Towards a Theoretical and Practical Understanding of One-Shot Federated Learning with Fisher Information

Divyansh Jhunjhunwala [1] Shiqiang Wang [2] Gauri Joshi [1]

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 computation 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 the Fisher information matrices computed at the local models of clients, motivated by a Bayesian perspective of FL. First, we theoretically analyze `FedFisher` for two-layer overparameterized 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 several competing baselines.

1. Introduction

Data collection and storage is becoming increasingly decentralized, both due to the proliferation of smart devices as well as privacy concerns stemming from transferring and storing data at a centralized location. Federated Learning (FL) is a framework that is designed to learn the parameters $\boldsymbol{w} \in \mathbb{R}^d$ of a model $f(\boldsymbol{w}, \cdot)$ on decentralized data that is distributed across a network of clients under the supervision of a central server (1; 2; 3). Our focus is particularly on the case where $f(\boldsymbol{w}, \cdot)$ is a neural network as is usually the case in practice. Formally, we formulate FL as the following distributed optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ L(\boldsymbol{w}) := \frac{1}{M}\sum_{i=1}^{M} L_i(\boldsymbol{w}) \right\} \tag{1}$$

where,

$$L_i(\boldsymbol{w}) = \frac{1}{n}\sum_{(\boldsymbol{x}_{ij},\boldsymbol{y}_{ij})\in\mathcal{D}_i} \ell(f(\boldsymbol{w},\boldsymbol{x}_{ij}),\boldsymbol{y}_{ij}). \tag{2}$$

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. The loss function $\ell(\cdot,\cdot)$ penalizes the difference between the prediction of the model $f(\boldsymbol{w}, \boldsymbol{x})$ and 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 data samples across clients.

Most standard FL algorithms such as `FedAvg` (4) and `FedProx` (5) require multiple rounds of communication between clients and server in order to train a global model in federated settings. However, a multi-round approach has several drawbacks for clients, including the need to frequently connect with the server, repeated investment of computational resources to update the global model in every round and increased susceptibility to privacy attacks. 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 works for one-shot FL can be broadly split into two categories: (i) Knowledge Distillation (KD) methods and (ii) Neuron Matching (NM) methods. KD methods treat the collection of client models as an ensemble and propose to distill the knowledge from this ensemble into a single global model. However, to perform the distillation step, these works either assume that the server has access to an auxiliary public dataset (6; 7; 8), or propose to train generative models such as GANs (9; 10) or variational autoen-

[*]Equal contribution [1]Carnegie Mellon University [2]IBM Research. Correspondence to: Divyansh Jhunjhunwala <djhunjhu@andrew.cmu.edu>.

Workshop of Federated Learning and Analytics in Practice, colocated with 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. Copyright 2023 by the author(s).

coders (11) to artificially generate data which raises privacy and computation concerns. NM 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 (12; 13; 14; 15; 16; 17). 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. While this idea has been shown to work well when combining simple models like feedforward NNs, the performance drops significantly for more complex models such as CNNs (18). Lastly, we note that none of these existing works provide any theoretical guarantees for their proposed methods.

Motivated by the limitations of multi-round FL and current approaches for one-shot FL, we ask the question: *Can we devise a one-shot FL method that is simultaneously communication and computation efficient, 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 (2) can alternatively be reformulated as a *posterior inference* problem, specifically finding the mode of a global posterior (19; 20). Some highlights of our contribution are as follows.

- We propose `FedFisher` and show how the problem of finding a mode of the global posterior can be solved approximately in a one-shot manner, using the local models at each client and the Fisher information matrices computed at these local models (Section 2).
- For the case of overparameterized two-layer neural networks, 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 amount of local training at clients increases (Section 3).
- We propose practical variants of `FedFisher` using the diagonal and K-FAC approximation (21) for the full Fisher which we term as `FedFisher(Diag)` and `FedFisher(K-FAC)`. 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 $5-10\%$ accuracy improvement compared to competing KD and NM one-shot baselines (Section 4).

2. Proposed Algorithm: `FedFisher`

We begin by stating 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 cross entropy loss.

Assumption 1. *Given $\boldsymbol{z} = f(\boldsymbol{w},\boldsymbol{x})$, we assume that $\ell(\boldsymbol{z},\boldsymbol{y})$ is the negative log likelihood of $\boldsymbol{y}$ under some exponential family probabilistic model, i.e., $\ell(\boldsymbol{z},\boldsymbol{y}) = -\log \mathbb{P}(\boldsymbol{y}|\boldsymbol{z})$ 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 $\mathbb{P}((\boldsymbol{x},\boldsymbol{y})|\boldsymbol{w}) = q(\boldsymbol{x})\exp(-\ell(f(\boldsymbol{w},\boldsymbol{x}_{ij}),\boldsymbol{y})$ where $q(\cdot)$ is some prior on $\boldsymbol{x}$, independent of $\boldsymbol{w}$. We can now adopt a Bayesian viewpoint and 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 (19)) 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 (2), 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 2 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 algorithm. 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.

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

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

$$\log\mathbb{P}(\boldsymbol{w}|\mathcal{D}_i) \approx \log\mathbb{P}(\tilde{\boldsymbol{w}}_i|\mathcal{D}_i) - \tfrac{n}{2}(\boldsymbol{w}-\boldsymbol{w})^\top\boldsymbol{H}_i(\boldsymbol{w}-\boldsymbol{w}) \quad (3)$$

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 $\tilde{\boldsymbol{w}}_i$.

Approximating Hessian with Fisher. The Fisher information matrix $\boldsymbol{F}_i$ (herein referred to as 'the Fisher') of the local model $\tilde{\boldsymbol{w}}_i$ at client i is defined as follows:

$$\boldsymbol{F}_i = \tfrac{1}{n}\textstyle\sum_{j=1}^{n}\mathbb{E}_{\boldsymbol{y}}\left[\nabla\log\mathbb{P}(\boldsymbol{y}|\boldsymbol{x}_{ij},\boldsymbol{w})\nabla\log\mathbb{P}(\boldsymbol{y}|\boldsymbol{x}_{ij},\boldsymbol{w})^\top\right]_{\boldsymbol{w}=\tilde{\boldsymbol{w}}_i}. \quad (4)$$

Now if $\tilde{\boldsymbol{w}}_i$ fits the data at client i perfectly, i.e., $f(\tilde{\boldsymbol{w}}_i,\boldsymbol{x}_{ij}) = \boldsymbol{y}_{ij}\ \forall j \in [n]$, then it can be shown that $\boldsymbol{H}_i = \boldsymbol{F}_i$ (22; 23). Unlike the Hessian, the Fisher is guaranteed to be positive semi-definite, a condition required for tractable inference at the global server. However, communi-

Algorithm 1 `FedFisher`

```
1:  Input: initial $w_0$, no. of iterations $K$, $T$, client and server step sizes $\eta$ and $\eta_S$ respectively
2:  Global server does:
3:      Communicate $w_0$ to all clients;
4:  Clients $i \in [N]$ in parallel do:
5:      Set $w_i^{(0)} \leftarrow w_0$;
6:      For $k = 0, \dots, K-1$ iterations:
7:          $w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta \nabla L_i(w_i^{(t)})$;
8:      Set $\tilde{w}_i \leftarrow w_i^{(K)}$;
9:      Compute $\tilde{F}_i$;        // Approximation to true Fisher
10:     Communicate $\tilde{w}_i$ and $\tilde{F}_i$ to the server;
11: Global server does:
12:     Set $w^{(0)} = \sum_{i=1}^M \tilde{w}_i / M$;
13:     For $t = 0, \dots, T-1$ iterations:
14:         $w^{(t+1)} \leftarrow w^{(t)} - \eta_S \left( \sum_{i=1}^M \tilde{F}_i w^{(t)} - \sum_{i=1}^M \tilde{F}_i \tilde{w}_i \right)$
```

cating the full Fisher doing 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 practical approximation $\tilde{\boldsymbol{F}}_i$ such as the diagonal Fisher or K-FAC (21) (see Section 4). Thus,

$$\boldsymbol{H}_i \approx \boldsymbol{F}_i \approx \tilde{\boldsymbol{F}}_i. \tag{5}$$

Computing Mode of Global Posterior. Assuming clients compute and send back $\tilde{\boldsymbol{w}}_i$ and $\tilde{\boldsymbol{F}}_i$ to the server, we can use Proposition 2, Equation (3) and Equation (5) to approximate the logarithm of the global posterior as $\log \mathbb{P}(\boldsymbol{w}|\mathcal{D}) \approx \sum_{i=1}^M \log \mathbb{P}(\tilde{\boldsymbol{w}}_i|\mathcal{D}_i) - \frac{n}{2}\sum_{i=1}^M (\boldsymbol{w}-\tilde{\boldsymbol{w}}_i)^\top \tilde{\boldsymbol{F}}_i (\boldsymbol{w}-\tilde{\boldsymbol{w}}_i)$. 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 find that any minimizer $\boldsymbol{w}$ of Equation (6) satisfies $(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i)\boldsymbol{w} = \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{w}}_i$. For overparameterized 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 minimizes $\sum_{i=1}^M \|\boldsymbol{w}-\tilde{\boldsymbol{w}}_i\|_2^2$. Such a constraint prevents the solution from drifting too far away from the local models of each client. Thus, we have,

$$\min_{\boldsymbol{w}\in\mathbb{R}^d : \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \boldsymbol{w} = \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{w}}_i} \tilde{L}(\boldsymbol{w}) = \sum_{i=1}^M \|\boldsymbol{w}-\tilde{\boldsymbol{w}}_i\|_2^2. \tag{7}$$

We refer to the minimizer $\boldsymbol{w}^*$ of the above objective as the `FedFisher` global model in the rest of our discussion. In Lemma 1 in Appendix B, we show that $\boldsymbol{w}^*$ can be easily computed using GD as follows: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta_S \left(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i \boldsymbol{w}^{(t)} - \sum_{i=1}^M \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{w}}_i \right)$ where $\eta_S \leq 1/\lambda_{\max}(\sum_{i=1}^M \tilde{\boldsymbol{F}}_i)$ and $\boldsymbol{w}^{(0)} = \sum_{i=1}^M \tilde{\boldsymbol{w}}_i / M$.

3. Theoretical Analysis for Two Layer Overparameterized ReLU Neural Network

In this section we present a bound on the global training loss i.e, $L(\boldsymbol{w}^*)$, when $f(\boldsymbol{w},\cdot)$ is a two layer overparameterized ReLU neural network. We begin by modeling a two-layer ReLU NN as follows,

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

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. Similar to (24), we consider the 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.

Definition 1. *(Minimum eigenvalue of Gram Matrix (24)) For $(\boldsymbol{x}, y) \in \mathcal{D}$, define matrix $\boldsymbol{H}^\infty \in \mathbb{R}^{N\times N}$ as $\boldsymbol{H}_{kl}^\infty = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\boldsymbol{x}_k^\top \boldsymbol{x}_l \mathbb{I}\left\{\boldsymbol{w}^\top \boldsymbol{x}_k \geq 0\right\}, \mathbb{I}\left\{\boldsymbol{w}^\top \boldsymbol{x}_l\right\} \geq 0\}\right]$ and $\lambda_0 = \lambda_{\min}(\boldsymbol{H}^\infty)$.*

Assumption 2. *(Data normalization and uniqueness) For any $(\boldsymbol{x}, y) \in \mathcal{D}$, we have $\|\boldsymbol{x}\|_2 = 1$ and $|y| \leq C$, where C is some positive constant. For any $(\boldsymbol{x}, y), (\boldsymbol{x}', y') \in \mathcal{D}$ we have $\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \geq \phi > 0$*

Assumption 3. *(Full Fisher) Clients compute and send their full Fisher, i.e, $\tilde{\boldsymbol{F}}_i = \boldsymbol{F}_i$.*

Definition 1 and Assumption 2 are standard in NN optimization literature (24; 25; 26; 27). We note that most commonly used approximations such as the diagonal and K-FAC are primarily empirically motivated and have little theoretical understanding. Therefore, we use Assumption 3 to simplify our analysis and focus on bounding the error introduced by Equation (3) and suboptimality of $\tilde{\boldsymbol{w}}_i$, i.e, $\nabla L_i(\tilde{\boldsymbol{w}}_i) \neq \boldsymbol{0}$.

Theorem 1. *Under Assumptions 2, 3, for $m \geq \mathrm{poly}(N, \lambda_0^{-1}, \delta^{-1}, \kappa^{-1})$, $\ell(z, y) = \frac{1}{2}(z-y)^2$ and i.i.d Gaussian initialization weights of $\boldsymbol{w}_0$ as $\boldsymbol{w}_{0,r} \sim \mathcal{N}(\boldsymbol{0}, \kappa\boldsymbol{I})$, and initializing $a_r = \{-1, 1\}$ 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 $1-\delta$*

$$L(\boldsymbol{w}^*) \leq \underbrace{\mathcal{O}\left((1-\eta\lambda_0/2)^K \frac{N}{\delta}\right)}_{\textit{local optimization error}} \tag{9}$$

$$+ \underbrace{\mathcal{O}\left((2-(1-\eta\lambda_0/2)^K)\frac{\mathrm{poly}(N, \kappa^{-1}, \lambda_0^{-1}, \delta^{-1})}{m}\right)}_{\textit{Laplace approximation error}}.$$

Takeaways. Theorem 2 shows that for sufficiently wide networks, $L(\boldsymbol{w}^*)$ can be decomposed into the local optimization error, which measures how well $\tilde{\boldsymbol{w}}_i$ fits the data at

Table 1: Test accuracy results on different datasets by keeping number of client $M = 5$ fixed and varying heterogeneity parameter α. Practical implementations of `FedFisher` consistently outperforms other baselines.

Dataset	Heterogeneity α	`FedAvg`	`PFNM`	`OTFusion`	`DENSE`	`FedFisher (Diag)`	`FedFisher (K-FAC)`
MNIST	0.05	49.33 ± 4.16	71.32 ± 3.00	44.18 ± 0.29	53.83 ± 1.53	66.88 ± 1.21	$\mathbf{78.27} \pm 3.09$
	0.075	55.90 ± 3.47	71.74 ± 2.62	49.85 ± 0.02	60.20 ± 0.50	71.40 ± 3.02	$\mathbf{83.15} \pm 2.20$
	0.1	55.67 ± 2.83	70.02 ± 6.06	50.40 ± 0.04	61.67 ± 1.29	69.24 ± 3.64	$\mathbf{84.43} \pm 1.63$
FashionMNIST	0.05	36.93 ± 3.92	50.86 ± 3.62	37.47 ± 0.22	44.17 ± 1.11	$\mathbf{54.40} \pm 2.77$	54.09 ± 3.18
	0.075	41.02 ± 3.68	48.97 ± 5.84	40.87 ± 0.77	47.59 ± 2.04	64.88 ± 3.32	$\mathbf{67.96} \pm 4.35$
	0.1	46.45 ± 1.47	52.36 ± 5.39	42.39 ± 0.39	53.22 ± 1.55	66.52 ± 4.90	$\mathbf{71.56} \pm 4.35$
SVHN	0.05	27.03 ± 0.71	45.62 ± 3.55	34.89 ± 0.77	49.06 ± 1.40	51.52 ± 2.92	$\mathbf{57.40} \pm 3.68$
	0.075	33.71 ± 2.34	51.15 ± 3.43	48.53 ± 0.67	57.39 ± 3.08	53.28 ± 7.48	$\mathbf{62.95} \pm 1.61$
	0.1	39.42 ± 2.52	53.19 ± 4.18	51.66 ± 0.26	55.93 ± 1.79	64.24 ± 3.07	$\mathbf{69.01} \pm 2.68$
CIFAR-10	0.05	29.75 ± 1.32	30.58 ± 3.65	27.00 ± 0.36	31.93 ± 2.70	31.12 ± 1.49	$\mathbf{40.39} \pm 3.51$
	0.075	34.72 ± 1.63	37.03 ± 3.72	34.69 ± 1.16	37.20 ± 2.50	39.44 ± 2.47	$\mathbf{44.07} \pm 0.71$
	0.1	36.43 ± 2.51	39.43 ± 1.80	39.72 ± 1.07	38.29 ± 2.61	40.00 ± 1.35	$\mathbf{47.58} \pm 1.11$
CINIC-10	0.05	10.01 ± 2.20	$\times$	8.08 ± 0.59	15.52 ± 0.80	11.56 ± 3.50	$\mathbf{18.12} \pm 1.97$
	0.075	12.13 ± 2.01	$\times$	8.73 ± 2.01	$\mathbf{20.53} \pm 1.84$	15.76 ± 1.79	20.39 ± 1.50
	0.1	12.68 ± 1.30	$\times$	10.97 ± 1.38	20.75 ± 1.17	16.92 ± 1.88	$\mathbf{22.66} \pm 3.00$

Table 2: Test accuracy results on different datasets by keeping heterogeneity parameter $\alpha = 0.3$ fixed and varying number of clients M. Practical implementations of `FedFisher` consistently outperforms other baselines.

Dataset	# Clients M	`FedAvg`	`PFNM`	`OTFusion`	`DENSE`	`FedFisher (Diag)`	`FedFisher (K-FAC)`
MNIST	10	83.68 ± 1.90	79.84 ± 4.80	56.46 ± 0.17	88.95 ± 0.65	91.88 ± 1.29	$\mathbf{91.32 \pm 0.29}$
	20	81.05 ± 2.33	62.44 ± 6.22	55.55 ± 0.26	88.08 ± 0.59	89.00 ± 1.29	$\mathbf{89.78} \pm 1.18$
	30	81.61 ± 2.85	49.88 ± 5.87	52.20 ± 0.23	87.12 ± 0.16	$\mathbf{90.08} \pm 0.72$	88.94 ± 1.35
FashionMNIST	10	75.20 ± 2.96	63.08 ± 8.41	50.86 ± 1.00	78.83 ± 1.68	$\mathbf{82.40} \pm 1.29$	78.17 ± 3.40
	20	74.61 ± 2.96	46.94 ± 6.11	49.45 ± 0.82	$\mathbf{79.54 \pm 1.54}$	77.84 ± 2.35	78.73 ± 2.46
	30	75.24 ± 2.28	40.93 ± 9.89	48.75 ± 1.24	77.52 ± 0.88	76.16 ± 1.38	$\mathbf{78.03} \pm 2.46$
SVHN	10	51.92 ± 3.89	63.24 ± 2.37	49.44 ± 0.65	63.27 ± 2.89	63.72 ± 2.73	$\mathbf{74.63} \pm 0.75$
	20	37.39 ± 3.73	51.96 ± 1.95	42.56 ± 3.30	57.34 ± 2.06	53.88 ± 4.16	$\mathbf{76.25} \pm 0.97$
	30	28.69 ± 4.70	38.59 ± 4.93	33.06 ± 1.24	64.56 ± 1.72	41.04 ± 8.82	$\mathbf{68.55} \pm 1.74$
CIFAR-10	10	43.12 ± 0.46	44.78 ± 0.96	35.27 ± 0.42	46.44 ± 1.02	44.92 ± 0.84	$\mathbf{51.92} \pm 1.18$
	20	37.96 ± 0.58	42.43 ± 0.81	29.47 ± 1.26	40.46 ± 2.09	39.16 ± 2.56	$\mathbf{48.07 \pm 1.61}$
	30	36.16 ± 2.19	40.65 ± 2.27	27.54 ± 1.36	40.74 ± 1.14	38.88 ± 2.07	$\mathbf{48.29} \pm 0.85$
CINIC-10	10	15.00 ± 2.40	$\times$	8.75 ± 2.22	19.85 ± 4.86	16.04 ± 2.22	$\mathbf{25.01} \pm 0.61$
	20	14.86 ± 0.63	$\times$	7.49 ± 2.04	22.62 ± 2.43	15.04 ± 2.03	$\mathbf{24.96} \pm 0.96$
	30	14.07 ± 0.92	$\times$	7.87 ± 0.23	19.48 ± 2.12	14.68 ± 0.83	$\mathbf{24.72} \pm 0.78$

the i-th client, and the error introduced by the Laplace approximation in Equation (3). 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 $\boldsymbol{w}^*$. Note by setting m and K appropriately, the error can be made arbitrarily small. In contrast, a similar bound for `FedAvg` has a fixed $\mathcal{O}(1)$ bound on the error (28; 29).

4. Practical Variants of `FedFisher` and Experiments

Two of the most popular approximations for the Fisher are the diagonal Fisher and the Kronecker Factored Approximate Curvature (K-FAC). Using these approximations as a substitute for the true Fisher in Equation (5), 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 privacy properties in Appendix C. In particular, we ensure that the communication cost of these variants matches that of `FedAvg`. Here we evaluate the performance of `FedFisher(Diag)` and `FedFisher(K-FAC)` in comparison to state-of-the-art (SOTA) one-shot FL baselines across a range of image recognition tasks in a FL setting. The datasets that we use are (i) MNIST (30), (ii) FashionMNIST (31) (iii) SVHN (32) (iv) CIFAR-10 (33) and (v) CINIC-10 (34).

Baselines. `FedAvg` is the de-facto baseline in all our experiments. The other baselines that we compare with are (i) `PFNM` (ii) `OTFusion` (12) and (iii) `DENSE` (9). `PFNM` and `OTFusion` are SOTA neuron matching methods that first permute the weights of the local models and then average them, instead of direct averaging. `DENSE`, is a SOTA method for one-shot FL using data-free knowledge distillation that uses GANs to artificially generate data for distillation at the server. We avoid comparing with baselines that need auxiliary data or maintain an ensemble of local models (35; 36) at the server to ensure fairness of comparison.

Models and Experimental Setup. For MNIST and FashionMNIST we use the MLPNet architecture proposed in (12); for SVHN and CIFAR10 we use a CNN model proposed in (18); for CINIC-10 we use ResNet-18 (37). We note that `PFNM` does not currently support ResNet18 architecture. To simulate data heterogeneity among the client datasets, we split our original image dataset into N partitions using a Dirichlet sampling procedure with parameter α, as is common in FL literature (38; 39), with a lower value of α implying a more heterogeneous split. Further details can be found in Appendix D.

`FedFisher` outperforms baselines across varying heterogeneity and number of clients. We evaluate `FedFisher` and other baselines in the range of $\alpha = \{0.05, 0.075, 0.1\}$ which can be considered as moderate to high data heterogeneity. Table 1 summarizes the results obtained the algorithms across the various datasets. We see that `FedFisher` variants comprehensively outperforms baselines in all regimes. In particular for MNIST, FashionMNIST and SVHN, `FedFisher(K-FAC)` gives almost 10% improvement compared to the nearest baseline. Table 2 summarizes the results obtained when we keep the heterogeneity parameter $\alpha = 0.3$ fixed and vary the number of clients M in the range $\{10, 20, 30\}$. Note that since 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. Nonetheless, we see that `FedFisher` variants, especially `FedFisher(K-FAC)` continue to outperform other baselines with up to 20% improvement in some cases like the SVHN and relatively lower drop in accuracy with larger M. This highlights the effectiveness of `FedFisher` as a one-shot algorithm which can tackle data heterogeneity in FL settings and scale to relatively large number of clients while being computation and communication efficient.

5. Conclusion

In this work, we propose `FedFisher`, 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 practical versions `FedFisher(Diag)` and `FedFisher(K-FAC)` that outperform current state-of-the-art one-shot methods while being computation and communication efficient. Future work involves extending the analysis of `FedFisher` for deeper neural networks and investigating the use of differential-privacy to improve privacy guarantees.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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 2020.

[6] 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.

[7] 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.

[8] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*, 2020.

[9] 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.

[10] 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.

[11] 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*.

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

[13] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

[14] 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.

[15] 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.

[16] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*, 2022.

[17] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[18] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[19] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. *arXiv preprint arXiv:2010.05273*, 2020.

[20] Han Guo, Philip Greengard, Hongyi Wang, Andrew Gelman, Yoon Kim, and Eric P Xing. Federated learning as variational inference: A scalable expectation propagation approach. *arXiv preprint arXiv:2302.04228*, 2023.

[21] 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.

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

[23] 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.

[24] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[25] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[26] 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, 2019.

[27] 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, 2019.

[28] 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.

[29] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Local sgd optimizes overparameterized neural networks in polynomial time. In *International Conference on Artificial Intelligence and Statistics*, pages 6840–6861. PMLR, 2022.

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

[31] 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.

[32] 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.

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. *CIFAR-10 (Canadian Institute for Advanced Research)*, 2009.

[34] 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.

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

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

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[38] 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.

[39] 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.

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

[41] 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.

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

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] Thomas George. Nngeometry: Easy and fast fisher information matrices and neural tangent kernels in pytorch. 2021.

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 (35) propose to just use the ensemble of client models at the server. The work (36) 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 (40; 41). However, the privacy guarantees of such methods is unclear. The work (42) 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 (43; 44); 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 (28; 45; 46) 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 (47; 48), that proposes to use NTK style Jacobian features to speed up FL training; however these works usually require multiple training rounds.

B. Proofs

B.1. Proof of Proposition 2.

Proposition 2. *(Global Posterior Decomposition (19)) 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 (2), 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,

$$\begin{aligned}
\mathbb{P}(\boldsymbol{w}|\mathcal{D}) &\propto \mathbb{P}(\mathcal{D}|\boldsymbol{w}) && (\because \mathbb{P}(\boldsymbol{w}) \propto 1) && (10)\\
&= \prod_{i=1}^{M} \mathbb{P}(\mathcal{D}_i|\boldsymbol{w}) && (\mathcal{D}_i \text{ are i.i.d generated}) && (11)\\
&\propto \prod_{i=1}^{M} \mathbb{P}(\boldsymbol{w}|\mathcal{D}_i). && && (12)
\end{aligned}$$

Also,

$$\begin{aligned}
\arg\max_{\boldsymbol{w}\in\mathbb{R}^d} \mathbb{P}(\boldsymbol{w}|\mathcal{D}) &= \arg\max_{\boldsymbol{w}\in\mathbb{R}^d} \log \mathbb{P}(\boldsymbol{w}|\mathcal{D}) && && (13)\\
&= \arg\max_{\boldsymbol{w}\in\mathbb{R}^d} \log \mathbb{P}(\mathcal{D}|\boldsymbol{w}) && && (14)\\
&= \arg\max_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{i=1}^{M}\sum_{j=1}^{n} (\log q(\boldsymbol{x}_{ij}) - \ell(f(\boldsymbol{w}, \boldsymbol{x}_{ij}), \boldsymbol{y}_{ij})) && (\text{Assumption } 1) && (15)\\
&= \arg\max_{\boldsymbol{w}\in\mathbb{R}^d} -\sum_{i=1}^{M}\sum_{j=1}^{n} \ell(f(\boldsymbol{w}, \boldsymbol{x}_{ij}), \boldsymbol{y}_{ij}) && (\text{Assumption } 1) && (16)\\
&= \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} L(\boldsymbol{w}). && && (17)
\end{aligned}$$

□

B.2. Lemma 1 and Proof.

Lemma 1. *Let $\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots$ be the iterates generated by running the following gradient descent (GD) procedure: $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_S \left(\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \boldsymbol{w}^{(t)} - \tilde{\boldsymbol{F}}_i \boldsymbol{w}_i\right)$ 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 eigen value of $\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i$. Then, $\lim_{T\to\infty} \boldsymbol{w}^{(T)} = \boldsymbol{w}^*$.*

Proof.

Recall,

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ \tilde{L}(\boldsymbol{w}) = \sum_{i=1}^{M} \|\boldsymbol{w} - \tilde{\boldsymbol{w}}_i\|_2^2 \text{ such that } \left(\sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i\right) \boldsymbol{w} = \sum_{i=1}^{M} \tilde{\boldsymbol{F}}_i \tilde{\boldsymbol{w}}_i \right\} \tag{18}$$

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

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ \tilde{L}(\boldsymbol{w}) = \|\boldsymbol{w} - \bar{\boldsymbol{w}}\|_2 \text{ such that } \boldsymbol{F}\boldsymbol{w} = \boldsymbol{b} \right\}. \tag{19}$$

Since $\boldsymbol{F}$ is symmetric and PSD (each $\tilde{\boldsymbol{F}}_i$ is symmetric and PSD), we have by the spectral decomposition of $\boldsymbol{F}$,

$$\boldsymbol{F} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^\top \\ \boldsymbol{V}_2^\top \end{bmatrix} = \boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top. \tag{20}$$

Here $\boldsymbol{V} \in \mathbb{R}^{(d\times d)}$ is an orthogonal matrix consisting of the eigenvectors of $\boldsymbol{F}$, and $\boldsymbol{\Sigma} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r, 0, \ldots, 0)$ is a diagonal matrix consisting of the eigen values of $\boldsymbol{F}$. We assume that $\mathrm{rank}(\boldsymbol{F}) = r$, therefore $\boldsymbol{V}_1 \in \mathbb{R}^{d\times r}$ consists of the first r eigenvectors of $\boldsymbol{F}$ and $\boldsymbol{\Sigma}_1 = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$. We also assume for our discussion that $\boldsymbol{b}$ lies in the column space of $\boldsymbol{F}$, i.e., $\boldsymbol{V}_1\boldsymbol{V}_1^\top\boldsymbol{b} = \boldsymbol{b}$.

Claim 1. *The set of* $\mathbf{w}$ *such that* $\boldsymbol{F}\boldsymbol{w} = \boldsymbol{b}$ *is given by,*

$$\mathcal{S} = \{\boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{w} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} | \boldsymbol{w} \in \mathbb{R}^d\}. \tag{21}$$

Proof. Let $\boldsymbol{w} = \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{x} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}$ for some $\boldsymbol{x} \in \mathbb{R}^d$. Firstly we see that,

$$\boldsymbol{F}\boldsymbol{w} = \boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top(\boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{x} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}) \tag{22}$$

$$= \boldsymbol{b} \tag{23}$$

Now let $\boldsymbol{w}^\diamond$ be such that $\boldsymbol{F}\boldsymbol{w}^\diamond = \boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top\boldsymbol{w}^\diamond = \boldsymbol{b}$. This implies $\boldsymbol{V}_1^\top\boldsymbol{w}^\diamond = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}$. We have,

$$\boldsymbol{w}^\diamond = \boldsymbol{V}\boldsymbol{V}^\top\boldsymbol{w}^\diamond \tag{24}$$

$$= \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{w}^\diamond + \boldsymbol{V}_1\boldsymbol{V}_1^\top\boldsymbol{w}^\diamond \tag{25}$$

$$= \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{w}^\diamond + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} \tag{26}$$

Combining Equation (26), Equation (26) we have,

$$\boldsymbol{w} \text{ such that } \boldsymbol{F}\boldsymbol{w} = \boldsymbol{b} \iff \boldsymbol{w} \in \mathcal{S}. \tag{27}$$

□

Claim 2.

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathcal{S}} \|\boldsymbol{w} - \bar{\boldsymbol{w}}\|_2^2 = \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} \tag{28}$$

Proof.

Case 1: $\bar{\boldsymbol{w}} \in \mathcal{S}$. In this case we have, $\boldsymbol{w}^* = \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} = \bar{\boldsymbol{w}}$, therefore $\arg\min_{\boldsymbol{w}\in\mathcal{S}} \|\boldsymbol{w} - \bar{\boldsymbol{w}}\|_2^2 = 0$.

Case 2: $\bar{\boldsymbol{w}} \notin \mathcal{S}$. Suppose $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathcal{S}} \|\boldsymbol{w} - \bar{\boldsymbol{w}}\|_2^2 = \boldsymbol{V}_2\boldsymbol{V}_2^\top\widetilde{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}$ for some $\widetilde{\boldsymbol{w}} \neq \bar{\boldsymbol{w}}$. Let $\boldsymbol{w}^\diamond = \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}$. We have,

$$\|\bar{\boldsymbol{w}} - \boldsymbol{w}^*\|_2^2 = \left\|\bar{\boldsymbol{w}} - \boldsymbol{V}_2\boldsymbol{V}_2^\top\widetilde{\boldsymbol{w}} - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}\right\|_2^2 \tag{29}$$

$$= \left\|\boldsymbol{V}_2\boldsymbol{V}_2^\top(\bar{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}) + \boldsymbol{V}_1\boldsymbol{V}_1^\top\bar{\boldsymbol{w}} - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}^\top\boldsymbol{b}\right\|_2^2 \tag{30}$$

$$= \left\|\boldsymbol{V}_2\boldsymbol{V}_2^\top(\bar{\boldsymbol{w}} - \widetilde{\boldsymbol{w}})\right\|_2^2 + \left\|\boldsymbol{V}_1\boldsymbol{V}_1^\top\bar{\boldsymbol{w}} - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}^\top\boldsymbol{b}\right\|_2^2 \tag{31}$$

$$= \left\|\boldsymbol{V}_2\boldsymbol{V}_2^\top(\bar{\boldsymbol{w}} - \widetilde{\boldsymbol{w}})\right\|_2^2 + \left\|\boldsymbol{V}_1\boldsymbol{V}_1^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} - \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}^\top\boldsymbol{b}\right\|_2^2 \tag{32}$$

$$= \left\|\boldsymbol{V}_2\boldsymbol{V}_2^\top(\bar{\boldsymbol{w}} - \widetilde{\boldsymbol{w}})\right\|_2^2 + \|\bar{\boldsymbol{w}} - \boldsymbol{w}^\diamond\|_2^2 \tag{33}$$

$$> \|\bar{\boldsymbol{w}} - \boldsymbol{w}^\diamond\|_2^2 \tag{34}$$

leading to a contradiction. Thus, $\boldsymbol{w}^* = \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b}$. □

Now the GD step in Lemma 1 can be written as,

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_S(\boldsymbol{F}\boldsymbol{w}^{(t)} - \boldsymbol{b}) \tag{35}$$

$$= (\boldsymbol{I} - \eta_S\boldsymbol{F})\boldsymbol{w}^{(t)} + \eta_S\boldsymbol{b}. \tag{36}$$

Therefore,

$$\boldsymbol{w}^{(T)} = (\boldsymbol{I} - \eta_S\boldsymbol{F})^T\boldsymbol{w}^{(0)} + \eta_S\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{F})^t\boldsymbol{b} \tag{37}$$

$$= \boldsymbol{V}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma})^T\boldsymbol{V}^\top\boldsymbol{w}^{(0)} + \eta_S\sum_{t=0}^{T-1}\boldsymbol{V}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma})^t\boldsymbol{V}^\top\boldsymbol{b} \tag{38}$$

$$= (\boldsymbol{V}_1(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{V}_2^\top)\boldsymbol{w}^{(0)} + \eta_S\left(\boldsymbol{V}_1\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{V}_2^\top\right)\boldsymbol{b}. \tag{39}$$

$$= (\boldsymbol{V}_1(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{V}_2^\top)\boldsymbol{w}^{(0)} + \eta_S\left(\boldsymbol{V}_1\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t\boldsymbol{V}_1\right)\boldsymbol{b}. \tag{40}$$

$$\tag{41}$$

where the last line uses the fact $\boldsymbol{b}$ lies in the column space of $\boldsymbol{F}$.

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

$$\lim_{T\to\infty}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^T = \boldsymbol{0} \text{ and } \lim_{T\to\infty}\sum_{t=0}^{T-1}(\boldsymbol{I} - \eta_S\boldsymbol{\Sigma}_1)^t = \frac{1}{\eta_S}\boldsymbol{\Sigma}_1^{-1}. \tag{42}$$

Thus,

$$\lim_{T\to\infty}\boldsymbol{w}^{(T)} = \boldsymbol{V}_2\boldsymbol{V}_2^\top\boldsymbol{w}^{(0)} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} \tag{43}$$

$$= \boldsymbol{V}_2\boldsymbol{V}_2^\top\bar{\boldsymbol{w}} + \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}_1^\top\boldsymbol{b} \tag{44}$$

$$= \boldsymbol{w}^* \tag{45}$$

which the last line follows from Equation (21) and Equation (28). This completes our proof. □

B.3. Proofs for 2 layer Overparameterized ReLU NN

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

$$f(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\boldsymbol{x}^\top\boldsymbol{w}_r\mathbb{I}\left\{\boldsymbol{x}^\top\boldsymbol{w}_r \geq 0\right\} \tag{46}$$

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

$$f(\boldsymbol{w},\boldsymbol{x}) = \phi(\boldsymbol{w},\boldsymbol{x})^\top\boldsymbol{w} \tag{47}$$

where $\phi(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{\sqrt{m}}[a_1\boldsymbol{x}\mathbb{I}\left\{\boldsymbol{x}^\top\boldsymbol{w}_1 \geq 0\right\}; a_2\boldsymbol{x}\mathbb{I}\left\{\boldsymbol{x}^\top\boldsymbol{w}_2 \geq 0\right\}; \ldots a_m\boldsymbol{x}\mathbb{I}\left\{\boldsymbol{x}^\top\boldsymbol{w}_m \geq 0\right\}] \in \mathbb{R}^{mp}$.

We see that,

$$\nabla_{\boldsymbol{w}}f(\boldsymbol{w},\boldsymbol{x}) = \phi(\boldsymbol{w},\boldsymbol{x}) \tag{48}$$

Also observe that,

$$\|\phi(\boldsymbol{w},\boldsymbol{x})\|_2^2 \leq \frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{x}\|_2^2 = 1 \tag{49}$$

Definition of $\boldsymbol{A}_{\boldsymbol{w}}$, $\tilde{\boldsymbol{A}}$, $\boldsymbol{H}_{\boldsymbol{w}}$, $\tilde{\boldsymbol{H}}$. We define the following matrices $\boldsymbol{A}_{\boldsymbol{w}}$ and $\tilde{\boldsymbol{A}}$ that will be used in the proof.

$$\boldsymbol{A}_{\boldsymbol{w}} = \begin{bmatrix} \phi^\top(\boldsymbol{w}, \boldsymbol{x}_{11}) \\ \vdots \\ \phi^\top(\boldsymbol{w}, \boldsymbol{x}_{Mn}) \end{bmatrix} \in \mathbb{R}^{(N \times mp)} \tag{50}$$

and,

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \phi^\top(\tilde{\boldsymbol{w}}_1, \boldsymbol{x}_{11}) \\ \vdots \\ \phi^\top(\tilde{\boldsymbol{w}}_M, \boldsymbol{x}_{Mn}) \end{bmatrix} \in \mathbb{R}^{(N \times mp)} \tag{51}$$

We also define $\boldsymbol{H}_{\boldsymbol{w}} = \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{A}_{\boldsymbol{w}}^\top$, $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{A}}^\top$.

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.

Simplification of Fisher. Under the squared loss $\ell(z, y) = \frac{1}{2}(y - z)^2$, it can be shown that the Fisher at each client **??** can be simplified as follows (22),

$$\boldsymbol{F}_i = \frac{1}{n} \sum_{j=1}^n [\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x}_{ij}) \nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x}_{ij})^\top]_{\boldsymbol{w} = \tilde{\boldsymbol{w}}_i} \tag{52}$$

$$= \frac{1}{n} \sum_{j=1}^n \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij}) \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top \qquad \text{(using Equation (48))} \tag{53}$$

True Model and Proxy Model Output: Let $\boldsymbol{f}(\boldsymbol{w}) = [f(\boldsymbol{w}, \boldsymbol{x}_{11}), f(\boldsymbol{w}), \boldsymbol{x}_{12}), \ldots, f(\boldsymbol{w}), \boldsymbol{x}_{Mn})]$ be the vector of true model responses. Let $\boldsymbol{y} = [y_{11}, y_{12}, \ldots, y_{Mn}]$ be the vector of true labels and let $\tilde{\boldsymbol{y}} = [f(\tilde{\boldsymbol{w}}_1, \boldsymbol{x}_{11}), f(\tilde{\boldsymbol{w}}_1, \boldsymbol{x}_{12}), \ldots, f(\tilde{\boldsymbol{w}}_M, \boldsymbol{x}_{Mn})]$ be the vector of predicted local model outputs.

For any $\boldsymbol{x}_{ij}$ and any weight vector $\boldsymbol{w}$, the proxy model output is defined as $\tilde{f}(\boldsymbol{w}, \boldsymbol{x}_{ij}) = \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top \boldsymbol{w}$. Let $\tilde{\boldsymbol{f}}(\boldsymbol{w}) = [\tilde{f}(\boldsymbol{w}, \boldsymbol{x}_{11}), \tilde{f}(\boldsymbol{w}, \boldsymbol{x}_{12}), \ldots, \tilde{f}(\boldsymbol{w}, \boldsymbol{x}_{Mn})]$ be the vector of proxy Fisher model responses.

We will now state some lemmas from previous work that we utilize in our proof.

Lemma 2. *(Theorem 3.1 and Lemma C.1 in (49)) 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*

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

Note that (49) provides the above guarantees for a single machine setting. In order for the above result to hold simultaneously for all $i \in [M]$, we set the failure probability $\delta' = \delta/M$, leading m to have a polynomial dependence on N instead of n.

Lemma 3. *For a given* R, *define the following event:*

$$A_{ijr}(R) = \{\exists \boldsymbol{w} : \|\boldsymbol{w} - \boldsymbol{w}_{0,r}\|_2 \leq R, \mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \geq 0\} \neq \mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w} \geq 0\}\} \tag{54}$$

We have $\Pr(A_{ijr}(R)) \leq \frac{2R}{\sqrt{2\pi}\kappa}$.

Lemma 4. *(Lemma 3.1 in (24)). If* $m = \Omega\left(\frac{N^2}{\lambda_0^2} \log \frac{N}{\delta}\right)$, *we have with probability* $1 - \delta$, $\|\boldsymbol{H}_{\boldsymbol{w}_0} - \boldsymbol{H}^\infty\|_2 \leq \frac{\lambda_0}{4}$ *and* $\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{w}_0}) \geq \frac{3\lambda_0}{4}$.

The following three lemmas 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 $\tilde{\boldsymbol{H}}$ is given by,

$$\tilde{\boldsymbol{H}}_{kl} = \frac{1}{m}\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{x}_{l'l''}\sum_{r=1}^{m}\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{k',r} \geq 0\right\}\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{l',r} \geq 0\right\} \tag{55}$$

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

We have,

$$\mathbb{E}\left[|\tilde{\boldsymbol{H}}_{kl} - (\boldsymbol{H}_{\boldsymbol{w}^{(0)}})_{kl}|\right] \tag{56}$$

$$= \mathbb{E}\left[\frac{1}{m}\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{x}_{l'l''}\sum_{r=1}^{m}\mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{k',r} \geq 0\right\}\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \tag{57}$$

$$\leq \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m}\mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{k',r} \geq 0\right\}\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \tag{58}$$

$$\leq \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m}\mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{k',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{k'k''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\right\} + \mathbb{I}\left\{\mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{l',r} \geq 0\right\} \neq \mathbb{I}\left\{\boldsymbol{x}_{l'l''}^{\top}\boldsymbol{w}_{0,r} \geq 0\right\}\right\}\right] \tag{59}$$

$$\leq \mathbb{E}\left[\frac{1}{m}\sum_{r=1}^{m}\mathbb{I}\left\{A_{k'k''r}(R_0)\right\} + \mathbb{I}\left\{A_{l'l''r}(R_0)\right\}\right] \tag{60}$$

$$= \frac{1}{m}\sum_{r=1}^{m}\Pr(A_{k'k''r}(R_0)) + \Pr(A_{l'l''r}(R_0)) \tag{61}$$

$$\leq \frac{4R_0}{\sqrt{2\pi}\kappa} \tag{62}$$

Equation (58) uses Cauchy-Schwartz and Assumption 2, Equation (59) follows from the definition of A_{ijr} in Lemma 3, Equation (62) uses the result in Lemma 3.

Thus we have,

$$\mathbb{E}\left[\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_{\boldsymbol{w}^{(0)}}\|_F\right] \leq \mathbb{E}\left[\sum_{k,l}|\tilde{\boldsymbol{H}}_{kl} - (\boldsymbol{H}_{\boldsymbol{w}^{(0)}})_{kl}|\right] \leq \frac{4N^2R}{\sqrt{2\pi}} \tag{63}$$

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

$$\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_{\boldsymbol{w}^{(0)}}\|_F \leq \frac{4N^2R_0}{\sqrt{2\pi}\delta\kappa} \tag{64}$$

Thus,

$$\left\|\tilde{\boldsymbol{H}} - \boldsymbol{H}_{\boldsymbol{w}^{(0)}}\right\|_2 \leq \|\tilde{\boldsymbol{H}} - \boldsymbol{H}_{\boldsymbol{w}^{(0)}}\|_F \leq \frac{4N^2R_0}{\sqrt{2\pi}\delta\kappa} \tag{65}$$

This implies,

$$\lambda_{\min}(\tilde{\boldsymbol{H}}) \geq \lambda_{\min}(\boldsymbol{H}_{\boldsymbol{w}^{(0)}}) - \frac{4N^2R_0}{\sqrt{2\pi}\delta\kappa} \geq \frac{\lambda_0}{2} \tag{66}$$

where the last line follows from definition of R_0 in Lemma 2 and $m = \Omega\left(\frac{N^6}{\lambda_0^4\delta^3\kappa^2}\right)$.

We also have,

$$\left\|\tilde{\boldsymbol{H}}\right\|_2 = \left\|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top\right\|_2 \tag{67}$$

$$= \left\|\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}}\right\|_2 \tag{68}$$

$$= \left\|\sum_{i=1}^{M}\sum_{j=1}^{n}\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top\right\|_2 \tag{69}$$

$$\leq \sum_{i=1}^{M}\sum_{j=1}^{n}\left\|\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top\right\|_2 \tag{70}$$

$$\leq N\max_{i,j}\|\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})\|_2 \tag{71}$$

$$\leq N \tag{72}$$

Equation (70) follows from the triangle inequality, for Equation (72) we use Equation (49) and Assumption 2.

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

Lemma 6. *(Theorem 3.1 and Lemma C.1 in (49)) 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 $\mathbf{w}_0$, we have*

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

Proof.

First part.

We have,

$$(\tilde{f}(\boldsymbol{w}^{(0)}, \boldsymbol{x}_{ij}) - \tilde{y}_{ij})^2 = (\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top\boldsymbol{w}^{(0)} - \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top\tilde{\boldsymbol{w}}_i)^2 \tag{73}$$

$$= \left(\frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\right\}\boldsymbol{x}_{ij}^\top\left(\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right)\right)^2 \tag{74}$$

$$\leq \sum_{r=1}^{m}\left(a_r\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top\tilde{\boldsymbol{w}}_{i,r}\right\}\boldsymbol{x}_{ij}^\top\left(\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right)\right)^2 \tag{75}$$

$$\leq \sum_{r=1}^{m}\left\|\sum_{l=1}^{M}\tilde{\boldsymbol{w}}_{l,r}/M - \tilde{\boldsymbol{w}}_{i,r}\right\|_2^2 \tag{76}$$

$$\leq 4mR_0^2 \tag{77}$$

$$= \mathcal{O}\left(\frac{n^2}{\lambda_0^2\delta}\right) \tag{78}$$

where Equation (75) follows from Jensen's inequality, Equation (76) uses Cauchy-Schwartz and Assumption 2, Equation (78) follows from Lemma 2.

Thus, $\left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(0)}) - \tilde{\boldsymbol{y}}\right\|_2^2 = \mathcal{O}\left(\frac{N}{\delta}\right) \leq \mathcal{O}\left(\frac{Nn^2}{\lambda_0^2\delta}\right)$

Second part.

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

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_S \sum_{i=1}^{M} \boldsymbol{F}_i \left(\boldsymbol{w}^{(t)} - \tilde{\boldsymbol{w}}_i\right) \tag{79}$$

$$= \boldsymbol{w}^{(t)} - \tilde{\eta_S} \sum_{i=1}^{M}\sum_{j=1}^{n} \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})\phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top \left(\boldsymbol{w}^{(t)} - \tilde{\boldsymbol{w}}_i\right) \qquad (\tilde{\eta_S} = \eta_S/n) \tag{80}$$

$$= \boldsymbol{w}^{(t)} - \tilde{\eta_S} \sum_{i=1}^{M}\sum_{j=1}^{n} \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij}) \left(\tilde{f}(\boldsymbol{w}^{(t)}, \boldsymbol{x}_{ij}) - \tilde{y}_{ij}\right) \tag{81}$$

$$= \boldsymbol{w}^{(t)} - \tilde{\eta_S} \tilde{\boldsymbol{A}}^\top (\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}) \tag{82}$$

Equation (81), Equation (82) follow from the definition of $\tilde{f}(\boldsymbol{w}, \boldsymbol{x}_{ij})$, $\tilde{y}_{ij}$, $\tilde{\boldsymbol{A}}$ in Appendix B.3.

We have,

$$\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t+1)}) - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) = \tilde{\boldsymbol{A}}(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}) \tag{83}$$

$$= -\tilde{\eta_S} \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}) \tag{84}$$

Therefore,

$$\left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t+1)}) - \tilde{\boldsymbol{y}}\right\|_2^2 = \left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t+1)}) - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) + \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}\right\|_2^2 \tag{85}$$

$$= \left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t+1)}) - \tilde{\boldsymbol{y}}\right\|_2^2 - 2\tilde{\eta_S}(\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}})\tilde{\boldsymbol{H}}(\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}) + \tilde{\eta}_S^2 \left\|\tilde{\boldsymbol{H}}(\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}})\right\|_2^2 \tag{86}$$

$$\leq (1 - \tilde{\eta_S}\lambda_0 + \tilde{\eta}_S^2 N^2) \left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}\right\|_2^2 \tag{87}$$

$$\leq \left(1 - \frac{\eta_S \lambda_0}{2n}\right) \left\|\tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)}) - \tilde{\boldsymbol{y}}\right\|_2^2 \tag{88}$$

where Equation (87) follows from Lemma 5 and Equation (88) follows from using $\tilde{\eta_S} \leq \frac{\lambda_0}{2N^2}$

Third part.

We have Equation (81),

$$\left\|\boldsymbol{w}_r^{(t+1)} - \boldsymbol{w}_r^{(t)}\right\|_2 = \left\|\frac{1}{\sqrt{m}} a_r \tilde{\eta_S} \sum_{i=1}^{M}\sum_{j=1}^{n} \boldsymbol{x}_{ij} \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{i,r}^* \geq 0\right\} (\tilde{y}_{ij} - \tilde{f}(\boldsymbol{w}^{(t)}, \boldsymbol{x}_{ij}))\right\|_2 \tag{89}$$

$$\leq \frac{\tilde{\eta_S}\sqrt{N}}{\sqrt{m}} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)})\right\|_2 \tag{90}$$

where the last inequality follows from Cauchy-Schwartz and Assumption 2.

Therefore,

$$\left\|\boldsymbol{w}_r^* - \boldsymbol{w}_r^{(0)}\right\|_2 \le \sum_{t=0}^{\infty} \left\|\boldsymbol{w}_r^{(t+1)} - \boldsymbol{w}_{F,r}^{(t)}\right\|_2 \tag{91}$$

$$\le \frac{\tilde{\eta}_S\sqrt{N}}{\sqrt{m}} \sum_{t=0}^{\infty} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(t)})\right\|_2 \tag{92}$$

$$\le \frac{\tilde{\eta}_S\sqrt{N}}{\sqrt{m}} \sum_{t=0}^{\infty} \left(1 - \frac{\tilde{\eta}_S\lambda_0}{4}\right)^t \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(0)})\right\|_2 \tag{93}$$

$$= \frac{4\sqrt{N}}{\sqrt{m}\lambda_0} \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{f}}(\boldsymbol{w}^{(0)})\right\|_2 \tag{94}$$

where Equation (93) follows from Equation (88). □

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

Proof.

We have,

$$\mathbb{E}\left[|S_{ij}^\perp|\right] = \sum_{i=1}^m \Pr(\mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \ge 0\} \ne \mathbb{I}\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \ge 0\}) \tag{95}$$

$$\le \Pr\left(\{\mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \ge 0\} \ne \mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \ge 0\}\} \cup \{\mathbb{I}\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \ge 0\} \ne \mathbb{I}\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{0,r} \ge 0\}\}\right) \tag{96}$$

$$\le \Pr(A_{ijr}(R_0+R_1)) + \Pr(A_{ijr}(R_0)) \tag{97}$$

$$\le \frac{4m(R_0+R_1)}{\sqrt{2\pi}} \tag{98}$$

Equation (97) follows from union-bound and Lemma 3, Lemma 2, Lemma 6.

Thus, using Markov's inequality, with probability at least $1-\delta$ we have,

$$|S_{ij}^\perp| = \mathcal{O}\left(\frac{m(R_0+R_1)}{\delta}\right) \tag{99}$$

Setting the failure probability of each $|S_{ij}|$ being large as δ/N, we have with probability $1-\delta$, for all $i \in [M]$ and $j \in [n]$ simultaneously,

$$|S_{ij}^\perp| = \mathcal{O}\left(\frac{mN(R_0+R_1)}{\delta}\right) \tag{100}$$

Thus with probability $1-\delta$ we have,

$$\sum_{i=1}^M \sum_{j=1}^n |S_{ij}|^2 \le \mathcal{O}\left(\frac{m^2N^2(R_0^2+R_1^2)}{\delta^2}\right) \tag{101}$$

□

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 2. *Under Assumptions 2, 3, for $m = \Omega\left(\frac{N^9}{\lambda_0^8\kappa^2\delta^4}\right)$, and i.i.d Gaussian initialization weights of $\boldsymbol{w}_0$ as $\boldsymbol{w}_{0,r} \sim \mathcal{N}(\boldsymbol{0}, \kappa)$, and initializing $a_r = \{-1, 1\}$ 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:*

$$L(\boldsymbol{w}^*) \le \underbrace{\mathcal{O}\left((1-\eta\lambda_0/2)^K \frac{N}{\delta}\right)}_{\textit{local optimization error}} + \underbrace{\mathcal{O}\left((2-(1-\eta\lambda_0/2)^K)\frac{N^9}{\lambda_0^8\delta^4 m}\right)}_{\textit{Laplace approximation error}}. \tag{102}$$

Proof.

Note that the conditions in all the preceding lemmas are satisfied by setting $m = \Omega\left(\frac{N^9}{\lambda_0^8\kappa^2\delta^4}\right)$, $\eta = \mathcal{O}(\lambda_0/N^2)$, $\eta_S = \mathcal{O}(\lambda_0/N^2)$ and hence we can now apply these lemma results for our proof.

From Lemma 6, we observe that

$$\tilde{y}_{ij} = \tilde{f}(\boldsymbol{w}^*, \boldsymbol{x}_{ij}) = \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^*$$

We have,

$$L(\boldsymbol{w}^*) = \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}(f(\boldsymbol{w}^*, \boldsymbol{x}_{ij}) - y_{ij})^2 \tag{103}$$

$$= \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left((\phi(\boldsymbol{w}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^* - \tilde{y}_{ij} + \tilde{y}_{ij} - y_{ij}\right)^2 \tag{104}$$

$$\leq \underbrace{\frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{w}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^* - \tilde{y}_{ij}\right)^2}_{T_1} + \underbrace{\frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{N}(\tilde{y}_{ij} - y_{ij})^2}_{T_2} \tag{105}$$

T_2 can be bounded as $\mathcal{O}\left((1-\eta\lambda_0/2)^K \frac{N}{\delta}\right)$ using the result from Lemma 2.

Our goal is to now bound T_1 as follows.

$$T_1 = \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{w}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^* - \tilde{y}_{ij}\right)^2 \tag{106}$$

$$= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\phi(\boldsymbol{w}^*, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^* - \phi(\tilde{\boldsymbol{w}}_i, \boldsymbol{x}_{ij})^\top \boldsymbol{w}^*\right)^2 \tag{107}$$

$$= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\frac{1}{\sqrt{m}}\sum_{r=1}^{m} a_r \boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* (\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} - \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\})\right)^2 \tag{108}$$

$$= \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\left(\frac{1}{\sqrt{m}}\sum_{r \in S_{ij}^\perp} a_r \boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* (\mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* \geq 0\right\} - \mathbb{I}\left\{\boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r} \geq 0\right\})\right)^2 \tag{109}$$

$$\leq \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|}{m}\sum_{r \in S_{ij}^\perp}\left(a_r \boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^*\right)^2 \tag{110}$$

$$\leq \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|}{m}\sum_{r \in S_{ij}^\perp}\left(\boldsymbol{x}_{ij}^\top \boldsymbol{w}_r^* - \boldsymbol{x}_{ij}^\top \tilde{\boldsymbol{w}}_{i,r}\right)^2 \tag{111}$$

$$\leq \frac{2}{N}\sum_{i=1}^{M}\sum_{j=1}^{n}\frac{|S_{ij}^\perp|^2 \max_{r\in[m]} \|\boldsymbol{w}_r^* - \tilde{\boldsymbol{w}}_{i,r}\|_2^2}{m} \tag{112}$$

$$\leq \mathcal{O}\left(\frac{m^2 N(R_0^4 + R_1^4)}{\delta^2 m}\right) \tag{113}$$

$$= \mathcal{O}\left(\frac{N^9}{\lambda_0^8\delta^4 m}(2 - (1-\eta\lambda_0/2)^K)\right) \tag{114}$$

Equation (110) uses Jensen's inequality and definition of S_{ij} in Lemma 7, Equation (111) uses the observation that since $r \in S_{ij}^\perp$ we have $\text{sign}(\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{F,r}) \neq \text{sign}(\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{i,r}^*)$ which implies $|\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{F,r}| \leq |\boldsymbol{x}_{ij}^\top \boldsymbol{w}_{F,r} - \boldsymbol{x}_{ij}^\top \boldsymbol{w}_{i,r}^*|$, Equation (112) uses Cauchy Schwartz, Equation (113) uses Lemma 7, Lemma 6, Lemma 2, Equation (114) substitutes $R_0 = \mathcal{O}\left(\frac{n}{\sqrt{m}\delta\lambda_0}(1-(1-\eta\lambda_0/4)^K)\right)$ and $R_1 = \mathcal{O}\left(\frac{N^2}{\sqrt{m}\delta\lambda_0^2}\right)$.

Thus we have,

$$L(\boldsymbol{w}^*) \leq \mathcal{O}\left((1-\eta\lambda_0/2)^K \frac{N}{\delta}\right) + \mathcal{O}\left((2-(1-\eta\lambda_0/2)^K)\frac{N^9}{\lambda_0^8\delta^4 m}\right). \tag{115}$$

This completes the proof. □

C. Computation and Communication Efficiency of `FedFisher(Diag)` and `FedFisher(K-FAC)`

C.1. 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 the bulk of the computation cost goes into computing $\tilde{\boldsymbol{w}}_i$, which needs multiple local epochs. We also experimentally verify this in Table 3, where we show that `FedFisher(Diag)` and `FedFisher(K-FAC)` add less than 12.5% of the total computational cost of `FedAvg` at clients for our FL setups.

Table 3: Time spent (in s) for client side computation for `FedFisher` variants vs `FedAvg`

Dataset/Model	`FedAvg`	`FedFisher (Diag)`	`FedFisher (K-FAC)`
MNIST/MLP	18.7	19.7(+5.3%)	19.9(+6.4%)
CIFAR10/CNN	36.1	40.5(+12.1%)	38.5(+6.6%)
CINIC10/ResNet18	314.5	353.8(+12.4%)	344.3(+9.4%)

C.2. Communication Efficiency

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,

$$[Q(\boldsymbol{x}, s_q)]_i = \|\boldsymbol{x}\|_\infty \text{sign}(x_i)\zeta_i(\boldsymbol{x}, s_q) \tag{116}$$

where $\|\boldsymbol{x}\|_\infty = \max_{i\in[d]} |x_i|$ and $\zeta_i(\boldsymbol{x}, s_q) = \left\lceil l_q \frac{|x_i|}{\|\boldsymbol{x}\|_\infty} \right\rceil / l_q$. Now 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,

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top \tag{117}$$

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 natural 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,

$$V(\boldsymbol{A}, s_v) = \boldsymbol{U}_{l_v}\boldsymbol{\Sigma}_{l_v}\boldsymbol{V}_{l_v}^\top \tag{118}$$

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 that the compression affects the accuracy of `FedFisher(Diag)` by less than 1% for the CIFAR-10 and MLP datasets.

Table 4: Test accuracy performance of `FedFisher(Diag)` and its compressed version on CIFAR10 and FashionMNIST with 5 clients and $\alpha = 0.5$

Dataset	`FedAvg`	`FedFisher(Diag)`	`FedFisher(Diag)`(Compressed)
FashionMNST	74.30	78.53	77.59
CIFAR10	44.12	45.32	44.67

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. 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 and Table 6 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 compression on FashionMNIST with 5 clients and $\alpha = 0.5$

Algorithm	s_q	s_v	Accuracy
`FedAvg`	$\times$	$\times$	77.81
`FedFisher(K-FAC)`	$\times$	$\times$	84.93
`FedFisher(K-FAC)`	2	2.5	82.70
`FedFisher(K-FAC)`	4	1.25	83.30
`FedFisher(K-FAC)`	8	1	79.71

Table 6: Test accuracy performance of `FedFisher(K-FAC)` with different levels of compression on CIFAR10 with 5 clients and $\alpha = 0.5$

Algorithm	s_q	s_v	Accuracy
`FedAvg`	$\times$	$\times$	48.93
`FedFisher(K-FAC)`	$\times$	$\times$	63.97
`FedFisher(K-FAC)`	2	8.1	60.51
`FedFisher(K-FAC)`	4	4.1	62.67
`FedFisher(K-FAC)`	8	2.1	48.57

D. Additional Experimental Details

The local optimization procedure is the same across all algorithms. In particular, clients perform T epochs of local training using the SGD optimizer with local learning rate $\eta = 0.01$, batch size 64 and momentum factor 0.9. We set $T = 100$ for the experiments on CINIC-10 and $T = 30$ for the others. To compute the Fisher diagonal and Fisher K-FAC we use the `nngeometry` package (50). 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 tune the σ, σ_0 and γ parameters as done in the official implementation available at `https://github.com/IBM/probabilistic-federated-neural-matching`. The grid for σ is $\{0.1, 0.5, 1.0\}$, grid for σ_0 is $\{1.0, 10.0\}$ and grid for γ is $\{0.001, 1, 50\}$.

`DENSE`. For `DENSE` we use the default settings as described in Section 3.1.4 of (9). 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. To ensure computational fairness among baselines we limit the number of distillation epochs $T = 20$. 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 hyperparameter and type of ground metric normalization affect performance most and hence we focus on tuning these hyperparameters for our experiments, while using the default settings for the other hyperparameters.