we use $R_0=\mathcal{O}\left(\frac{N}{\sqrt{m\delta}\lambda_0}(1-(1-\eta\lambda_0/4)^K)\right)$ from Lemma 2 and $R_1=\mathcal{O}\left(\frac{N^2}{\sqrt{m\delta}\lambda_0^2}\right)$ from Lemma 6.

Now substituting the bounds in Equation (62) and Equation (63) in Equation (61), we have,

$$
L(\boldsymbol{W}^*)\leq\mathcal{O}\left((1-\eta\lambda_0/2)^K\frac{N}{\delta}\right)+\mathcal{O}\left((2-(1-\eta\lambda_0/2)^K)\frac{N^9}{\lambda_0^8\delta^4\kappa^2 m}\right) \tag{64}
$$

which completes the proof. □

B.5 Generalization Guarantees

In this subsection we provide generalization guarantees for the `FedFisher` global model. To do so, we first introduce some additional assumptions and definitions.

B.5.1 Additional Assumptions and Definitions

Definition 5. *A distribution ξ over $\mathbb{R}^d \times \mathbb{R}$ is (λ_0, δ, N)-non degenerate if for N i.i.d samples $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$ from $\mathcal{D}$, with probability at least $1-\delta$, we have $\lambda_{\min}(\boldsymbol{H}^\infty) \geq \lambda_0 > 0$.*

Assumption 5. *Let $\bar{\xi}$ be the distribution from which the test data points are sampled. $\bar{\xi}$ is a $(\lambda_0, \delta/3, N)$-non-degenerate distribution and the data samples $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^N \in \mathcal{D}$ are i.i.d samples from $\bar{\xi}$.*

Remark 1. *Note that in Assumption 5, we are only assuming that the collection of local data across clients is drawn i.i.d from some distribution $\bar{\xi}$. This does not imply that the data at any particular client i is drawn from $\bar{\xi}$, i.e., $\mathcal{D}_i \sim \bar{\xi}$ or that the data at client i is i.i.d with the data at client j for $i \neq j$. The data in $\mathcal{D}$ can be partitioned arbitrarily across clients and we make no assumptions on this. For instance, consider the case with $M = 2$ clients, $\mathcal{D} = \{(x_k, y_k) \sim \mathcal{N}(0, \boldsymbol{I})\}_{k=1}^N$, $\mathcal{D}_1 = \{(x, y) \in \mathcal{D}$ such that $y \geq 0\}$ and $\mathcal{D}_2 = \{(x, y) \in \mathcal{D}$ such that $y < 0\}$. Then clearly the data in $\mathcal{D}_1$ and $\mathcal{D}_2$ are not sampled from $\mathcal{N}(0, \mathbf{I})$ and are not i.i.d with each other.*

Given the test data distribution $\bar{\xi}$, for a function $f : \mathbb{R}^d \to \mathbb{R}$, we have the following definitions of population loss $L_{\bar{\xi}}(f)$ and empirical loss $L(f)$:

$$L_{\bar{\xi}}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\bar{\xi}}\left[\frac{1}{2}(f(\boldsymbol{x}) - y)^2\right], \tag{65}$$

$$L(f) = \frac{1}{2N}\sum_{k=1}^{N}(f(\boldsymbol{x}_k) - y_k)^2. \tag{66}$$

Definition 6. *(Rademacher complexity) Given N samples, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, the Rademacher complexity of a function class $\mathcal{F}$ that maps from $\mathbb{R}^d$ to $\mathbb{R}$ is defined as,*

$$\mathcal{R}_S(\mathcal{F}) = \frac{1}{N}\mathbb{E}_{\epsilon_1,\epsilon_2,\ldots,\epsilon_N}\left[\sup_{f\in\mathcal{F}}\sum_{k=1}^{N}\epsilon_k f(\boldsymbol{x}_k)\right] \tag{67}$$

where each ϵ_i is sampled independently from the Rademacher distribution $\mathrm{unif}(\{-1, 1\})$.

B.5.2 Key Lemmas

We now state some key lemmas that will be used in the generalization proof.

We begin with the following lemma from Arora et al. (2019) which bounds the Rademacher complexity of the class of two-layer neural network functions with bounded distance from initialization.

Lemma 8. *(Lemma 5.4 in Arora et al. (2019)) Given $R > 0$, with probability at least $1-\delta$ over the random initialization of $(\boldsymbol{W}_0, \boldsymbol{a})$, simultaneously for every $B > 0$, the function class*

$$\mathcal{F}_{R,B}^{\boldsymbol{W}_0,\boldsymbol{a}} = \{f_{\boldsymbol{W},\boldsymbol{a}} : \|\boldsymbol{w}_r - \boldsymbol{w}_{0,r}\|_2 \leq R\ \forall r \in [m], \|\boldsymbol{W} - \boldsymbol{W}_0\|_2 \leq B\}$$

has Rademacher complexity bounded as,

$$\mathcal{R}_S(\mathcal{F}_{R,B}^{\boldsymbol{W}_0,\boldsymbol{a}}) \leq \frac{B}{\sqrt{2N}}\left(1 + \left(\frac{2\log 2/\delta}{m}\right)^{1/4}\right) + \frac{2R^2\sqrt{m}}{\kappa} + R\sqrt{2\log\frac{2}{\delta}}. \tag{68}$$

Lemma 9 is standard in generalization theory and used to bound the population loss in terms of the empirical loss and Rademacher complexity of the class of functions to which our estimator belongs.

Lemma 9. *(Mohri et al. (2018)) Suppose $\frac{1}{2}(f(\boldsymbol{x})-y)^2$ is bounded in the range $[0,c]$. With probability $1-\delta$ over the random sampling of $\mathcal{D}$ we have,*

$$\sup_{f\in\mathcal{F}}\{L_{\bar{\xi}}(f)-L(f)\}\leq 2\mathcal{R}_S(\mathcal{F})+3c\sqrt{\frac{\log 2/\delta}{2N}}. \tag{69}$$

Similar to Lemma 2, we can use existing results in the centralized setting to bound the total distance moved by the local model of a client from its initialization.

Lemma 10. *(Lemma C.3 and Lemma 5.3 in Arora et al. (2019)) If we set $m\geq\kappa^{-2}\text{poly}(N,\delta^{-1},\lambda_0^{-1})$ and $\eta=\mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$, then with probability at least $1-\delta$ over the random initialization, we have*

1. $\|\boldsymbol{H}_0-\boldsymbol{H}^\infty\|_2\leq\mathcal{O}\left(\frac{N\sqrt{\log N/\delta}}{\sqrt{m}}\right)$
2. $\left\|\tilde{\boldsymbol{W}}_i-\boldsymbol{W}_0\right\|_2\leq\sqrt{\boldsymbol{y}_i^\top(\boldsymbol{H}_i^\infty)^{-1}\boldsymbol{y}_i}+\mathcal{O}\left(\frac{N\kappa}{\lambda_0\delta}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/4}\kappa^{1/2}}\qquad\forall i\in[M]$

where $\boldsymbol{y}_i=\text{vec}(y_{i1},y_{i2},\ldots,y_{in})$ and $\boldsymbol{H}_i^\infty$ is defined in Claim 3.

Note that in order to ensure that part (2) holds simultaneously for all $i\in[M]$ we set the failure probability to be δ/M in Lemma 5.3 in Arora et al. (2019).

The following lemmas are our contribution and used to bound the distance of the `FedAvg` model from initialization and the `FedFisher` model from the `FedAvg` model respectively.

Lemma 11. *(Bounding distance of `FedAvg` model from initialization)*

Assuming the conditions in Lemma 10 hold true we have,

$$\left\|\boldsymbol{W}^{(0)}-\boldsymbol{W}_0\right\|_2\leq\sqrt{2\boldsymbol{y}^\top(\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}}+\mathcal{O}\left(\frac{N\kappa}{\lambda_0\delta}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/4}\kappa^{1/2}} \tag{70}$$

Proof. We have,

$$\begin{aligned}
\left\|\boldsymbol{W}^{(0)}-\boldsymbol{W}_0\right\|_2^2&=\left\|\sum_{i=1}^M\tilde{\boldsymbol{W}}_i/M-\boldsymbol{W}_0\right\|_2^2\\
&\leq\frac{1}{M}\sum_{i=1}^M\left\|\tilde{\boldsymbol{W}}_i-\boldsymbol{W}_0\right\|_2^2\\
&\leq\left(\frac{1}{M}\sum_{i=1}^M2\boldsymbol{y}_i^\top(\boldsymbol{H}_i^\infty)^{-1}\boldsymbol{y}_i\right)+\mathcal{O}\left(\frac{N^2\kappa^2}{\lambda_0^2\delta^2}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/2}\kappa}&&\text{(using Lemma 10)}\\
&=2\boldsymbol{y}^\top(M\text{diag}(\boldsymbol{H}_1^\infty,\boldsymbol{H}_2^\infty,\ldots,\boldsymbol{H}_M^\infty))^{-1}\boldsymbol{y}+\mathcal{O}\left(\frac{N^2\kappa^2}{\lambda_0^2\delta^2}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/2}\kappa}\\
&\leq2\boldsymbol{y}^\top(\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}+\mathcal{O}\left(\frac{N^2\kappa^2}{\lambda_0^2\delta^2}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/2}\kappa}&&\text{(using Claim 5)}
\end{aligned} \tag{71}$$

Thus we have,

$$\left\|\boldsymbol{W}^{(0)}-\boldsymbol{W}_0\right\|_2\leq\sqrt{2\boldsymbol{y}^\top(\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}}+\mathcal{O}\left(\frac{N\kappa}{\lambda_0\delta}\right)+\frac{\text{poly}(N,\lambda_0^{-1},\delta^{-1})}{m^{1/4}\kappa^{1/2}}\qquad(\sqrt{a+b}\leq\sqrt{a}+\sqrt{b}) \tag{72}$$

This completes the proof. □

Lemma 12. *(Bounding distance of `FedFisher` model from `FedAvg` model) If we set $m \geq \kappa^{-2}\text{poly}(N, \delta^{-1}, \lambda_0^{-1})$, $\eta = \mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$ and $\kappa = \mathcal{O}\left(\frac{\lambda_0 \delta}{N}\right)$ then with probability at least $1-\delta$ over the random initialization, we have,*

$$\left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2 = \mathcal{O}\left(\frac{N}{\lambda_0}\right) \tag{73}$$

Proof.

We have,

$$\begin{aligned}
\left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2^2 &= \left\|\sum_{t=0}^{\infty} \boldsymbol{W}^{(t+1)} - \boldsymbol{W}^{(t)}\right\|_2^2 \\
&= \left\|-\tilde{\eta}_S \sum_{t=0}^{\infty} \tilde{\boldsymbol{A}}^\top (\tilde{\boldsymbol{f}}(t) - \tilde{\boldsymbol{y}})\right\|_2^2 && \text{(Equation (50))} \\
&= \left\|\tilde{\eta}_S \sum_{t=0}^{\infty} \tilde{\boldsymbol{A}}^\top (\boldsymbol{I} - \tilde{\eta}_S \tilde{\boldsymbol{H}})^t (\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}})\right\|_2^2 && \text{(Equation (51))} \\
&= \left\|\tilde{\boldsymbol{A}}^\top \boldsymbol{T} (\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}})\right\|_2^2 && \left(\boldsymbol{T} := \tilde{\eta}_S \sum_{t=0}^{\infty} (\boldsymbol{I} - \tilde{\eta}_S \tilde{\boldsymbol{H}})^t\right) \\
&= (\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}})^\top \boldsymbol{T} \tilde{\boldsymbol{H}} \boldsymbol{T} (\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}) && (\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{A}}^\top) \\
&\overset{(a)}{\leq} \frac{2\left\|\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}\right\|_2^2}{\lambda_0} \\
&\overset{(b)}{=} \mathcal{O}\left(\frac{N^2}{\lambda_0^2}\right).
\end{aligned} \tag{74}$$

For (a) we use the following argument. Let $\tilde{\boldsymbol{H}} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top$ be the eigen decomposition of $\tilde{\boldsymbol{H}}$. We see that $\boldsymbol{T} = \tilde{\eta}_S \boldsymbol{V} \sum_{t=0}^{\infty} (\boldsymbol{I} - \tilde{\eta}_S \boldsymbol{D}) \boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{V}^\top$. Thus $\boldsymbol{T}\tilde{\boldsymbol{H}}\boldsymbol{T} = \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{V}^\top = \tilde{\boldsymbol{H}}^{-1}$. Furthermore $\left\|\tilde{\boldsymbol{H}}^{-1}\right\|_2 \leq \frac{2}{\lambda_0}$ which follows from Lemma 5.

For (b) we use the following argument,

$$\begin{aligned}
\left\|\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}\right\|_2^2 &= \sum_{i=1}^{M}\sum_{j=1}^{n} (\tilde{f}(\boldsymbol{W}^{(0)}, \boldsymbol{x}_{ij}) - \tilde{y}_{ij})^2 \\
&= \sum_{i=1}^{M}\sum_{j=1}^{n} (\phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})^\top \boldsymbol{W}^{(0)} - \phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})^\top \tilde{\boldsymbol{W}}_i)^2 \\
&\leq \sum_{i=1}^{M}\sum_{j=1}^{n} \left\|\phi(\tilde{\boldsymbol{W}}_i, \boldsymbol{x}_{ij})\right\|_2^2 \left\|\boldsymbol{W}^{(0)} - \tilde{\boldsymbol{W}}_i\right\|_2^2 && \text{(Cauchy-Schwartz)} \\
&\leq \sum_{i=1}^{M}\sum_{j=1}^{n} \left\|\boldsymbol{W}^{(0)} - \tilde{\boldsymbol{W}}_i\right\|_2^2 && \text{(Equation (33))} \\
&\leq 2\sum_{i=1}^{M}\sum_{j=1}^{n} \left\|\boldsymbol{W}^{(0)} - \boldsymbol{W}_0\right\|_2^2 + \left\|\tilde{\boldsymbol{W}}_i - \boldsymbol{W}_0\right\|_2^2 \\
&\overset{(c)}{=} \sum_{i=1}^{M}\sum_{j=1}^{n} \mathcal{O}\left(\frac{N}{\lambda_0}\right) \\
&= \mathcal{O}\left(\frac{N^2}{\lambda_0}\right).
\end{aligned} \tag{75}$$

where (c) follows from Lemma 10 part (2), Lemma 11 and setting $\kappa = \mathcal{O}\left(\frac{\lambda_0 \delta}{N}\right)$.

Thus,

$$\left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2 = \mathcal{O}\left(\frac{N}{\lambda_0}\right).$$

□

B.5.3 Generalization Theorem and Proof

We first state the generalization bound and then provide the proof below.

Theorem 2. *Fix a failure probability $\delta \in (0,1)$. Under Assumptions 2, 3, 5, if we let $\kappa = \mathcal{O}\left(\frac{\lambda_0 \delta}{N}\right)$, $m \geq \kappa^{-2}\text{poly}(N, \lambda_0^{-1}, \delta^{-1})$, $K = \Omega\left(\frac{1}{\eta\lambda_0}\log(N/\delta)\right)$ then with probability at least $1-\delta$ over the random initialization and sampling of training data points, we have*

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\bar{\xi}}[(f(\boldsymbol{W}^*, \boldsymbol{x}) - y)^2] \leq \mathcal{O}\left(\frac{\sqrt{N}}{\lambda_0}\right) + \sqrt{\frac{\boldsymbol{y}^\top (\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}}{N}} + \mathcal{O}\left(\sqrt{\frac{\log \frac{N}{\lambda_0 \delta}}{N}}\right) \tag{76}$$

Remark 2. *We see that the last two terms in our bound match the generalization bound of a model trained in the centralized setting on $\mathcal{D}$ (Arora et al., 2019). The first term of order $\mathcal{O}\left(\frac{\sqrt{N}}{\lambda_0}\right)$ comes from bounding the distance of the* `FedFisher` *model from* `FedAvg`*, i.e., $\left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2$ and can be seen as the additional error incurred in the federated setting by* `FedFisher`*. In particular, to bound $\left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2$ we need to bound $\left\|\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}\right\|_2$ (see Lemma 11). Clearly, if the data across clients is similar then $\left\|\tilde{\boldsymbol{f}}(0) - \tilde{\boldsymbol{y}}\right\|_2$ will be small. However, our current approach does not make any bounded heterogeneity assumptions leading to a pessimistic $\mathcal{O}\left(\frac{\sqrt{N}}{\lambda_0}\right)$ bound. We conjecture that this bound can be improved by explicitly incorporating bounded data heterogeneity assumptions; however we leave the investigation of this as future work.*

Proof.

As a first step, we bound the distance of the $\boldsymbol{W}^*$ from $\boldsymbol{W}_0$. We have,

$$\begin{aligned}
\|\boldsymbol{W}^* - \boldsymbol{W}_0\|_2 &\leq \left\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\right\|_2 + \left\|\boldsymbol{W}^{(0)} - \boldsymbol{W}_0\right\|_2 \\
&\leq \mathcal{O}\left(\frac{N}{\lambda_0}\right) + \sqrt{2\boldsymbol{y}^\top (\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}} + \mathcal{O}\left(\frac{N\kappa}{\lambda_0 \delta}\right) + \frac{\text{poly}(N, \lambda_0^{-1}, \delta^{-1})}{m^{1/4}\kappa^{1/2}} \quad \text{(Lemma 10, Lemma 11)} \\
&= \mathcal{O}\left(\frac{N}{\lambda_0}\right)
\end{aligned} \tag{77}$$

where the last line follows from setting $\kappa = \mathcal{O}\left(\frac{\lambda_0 \delta}{N}\right)$, $m \geq \kappa^{-2}\text{poly}(N, \lambda_0^{-1}, \delta^{-1})$ and observing that $\sqrt{2\boldsymbol{y}^\top (\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}} = \mathcal{O}(\sqrt{N/\lambda_0})$.

Next we aim to bound $\|\boldsymbol{w}_r^* - \boldsymbol{w}_{0,r}\|_2$ for any $r \in [m]$. Recall we already bounded this quantity in Equation (63) as follows,

$$\begin{aligned}
\|\boldsymbol{w}_r^* - \boldsymbol{w}_{0,r}\|_2 &\leq \left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 + \left\|\boldsymbol{w}_r^{(0)} - \boldsymbol{w}_{0,r}\right\|_2 \\
&\leq \left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 + \frac{1}{M}\sum_{i=1}^{M} \|\tilde{\boldsymbol{w}}_{i,r} - \boldsymbol{w}_{0,r}\|_2 \\
&= R_1 + R_0 \quad \text{(Lemma 2, Lemma 6)} \\
&= \mathcal{O}\left(\frac{N^2}{\sqrt{m\delta}\lambda_0^2}\right) + \mathcal{O}\left(\frac{N}{\sqrt{m\delta}\lambda_0}(1 - (1 - \eta\lambda_0/4)^K)\right) \\
&= \mathcal{O}\left(\frac{\text{poly}(N, \delta^{-1}, \lambda_0^{-1})}{m^{1/2}}\right)
\end{aligned} \tag{78}$$

We now use a similar argument as the proof of theorem 5.1 in Arora et al. (2019) to provide the generalization guarantee. Under Assumption 5 we have $\mathcal{D} \sim \bar{\xi}$, where $\bar{\xi}$ is a $(\lambda_0, \delta/3, N)$ non-degenerate distribution. This implies with probability $1-\delta/3$ we have $\lambda_{\min}(\boldsymbol{H}^\infty) \geq \lambda_0 > 0$. Assuming this event holds, we note that the following events hold with probability $1-\delta$ over the random initialization.

i)

$$L(\boldsymbol{W}^*) \leq \frac{1}{\sqrt{N}} \tag{79}$$

which follows from Theorem 1 by setting $m = \Omega\left(\text{poly}(N, \lambda_0^{-1}, \delta^{-1}, \kappa^{-1})\right)$, $K = \Omega\left(\frac{1}{\eta\lambda_0}\log(N/\delta)\right)$, $\eta = \mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$ and $\eta_S = \mathcal{O}\left(\frac{\lambda_0}{N^2}\right)$.

ii)

$$\|\boldsymbol{W}^* - \boldsymbol{W}_0\|_2 \leq B \text{ and } \|\boldsymbol{w}_r^* - \boldsymbol{w}_{0,r}\|_2 \leq R \text{ where } B = \mathcal{O}\left(\frac{N}{\lambda_0}\right) \text{ and } R = \mathcal{O}\left(\frac{\text{poly}(N, \delta^{-1}, \lambda_0^{-1})}{m^{1/2}}\right) \tag{80}$$

which follows from Equation (77) and Equation (78) respectively.

iii) For $k = 1, 2, \ldots$, let $B_k = k$. Let k^* be the smallest integer such that $B_{k^*} = k^* \geq B$. This implies $B_{k^*} \leq B+1$ and $k^* = \mathcal{O}(N/\lambda_0)$. Using Lemma 8 we have,

$$\begin{aligned}
\mathcal{R}_S(\mathcal{F}_{R,B_{k^*}}^{\boldsymbol{W}_0,\boldsymbol{a}}) &\leq \frac{B+1}{\sqrt{2N}}\left(1 + \left(\frac{2\log 2/\delta}{m}\right)^{1/4}\right) + \frac{2R^2\sqrt{m}}{\kappa} + R\sqrt{2\log\frac{2}{\delta}} \\
&\leq \mathcal{O}\left(\frac{\sqrt{N}}{\lambda_0}\right) + \sqrt{\frac{\boldsymbol{y}^\top(\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}}{N}} + \frac{2}{\sqrt{N}}
\end{aligned} \tag{81}$$

where the last inequality follows from setting $m \geq \text{poly}(N, \lambda_0^{-1}, \delta^{-1}, \kappa^{-1})$, $\kappa = \mathcal{O}\left(\frac{\lambda\delta}{N}\right)$ and Equation (77). Also note that $f_{\boldsymbol{W}^*,\boldsymbol{a}} \in \mathcal{F}_{R,B_k^*}^{\boldsymbol{W}_0,\boldsymbol{a}}$.

iv) Using Lemma 9 and a union bound over all $k \in \{0, 1, 2, \mathcal{O}(N/\delta)\}$ we have,

$$\sup_{f \in \mathcal{F}_{R,B_k}^{\boldsymbol{W}_0,\boldsymbol{a}}} \{L_{\bar{\xi}}(f) - L(f)\} \leq 2\mathcal{R}_S(\mathcal{F}_{R,B_k}^{\boldsymbol{W}_0,\boldsymbol{a}}) + \mathcal{O}\left(\sqrt{\frac{\log(N/\delta\lambda_0)}{N}}\right) \quad \forall k \in \{0, 1, 2, \mathcal{O}(N/\delta)\}. \tag{82}$$

Note that we can scale down the failure probabilities in each of the above events to ensure that all the above events hold simultaneously with probability $1-\delta$, via union bound. Now assuming all the above conditions hold simultaneously, we have,

$$\begin{aligned}
L_{\bar{\xi}}(f_{\boldsymbol{W}^*,\boldsymbol{a}}) &\leq L(f_{\boldsymbol{W}^*,\boldsymbol{a}}) + 2\mathcal{R}_S(\mathcal{F}_{R,B_{k^*}}^{\boldsymbol{W}_0,\boldsymbol{a}}) + \mathcal{O}\left(\sqrt{\frac{\log\frac{N}{\lambda_0\delta}}{N}}\right) && \text{(from Equation (82))} \\
&\leq \frac{1}{\sqrt{N}} + 2\mathcal{R}_S(\mathcal{F}_{R,B_{k^*}}^{\boldsymbol{W}_0,\boldsymbol{a}}) + \mathcal{O}\left(\sqrt{\frac{\log\frac{N}{\lambda_0\delta}}{N}}\right) && \text{(from Equation (79))} \\
&\leq \mathcal{O}\left(\frac{\sqrt{N}}{\lambda_0}\right) + \sqrt{\frac{\boldsymbol{y}^\top(\boldsymbol{H}^\infty)^{-1}\boldsymbol{y}}{N}} + \mathcal{O}\left(\sqrt{\frac{\log\frac{N}{\lambda_0\delta}}{N}}\right) && \text{(from Equation (80) and Equation (81))}
\end{aligned} \tag{83}$$

which completes the proof. □

C Communication Efficiency of FedFisher Variants

We assume that the number of bits used to represent a scalar is 32 by default. For FedAvg, clients just need to transfer $\tilde{\boldsymbol{W}}_i$, making the total communication cost $32d$ bits. Our goal in this section is to show that we can introduce compression techniques in FedFisher(Diag) and FedFisher(K-FAC) to match the communication cost of FedAvg while having similar accuracy as the uncompressed version of these algorithms. To do so, we use standard uniform quantization and SVD compression as described below.

Uniform Quantization. Let $s_q \in \{1, 2, \ldots, 16\}$ be the factor by which we want to compress our information. We define number of quantization levels as $l_q = 2^{\lfloor 32/s_q \rfloor - 1} - 1$. Now given a vector $\boldsymbol{x} \in \mathbb{R}^d$, we define each element of the quantized $\boldsymbol{x}$ as follows,

$$[Q(\boldsymbol{x}, s_q)]_i = \|\boldsymbol{x}\|_\infty \text{sign}(x_i)\zeta_i(\boldsymbol{x}, s_q) \tag{84}$$

where $\|\boldsymbol{x}\|_\infty = \max_{i\in[d]} |x_i|$ and $\zeta_i(\boldsymbol{x}, s_q) = \left\lceil l_q \frac{|x_i|}{\|\boldsymbol{x}\|_\infty} \right\rceil / l_q$. Note that $\zeta_i(\boldsymbol{x}, s_q)$ can take only $l_q + 1 = 2^{\lfloor 32/s_q \rfloor - 1}$ distinct values and therefore to communicate $\zeta_i(\boldsymbol{x}, s_q)$ we only need $\lfloor 32/s_q \rfloor - 1$ bits. To communicate $\text{sign}(x_i)$ we need 1 bit and to communicate $\|\boldsymbol{x}\|_\infty$ we need 32 bits. Thus the communication cost of $Q(\boldsymbol{x}, s_q)$ becomes $d(\lfloor 32/s_q \rfloor - 1) + d + 32 = d(\lfloor 32/s_q \rfloor) + 32$ bits.

Singular Value Decomposition Compression. Let $\boldsymbol{A} = \mathbb{R}^{(m\times m)}$ matrix. The SVD decomposition of $\boldsymbol{A}$ can be written as,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{85}$$

where $\boldsymbol{U} = \mathbb{R}^{(m\times m)}$ is the matrix of left singular vectors, $\boldsymbol{\Sigma} \in \mathbb{R}^{(m\times m)}$ is a diagonal matrix with each element corresponding to a singular value and $\boldsymbol{V} = \mathbb{R}^{(m\times m)}$ is the matrix of right of singular vectors. The singular values are assumed to be sorted by magnitude, i.e., $|\Sigma_{1,1}| \geq |\Sigma_{2,2}| \ldots, |\Sigma_{m,m}|$. We see that the total cost for communicating $\boldsymbol{A}$ will $32m^2$ bits. To reduce this cost, a standard idea is to send only a limited number of singular values and singular vectors obtained by the SVD decomposition of $\boldsymbol{A}$. Specifically let s_v be the factor by which we want to compress the information in $\boldsymbol{A}$. We define $l_v = \lfloor m/2s_v \rfloor$. Then the SVD decompression of $\boldsymbol{A}$ can be defined as,

$$V(\boldsymbol{A}, s_v) = \boldsymbol{U}_{l_v}\boldsymbol{\Sigma}_{l_v}\boldsymbol{V}_{l_v}^\top \tag{86}$$

where $\boldsymbol{U}_{l_v} \in \mathbb{R}^{(m\times l_v)}$ corresponds to first l_v columns of $\boldsymbol{U}$, $\boldsymbol{\Sigma}_{l_v} \in \mathbb{R}^{(l_v\times l_v)}$ is a diagonal matrix corresponding to the first l_v elements of $\boldsymbol{\Sigma}$ and $\boldsymbol{V} \in \mathbb{R}^{(m\times l_v)}$ corresponds to the first l_v columns of $\boldsymbol{V}$. The communication cost of $V(\boldsymbol{A}, s_v)$ becomes $32(ml_v + l_v + ml_v) \approx 64ml_v \leq 32m^2/s_v$ bits, thereby achieving close to s_v compression.

Compression in FedFisher(Diag). For FedFisher(Diag) , clients need to communicate $\tilde{\boldsymbol{W}}_i$ and $\tilde{\boldsymbol{F}}_i$ where the number of parameters in $\tilde{\boldsymbol{F}}_i$ is exactly d. To ensure comparable communication to FedAvg, we quantize the weights corresponding to each layer of a neural network in $\tilde{\boldsymbol{W}}_i$ and $\tilde{\boldsymbol{F}}_i$ by a factor of 2, i.e, $s_q = 2$. This ensures that the communication of FedFisher(Diag) is $32(d + 2L)$ bits where L is the number of layers in the neural network. We note that there is a small overhead of $64L$ bits; however is negligible since $d \gg 2L$ for our neural networks. Table 4 shows the effects of compression on the accuracy of FedFisher(Diag) for CIFAR-10 and FashionMNIST datasets. For FashionMNIST the compressed version even slightly improves upon the uncompressed version which can be attributed to an additional regularization effect of compression. For CIFAR-10, the drop in accuracy is less than 1%.

Compression in FedFisher(K-FAC). Let $\{m_0, m_1, m_2, \ldots, m_L\}$ be the dimensions of each layer of a L layer neural network with m_0 corresponding to the dimension of the input. For FedFisher(K-FAC), $\tilde{\boldsymbol{F}}_i$ can be represented as $\{(\boldsymbol{A}_1 \otimes \boldsymbol{B}_1), (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2), \ldots, (\boldsymbol{A}_L \otimes \boldsymbol{B}_L)\}$ where $\boldsymbol{A}_l \in \mathbb{R}^{(m_{l-1}\times m_{l-1})}$ and $\boldsymbol{B}_l \in \mathbb{R}^{(m_l\times m_l)}$ represents the Kronecker factors of the l-th layer. Thus, the communication cost of $\tilde{\boldsymbol{F}}_i$ in this case is $\sum_{l=1}^{L} 32(m_{l-1}^2 + m_l^2)$ bits.

Our goal is to ensure that the communication cost of compressed $\tilde{\boldsymbol{F}}_i$ is less than $16d$ bits (we compress $\tilde{\boldsymbol{W}}_i$ to $16d$ bits using quantization, similar to FedFisher(Diag)). To do so, we use a mix of quantization and SVD compression.

Table 4: Test accuracy performance of `FedFisher(K-FAC)` with different levels of s_q compression on FashionMNIST and CIFAR10 with 5 clients and $\alpha = 0.1$.

Dataset	s_q	Compression	Test Accuracy
FashionMNIST	1	–	54.64±4.91
	2	2×	56.56±5.57
CIFAR-10	1	–	39.68±2.45
	2	2×	39.44±2.31

Specifically each $\boldsymbol{A}_l$ and $\boldsymbol{B}_l$ is first compressed to the SVD decomposition corresponding to maintaining the top l_v vectors. This ensures that the communication cost of $(\boldsymbol{A}_l, \boldsymbol{B}_l)$ is $32(2m_{l-1}l_v + 2m_l l_v + 2l_v)$. This SVD decomposition is then further compressed using s_q quantization compression to ensure that the communication cost is $(32/s_q)(2m_{l-1}l_v + 2m_l l_v + 2l_v)$. We set s_q and l_v such that $\sum_{l=1}^{L}(32/s_q)(2m_{l-1}l_v + 2m_l l_v + 2l_v) \leq 16d$. The corresponding s_v is then defined as $\lceil m/2l_v \rceil$. Table 5 summarizes the results obtained by different levels of s_q and s_v for the FashionMNIST and CIFAR datasets respectively. We see that keeping $s_q = 4$ and setting s_v accordingly usually gives the best performance and hence we use this setting for all our experiments.

Table 5: Test accuracy performance of `FedFisher(K-FAC)` with different levels of $s_q \times s_v$ compression on FashionMNIST and CIFAR10 with 5 clients and $\alpha = 0.1$.

Dataset	s_q	s_v	Compression	Test Accuracy
FashionMNIST	1	1	–	68.47±2.73
	2	3	6×	65.87±3.42
	4	1.5	6×	68.96±2.71
	6	1	6×	63.90±4.34
CIFAR-10	1	1	–	49.13±1.25
	2	4	8×	45.29±1.30
	4	2	8×	47.60±0.84
	6	1.33	8×	43.53±1.85

D Additional Experimental and Details

D.1 Details on Synthetic Experiment

Our setup consists of $M = 2$ clients and $n = 100$ data points at each client with $\boldsymbol{x} \in \mathbb{R}^2$ and $y \in \mathbb{R}$. The data at each client is generated following a similar procedure as Li et al. (2020a). For each $i \in [M]$, we first sample $w_i \sim \mathcal{N}(0, 1)$, $b_i \sim \mathcal{N}(0, 1)$, $\boldsymbol{w}_i \sim \mathcal{N}(w_i, \boldsymbol{I}) \in \mathbb{R}^2, \boldsymbol{b}_i \sim \mathcal{N}(b_i, \boldsymbol{I}) \in \mathbb{R}^2$. Then for all $j \in [n]$ we have $\tilde{\boldsymbol{x}}_{ij} \sim \mathcal{N}(\boldsymbol{b}_i, \boldsymbol{\Sigma})$, $\boldsymbol{x}_{ij} = \tilde{\boldsymbol{x}}_{ij} / \left\|\tilde{\boldsymbol{x}}_{ij}\right\|_2$ and $y_{ij} = \boldsymbol{w}_i^\top \boldsymbol{x}_{ij}$ where $\boldsymbol{\Sigma} = \text{diag}(1, 2^{-1.2})$.

For our two layer neural network, we initialize the weights in the second layer as $a_r = \frac{1}{\sqrt{m}}$ with probability $1/2$ or $a_r = -\frac{1}{\sqrt{m}}$ otherwise for all $r \in [m]$. We keep the weights in the second layer to be fixed as assumed in our analysis. The weights in the first layer are initialized as $\boldsymbol{w}_r \sim \mathcal{N}\left(0, \frac{1}{2}\right)$ for all $r \in [m]$. We set $\eta = 0.1$ for the local optimization and $\eta_S = 0.001$ for the global optimization. Results in Figure $1(a)$ were averaged over 50 seeds and results in Figure $1(b)$ were averaged over 10 random seeds.

D.2 Experiment on Varying Number of Clients

Figure 3 shows the result our experiment where we keep the heterogeneity to be fixed as $\alpha = 0.3$ and vary the number of clients $M = \{10, 20, 30\}$. Note that as the total dataset size is fixed, as we increase M, each client gets assigned fewer data samples. Thus as M increases, local models have a larger tendency to overfit the data they are trained on, making it harder to aggregate such models to achieve a global model with good performance. Nonetheless we see that `FedFisher` variants, especially `FedFisher(K-FAC)`, continue to outperform baselines even for large M with up to 8% improvement in some cases such as CIFAR-10.

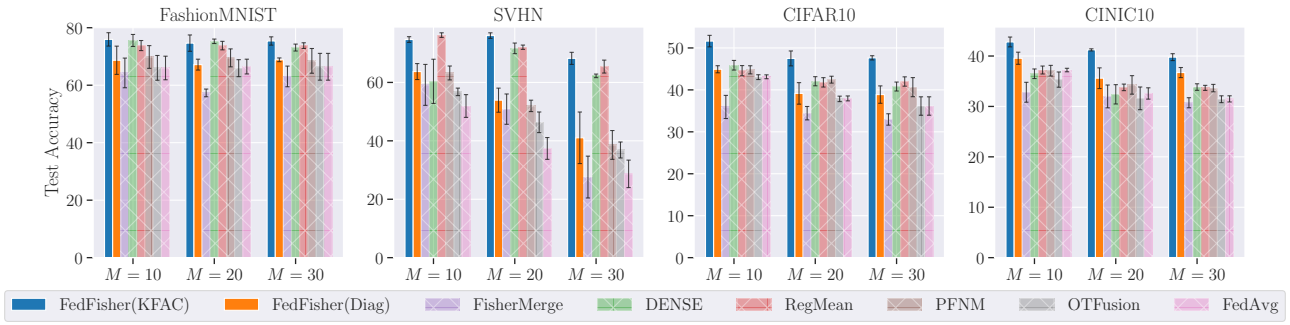


Figure 3: Test accuracy results on different datasets by keeping $\alpha = 0.3$ fixed and varying number of clients M. `FedFisher` variants, especially `FedFisher(K-FAC)`, consistently outperforms other baselines.

D.3 Hyperparameter Details

For hyperparameter tuning we assume that the server has access to a dataset of 500 samples, sampled uniformly from the original training set. We describe the hyperparameters tuned for each of the algorithms below.

`FedFisher(Diag)` and `FedFisher(K-FAC)`. While Algorithm 1 performs the server optimization with GD, this can be replaced with any other suitable GD optimizer. We choose to use the `Adam` optimizer here. We set $\eta_S = 0.01, \beta_1 = 0.9, \beta_2 = 0.99$ and $\epsilon = 0.01$ for the `Adam` optimizer and number of steps $T = 2000$ for all our experiments. We measure the validation performance after every 100 steps and use the model which achieves the best validation performance as the final `FedFisher` model.

`PFNM`. For `PFNM` we use the version applicable for CNNs whose code is available at `https://github.com/IBM/FedMA`. We keep the default values of the hyperparameters $\sigma = 1, \sigma_0 = 1$ and $\gamma = 7$ as in the original code in all our experiments.

`DENSE`. For `DENSE` we use the default settings as described in Section 3.1.4 of Zhang et al. (2022) and official code implementation available at `https://github.com/zj-jayzhang/DENSE`. In particular we set $T_G = 30, \lambda_1 = 1, \lambda_2 = 0.5$ and train the server model with SGD optimizer with learning rate $\eta_S = 0.01$ and momentum factor

0.9. We use the validation data to determine the model which achieves the best validation performance during the server training and use this as the final model.

`OTFusion.` The official code for `OTFusion` available at `https://github.com/sidak/otfusion` contains more than 15 hyperparameters, making it hard to tune each of these parameters. Among these we found that the `correction` and `past-correction` hyperparameter affect performance most and hence we focus on tuning these hyperparameters for our experiments. In particular we vary `correction` $\in \{\text{True}, \text{False}\}$ and `past-correction` $\in \{\text{True}, \text{False}\}$ in our experiments and choose the model which achieves the best validation performance.

`RegMean.` For `RegMean`, we keep all parameters the same as the official code available at `https://github.com/bloomberg/dataless-model-merging` and only tune α in the range $\{0.1, 0.9\}$ using validation data as suggested by the authors.

`FisherMerge.` For `FisherMerge`, we compute the diagonal Fisher at each client using the entire dataset available at each client with a batch size of 1 and set $\lambda_i = p_i$ where p_i is the proportion of data available at client i. We also keep the `fisher-floor` variable to be 10^{-6} as in the official code implementation available at `https://github.com/mmatena/model_merging/tree/master/model_merging`.

D.4 Computation Details

Experiments in Table 2, Table 4, Table 5 and Figure 3 were averaged over 5 seeds and run on a NVIDIA TitanX GPU. Results in Figure 2 and Table 3 were averaged over 3 seeds and run on NVIDIA A100 GPU.

E Measuring Privacy Using Inversion Attacks

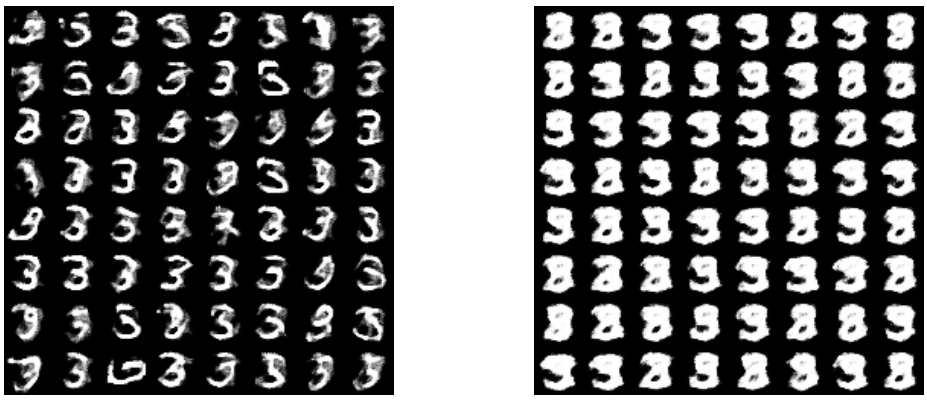
Measuring Privacy using Inversion Attacks. As discussed in Section 4, for `FedFisher(K-FAC)`, while the local models can be securely aggregated, the server does need access to individual K-FAC information at clients to perform the global optimization. We argue, however, that this is more privacy-preserving than having access to the local models which is needed for knowledge distillation and neuron matching baselines. We demonstrate this empirically via the following inversion attack on the MNIST dataset LeCun (1998).

We consider a setup where client 1 has all the images corresponding to label 3 along with 1000 randomly sampled images. The remaining data is then split among 9 clients with $\alpha = 0.1$. The objective at the server is to reconstruct the data at client 1 corresponding to label 3. To do so, we adopt the attack strategy outlined in Zhang et al. (2020), a popular model inversion attack using GANS. We assume that the server has access to auxiliary data of size 5000 that is randomly sampled from the data belonging to clients 2 to 10. Note that this ensures that the server does not have any samples corresponding to label 3. We now consider three cases **(i)** server has access to the local model at client 1 (`DENSE`, `OTFusion`, `PFNM`) **(ii)** server has access to the aggregate of local models (`FedAvg`) and **(iii)** server has access to the aggregate of local model plus the K-FAC computed by client 1 (`FedFisher(K-FAC)`). To modify the attack to include the K-FAC information, we add an additional loss term that penalizes the difference between the K-FAC on the generated data and the given K-FAC. In particular, we use the squared loss between the elements of the generated K-FAC and given K-FAC, summed over all elements.

Table 6 summarizes the attack accuracy in the different cases. Figure 4 shows the generated images for the first and third case. We see that while adding the K-FAC information slightly improves the attack accuracy compared to case **(ii)**, the overall accuracy is still much lower than case **(i)** as also evidenced by the generated images. This hints at the difficulty of inverting the K-FAC to generate meaningful client data, especially when we do not have access to the local models themselves. We acknowledge that there could be more sophisticated attacks designed especially for inverting the K-FAC; however we leave a investigation of such methods to future work.

Table 6: Inversion attack accuracy at server corresponding to different algorithms. While `FedFisher(K-FAC)` improves attack accuracy compared to `FedAvg`, it is still lower than other one-shot baselines.

Algorithm	Attack Accuracy
`OTFusion`, `PFNM`, `DENSE`	$78.4_{\pm 3.0}$
`FedAvg`	$5.0_{\pm 1.0}$
`FedFisher(K-FAC)`	$9.6_{\pm 2.9}$



(a) (b)

Figure 4: Reconstructed images when (a) server has access to the local model at first client and (b) server has access to the global model and K-FAC information of first client. The goal is to generate images corresponding to digit 3.