Fed-ensemble: Improving Generalization through Model Ensembling in Federated Learning

Naichen Shi [*], Fan Lai [†], Raed Al Kontar [‡], Mosharaf Chowdhury [§]

July 23, 2021

Abstract

In this paper we propose Fed-ensemble: a simple approach that brings model ensembling to federated learning (FL). Instead of aggregating local models to update a single global model, Fed-ensemble uses random permutations to update a group of K models and then obtains predictions through model averaging. Fed-ensemble can be readily utilized within established FL methods and does not impose a computational overhead as it only requires one of the K models to be sent to a client in each communication round. Theoretically, we show that predictions on new data from all K models belong to the same predictive posterior distribution under a neural tangent kernel regime. This result in turn sheds light on the generalization advantages of model averaging. We also illustrate that Fed-ensemble has an elegant Bayesian interpretation. Empirical results show that our model has superior performance over several FL algorithms, on a wide range of data sets, and excels in heterogeneous settings often encountered in FL applications.

1 Introduction

The rapid increase in computational power on edge devices has set forth federated learning (FL) as an elegant alternative to traditional cloud/data center based analytics. FL brings training to the edge, where devices collaboratively extract knowledge and learn complex models (most often deep learning models) with the orchestration of a central server while keeping their personal data stored locally. This paradigm shifts not only reduces privacy concerns but also sets forth many intrinsic advantages including cost efficiency, diversity, and reduced communication, amongst many others [39, 18].

The earliest and perhaps most popular FL algorithm is FederatedAveraging (fedavg) [28]. In fedavg, the central server broadcasts a global model (set of weights) to selected edge devices, these devices run updates based on their local data, and the server then takes a weighted average of the resulting local models to update the global model. This process iterates over hundreds of training rounds to maximize the performance for all devices. Fedavg has seen prevailing empirical successes in many real-world applications [4, 11]. The caveat, however, is that aggregating local models is prone to overfitting and suffers from high variance in learning and prediction when local datasets are heterogeneous be it in size or distribution or when clients have limited bandwidth, memory, or unreliable connection that effects their participation in the training process [31, 30]. Indeed, in the past few years, multiple papers have shown the high variance in the performance of FL algorithms and their vulnerability to overfitting, specifically in the presence of data heterogeneity or unreliable devices [17, 38, 36, 31, 30]. Since then some notable algorithms have attempted to improve the generalization performance of fedavg and tackle the above challenges as discussed in Sec. 2.

In this paper, we adopt the idea of ensemble training to FL and propose *Fed-ensemble* which iteratively updates an ensemble of models to improve the generalization performance of FL methods. We show,

[*]IOE, University of Michigan
[†]CSE, Univeristy of Michigan
[‡]IOE, University of Michigan
[§]CSE, University of Michigan

both theoretically and empirically, that ensembling is efficient in reducing variance and achieving better generalization performance in FL. Our approach is orthogonal to current efforts in FL aimed at reducing communication cost, heterogeneity or finding better fixed points as such approaches can be directly integrated into our ensembling framework. Specifically, we propose an algorithm to train K different models. Predictions are then obtained by ensembling all trained modes, i.e. model averaging. Our contributions are summarized below:

- **Model:** We propose an ensemble treatment to FL which updates K models over local datasets. Predictions are obtained by model averaging. Our approach does not impose additional burden on clients as only one model is assigned to a client at each communication round. We then show that *Fed-ensemble* has an elegant Bayesian interpretation.
- **Convergence and Generalization:** We motivate the generalization advantages of ensembling under a bias-variance decomposition. Using neural tangent kernels (NTK), we then show that predictions at new data points from all K models converge to samples from the same limiting Gaussian process in sufficiently overparameterized regimes. This result in turn highlights the improved generalization and reduced variance obtained from averaging predictions as all K models converge to samples from that same limiting posterior. To the best of our knowledge, this is the first theoretical proof for the convergence of general multilayer neural network in FL in the kernel regime, as well as the first justification for using model ensembling in FL. Our proof also offers standalone value as it extends NTK results to FL settings.
- **Numerical Findings:** We demonstrate the superior performance of *Fed-ensemble* over several FL techniques on a wide range of datasets, including realistic FL datasets in FL benchmarks [21]. Our results highlight the capability of *Fed-ensemble* to excel in heterogeneous data settings.

2 Related Work

Single-mode FL Many approaches have been proposed to tackle the aformentioned FL challenges. Below we list a few, yet this is by no means an extensive list. *Fedavg* [28] allows inexact local updating to balance communication vs. computation on resource-limited edge devices, while reporting decent performance on mitigating performance bias on skewed client data distributions. *Fedprox* [23] attempts to solve the heterogeneity challenge by adding a proxy term to control the shift of model weights between the global and local client model. This proxy term can be viewed as a normal prior distribution on model weights. There are several influential works aiming at expediting convergence, like *FedAcc* [40], *FedAdam* and *FedYogi* [34], reducing communication cost, like *DIANA* [29], *DORE* [25], or find better fixed points, like *FedSplit* [32], *FedPD* [43], *FedDyn* [1]. *As aformentioned, these efforts are complementary to our work, and they can be integrated into our ensembling framework.*

Ensemble of deep neural nets Recently ensembling methods in conventional, non-federated, deep learning has seen great success. Amongst them, [8] analyzes the loss surface of deep neural networks and uses cyclic learning rate to learn multiple models and form ensembles. [7] visually demonstrates the diversity of randomly initialized neural nets and empirically shows the stability of ensembled solutions. Also, [10] connects ensembling to Bayesian deep neural networks and highlights the benefits of ensembling.

Bayesian methods in FL and beyond There are also some recent attempts to exploit Bayesian philosophies in FL. Very recently, *Fedbe* was proposed [5] to couple Bayesian model averaging with knowledge distillation on the server. Fedbe formulates the Gaussian or Dirichlet distribution of local models and then uses knowledge distillation on the server to update the global model. This procedure however requires additional datasets on the server and a significant computational overhead, thus being demanding in FL. Besides that, Bayesian non-parametrics have been investigated for advanced model aggregation through matching and re-permuting neurons using neuron matching algorithms [37, 41]. Such approaches intend to address the permutation invariance of parameters in neural networks, yet suffer from a large computational burden. We also note that Bayesian methods have been also exploited in meta-learning which can achieve personalized FL. For instance, [19] proposes a Bayesian version of MAML [6] using Stein variational gradient descent [24]. An amortized version of this [33] utilizes variational inference (VI) across tasks to learn a prior distribution over neural network weights.

3 Fed-ensemble

3.1 Parameter updates through Random Permutation

Let N denote the number of clients where the local dataset of client i is given as $\boldsymbol{D}_i = \{x_{ij}, y_{ij}\}_{j\in[1,2,..,n_i]}$ and n_i is the number of observations for client i. Also, let $\boldsymbol{D}$ be the union of all local datasets $\boldsymbol{D} = \bigcup_{i=1}^{N} \boldsymbol{D}_i$, and $f_{\boldsymbol{w}}(\cdot)$ denote the model to be learned parametrized by weight vector $\boldsymbol{w}$.

Design principle Our goal is to get multiple models $(f_{\boldsymbol{w}_1}, f_{\boldsymbol{w}_2}, .., f_{\boldsymbol{w}_K})$ engaged in the training process. Specifically, we use K models in the ensemble, where K is a predetermined number. The K models are randomly initialized by standard initialization, e.g. Xavier [9] or He [12]. In each training round, *Fed-ensemble* assigns one of the K models to the selected clients to train on their local data. The server then aggregates the updated models from the same initialization. All K models eventually learn from the entire dataset, and allow an improvement to the overall inference accuracy by averaging over K predictions produced by each model. Hereon, we use mode or weight to refer to $\boldsymbol{w}_k$ and model to refer to the corresponding $f_{\boldsymbol{w}_k}$.

Objective of ensemble training Since we aim to learn K modes, the objective of FL training can be simply defined as:

$$\min_{\mathcal{W}} \sum_{k=1}^{K} \sum_{i=1}^{N} p_i \ell_i(\boldsymbol{w}_k) \tag{1}$$

where $\mathcal{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_K\}$, ℓ_i is the local empirical loss on client i, $\ell_i(\boldsymbol{w}_k) = \frac{1}{n_i} \sum_{(x_{ij},y_{ij})\in \boldsymbol{D}_i} L(f_{\boldsymbol{w}_k}(x_{ij}), y_{ij})$, p_i is a weighting factor for each client and L is a loss function such as cross entropy.

Algorithm 1 Fed-ensemble: Fed-ensemble using random permutations

Input: Client datasets $\{\boldsymbol{D}_i\}_{i=1}^{N}$, T, K, M, initialization for $\{\boldsymbol{w}_k\}_{k=1}^{K}$
Randomly divide N clients into Q strata $\{S_q\}_{q=1}^{Q}$
for $t = 1, 2, \cdots T$ **do**
  Index Permutation: $\boldsymbol{P}_{q,\cdot} = shuffle\_list[1, 2, \cdots K]$
  for $r = 1, 2, \cdots K$ **do**
    for $q = 1, 2, \cdots Q$ **do**
      Randomly select clients C_q from S_q
      Server broadcasts mode $\boldsymbol{w}_{\mathbf{P}_{q,r}}$ to clients C_q
      for each client i in C_q **do**
        $\boldsymbol{w}^{(i)} = local\_training[\boldsymbol{w} = \boldsymbol{w}_{\mathbf{P}_{q,r}}, \boldsymbol{D}_i]$
        Client i sends $\boldsymbol{w}_i$ to server
      end for
    end for
    $server\_update$ $[\boldsymbol{w}^{(1)}, .., \boldsymbol{w}^{(M)}]$.
  end for
end for

Model training with Fed-ensemble We now introduce our algorithm *Fed-ensemble* (shown in Algorithm 1), which is inspired by random permutation block coordinate descent used to optimize (1). We first, randomly divide all clients into Q strata, $\{S_q\}_{q=1}^{Q}$. Now let $r \in [1, .., K]$ denote an individual communication round where M clients are sampled in each round. Also, let $t \in [1, .., T]$ denote the training age, where each age consists of K communication rounds. Hence the total number of communication rounds for T ages is $R = T \times K$. At the beginning of each training age, every stratum will decide the order of training in this age. To do so, define a permutation matrix $\boldsymbol{P}$ of size $Q \times K$ such that at each age t the rows of $\boldsymbol{P}$ are given as $\boldsymbol{P}_{q,\cdot} = shuffle\_list[1, 2, \cdots K]$. More specifically, in the r-th communication

rounds in this age, the server samples some clients from each stratum. Clients from stratum q get assigned mode $\boldsymbol{w}_{\mathbf{P}_{q,r}}$ as their initialization for $\boldsymbol{w}_i$ and then do a training procedure on their local data, denoted as $\boldsymbol{w}^{(i)} = local\_training[\boldsymbol{w} = \boldsymbol{w}_{\mathbf{P}_{q,r}}, \boldsymbol{D}_i]$. Note that the use of the random permutation matrix $\boldsymbol{P}$, not only ensures that modes are trained on diverse clients but also guarantees that every mode is downloaded and trained on all strata in one age. Upon receiving updated models from all M clients, the server activates *server_ update* to calculate the new $[\boldsymbol{w}^{(1)}, .., \boldsymbol{w}^{(M)}]$.

Remark 3.1. *The simplest form of the server_ update function is to average modes from the same initialization:* $\boldsymbol{w}_k \leftarrow \frac{\sum_{i=1}^{M} \delta_{ik} \boldsymbol{w}^{(i)}}{\sum_{i=1}^{M} \delta_{ik}}$ $\forall k$, *where* $\delta_{ik} = 1$ *if client* i *downloads model* k *at communication round* r, *and 0 otherwise.* δ_{ik} *can also be obtained from* $\boldsymbol{P}$. *This approach is an extension of fedavg. However, one can directly utilize any other scheme in server_ update to aggregate modes from the same initialization. Similarly different local training schemes can be used within local_ training.*

Remark 3.2. *The use of multiple modes does not increase computation or communication overhead on clients compared with single-mode FL algorithms, as for every round, only one mode is sent to and trained on each client. However, after carefully selecting modes, models trained by Fed-ensemble can efficiently learn statistical patterns on all clients as proven in the convergence and generalization results in Sec. 4.*

Model prediction with Fed-ensemble: After the training process is done, all K modes are sent to the client, and model prediction at a new input point $x^\star$ is achieved via ensembling.

$$f_{\mathcal{W}}(x^\star) = \frac{1}{K} \sum_{k=1}^{K} f_{\boldsymbol{w}_k}(x^\star)$$

3.2 Bayesian Interpretation of Fed-ensemble

Interestingly, *Fed-ensemble* has an elegant Bayesian interpretation as a variational inference (VI) approach to estimate a posterior Gaussian mixture distribution over model weights. To view this, let model parameters $\boldsymbol{w}$ admit a posterior distribution $p(\boldsymbol{w}|\boldsymbol{D})$. Under VI, a variational distribution $q(\boldsymbol{w})$ is used to approximate $p(\boldsymbol{w}|\boldsymbol{D})$ by minimizing the KL-divergence between the two:

$$\min D_{KL}\left[q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\right] = \mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}\left[ln\, q(\boldsymbol{w}) - ln\, p(\boldsymbol{w}|\boldsymbol{D})\right], \tag{2}$$

Now, if we take $q(\boldsymbol{w})$ as a mixture of isotropic Gaussians, $q(\boldsymbol{w}) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}\left(\boldsymbol{w}_k, \sigma^2 I\right)$, where $\boldsymbol{w}_k$'s are the variational parameters to be optimized. When $\boldsymbol{w}_k$'s are well separated, $\mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}\left[ln\, q(\boldsymbol{w})\right]$ does not depend on $\boldsymbol{w}_i$. In the limit $\sigma \to 0$, $q(\boldsymbol{w})$ becomes the linear combination of K Dirac-delta functions $\frac{1}{K}\sum_{k=1}^{K}\delta(\boldsymbol{w_i})$, then $\mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}\left[ln\, p(\boldsymbol{w}|\boldsymbol{D})\right] = \frac{1}{K}\sum_{k=1}^{K} ln\, p(\boldsymbol{w}_k|\boldsymbol{D})$. Finally, notice that data on different clients are usually independent, therefore the log posterior density factoririzes as: $ln\, p(\boldsymbol{w}_k|\boldsymbol{D}) = ln\, p(\boldsymbol{D}|\boldsymbol{w}_k) + ln\, p(\boldsymbol{w}_k) = ln\, p(\boldsymbol{w}_k) + \sum_i ln\, p\left(\boldsymbol{D_i}|\boldsymbol{w}\right)$. If we take the loss function in (1) to be $\ell_i(\boldsymbol{w}_k) = \frac{1}{n_i} ln\, p(\boldsymbol{D}_i|\boldsymbol{w}_k) + \frac{1}{\sum_i n_i} p(\boldsymbol{w}_k)$, we then recover (1). This Bayesian view highlights the ensemble diversity as the K modes can be viewed as modes of a mixture Gaussian. Further details on this Bayesian interpretation can be found in the Appendix.

4 Convergence and limiting behavior of sufficiently over parameterized neural networks

In this section we present theoretical results on the generalization advantages of ensembling. We show that one can improve generalization performance by reducing predictive variance. Then we analyze sufficiently overparametrized networks trained by *Fed-ensemble*. **We prove the convergence of the training loss, and derive the limiting model after sufficient training, from which we show how generalization can be improved using *Fed-ensemble*.**

4.1 Variance-bias decomposition

We begin by briefly reviewing the bias-variance decomposition of a general regression problem. We use $\boldsymbol{\theta}$ to parametrize the hypothesis class $h_{\boldsymbol{\theta}}$, $\overline{h_{\boldsymbol{\theta}}}(x)$ to denote the average of $h_{\boldsymbol{\theta}}(x)$ under some distribution $q(\boldsymbol{\theta})$, i.e. $\overline{h_{\boldsymbol{\theta}}}(x) = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[h_{\boldsymbol{\theta}}(x)\right]$. Similarly $\overline{y}(x)$ is defined as $\overline{y}(x) = \mathbb{E}_y\left[y|x\right]$, then:

$$\begin{aligned}\mathbb{E}_{y,x,\boldsymbol{\theta}}\left[\left(y - h_{\boldsymbol{\theta}}(x)\right)^2\right] = \mathbb{E}_{y,x,\boldsymbol{\theta}}\left[\left(y - \overline{y}(x)\right)^2\right] + \mathbb{E}_{x,\boldsymbol{\theta}}\left[\left(\overline{y}(x) - \overline{h_{\boldsymbol{\theta}}}(x)\right)^2\right] + \\ \mathbb{E}_{x,\boldsymbol{\theta}}\left[\left(\overline{h_{\boldsymbol{\theta}}}(x) - h_{\boldsymbol{\theta}}(x)\right)^2\right]\end{aligned} \tag{3}$$

where the expectation over x is taken under some input distribution $p(x)$, and that over $\boldsymbol{\theta}$ is taken under $q(\theta)$. In (3), the first term represents the intrinsic noise in data, referred to as data uncertainty. The second term is the bias term which represents the difference between expected predictions and the expected predicted variable y. The third term characterizes the variance from the discrepancies of different functions in the hypothesis class, also referred to as knowledge uncertainty in [26]. In FL, this variance is often large due to heterogeneity, partial participation, etc. However, we will show that this variance decreases through model ensembling.

4.2 Convergence and variance reduction using neural tangent kernels

Inspired by recent work on neural tangent kernels [16, 22], we analyze sufficiently overparametrized neural networks. We focus on regression tasks and define the local empirical loss in (1) as:

$$\ell_i(\boldsymbol{w}_k) = \frac{1}{2n_i}\sum_{(x_{ij},y_{ij})\in\boldsymbol{D}_i}\left(f_{\boldsymbol{w}_k}(x_{ij}) - y_{ij}\right)^2 \tag{4}$$

For this task, we will prove that when overparameterized neural networks are trained by *Fed-ensemble*, the training loss converges exponentially to a small value determined by stepsize, and $f_{\boldsymbol{w}_k}(x)$ for all $k \in [1, .., K]$ converge to samples from the same posterior Gaussian Process ($\mathcal{GP}$) defined via a neural tangent kernel. Note that for space limitations we only provide an informal statement of the theorems, while details are relegated to the Appendix. Prior to stating our result we introduce some needed notations.

For notational simplicity we drop the subscript k in $\boldsymbol{w}_k$, and use $\boldsymbol{w}$ instead, unless stated otherwise. We let $\boldsymbol{w}(0)$ denote the initial value of $\boldsymbol{w}$, and $\boldsymbol{w}(t)$ denote $\boldsymbol{w}$ after t local epochs of training. We also let p_{init} denote the initialization distribution for weights $\boldsymbol{w}$. Conditions for p_{init} are found in the Appendix. We define an initialization covariance matrix for any input x as $\mathcal{K}(x, x') = \mathbb{E}_{\boldsymbol{w}\sim p_{\text{init}}(\boldsymbol{w})}[f_{\boldsymbol{w}}(x)f_{\boldsymbol{w}}(x')]$. Also we denote the neural tangent kernel of a neural network to be $\Theta^{(l)}(\cdot, \star) = \sum_{u=1}^{U}\partial_{w_u}f_{\boldsymbol{w}}(\cdot)\partial_{w_u}f_{\boldsymbol{w}}(\star)$, where l represents the minimum width of neural network $f_{\boldsymbol{w}}$ in each layer and U denotes the number of trainable parameters. Indeed, [16] shows that this limiting kernel Θ remains fixed during training if the width of every layer of a neural network approaches infinity and when the stepsize η scales with l^{-1}: $\eta = \frac{\eta_0}{l}$. We adopt the notation in [16] and extend the analysis to FL settings.

Below we state an informal statement of our convergence result.

Theorem 4.1. *(Informal) For the least square regression task* $\min \frac{1}{2}\mathbb{E}_{(x,y)\sim\mathbf{D}}\left|f(x) - y\right|^2$*, where* $f(x)$ *is a neural network whose width* l *goes to infinity,* $l \to \infty$*, then under the following assumptions*

(i) Θ *is full rank i.e.,* $\lambda_{min}(\Theta) > 0$

(ii) The norm of every input x *is smaller than 1:* $||x|| \leq 1$

(iii) The stationary points of all local losses coincide: $\nabla_{\boldsymbol{w}}\mathbb{E}_{(x,y)\sim\mathbf{D}}\left|f_{\boldsymbol{w}}(x) - y\right|^2 = 0$ *leads to* $\nabla_{\boldsymbol{w}}\mathbb{E}_{(x,y)\sim\mathbf{D}_i}\left|f_{\boldsymbol{w}}(x) - y\right|^2 = 0$ *for all clients*

(iv) The total number of data points in one communication round is a constant, $\overline{n}$

when we use Fed-ensemble and local clients train via gradient descent with stepsize $\eta = \frac{\eta_0}{l}$, the training error associated to each mode decreases exponentially

$$\mathbb{E}_{(x,y)\sim\mathbf{D}} \left| f_{\boldsymbol{w}(t)}(x) - y \right|^2 \leq e^{-\frac{\eta_0 \lambda_{min}(\Theta) t}{3n}} \mathbb{E}_{(x,y)\sim\mathbf{D}} \left| f_{\boldsymbol{w}(0)}(x) - y \right|^2 + o(\eta_0^2)$$

if the learning rate η is smaller than a threshold.

Remark 4.1. *Assumptions (i) and (ii) are standard for theoretical development in NTK. Assumption (iii) can be derived directly from B-local dissimilarity condition in [23]. It is actually an overparametrization condition: it says that if the gradient of the loss evaluated on the entire union dataset is zero, the gradient of the loss evaluated on each local dataset is also zero. Here we note that recent work [43, 1] have tried to provide FL algorithms that work well when this assumption does not hold. As aforementioned, such algorithms can be utilized within our ensembling framework. Assumption (iv) is added only for simplicity: it can be removed if we choose a stepsize according to number of datapoints in each round.*

We would like to note that after writing this paper, we find that [14] show convergence of the training loss in FL under a kernel regime. Their analysis, however, is limited to a special form of 2-layer *relu* activated networks with the top layer fixed, while we study general multi-layer networks. Also this work is mainly concerned with theoretical understanding of FL under NTK regimes, while our overarching goal is to propose an algorithm aimed at ensembling, *Fed-ensemble*, and motivate its use through NTK.

More importantly and beyond convergence of the training loss, we can analytically calculate the limiting solution of sufficiently overparametrized neural networks. The following theorem shows that models in the ensemble will converge into independent samples in a Gaussian Process:

Theorem 4.2. *(Informal) If Fed-ensemble in Algorithm 1 is used to train $\{\boldsymbol{w}_k\}_{k=1,\dots K}$ for the regression task (1) and (4), then after sufficient communication rounds, functions $f_{\boldsymbol{w}_k}(x)$ can be regarded as independent samples from a $\mathcal{GP}\left(m(x), k(x,x')\right) + o(\eta_0^2)$, with mean and variance defined as*

$$m(x) = \Theta(x, X)\Theta^{-1}(X, X)Y$$

, and

$$k(x,x') = \mathcal{K}\left(x,x'\right) + \Theta\left(x,X\right)\Theta^{-1}\mathcal{K}\Theta^{-1}\Theta\left(X,x'\right) - \left(\Theta\left(x,X\right)\Theta^{-1}\mathcal{K}\left(X,x'\right) + \Theta\left(x',X\right)\Theta^{-1}\mathcal{K}\left(X,x\right)\right)$$

, and (X, Y) represents the entire dataset $\boldsymbol{D}$.

Remark 4.2. *The result in Theorem 4.2 is illustrated in Fig. 1. The central result is that training K modes with Fed-ensemble will lead to predictions $f_{\boldsymbol{w}_k}(x)$, all of which are samples from the same posterior $\mathcal{GP}$. The mean of this $\mathcal{GP}$ is the exact result of kernel regression using Θ, while the variance is dependent on initialization. Hence via Fed-ensemble one is able to obtain multiple samples from the posterior predictive distribution. This result is similar to the simple sample-then-optimize method by [27] which shows that for a Bayesian linear model the action of sampling from the prior followed by deterministic gradient descent of the squared loss provides posterior samples given normal priors.*

A direct consequence of the result above in given in the corollary below.

Corollary 4.1. *Let $\epsilon > 0$ be a positive constant, if the assumptions in theorem 4.1 are satisfied, when we train with Fed-ensemble with K modes, after sufficient iterations, we have that:*

$$\mathbb{P}\left(\left|\frac{1}{K}\sum_{k=1}^{K} f_{\boldsymbol{w}_k} - \tilde{f}\right| \geq \epsilon\right) \leq O\left(\frac{1}{\epsilon^2 K}\right) \tag{5}$$

where $\tilde{f}$ is the *maximum a posteriori* solution of the limiting Gaussain Process obtained in Theorem 4.2. This corollary is obtained by simply combining Chebyshev inequality with Theorem 4.2 and shows that since the variance shrinks at the rate of $\frac{1}{K}$, averaging over multiple models is capable of getting closer to $m(x)$.

We finally should note that there is a gap between neural tangent kernel and the actual neural network [2]. Yet this analysis still serves to highlight the generalization advantages of FL inference with multiple modes. Also, experiments show Fed-ensemble works well beyond the kernel regime assumed in Theorem 4.1.

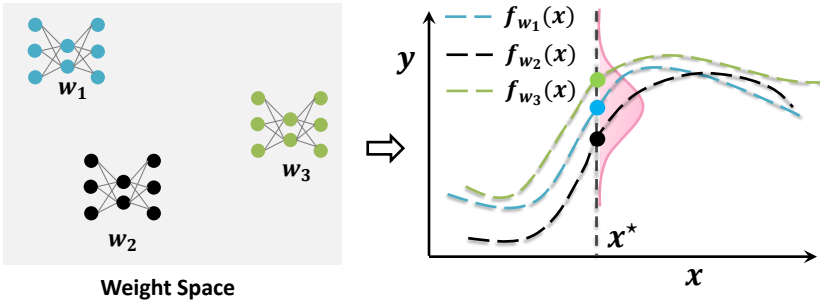
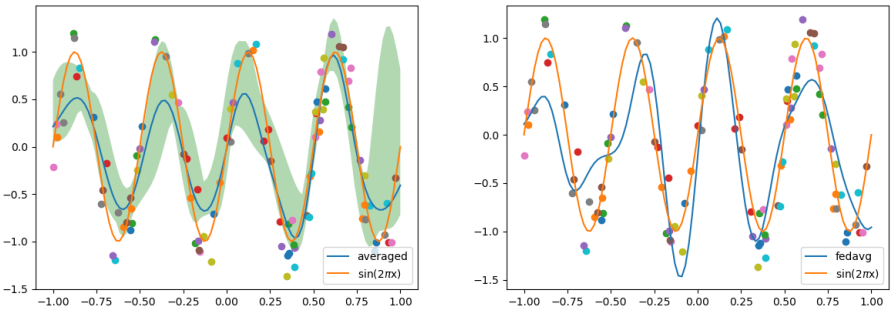


Figure 1: An illustration of an ensemble of 3 samples from the posterior distribution.

5 Experiments

In this section we provide empirical evaluations of *Fed-ensemble* on five representative datasets of varying sizes. We start with a toy example to explain the bias-variance decomposition, then move to realistic tasks. We note that since ensembling is an approach yet to be fully investigated in FL, we dedicate many experiments to a proof of concept.

A simple toy example with kernels We start with a toy example on kernel methods that illustrates the benefits of using multiple modes and reinforces the key conclusions from Theorem 4.2. We create 50 clients and generate the data of each client following a noisy sine function $y = a\sin(2\pi x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.2^2)$, and a denotes the parameter unique for each client that is sampled from a random distribution $\mathcal{N}(1, 0.2^2)$. On each client we sample 2 points of x uniformly in $[-1, 1]$. We use the linear model $f(x; \mathcal{W}) = \sum_{k=1}^{100} w_k \varphi_k(x)$, where $\varphi_k(x)$ is a radial basis kernel defined as: $\phi_k(x) = \exp\left(-\frac{(x-\nu_k)^2}{2b^2}\right)$. In this badis function, $b = 0.08$ and ν_k's are 100 uniformly randomly sampled parameters from $[-1, 1]$ that remain constant during training. Note that the expectation of the generated function is $\mathbb{E}(y|x) = \sin(2\pi x)$.



(a) Prediction performance of Fed-ensemble. (b) Prediction performance of Fedavg.

Figure 2: Linear model on toy dataset. Dots represent datapoint from a subset of clients. The green area denotes predictive interval obtained from individual models, $f_{\boldsymbol{w}_k}(x)$, estimated using *Fed-ensemble*. The "averaged" reports the final prediction obtained after model averaging in *Fed-ensemble*.

We report our predictive results in Table 1 and Fig. 2. Despite the diversity of individual mode predictions (green area), upon averaging, the averaged result becomes more accurate than a fully trained single mode (Fig. 2 (a,b)). Hence, *Fed-ensemble* is able to average out the noise introduced by individual modes. As a more quantitative measurement, in Table 1 we vary K and calculate the bias-variance decomposition in (3) in each case. As seen in the table, the variance of *Fed-ensemble* can be efficiently reduced compared with a single mode approach such as FedAvg.

	K=1(FedAvg)	K=2	K=10	K=20	K=40
Bias	0.109	0.117	0.112	0.113	0.112
Variance	0.0496	0.0115	0.0063	0.0045	0.0042

Table 1: Bias-variance decomposition on the toy regression model. We fix training data, and run each algorithm 100 times from random initializations. We take the average of 100 models as $\overline{h}_{\boldsymbol{\theta}}$ in (3). Bias and variance are calculated accordingly. When we increase K from 1 to 40, bias almost remains in the same level, while variance decays at a rate slightly slower than $\frac{1}{K}$. Variance ceases to decrease after K is larger than 20. This is intuitively understandable as when variance is very low, a larger dataset and number of communication rounds are needed to decrease it further.

Experimental setup In our evaluation, we show that *Fed-ensemble* outperforms its counterparts across two popular but different settings: (i) *Homogeneous setting*: data are randomly shuffled and then uniformly randomly distributed to clients; (ii) *Heterogeneous setting*: data are distributed in a non-i.i.d manner: we firstly sort the images by label and then assign them to different clients to make sure that each client has images from exactly 2 labels. We experiment with five different datasets of varying sizes:

- *MNIST*: a popular image classification dataset with 55k training samples across 10 categories.
- *CIFAR10*: a dataset containing 60k images from 10 classes.
- *CIFAR100*: a dataset with the similar images of CIFAR10 but categorized into 100 classes.
- *Shakespeare*: the complete text of William Shakespeare with 3M characters for next work prediction.
- *OpenImage*: a real-world image dataset with 1.1M images of 600 classes from 13k image uploaders [20]. We use the realistic distribution of client data in FedScale benchmark [21].

In our experiments, we use $K = 5$ modes by default, except in the sensitivity analysis, where we vary K. Initial learning rates for clients are chosen based on the existing literature for each method.

We compare our model with *Fedavg, Fedprox, Fedbe* and *Fedbe* without knowledge distillation, where we use random sampling to replace knowledge distillation. Some entries in Fedbd/Fedbe-noKD columns are missing either because it's impractical (dataset on the central server does not exist) or we cannot finetune the hyper-parameters to achieve reasonable performance.

MNIST dataset. We train a 2-layer convolutional neural network to classify different labels. Results in Table 2 show that *Fed-ensemble* outperforms all benchmarked single mode FL algorithms. This confirms the effectiveness of ensembling over single mode methods in improving generalization. Fedbe turns out to perform closely to single mode algorithms eventually. We conjecture that this happens as the diversity of local models is lost in the knowledge distillation step. To show diversity of models in the ensemble, we plot the the projection of the loss surface on a plane spanned by 3 modes from the ensemble at the end of training in Fig 3. Fig 3 shows that modes ($\boldsymbol{w}_k$) have rich diversity: different modes correspond to different local minimum with a high loss barrier on the line connecting them. In the non-i.i.d setting, the performance gap between *Fed-ensemble* and single mode algorithms becomes larger compared with i.i.d settings. This shows that *Fed-ensemble* can better fit more variant data distributions compared with Fedavg and Fedprox. This result is indeed expected as ensembling excels in stabilizing predictions [10].

CIFAR/Shakespeare/OpenImage dataset. We use ResNet [13] for CIFAR, a character-level LSTM model for the sequence prediction task, and ShuffleNet [42] and MobileNet [35] for OpenImage. Results in Table 2 show that *Fed-ensemble* can indeed improve the generalization performance. Further details on the experimental setup can be found in the Appendix.

Effect of non-i.i.d client data distribution We change the number of classes (N.o.C.) assigned to each client from 10 to 2 on the CIFAR-10 dataset. Conceivably, when each client has fewer categories, the data distribution is more variant. The results are shown in Table 3. As expected, the performance of all algorithms degrades with such heterogeneity. Further *Fed-ensemble* outperforms its counterparts

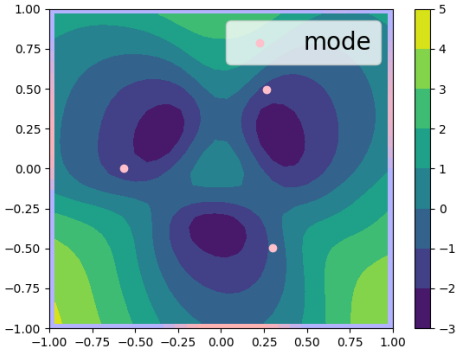


Figure 3: Projection of loss surface to a plane spanned by 3 modes of the ensemble trained by *Fed-ensemble*. Color represents the logarithm of cross-entropy loss on the entire training set.

Testing acc (%)	Fedavg	Fedprox	Fedbe	Fedbe-nokd	Fed-ensemble
MNIST-iid	95.08 ± 0.3	95.76 ± 0.12	95.21 ± 0.11	-	**97.75**±0.04
MNIST-noniid	90.17 ± 0.19	90.75 ± 0.34	-	-	**95.44**±0.14
CIFAR10-iid	86.43 ± 0.11	86.77 ± 0.06	83.13 ± 0.34	86.78 ± 0.4	**87.99**±0.09
CIFAR10-noniid	66.11 ± 0.17	66.62 ± 0.24	-	-	**71.18**±0.4
CIFAR100	56.25 ± 0.22	55.91 ± 0.25	49.24 ± 2	-	**58.09**±0.12
Shakespeare	60.05 ± 0.02	60.18 ± 0.02	-	61.52 ± 0.22	**62.49**±0.08
OpenImage mobile net	51.85±0.17	52.93±0.14	-	-	**53.92**±0.19
OpenImage shuffle net	53.98±0.16	54.42±0.22	-	-	**55.75**±0.25

Table 2: Testing accuracy of models trained by different FL algorithms on five datasets. Fedbe-Nokd denotes Fedbe without knowledge distillation.

N.o.C.	Fedavg	Fedprox	Fed-ensemble	Gap
2	66.19	66.87	**71.45**	4.58
4	83.90	84.40	**86.12**	1.72
6	85.90	86.10	**87.14**	1.04
8	85.90	86.06	**87.33**	1.27
10	86.52	86.77	**87.94**	1.17

Table 3: Sensitivity analysis with different assigned N.o.C on CIFAR-10. Gap denotes the difference in testing accuracy between *Fed-ensemble* and Fedprox.

and the performance gap becomes significantly clear as variance increases (i.e., smaller N.o.C.). This highlights the ability of *Fed-ensemble* to improve generalization specifically with heterogeneous clients.

Effect of number of modes K. Since the number of modes, K, in the ensemble is an important hyperparameter, we choose different values of K and test the performance on MNIST. We vary K from 3 to 80. The results are shown in Table 4. Besides testing accuracy of ensemble predictions, we also calculate the accuracy and the entropy of the predictive distribution of each mode in the ensemble. As shown in Table 4, as K increases from 3 to 40, the ensemble prediction accuracy increases as a result of variance reduction. However, when K is very large, entropy increases and model accuracy drops slightly, suggesting that model prediction is less certain and accurate. The reason here is that when $K = 80$, the number of clients, hence datapoints, assigned to each mode significantly drop. This in turn decreases learning accuracy specifically when datasets are relatively small and a limited budget exists for communication rounds.

	K=3	K=5	K=20	K=40	K=80
Test acc	97.49 ± 0.2	97.85 ± 0.02	98.19 ± 0.08	98.17 ± 0.01	97.98 ± 0.03
Acc max	95.50	95.60	95.57	95.53	94.84
Acc min	94.97	95.00	94.13	93.54	92.74
Avg entropy	0.16	0.15	0.16	0.18	0.20

Table 4: Sensitivity of number of modes K on MNIST. Acc max/min is the maximum/minimum testing accuracy across all modes in the ensemble. Avg Entropy is the average of the entropy of the empirical predictive distribution across all modes.

6 Conclusion & potential extensions

This paper proposes *Fed-ensemble*: an ensembling approach for FL which learns K modes and obtains predictions via model averaging. We show, both theoretically and empirically, that *Fed-ensemble* is efficient in reducing variance and achieving better generalization performance in FL compared to single mode approaches. Beyond ensembling, *Fed-ensemble* may find value in meta-learning where modes acts as multiple learned initializations and clients download ones with optimal performance based on some training performance metric. This may be a route worthy of investigation. Another potential extension is instead of using random permutations, one may send specific modes to individual clients based on training loss or gradient norm, etc., from previous communication rounds. The idea is to increase assignments for modes with least performance or assign such modes to clients were they mal-performed in an attempt to improve the worst performance cases. Here it is interesting to understand if theoretical guarantees still hold in such settings.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8141–8150. Curran Associates, Inc., 2019.

[3] Keith Bonawitz, Hubert Eichner, and et al. Towards federated learning at scale: System design. In *MLSys*, 2019.

[4] Walter De Brouwer. The federated future is ready for shipping. `https://medium.com/_doc_ai/the-federated-future-is-ready-for-shipping-d17ff40f43e3`. Accessed: 2020-07-18.

[5] Hong-You Chen and Wei-Lun Chao. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[7] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. In *Arxiv*, 2020.

[8] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *32nd Conference on Neural Information Processing Systems*, 2018.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[10] Andrew Gordon and Wilson Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Arxiv*, 2020.

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

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[14] Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning convergence analysis. In *Arxiv*, 2021.

[15] Arthur Jacot, Franck Gabriel, and Clement Hongler. The asymptotic spectrum of the hessian of dnn throughout training. In *International Conference on Learning Representations*, 2020.

[16] Arthur Jacot, Frank Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

[17] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[18] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[19] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *NeurIPS*, 2018.

[20] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, pages 1–26, 2020.

[21] Fan Lai, Yinwei Dai, Xiangfeng Zhu, and Mosharaf Chowdhury. FedScale: Benchmarking model and system performance of federated learning. In *Arxiv arxiv.org/abs/2105.11367*, 2021.

[22] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *33rd Conference on Neural Information Processing Systems*, 2019.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, and Virginia Smith Ameet Talwalkar. Federated optimization in heterogeneous networks. *Proceedings of the 3rd MLSys Conference*, 2018.

[24] Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose bayesian inference algorithm. In *NeurIPS*, 2016.

[25] Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan. A double residual compression algorithm for efficient distributed learning. In *Arxiv*, 2019.

[26] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *International Conference on Learning Representations*, 2021.

[27] Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. 2017.

[28] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agueray Arcas. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017.

[29] Konstantin Mishchenko, Eduard Gorbunov, Martin Takac, and Peter Richtarik. Distributed learning with compressed gradient differences. In *Arxiv*, 2019.

[30] Jose G Moreno-Torres, Troy Raeder, RocíO Alaiz-RodríGuez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[31] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[32] Reese Pathak and Martin J. Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. In *NeurIPS*, 2020.

[33] Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. *ICLR*, 2019.

[34] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[35] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[36] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[37] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

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

[39] 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):1–19, 2019.

[40] Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. In *NeurIPS*, 2020.

[41] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. *ICLR*, 2019.

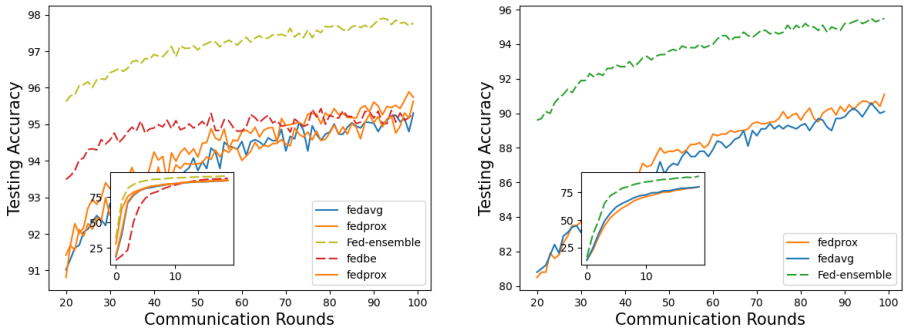
[42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[43] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. In *Arxiv*, 2020.

7 Additional Experimental Details

In this section we articulate experimental details, and present round-to-accuracy figures to better visualize the training process. In all experiments, we decay learning rates by a constant of 0.99 after 10 communication rounds. On image datasets, we use stochastic gradient descent (SGD) as the client optimizer, while on the Shakespear dataset, we train using Adam. Every set of experiment can be done on single Tesla V100 GPU.

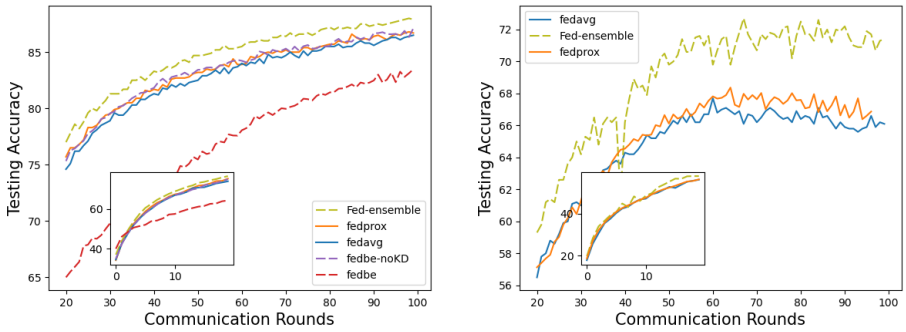
Fig 4 is the round-to-accuracy of several algorithms on MNIST dataset. Fed-ensemble outperforms all single mode federated algorithms and Fedbe.



(a) MNIST iid. (b) MNIST noniid.

Figure 4: Training process on MNIST

Fig 6 shows the result on CIFAR10. Fed-ensemble has better generalization performance from the very beginning.



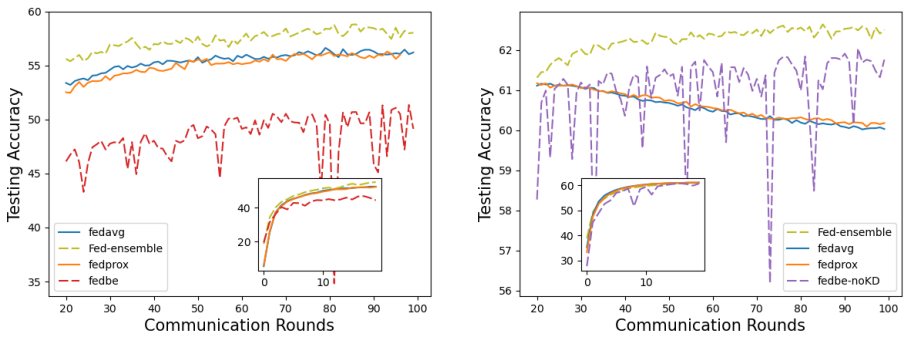
(a) CIFAR10 iid. (b) CIFAR10 noniid.

Figure 5: Training process on CIFAR10

Fig 6(a) plots round-to-accuracy on CIFAR100. Also, there is a clear gap between Fed-ensemble and FedAvg/FedProx.

The comparison between Fed-ensemble and single mode algorithms is even more conspicuous on Shakespeare. The next-word-prediction model we use has 2 LSTM layers, each with 256 hidden states. Fig 6(b) shows that FedAvg and FedProx suffers from severe overfitting when communication rounds exceed 20, while Fed-ensemble continues to improve. Fedbe is unstable on this dataset.

On OpenImage dataset, we simulate the real FL deployment in Google [3], where we select 130 clients to participate in training, but only the updates from the first 100 completed clients are aggregated in order to mitigate stragglers.



(a) Training process on CIFAR100 (b) Training process on Shakespeare

Figure 6: Larger datasets

8 Additional Details on Deriving the Variational Objective

As discussed in Sec. 3.2 the objective is to minimize the KL-divergence between the variational and true posterior distributions.

$$\begin{aligned} &\min D_{KL}\left[q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\right] \\ &= \mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}\left[ln\, q(\boldsymbol{w}) - ln\, p(\boldsymbol{w}|\boldsymbol{D})\right], \end{aligned} \tag{6}$$

In this section, we use Gaussian mixture as variational distribution and then take limit $\sigma \to 0$ to the result. Given (6), $D_{KL}\left[q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\right]$ can be decomposed by conditional expectation law:

$$\mathbb{E}_{k\sim Uniform(K)}\left[\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\left[ln\, q(\boldsymbol{w}) - ln\, p(\boldsymbol{D}|\boldsymbol{w})\right]\right],$$

where *Uniform(K)* denotes a uniform distribution on $[K]$. When σ is very small, different mode centers $(\boldsymbol{w}_k)$ are well-separated such that $q(\boldsymbol{w})$ is close to zero outside a small vicinity of $\boldsymbol{w}_k$. Thus,

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\left[\ln\, q(\boldsymbol{w})\right] \\ &\approx \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\left[-\frac{1}{2\sigma^2}\left\|\boldsymbol{w}-\boldsymbol{w}_k\right\|^2 - d\ln\sigma\right] + C \\ &= -\frac{1}{2} - d\ln\ \sigma + C \end{aligned} \tag{7}$$

where C is a constant. Thus this term is independent of k. Now taking a first order Taylor expansion of $ln\, p(\boldsymbol{w}|\boldsymbol{D})$ around $\boldsymbol{w}_k$, we have

$$ln\, p(\boldsymbol{w}|\boldsymbol{D}) \approx ln\, p(\boldsymbol{w}_k|\boldsymbol{D}) + (\boldsymbol{w}-\boldsymbol{w}_k)^T \nabla ln\, p(\boldsymbol{w}_k|\boldsymbol{D}).$$

After taking expectation on $\boldsymbol{w}$, we have $\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\left[ln\, p(\boldsymbol{w}|\boldsymbol{D})\right] \approx ln\, p(\boldsymbol{w}_k|\boldsymbol{D})$ in the small σ limit. As a result, by conditional expectation, the KL-divergence reduces to:

$$\begin{aligned} &D_{KL}\left(q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\right) \\ &= \mathbb{E}_{k\sim Uniform(K)}\left[-ln\, p(\boldsymbol{w}_k|\boldsymbol{D})\right] + C \\ &= \frac{1}{K}\sum_{k=1}^{K} -\ln\, p(\boldsymbol{w}_k) - \ln\, p(\boldsymbol{D}|\boldsymbol{w}_k) + C'. \end{aligned} \tag{8}$$

where $p(\boldsymbol{D}|\boldsymbol{w}_k)$ is a function of $\boldsymbol{w}_k$ and $p(\boldsymbol{w}_k)$ is the prior p.d.f. of $p(\boldsymbol{w})$ when $\boldsymbol{w} = \boldsymbol{w}_k$.

In (8), $p(\boldsymbol{w}_k)$ acts as a regularizer on negative log-likelihood loss $p(\boldsymbol{D}|\boldsymbol{w}_k)$. When the prior is Gaussian centered at 0, the prior term simply becomes L_2 regularization.

9 Proof of theorem 4.1 and 4.2

In section, we discuss the training of linear models and neural networks in the sufficiently overparametrized regime, and prove theorem 4.1 and 4.2. This section is organized as following: in sec 9.1, we introduce some needed notations and prove some helper lemmas. In sec 9.2, we describe the dynamics of training linear model using Fed-ensemble. In sec 9.3, we prove the global convergence of training loss of Fed-ensemble, and show that neural tangent kernel is fixed during training of Fed-ensemble at the limit of width $l \to \infty$. The result is a refined version of theorem 4.1. In sec 9.4 we will show that the training can indeed be well approximated by kernel method in the extreme wide limit. Then combining sec 9.4 with sec 9.3, we can prove theorem 4.2.

9.1 Notations and Some lemmas

In this section we will introduce some notations, followed by some lemmas.

For an operator A, we use $\|A\|_{op}$ to denote its operator norm, i.e. largest eigenvalue of A. If A is a matrix, $\|A\|_F$ is its Frobenius norm. Generally we have $\|A\|_F > \|A\|_{op}$. $\lambda_{min}(A)$ is the smallest eigenvalue of A.

The update of K different modes in the ensemble described by algorithm 1 in the main text is decoupled, thus we can monitor the training of each mode separately and drop the subscript k. We focus on one mode and use $\boldsymbol{w}$ to denote the weight vector of this mode. We define $g(\boldsymbol{w})$ to be an n-dimensional error vector of training data on all clients, $g(\boldsymbol{w}) = f_{\boldsymbol{w}}(X) - Y$. Its j-th component is $f_{\boldsymbol{w}}(x_j) - y_j$. Notice that g contains the datapoints from all clients. Similarly we can define g_i to be the error vector of the i-th client $g_i(\boldsymbol{w}) = f_{\boldsymbol{w}}(X_i) - Y_i$. $g(\boldsymbol{w})$ is thus the concatenation of all $g_i(\boldsymbol{w})$. Also, we can define a $n_i \times n$ projection matrix $\mathcal{P}_i$:

$$\mathcal{P}_i = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ & \vdots & & & & \ddots & & & \vdots & \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix}$$

It is straightforward to verify that $g_i(\boldsymbol{w}) = \mathcal{P}_i g(\boldsymbol{w})$. As introduced in the main text, we use $\boldsymbol{w}(t)$ to denote the weight vector $\boldsymbol{w}$ after t iterations of local gradient descent. Suppose in each communication round, every client performs τ steps of gradient descent, then $\boldsymbol{w}(tK\tau + r\tau)$ naturally means the weight vector at the beginning of r-th communication round of t-th age. We use $J(\boldsymbol{w})$ to denote the $d \times n$ gradient matrix, whose ij-th element $\frac{\partial}{\partial w_i} f_{\boldsymbol{w}}(x_j)$. Similarly, $J_i(\boldsymbol{w})$ is a $d \times n_i$ gradient matrix on client i, whose kj-th element is $\frac{\partial}{\partial w_k} f_{\boldsymbol{w}}(x_j)$.

We use η to denote the constant stepsize of gradient descent in local client training. For NTK parametrization, stepsize should decrease with the increase of network width, hence we rewrite η as $\eta = \frac{\eta_0}{l}$, where η_0 is a new constant independent of neural network width l. We use $\mathcal{S}(t, r)$ to denote the subset of all clients participating in training of the r-th communication round of the t-th age. We will assume that in one communication round, the number of datapoints on all clients that participated in training is the same: $\sum_{i \in \mathcal{S}(t,r)} n_i = \bar{n}, \forall r$. This assumption is roughly satisfied in our experiments. With some modifications in the weighting parameters in algorithm 1, the equal-datapoint assumption can be alleviated.

The following lemma is proved as Lemma 1 in [22]:

Lemma 9.1. *Local Lipschitz of Jacobian. If (i) the training set is consistent: for any $x_i \neq x_{i'}$, $y_i \neq y_{i'}$. (ii) activation function of the neural nets ϕ satisfies $\phi(0)$, $\|\phi'\|_\infty$, $\frac{|\phi'(x) - \phi'(x')|}{|x - x'|} < \infty$. (iii) $\|x\| \leq 1$ for all input x, then there are constants $C^{(l)}, C_1^{(l)}, ...C_N^{(l)}$ such that for every $C > 0$, with high probability over random initialization the following holds:*

$$\begin{cases} \frac{1}{\sqrt{l}} \|J_i(\boldsymbol{w}) - J_i(\tilde{\boldsymbol{w}})\|_F & \leq C_i^{(l)} \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \\ \frac{1}{\sqrt{l}} \|J_i(\boldsymbol{w})\|_F & \leq C_i^{(l)} \end{cases}$$

$\forall \boldsymbol{w}, \tilde{\boldsymbol{w}} \in B\left(\boldsymbol{w}_0, Cl^{-\frac{1}{2}}\right)$ *for every i=1,2,...N, and also:*

$$\begin{cases} \frac{1}{\sqrt{l}}\|J(\boldsymbol{w}) - J(\tilde{\boldsymbol{w}})\|_F & \leq C^{(l)}\|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|_2 \\ \frac{1}{\sqrt{l}}\|J(\boldsymbol{w})\|_F & \leq C^{(l)} \end{cases}$$

$\forall \boldsymbol{w}, \tilde{\boldsymbol{w}} \in B\left(\boldsymbol{w}_0, Cl^{-\frac{1}{2}}\right)$, *where*

$$B\left(\boldsymbol{w}_0, R\right) := \{\boldsymbol{w} : \|\boldsymbol{w} - \boldsymbol{w}_0\|_2 < R\}$$

Lemma 9.1 discusses the sufficient conditions for the Lipschitz continuity of Jacobian. In the following derivations, we use the result of Lemma 9.1 directly.

In the proof, we extensively use Taylor expansions over the product of local stepsize and local training epochs τ. As we will show, the first order term can yield the desired result. We will bound the second order term and neglect higher order terms by incorporating them into an $o(\eta_0^2\tau^2)$ residual. We abuse this notation a little: $o\left(\eta_0^2\tau^2\right)$ sometimes denote a constant containing orders higher than $\eta_0^2\tau^2$, sometimes a vector whose L_2 norm is $o\left(\eta_0^2\tau^2\right)$, sometimes an operator whose operator norm is $o\left(\eta_0^2\tau^2\right)$. The exact meaning should be clear depending on the context. Taylor expansion is feasible when $\eta_0\tau$ is small enough. Extensive experimental results confirm the validity of Taylor expansion treatment.

Lemma 9.2. *For K operators A_1, A_2,...A_K, if their operator norm is bounded by ρ_i, $i = 1 \cdots K$ respectively, we have:*

$$\begin{aligned} & e^{-\eta\tau A_K} e^{-\eta\tau A_{K-1}} ... e^{-\eta\tau A_1} \\ & = e^{-\eta\tau(A_K + A_{K-1} + ... + A_1)} + \frac{1}{2}\eta^2\tau^2 \sum_{i>j} [A_i, A_j] + o\left(\eta^2\tau^2\right) \end{aligned} \tag{9}$$

where $[A_i, A_j]$ is defined as the commutator of operator A_i and A_j: $[A_i, A_j] = A_iA_j - A_jA_i$. Here $o\left(\eta^2\tau^2\right)$ denotes some operator whose operator norm is $o\left(\eta\tau^2\right)$.

Lemma 9.2 connects the product of exponential to the exponential of sum. This is an extended version of Baker Campbell Hausdorff formula. We still give a simple proof at the end of this section for completeness.

Lemma 9.3 bounds the difference between the average of exponential and exponential of average:

Lemma 9.3. *For K operators A_1, A_2,...A_K, and K non-negative values $p_1, ...p_K$ satisfying p_1+...+$p_K = 1$, we have:*

$$\begin{aligned} & \sum_{i=1}^{K} p_i e^{-\eta\tau A_i} \\ & = e^{-\eta\tau \sum_{i=1}^{K} p_i A_i} + \frac{\eta^2\tau^2}{2}\left(\sum_{i=1}^{K} p_i A_i^2 - \left(\sum_{i=1}^{K} p_i A_i\right)^2\right) + o\left(\eta^2\tau^2\right) \end{aligned} \tag{10}$$

Proof. The proof is straightforward by using Taylor expansion on both sides and retain only highest order terms in residual. For an operator A, its exponential is defined as:

$$\begin{aligned} & \exp\left(-\eta\tau A\right) \\ & = \sum_{k=0}^{\infty} (-1)^k \frac{(\eta\tau A)^k}{k!} \\ & = 1 - \eta\tau A + \frac{\eta^2\tau^2}{2}A^2 + \eta^3\tau^3\Delta_A \end{aligned} \tag{11}$$

where Δ_A is defined as:

$$\Delta_A = \sum_{k=3}^{\infty} (-1)^k \frac{(\eta\tau)^{k-3} A^k}{k!} \tag{12}$$

When A's operator norm is bounded by ρ_A, we can also bound the operator norm of Δ_A as:

$$\begin{aligned}\|\Delta_A\| &\leq \sum_{k=3}^{\infty} \frac{(\eta\tau)^{k-3}\|A\|^k}{k!} \\ &= \|A\|^3 \sum_{k=3}^{\infty} \frac{(\eta\tau)^{k-3}\|A\|^{k-3}}{k!} \\ &\leq \frac{\|A\|^3}{6} \sum_{k=3}^{\infty} \frac{(\eta\tau)^{k-3}\|A\|^{k-3}}{(k-3)!} \\ &= \frac{\rho_A^3}{6}\exp\left(\eta\tau\rho_A\right)\end{aligned}$$

Then we can rewrite two sides of the equations using Δ_{A_i} as below:

$$\begin{aligned}&\sum_{i=1}^{K} p_i e^{-\eta\tau A_i} - e^{-\eta\tau\sum_{i=1}^{K} p_i A_i} \\ &= \sum_{i=1}^{K} p_i \left(1 - \eta\tau A_i + \frac{1}{2}\eta^2\tau^2 A_i^2 + \eta^3\tau^3\Delta_{A_i}\right) \\ &- \left(1 - \eta\tau\sum_{i=1}^{K} p_i A_i + \frac{1}{2}\eta^2\tau^2\left(\sum_{i=1}^{K} p_i A_i\right)^2 + \eta^3\tau^3\Delta_{\sum_{i=1}^{K} p_i A_i}\right) \\ &= \frac{1}{2}\eta^2\tau^2\left(\sum_{i=1}^{K} p_i A_i^2 - \left(\sum_{i=1}^{K} p_i A_i\right)^2\right) + \eta^3\tau^3\left(\sum_{i=1}^{K} p_i\Delta_{A_i} - \Delta_{\sum_{i=1}^{K} p_i A_i}\right)\end{aligned}$$

Since we have shown $\|\Delta_{A_i}\|$'s are bounded, the last term $\eta^3\tau^3\left(\sum_{i=1}^{K} p_i\Delta_{A_i} - \Delta_{\sum_{i=1}^{K} p_i A_i}\right) = O(\eta^3\tau^3)$ □

Proof of Lemma 9.2

Proof. We adopt the notations in the proof of lemma 9.3 from (11) and (12). Then we have:

$$\begin{aligned}&e^{-\eta\tau A_K}e^{-\eta\tau A_{K-1}}...e^{-\eta\tau A_1} - e^{-\eta\tau(A_K + A_{K-1} + ... + A_1)} \\ &= \left(1 - \eta\tau A_K + \frac{\eta^2\tau^2}{2}A_K^2 + \eta^3\tau^3\Delta_{A_K}\right)\cdots\left(1 - \eta\tau A_1 + \frac{\eta^2\tau^2}{2}A_1^2 + \eta^3\tau^3\Delta_{A_1}\right) \\ &- \left(1 - \eta\tau\left(A_K + \cdots + A_1\right) + \frac{\eta^2\tau^2}{2}\left(A_K + \cdots + A_1\right)^2 + \eta^3\tau^3\Delta_{A_K + \cdots + A_1}\right) \\ &= \frac{1}{2}\eta^2\tau^2\sum_{i>j}\left[A_i, A_j\right] + o\left(\eta^2\tau^2\right)\end{aligned}$$

□

9.2 Training of Linearized Neural Networks

The derivation in this section is inspired by the kernel regression model from [16]. Now $\Theta(\cdot,\star)$ can be any positive definite symmetric kernel, later we will specify it to be neural tangent kernel. We define the gram operator of client i to be a map from function $f(x)$ into function

$$\Pi_i(f)(x) = \frac{1}{n_i}\sum_{(x_{ij}, y_{ij})\in\mathbf{D_i}} f\left(x_{ij}\right)\Theta\left(x_{ij}, x\right)$$

and also, the gram operator of the entire dataset is defined as:

$$\Pi(f)(x) = \frac{1}{n} \sum_{(x_j, y_j) \in \mathbf{D}} f(x_j)\, \Theta(x_j, x)$$

In the beginning of r-th communication round of age t, $\boldsymbol{w}(tK\tau + r\tau)$ is sent to all clients in some selected strata. As discussed before, we use $S(t, r)$ to denote the set of clients that download mode $\boldsymbol{w}$ in the r-th communication round of age t. Since client i uses gradient descent to minimize the objective, we approximate it with gradient flow:

$$\frac{\partial}{\partial s} f_{\boldsymbol{w}^{(i)}(tK\tau + r\tau + s)}(x) = -\eta \Pi_i f_{\boldsymbol{w}^{(i)}(tK\tau + r\tau + s)}(x)$$

where $\boldsymbol{w}^{(i)}(tK\tau + r\tau + s)$ is the local weight vector on client i starting from $\boldsymbol{w}^{(i)}(tK\tau + r\tau) = \boldsymbol{w}(tK\tau + r\tau)$.

Since Π_i does not evolve over time, after τ epochs, the updated function on client i is:

$$f_{\boldsymbol{w}^{(i)}(tK\tau + (r+1)\tau)}(x) - f^\star(x) = e^{-\eta \Pi_i \tau}(f_{\boldsymbol{w}^{(i)}(tK\tau + r\tau)}(x) - f^\star(x))$$

where $f^\star$ is the ground truth function satisfying $\Pi_i f^\star = 0$ for every i, i.e. it achieves zero training error on every client. After τ epochs, all clients send updated model back to server, and server use *server_update* to calculate new $\boldsymbol{w}$. Since we are training a linear model:

$$f_{\boldsymbol{w}(tK\tau + (r+1)\tau)}(x) = \frac{1}{\sum_{i \in S(t,r)} n_i} \sum_{i \in S(t,r)} n_i f_{\boldsymbol{w}^{(i)}(tK\tau + (r+1)\tau)}(x)$$

Plugging in the equation of $f_{\boldsymbol{w}^{(i)}(tK\tau + (r+1)\tau)}$, we can use lemma 9.3 to derive:

$$\begin{aligned}
&f_{\boldsymbol{w}(tK\tau + (r+1)\tau)}(x) - f^\star(x) \\
&= \frac{1}{\sum_{i \in S(t,r)} n_i} \sum_{i \in S(t,r)} n_i e^{-\eta \Pi_i \tau}(f_{\boldsymbol{w}(tK\tau + r\tau)}(x) - f^\star) \\
&= e^{-\eta \frac{1}{\sum_{i \in S(t,r)} n_i} \sum_{i \in S(t,r)} n_i \Pi_i \tau}(f_{\boldsymbol{w}(tK\tau + r\tau)}(x) - f^\star) + \\
&\frac{1}{2} \mathbf{Var}(\Pi_i)\, \tau^2 \eta^2 (f_{\boldsymbol{w}(tK\tau + r\tau)}(x) - f^\star) + o\left(\eta^2 \tau^2\right)
\end{aligned}$$

where $\mathbf{Var}(\Pi_i)$ is defined as:

$$\mathbf{Var}(\Pi_i) = \frac{1}{\sum_{i \in S(t,r)} n_i} \sum_{i \in S(t,r)} n_i \Pi_i^2 - \left(\frac{1}{\sum_{i \in S(t,r)} n_i} \sum_{i \in S(k,t)} n_i \Pi_i \right)^2$$

We call this term the variance of gram operator Π. When Π_i's are very different from each other, i.e. data distribution is very heterogeneous, this term would be large and when Π_i's are similar, this term would be small. $o\left(\eta^2 \tau^2\right)$ are terms that contain $\eta\tau$ order higher than 2. When $\eta\tau$ are chosen to be not too large, we neglect higher order residuals and retain only the leading terms in further derivations.

We then consider the training of K consecutive communication rounds:

$$\begin{aligned}
&f_{\boldsymbol{w}_{(t+1)K\tau}}(x) - f^\star(x) \\
&= \prod_{l=1}^{K} e^{-\eta \mathbb{E}[\Pi_l] \tau}(f_{\boldsymbol{w}(tK\tau)}(x) - f^\star) + \\
&\frac{1}{2} \tau^2 \eta^2 \sum_j \prod_{l<j} e^{-\eta\tau \mathbb{E}[\Pi_l]} \mathbf{Var}[\Pi_j] \prod_{l>j} e^{-\eta\tau \mathbb{E}[\Pi_l]} (f_{\boldsymbol{w}(tK\tau)}(x) - f^\star) + o\left(\tau^2 \eta^2\right)
\end{aligned} \tag{13}$$

where $\mathbb{E}[\Pi_l]$ is defined as:

$$\mathbb{E}[\Pi_l] = \frac{1}{\sum_{i \in S(t,l)} n_i} \sum_{i \in S(t,l)} n_i \Pi_i$$

We can reorganize operators that apply on $f_{\boldsymbol{w}} - f^\star$. We introduce a new operator U as:

$$U = \left(1 + \frac{1}{2}\tau^2\eta^2 \left(\sum_{j=1}^{K} \mathbf{Var}[\Pi_j] + \sum_{i>j} [\mathbb{E}[\Pi_i], \mathbb{E}[\Pi_j]]\right)\right) e^{-\eta\tau \sum_{l=1}^{K} \mathbb{E}[\Pi_l]}$$

By definition of matrix Π, we know that $\sum_{l=1}^{K} \mathbb{E}\left[\Pi_i\right] = K\Pi$, therefore, we can replace $\sum_{l=1}^{K} \mathbb{E}\left[\Pi_i\right]$ by $K\Pi$. By applying lemma 9.2 and neglecting higher order terms, equation (13) can be rewritten as:

$$\begin{aligned} &f_{\boldsymbol{w}(tK\tau+(r+1)\tau)}(x) - f^\star(x) \\ &= U(f_{\boldsymbol{w}(tK\tau+r\tau)}(x) - f^\star) + o\left(\tau^2\right) \end{aligned} \tag{14}$$

We now derive an upper bound on the eigenvalue of U when the stepsize is small enough. Since the range of each Π_i has finite dimension in the function space, the operator norm of Π_i is finite, and so is the operator norm of the multiplication of Π_i. We use B_1 to denote the maximum operator norm of $\sum_{j=1}^{K} \mathbf{Var}[\Pi_j] + \sum_{i>j} \left[\mathbb{E}\left[\Pi_i\right], \mathbb{E}\left[\Pi_j\right]\right]$ among all possible Π operators.

For any function f in the null space of Π, by assumption (v) in theorem 4.1, it is also in the null space of each Π_i, thus $\mathbf{Var}\left[\Pi_k\right] f = \left[\mathbb{E}\left[\Pi_i\right], \mathbb{E}\left[\Pi_j\right]\right] f = 0$ for any i, j, k. As a result, $Uf = f$.

For any function in the orthogonal space of the null space of Π_i

$$\|Uf\| \leq \left(1 + \frac{1}{2}\tau^2\eta^2 B_1\right) e^{-\eta\tau K\lambda_{min}(\Pi)} \|f\|$$

where $\lambda_{min}(\cdot)$ is the smallest nonzero eigenvalue of an operator. If η and τ is small enough such that

$$\left(1 + \frac{1}{2}\tau^2\eta^2 B_1\right) e^{-\eta\tau K\lambda_{min}(\Pi)} < 1 \tag{15}$$

f will decay to zero in the limit $\lim_{k\to\infty} U^k f = 0$.

As a simple estimate of (15), when $\tau\eta$ is small enough, the leading term solution of equation (15) is $\eta\tau \leq \frac{2K\lambda_{min}(\Pi)}{K\lambda_{min}(\Pi)+B_1}$.

By orthogonal decomposition, every function f can be decomposed into $f = \Delta_0(f) + \Delta_\perp(f)$, where Δ_0 is in the projection into the null space of Π and $\Delta_\perp$ is in the projection into the orthogonal space of the null space of Π. By using relation (14) iteratively, $\Delta_\perp$ will shrink to zero, then we can obtain the limiting result:

$$f_{\boldsymbol{w}_\infty}(x) - f^\star(x) = \Delta_0(f_{\boldsymbol{w}_0}(x) - f^\star(x)) + o\left(\eta^2\tau^2\right) \tag{16}$$

We thus have:

$$f_{\boldsymbol{w}_\infty}(x) = f_{\boldsymbol{w}_0}(x) + \Theta(x, X)\Theta^{-1}(X, X)(y^\star - f_{\boldsymbol{w}_0}(X)) + o\left(\tau^2\right) \tag{17}$$

where $f_{\boldsymbol{w}_0}(X)$ is a vector of dimension n whose i-th component is $f_{\boldsymbol{w}_0}(x_i)$, and $\Theta(X, X)$ is an n by n matrix whose ij-th entry is $\left[\Theta(X, X)\right]_{ij} = \Theta\left(x_i, x_j\right)$. This result shows that although each mode is trained only on a part of clients, when we use block coordinate descent rule, the limiting behavior of individual mode is similar to the case where we train each mode on the entire dataset, with a small higher order discrepancy term that depends on local epochs and data heterogeneity.

As a result, each mode can be regarded as independent draw from Gaussian $\mathcal{GP}\left(m(x), k(x, x')\right)$

$$m(x) = \Theta(x, X)\Theta^{-1}(X, X)y^\star$$

and

$$\begin{aligned} k\left(x, x'\right) &= \mathcal{K}\left(x, x'\right) + \Theta(x, X)\Theta^{-1}\mathcal{K}\Theta^{-1}\Theta(X, x') \\ &- \left(\Theta(x, X)\Theta^{-1}\mathcal{K}\left(X, x'\right) + \Theta(x', X)\Theta^{-1}\mathcal{K}\left(X, x\right)\right) \end{aligned}$$

where $\mathcal{K}(\cdot, \star) = \mathbb{E}_{\boldsymbol{w}(0)\sim p_{init}}\left[f_{\boldsymbol{w}_0}(\cdot) f_{\boldsymbol{w}_0}(\star)\right]$ is the kernel at initialization. As a special case, if $\mathcal{K}(\cdot, \star) = \sigma^2\Theta(\cdot, \star)$, which is the case of toy example in section 5.1 of the main text, the posterior variance is given by:

$$k(x, x') = \sigma^2\left(\Theta\left(x, x'\right) - \Theta(x, X)\Theta^{-1}(X, X)\Theta(X, x')\right).$$

This completes our discussion on training linearized model.

9.3 Convergence and change of neural tangent kernel

Training infinite width neural network by small-stepsize gradient descent is equivalent to training a linear model with fixed kernel. In this subsection, we prove similarly that the tangent kernel remains fixed during Fed-ensemble, following the derivations of [22]. If the neural tangent kernel is fixed, we can adopt the analysis of sec 9.2 to neural network by taking Θ to be the limiting kernel. From the result in sec 9.2, we can prove theorem 4.2 in the main text.

For NTK initialization [16] $f_{\boldsymbol{w}_0}$ converges to a Gaussian process whose mean is zero and whose kernel is denoted as $\mathcal{K}$. Then for any δ_0, there exists R_0, and n_0 such that for every $l \geq l_0$, $\|g(\boldsymbol{w}_0)\|_2 \leq R_0$ with probability at least $1-\delta_0$. Also, in NTK setting, it is proved as Lemma 1 in [15] that the operator norm of Hessian of function $\left\|\frac{1}{\sqrt{l}}\nabla^2 g(\boldsymbol{w})\right\|_{op}$ is bounded above by C_1, and that $C_1 \to 0$ in the limit $l \to \infty$.

Theorem 9.1. *(Convergence of global training error) Under the assumptions listed below:*

1. *$\eta_0\tau$ is small enough such that κ_2 defined in (26) is smaller than 1.*
2. *The width of neural network l is large enough.*

for positive constants $R_0,\ \delta_0 > 0$, the following holds with probability $1-\delta_0$:

$$\begin{cases} \|g(\boldsymbol{w}(tK\tau))\|_2 \leq e^{-\frac{\eta_0\lambda_m}{3\bar{n}}tK\tau}R_0 + o(\eta_0^2\tau^2) \\ \|\boldsymbol{w}(tK\tau) - \boldsymbol{w}(0)\|_2 \leq \dfrac{3\eta_0\bar{n}R_0 l^{-\frac{1}{2}}(1+\kappa_2)C_2}{\lambda_m}\left(1 - e^{-\frac{\eta_0\lambda_m}{3\bar{n}}t\tau}\right) + o(\eta_0^2\tau^2) \end{cases} \tag{18}$$

$\forall t = 1, 2, ...$ And also, $\Theta(\boldsymbol{w}(tK\tau))$ is not very far away from Θ:

$$\|\Theta(\boldsymbol{w}(tK\tau)) - \Theta\| \leq O\left(l^{-\frac{1}{2}}\right)$$

The notations is defined as following: $\bar{n}$ is the number of datapoints in one communication round, C_2 is a constant defined as the summation of all $C_i^{(l)}$'s in lemma 9.1, $C_2 = \sum_{i=1}^N C_i^{(l)}$, and κ_2 is an $O(\eta_0\tau)$ constant defined in (26):

$$\begin{aligned} \kappa_2 &= -1 + \exp\left(\eta_0\tau C_2 + \eta_0^2\tau^2\left(2C_2^4 + 8\frac{R_0}{\sqrt{l}}C_1C_2^2\right)\right) \\ &= O(\tau\eta_0) \end{aligned} \tag{19}$$

Note that the second equation in (18) is the result of theorem 4.1.

Proof. We will prove (18) by induction. At $t = 0$, the second equation is trivially true, and the first reduces to $\|g(\boldsymbol{w}(0))\|_2 \leq R_0$, which is also true with probability at least $1 - \frac{\delta_0}{10}$. We choose proper constants so that Lemma 9.1 holds with probability at least $1 - \frac{\delta}{10}$ over random initialization. Now assume (18) hold for $t = 1, 2, ...\xi$, we will prove they still hold for $t = \xi + 1$. For clarity, we call this induction the induction across ages.

Communication rounds within one age We will prove the following by another level of induction:

$$\begin{cases} \|g(\boldsymbol{w}(\xi K\tau + r\tau))\|_2 \leq (1+\kappa_2)\|g(\boldsymbol{w}(\xi K\tau))\|_2 + o(\tau^2) \\ \|\boldsymbol{w}(\xi K\tau + r\tau) - \boldsymbol{w}(0)\|_2 \leq \dfrac{3\eta_0 R_0 l^{-\frac{1}{2}}(1+\kappa_2)C_2}{\lambda_m}\left(1 - e^{-\frac{\eta_0\lambda_m}{3\bar{n}}\xi\tau}\right) + o(\tau^2) \end{cases} \quad \forall r = 1, 2, ...K \tag{20}$$

We call this induction the induction across communication rounds.

Obviously, (20) holds for $r = 0$, which is the inductive assumption of induction across different ages. Now we assume that (20) hold for communication rounds $r' = 1, 2, ...r$, and prove the inequalities for $r' = r + 1$.

As discussed before, we denote $\boldsymbol{w}(\xi K\tau + r\tau)$ the weights after the server update step of the r-th communication round of age ξ. In the beginning of next communication round, $\boldsymbol{w}(\xi K\tau + r\tau)$ is sent to all clients for some selected strata. Under gradient descent rule, the client update can be approximated by the differential equation:

$$\begin{aligned}&\frac{d}{dt_i}\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i)\\&= -\frac{\eta}{n_i}J_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))g_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))\end{aligned} \tag{21}$$

where t_i is the local training epoch of client i at communication round r, $\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i)$ is the weight on client i starting from $\boldsymbol{w}^{(i)}(\xi K\tau + r\tau) = \boldsymbol{w}(\xi K\tau + r\tau)$, and other notations are defined accordingly. As a result, the dynamics of g is given by:

$$\begin{aligned}&\frac{d}{dt_i}g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))\\&= -\frac{\eta}{n_i}J(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))^T J_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))g_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))\end{aligned} \tag{22}$$

We denote Θ_i as $\Theta_i(t_i) = \frac{1}{l}J(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))^T J_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i))\mathcal{P}_i$, then the solution to the differential equation is:

$$g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t_i)) = \exp\left(-\frac{\eta_0}{n_i}\int_0^\tau \Theta_i(t_i)dt_i\right)g(\boldsymbol{w}(\xi K\tau + r\tau))$$

By integral mean value theorem, there exists a $\bar{t_i} \in [0, \tau]$ such that $\int_0^\tau \Theta_i(t_i)dt_i = \tau\Theta_i(\bar{t_i})$, thus

$$g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)) = \exp\left(-\frac{\eta_0}{n_i}\Theta_i(\bar{t_i})\tau\right)g(\boldsymbol{w}(\xi K\tau + r\tau))$$

After τ local epochs, all clients will send model to server, and the server performs *server_update* to calculate $\boldsymbol{w}(\xi K\tau + (r+1)\tau)$ based on received local modes. As defined before, $S(\xi, r)$ is the indices of clients that download model $\boldsymbol{w}(\xi K\tau + r\tau)$ in communication round r of age ξ, then the server update function can be written as:

$$\boldsymbol{w}(\xi K\tau + (r+1)\tau) = \frac{1}{\sum_{i\in S(\xi,r)} n_i}\sum_{i\in S(\xi,r)} n_i\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)$$

The averaged error function is given by:

$$\begin{aligned}&\frac{1}{\sum_{i\in S(\xi,r)} n_i}\sum_{i\in S(\xi,r)} n_i g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau))\\&= g(\boldsymbol{w}(\xi K\tau + (r+1)\tau)) + \eta_0^2\tau^2 v_1^{(\xi,r)} + o(\eta_0^2\tau^2)\end{aligned} \tag{23}$$

We then try to estimate the averaged error vector by the function parametrized by averaged weights and bound their difference. To do so, we use a simple notation $v_1^{(\xi,r)}$ to denote all the second order terms in (23):

$$v_1^{(\xi,r)} = \frac{1}{\eta_0^2\tau^2\sum_{i\in S(\xi,r)} n_i}\sum_{i\in S(\xi,r)} n_i\Delta\boldsymbol{w}_i^T\nabla^2 g(\boldsymbol{w}(\xi K\tau + (r+1)\tau))\Delta\boldsymbol{w}_i$$

where

$$\begin{aligned}\Delta\boldsymbol{w}_i &= \boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau) - \boldsymbol{w}(\xi K\tau + (r+1)\tau)\\&= \boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau) - \boldsymbol{w}(\xi K\tau + r\tau) - (\boldsymbol{w}(\xi K\tau + (r+1)\tau) - \boldsymbol{w}(\xi K\tau + r\tau))\end{aligned}$$

For each client i, by integral mean value theorem, there exists a $\tilde{t_i} \in [0, \tau]$, such that:

$$\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau) - \boldsymbol{w}(\xi K\tau + r\tau) = -\frac{\eta\tau}{n_i}J_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tilde{t_i}))g_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tilde{t_i}))$$

As a result:

$$\begin{aligned}
\Delta \boldsymbol{w}_i &= -\frac{\eta\tau}{n_i} J_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tilde{t}_i)) g_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tilde{t}_i)) \\
&+ \frac{\eta\tau}{\sum_{i\in S(\xi,r)} n_i} \left(\sum_{j\in S(\xi,r)} J_j(\boldsymbol{w}^{(j)}(\xi K\tau + r\tau + \tilde{t}_i)) g_j(\boldsymbol{w}^{(j)}(\xi K\tau + r\tau + \tilde{t}_i)) \right) \\
&= -\frac{\eta\tau}{n_i} J_i(\boldsymbol{w}(\xi K\tau + r\tau)) g_i(\boldsymbol{w}(\xi K\tau + r\tau)) \\
&+ \frac{\eta\tau}{\sum_{i\in S(\xi,r)} n_i} \left(\sum_{j\in S(\xi,r)} J_j(\boldsymbol{w}(\xi K\tau + r\tau)) g_j(\boldsymbol{w}(\xi K\tau + r\tau)) \right) + o(\tau) \\
&= \frac{\eta_0\tau}{\sqrt{l}} D_1^{(\xi,r,i)} g(\boldsymbol{w}(\xi K\tau + r\tau)) + o(\tau)
\end{aligned}$$

where $D_1^{(\xi,r,i)}$ is an operator defined as:

$$D_1^{(\xi,r,i)} = -\frac{1}{n_i\sqrt{l}} J_i(\boldsymbol{w}(\xi K\tau + r\tau))\mathcal{P}_i + \frac{1}{\sum_{i\in S(\xi,r)} n_i\sqrt{l}} \left(\sum_{j\in S(\xi,r)} J_j(\boldsymbol{w}(\xi K\tau + r\tau))\mathcal{P}_j \right)$$

And its operator norm is upper bounded by $2C_2$. We replace $\tilde{t}_i$ in the third equality by 0. Due to continuity, the difference is proportional to $\tilde{t}_i \le \tau$, thus can be absorbed into $o(\tau)$ term after multiplying τ. Therefore, v_1 is

$$v_1^{(\xi,r)} = \frac{1}{\sum_{i\in S(\xi,r)} n_i l} \sum_{i\in S(\xi,r)} n_i \left(g(\boldsymbol{w}(\xi K\tau + r\tau))\right)^T \left(D_1^{(\xi,r,i)}\right)^T \nabla^2 g D_1^{(\xi,r,i)} g(\boldsymbol{w}(\xi K\tau + r\tau))$$

We can define a new operator $D_1^{(\xi,r)}$ as:

$$D_1^{(\xi,r)} = \frac{\left(g(\boldsymbol{w}(\xi K\tau + r\tau))\right)^T}{\sum_{i\in S(\xi,r)} n_i} \sum_{i\in S(\xi,r)} n_i \left(D_1^{(\xi,r,i)}\right)^T \nabla^2 g D_1^{(\xi,r,i)}$$

The operator norm of $D_1^{(\xi,r)}$ is thus bounded by:

$$\left\|D_1^{(\xi,r)}\right\|_{op} \le \left\|g(\boldsymbol{w}(\xi K\tau + r\tau))\right\|_2 \frac{1}{\sum_{i\in S(\xi,r)} n_i} \sum_{i\in S(\xi,r)} n_i \left\|\left(D_1^{(\xi,r,i)}\right)^T \nabla^2 g D_1^{(\xi,r,i)}\right\|_{op}$$

By inductive assumption (20), both $\left\|g(\boldsymbol{w}(\xi K\tau + r\tau))\right\|_2$ and $\left\|\frac{1}{\sqrt{l}}\nabla^2 g\right\|$ are upper bounded, also by lemma 9.1, $\left\|D_1^{(\xi,r,i)}\right\|_{op}$ is upper bounded by $2C_2$, thus $\|D_1^{(\xi,r)}\|_{op}$ is upper bounded by:

$$\|D_1^{(\xi,r)}\|_{op} \le \frac{2}{\sqrt{l}} R_0 C_1 \left(2C_2\right)^2$$

As a result, error vector $g(\boldsymbol{w}(\xi K\tau+(r+1)\tau))$ can be calculated as:

$$\begin{aligned}
&g(\boldsymbol{w}(\xi K\tau+(r+1)\tau))\\
&=\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}n_i\exp\left(-\frac{\eta_0}{n_i}\Theta_i(\bar{t_i})\tau\right)g(\boldsymbol{w}(\xi K\tau+r\tau))\\
&-\eta^2\tau^2D_1^{(\xi,r)}g(\boldsymbol{w}(\xi K\tau+r\tau))+o(\tau^2)\\
&=\exp\left(-\eta_0\tau\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right)g(\boldsymbol{w}(\xi K\tau+r\tau))\\
&+\eta_0^2\tau^2D_2^{(\xi,r)}g(\boldsymbol{w}(\xi K\tau+r\tau))+o(\tau^2)\\
&=\left(1+\eta_0^2\tau^2D_2^{(\xi,r)}\right)\exp\left(-\eta_0\tau\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right)g(\boldsymbol{w}(\xi K\tau+r\tau))+o(\tau^2)
\end{aligned}\tag{24}$$

where $D_2^{(\xi,r)}$ is defined as

$$D_2^{(\xi,r)}=\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}n_i\Theta_i^2(\bar{t_i})-\left(\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}n_i\Theta_i(\bar{t_i})\right)^2-D_1^{(\xi,r)}\tag{25}$$

The second equality of (24) comes from lemma 9.3, and the third combines $\eta_0^2\tau^2D_2^{(\xi,r)}$ into product term.

We can replace $\bar{t_i}$ in (25) by 0, since the difference can be absorbed into $o(\tau^2)$ terms. By induction hypothesis, $\|\Theta_i(0)\|_{op}$ is bounded by $\left(C_i^{(l)}\right)^2\le C_2^2$. Hence, the operator norm of $D_2^{(\xi,r)}$ is also bounded by constants $\kappa_2^{(\xi,r)}=2C_2^4+8\frac{R_0}{\sqrt{l}}C_1C_2^2$. Therefore:

$$\begin{aligned}
&\|g(\boldsymbol{w}(\xi K\tau+(r+1)\tau))\|_2\\
&\le(1+\eta_0^2\tau^2\kappa_2^{(\xi,r)})\left\|\exp\left(-\eta_0\tau\frac{1}{\sum_{i\in S(\xi,r)}n_i}\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right)\right\|_{op}\\
&\|g(\boldsymbol{w}(\xi K\tau+r\tau))\|_2+o(\tau^2)\\
&\le\exp\left(\eta_0^2\tau^2\kappa_2^{(\xi,r)}\right)\exp\left(\eta_0\tau\sum_{i\in S(\xi,r)}C_i^{(l)}\right)\|g(\boldsymbol{w}(\xi K\tau+r\tau))\|_2+o(\tau^2)\\
&\le\cdots\\
&\le\exp\left(\eta_0\tau\sum_{r'}^{r}\sum_{i\in S(\xi,r)}C_i^{(l)}+\eta_0^2\tau^2\kappa_2^{(\xi,r')}\right)\|g(\boldsymbol{w}(\xi K\tau))\|_2+o(\tau^2)\\
&\le(1+\kappa_2)\|g(\boldsymbol{w}(\xi K\tau))\|_2+o(\tau^2)
\end{aligned}$$

where

$$\kappa_2=-1+\exp\left(\eta_0\tau C_2+\eta_0^2\tau^2\left(2C_2^4+8\frac{R_0}{l}C_1C_2^2\right)\right)\tag{26}$$

We thus prove the first equation in (20).

For the second one, we should consider the dynamics of $\boldsymbol{w}$. On each client's side,

$$\frac{d}{dt_i}\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i)=-\frac{\eta}{n_i}J_i(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))g_i(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))$$

as a result:

$$\begin{aligned}
&\frac{d}{dt_i}\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i)-\boldsymbol{w}_0\right\|_2\\
&\leq\left\|\frac{d}{dt_i}\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i)\right\|_2\\
&=\frac{\eta_0}{ln_i}\left\|J_i(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))\mathcal{P}_i g(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))\right\|_2\\
&\leq\frac{\eta_0}{ln_i}\left\|J_i(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))\mathcal{P}_i\right\|_{op}\left\|g(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i))\right\|_2
\end{aligned}$$

We use another layer of induction here. If $\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau)(t_i')-\boldsymbol{w}_0\right\|_2\leq\frac{3R_0l^{-\frac{1}{2}}(1+\kappa_2)C_2}{\lambda_m}\left(1-e^{-\frac{\eta_0\lambda_m}{3}\xi K\tau}\right)+\eta_0R_0C_i^{(l)}t_i'l^{-\frac{1}{2}}n_i^{-1}(1+\kappa_2)\exp\left(-\frac{1}{3}\eta_0\lambda_m\xi K\tau\right)$, for all $t_i'<t_i$, then it's also true that $\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i)-\boldsymbol{w}_0\right\|_2\leq 3R_0l^{-\frac{1}{2}}(1+\kappa_2)C_2\lambda_m^{-1}$, by lemma 9.1, we know:

$$l^{-\frac{1}{2}}\left\|J_i(\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i'))\mathcal{P}_i\right\|_{op}\leq C_i^{(l)}$$

for any $t_i'<t_i$. By integral mean value theorem, we finally have:

$$\begin{aligned}
&\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+t_i)-\boldsymbol{w}(0)\right\|_2-\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau)-\boldsymbol{w}(0)\right\|_2\\
&\leq\eta_0R_0C_i^{(l)}t_il^{-\frac{1}{2}}n_i^{-1}(1+\kappa_2)\exp\left(-\frac{1}{3\bar{n}}\eta_0\lambda_m\xi\tau\right)
\end{aligned}$$

The upper bound thus also holds for t_i, $\forall t_i\in[0,\tau]$. We can set $t_i=\tau$:

$$\begin{aligned}
&\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau+\tau)-\boldsymbol{w}(0)\right\|_2-\left\|\boldsymbol{w}^{(i)}(\xi K\tau+r\tau)-\boldsymbol{w}(0)\right\|_2\\
&\leq\eta_0R_0C_i^{(l)}\tau l^{-\frac{1}{2}}n_i^{-1}(1+\kappa_2)\exp\left(-\frac{1}{3\bar{n}}\eta_0\lambda_m\xi\tau\right)
\end{aligned}$$

By convexity of the ℓ_2 norm, we know that

$$\begin{aligned}
&\|\boldsymbol{w}(\xi K\tau+(r+1)\tau)-\boldsymbol{w}(0)\|_2-\|\boldsymbol{w}(\xi K\tau+r\tau)-\boldsymbol{w}(0)\|_2\\
&\leq\eta_0\tau R_0l^{-\frac{1}{2}}(1+\kappa_2)\frac{\sum_{i\in S(\xi,r)}C_i^{(l)}}{\sum_{i\in S(\xi,r)}n_i}\exp\left(-\frac{1}{3\bar{n}}\eta_0\lambda_m\xi\tau\right)\\
&\leq\eta_0\tau R_0l^{-\frac{1}{2}}(1+\kappa_2)C_2\exp\left(-\frac{1}{3\bar{n}}\eta_0\lambda_m\xi\tau\right)
\end{aligned}$$

where $C_2=\sum_{i=1}^N C_i^{(l)}$.

As a result:

$$\begin{aligned}
&\|\boldsymbol{w}(\xi K\tau+(r+1)\tau)-\boldsymbol{w}(0)\|_2\\
&\leq r\eta_0\tau R_0l^{-\frac{1}{2}}(1+\kappa_2)C_2\exp\left(-\frac{1}{3\bar{n}}\eta_0\lambda_m\xi\tau\right)+\|\boldsymbol{w}(\xi K\tau)-\boldsymbol{w}(0)\|_2\\
&\leq\frac{3\eta_0R_0l^{-\frac{1}{2}}(1+\kappa_2)C_2}{\lambda_m}\left(1-e^{-\frac{\eta_0\lambda_m}{3\bar{n}}(\xi+1)\tau}\right)
\end{aligned}$$

Thus the second equation in (20) holds. Also, the second equation in (18) holds.

Exponential decrease of error Now we come to the induction across different ages, in which we will prove that the norm of error vector decreases at an exponential rate. From (24):

$$\begin{aligned}
&g(\boldsymbol{w}((\xi+1)K\tau)) \\
&= \left(\prod_{r=1}^{K}\left(1+\eta_0^2\tau^2 D_2^{(\xi,r)}\right)\exp\left(-\eta_0\tau\frac{1}{\sum_{i\in S(\xi,r)} n_i}\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right)\right) g(\boldsymbol{w}(\xi K\tau)) + o(\tau^2) \\
&= \left(1+\eta_0^2\tau^2 D_3^{(\xi)}\right)\exp\left(-\eta_0\tau\sum_r\frac{1}{\sum_{i\in S(\xi,r)} n_i}\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right) g(\boldsymbol{w}(\xi K\tau)) + o(\tau^2)
\end{aligned}$$

where $D_3^{(\xi)}$ is defined as:

$$D_3^{(\xi)} = \sum_{r=1}^{K} D_2^{(\xi,r)} + \frac{1}{2}\sum_{\beta>\gamma}\left[\frac{1}{\sum_{i\in S(\xi,r_\beta)} n_i}\sum_{i\in S(\xi,r_\beta)}\Theta_i(\bar{t_i}), \frac{1}{\sum_{i\in S(\xi,r_\gamma)} n_i}\sum_{i\in S(\xi,r_\gamma)}\Theta_i(\bar{t_i})\right]$$

We have shown that $\|D_2\|$ is upper bounded, also by lemma 9.1 and second equation in (18), $\left\|D_3^{(\xi)}\right\|_{op}$ is upper bounded by constant $\kappa_3^{(\xi)}$: $\left\|D_3^{(\xi)}\right\|_{op} \leq \kappa_3^{(\xi)}$.

$\kappa_3 = \max_\xi \kappa_3^{(\xi)}$ is a $O(\tau^2)$ constant defined as:

$$\begin{aligned}
\kappa_3 &= \eta_0^2\tau^2\left(2C_2^4 + 8\frac{R_0}{\sqrt{l}}C_1C_2^2\right) + \eta_0^2\tau^2\frac{N(N-1)}{2}C_2^4 \\
&= O(\tau^2\eta_0^2)
\end{aligned}$$

By assumption $\sum_{i\in S(\xi,r)} n_i = \bar{n}$ is a constant, thus we can rewrite $g(\boldsymbol{w}((\xi+1)K\tau))$ as:

$$\begin{aligned}
&g(\boldsymbol{w}((\xi+1)K\tau)) \\
&= \left(1+\eta_0^2\tau^2 D_3^{(\xi)}\right)\exp\left(-\frac{\eta_0\tau}{\bar{n}}\sum_r\sum_{i\in S(\xi,r)}\Theta_i(\bar{t_i})\right) g(\boldsymbol{w}(\xi K\tau)) + o(\tau^2) \\
&= \left(1+\eta_0^2\tau^2 D_3^{(\xi)}\right)\exp\left(-\frac{\eta_0\tau}{\bar{n}}\sum_{i=1}^{N}\Theta_i(\bar{t_i})\right) g(\boldsymbol{w}(\xi K\tau)) + o(\tau^2)
\end{aligned} \tag{27}$$

We will now prove that, with high probability,

$$\lambda_{min}\left(\sum_{i=1}^{N}\Theta_i(\bar{t_i})\right) \geq \frac{\lambda_m}{2}$$

The smallest eigenvalue of summation of kernel is:

$$\begin{aligned}
&\lambda_{min}\left(\sum_{i=1}^{N}\Theta_i(\bar{t_i})\right) \\
&= \lambda_{min}\left(\Theta + \frac{1}{l}J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T - \Theta + \sum_{i=1}^{N}\Theta_i(\bar{t_i}) - \frac{1}{l}J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T\right) \\
&\geq \lambda_{min}(\Theta) - \left\|\frac{1}{l}J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T - \Theta\right\|_{op} - \left\|\sum_{i=1}^{N}\Theta_i(\bar{t_i}) - \frac{1}{l}J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T\right\|_{op}
\end{aligned}$$

The first term is just λ_m, the second term $\left\|J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T - \Theta\right\|_{op}$ can be smaller than $\frac{\lambda_m}{3}$ by [16] when the width l is large enough with probability at least $1 - \frac{\delta_0}{5}$. Now we will bound the third term:

$$
\begin{aligned}
&\left\|\sum_{i=1}^{N} \Theta_i(\bar{t_i}) - \frac{1}{l} J(\boldsymbol{w}(0))J(\boldsymbol{w}(0))^T\right\|_{op} \\
&= \frac{1}{l}\left\|\sum_{i=1}^{N} J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(\bar{\bar{t_i}}))^T\mathcal{P}_i - J(\boldsymbol{w}(0))\sum_{i=1}^{N} J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right\|_{op} \\
&= \frac{1}{l}\left\|\sum_{i=1}^{N} \left(J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(\bar{\bar{t_i}}))^T\mathcal{P}_i - J(\boldsymbol{w}(0))J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right)\right\|_{op} \\
&\leq \frac{1}{l}\sum_{i=1}^{N} \left\|J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(\bar{\bar{t_i}}))^T\mathcal{P}_i - J(\boldsymbol{w}(0))J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right\|_{op} \\
&\leq \frac{1}{l}\sum_{i=1}^{N} \left\|J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(\bar{\bar{t_i}}))^T\mathcal{P}_i - J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right\|_{op} \\
&+ \left\|J(\boldsymbol{w}(\bar{\bar{t_i}}))J_i(\boldsymbol{w}(0))^T\mathcal{P}_i - J(\boldsymbol{w}(0))J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right\|_{op} \\
&\leq \frac{1}{l}\sum_{i=1}^{N} \left\|J(\boldsymbol{w}(\bar{\bar{t_i}}))\right\|_F C_i^{(l)} \left\|\boldsymbol{w}(\bar{\bar{t_i}}) - \boldsymbol{w}_0\right\|_2 + \left\|J_i(\boldsymbol{w}(0))^T\mathcal{P}_i\right\|_F C^{(l)} \left\|\boldsymbol{w}(\bar{\bar{t_i}}) - \boldsymbol{w}_0\right\|_2 \\
&\leq \frac{\lambda_m}{6}
\end{aligned}
$$

when the width l is large enough. We use $\bar{\bar{t_i}}$ to denote $\bar{t_i} + \xi K\tau + r\tau$

Thus we have:

$$\|g(\boldsymbol{w}((n+1)K\tau))\|_2 \leq (1+\kappa_3)\exp\left(-\frac{\eta_0\tau}{\bar{n}}\frac{\lambda_m}{2}\right)\|g(\boldsymbol{w}(\xi K\tau))\|_2$$

with probability at least $1 - \frac{\delta_0}{2}$. Since κ_3 is $O(\eta^2\tau^2)$, when $\eta_0\tau$ is small enough we have:

$$\|g(\boldsymbol{w}((n+1)K\tau))\|_2 \leq \exp\left(-\frac{\eta_0\tau}{\bar{n}}\frac{\lambda_m}{3}\right)\|g(\boldsymbol{w}(\xi K\tau))\|_2$$

By union bound of probabilities, we thus complete the proof of the first equation in (18).

The upper bound of change of neural tangent kernel during training follows from directly Lemma 9.1 and second equation of (18).

□

9.4 Difference of neural network training and linear model training

In this section we will bound the difference between the training of neural network and that of linear model, and show that the bound goes to zero in the limit of infinitely wide network. Then combining with derivations in Sec. 9.2, we can prove theorem 4.2.

We use g^{lin} to denote the error vector when training linear model with limiting kernel Θ.

We start from the dynamics on individual client. We denote Θ_{i0} as $\Theta_{i0} = \frac{1}{l}J(\boldsymbol{w}(0))^T J_i(\boldsymbol{w}(0))\mathcal{P}_i$. Similar to [22], on client i, we have:

$$
\begin{aligned}
&\frac{d}{dt}\left(\exp\left(\eta_0\frac{\Theta_{i0}}{n_i}t\right)\left(g^{lin}(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t)) - g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t))\right)\right) \\
&= \eta_0\exp\left(\eta_0\frac{\Theta_{i0}}{n_i}t\right)\left(\Theta_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + t)) - \Theta_{i0})g(t)\right)
\end{aligned}
$$

Integrating both sides from 0 to τ, we have:

$$\begin{aligned}
&g^{lin}(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)) - g(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)) \\
&= \exp\left(-\eta_0 \frac{\Theta_{i0}}{n_i}\tau\right)\left(g^{lin}(\boldsymbol{w}(\xi K\tau + r\tau)) - g(\boldsymbol{w}(\xi K\tau + r\tau))\right) \\
&+ \exp\left(-\eta_0 \frac{\Theta_{i0}}{n_i}\tau\right)\eta_0 \int_0^\tau \exp\left(\eta_0 \frac{\Theta_{i0}}{n_i}s\right)\left(\Theta_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + s)) - \Theta_{i0})g(s)\right) ds
\end{aligned}$$

Then we take weighted average on both sides. Since

$$\boldsymbol{w}(\xi K\tau + (r+1)\tau) = \frac{1}{\sum_{i\in S(\xi,r)} n_i} \sum_{i\in S(\xi,r)} n_i \boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)$$

We know that for linear model g^{lin}:

$$g^{lin}\left(\boldsymbol{w}(\xi K\tau + (r+1)\tau)\right) = \frac{1}{\sum_{i\in S(\xi,r)} n_i} \sum_{i\in S(\xi,r)} n_i g^{lin}\left(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)\right)$$

However there will be a second order residual for a general nonlinear model:

$$\begin{aligned}
&g\left(\boldsymbol{w}(\xi K\tau + (r+1)\tau)\right) \\
&= \frac{1}{\sum_{i\in S(\xi,r)} n_i} \sum_{i\in S(\xi,r)} n_i g\left(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + \tau)\right) - \eta_0^2\tau^2 v_1^{(\xi,\tau)} + o\left(\tau^2\right)
\end{aligned}$$

The residual term $v_1^{(\xi,\tau)}$ is defined as:

$$v_1^{(\xi,\tau)} = \frac{g(\boldsymbol{w}(\xi K\tau + r\tau))^T}{\bar{n}l} \sum_{i\in\mathcal{S}(\xi,r)} n_i \left(D_1^{(\xi,r,i)}\right)^T \nabla^2 g D_1^{(\xi,r,i)} g(\boldsymbol{w}(\xi K\tau + r\tau))$$

It's norm is bounded by

$$\left\|v_1^{(\xi,\tau)}\right\| \leq 8R_0^2 \frac{C_1}{\sqrt{l}} C_2^2$$

On the right hand side, we can use lemma 9.3 to derive:

$$\begin{aligned}
&\frac{1}{\sum_{i\in S(\xi,r)} n_i} \sum_i n_i \exp\left(-\eta_0 \frac{\Theta_{i0}}{n_i}t\right)\left(g^{lin}(\boldsymbol{w}(\xi K\tau + r\tau)) - g(\boldsymbol{w}(\xi K\tau + r\tau))\right) \\
&= \left(1 + \eta_0^2\tau^2 \mathrm{Var}\left(\Theta_{i0}\right)\right) \exp\left(-\eta_0\tau \frac{1}{\bar{n}} \sum_{i\in S(\xi,r)} \Theta_{i0}\right)\left(g^{lin}(\boldsymbol{w}(\xi K\tau + r\tau)) - g(\boldsymbol{w}(\xi K\tau + r\tau))\right) \\
&+ o\left(\eta_0^2\tau^2\right)
\end{aligned}$$

As a result, the update can be written as:

$$\begin{aligned}
&g^{lin}(\boldsymbol{w}(\xi K\tau + r\tau + \tau)) - g(\boldsymbol{w}(\xi K\tau + r\tau + \tau)) \\
&= \left(1 + \eta_0^2\tau^2 \mathrm{Var}\left(\Theta_{i0}\right)\right) \exp\left(-\eta_0\tau \frac{1}{\bar{n}} \sum_{i\in S(\xi,r)} \Theta_{i0}\right)\left(g^{lin}(\boldsymbol{w}(\xi K\tau + r\tau)) - g(\boldsymbol{w}(\xi K\tau + r\tau))\right) \\
&+ \eta^2\tau^2 v_1^{(\xi,r)} \\
&+ \frac{1}{\bar{n}} \sum_{i\in S(\xi,r)} n_i \exp\left(-\eta_0 \frac{\Theta_{i0}}{n_i}t\right)\eta_0 \int_0^\tau \exp\left(\eta_0 \frac{\Theta_{i0}}{n_i}s\right)\left(\Theta_i(\boldsymbol{w}^{(i)}(\xi K\tau + r\tau + s)) - \Theta_{i0}\right) g(s) ds \\
&+ o(\tau^2)
\end{aligned}$$

When we apply this relation iteratively for r, and still retain only leading order terms, we have:

$$
\begin{aligned}
&g^{lin}(\boldsymbol{w}_{\xi K\tau+K\tau+\tau}) - g(\boldsymbol{w}_{\xi K\tau+K\tau+\tau}) \\
&= \left(1+\eta_0^2\tau^2 D_4^{(\xi,r)}\right)\exp\left(-\eta_0\tau\frac{1}{\bar{n}}\Theta_0\right)\left(g^{lin}(\boldsymbol{w}(\xi K\tau)) - g(\boldsymbol{w}(\xi K\tau))\right) \\
&+ \sum_{i=1}^{K}\left(\prod_{j=K}^{i+1}\exp\left(-\eta_0\tau\frac{\sum_{k\in S(\xi,j)}\Theta_{0j}}{\bar{n}}\right)\right)\left(\eta_0^2\tau^2 v_1^{(\xi,i)}\right. \\
&\left.+ \frac{1}{\bar{n}}\sum_{k\in\mathcal{S}(\xi,i)}\exp\left(-\eta_0\frac{\Theta_{k0}}{n_k}\tau\right)\eta_0\int_0^{\tau}\exp\left(\eta_0\frac{\Theta_{k0}}{n_k}s\right)\left(\Theta_k(\boldsymbol{w}^{(k)}(\xi K\tau+r\tau+s)) - \Theta_{k0})g(s)\right)ds\right) \\
&+ o(\tau^2)
\end{aligned}
$$

And we can take the norm on both sides:

$$
\begin{aligned}
&\left\|g^{lin}(\boldsymbol{w}(\xi K\tau+r\tau+\tau)) - g(\boldsymbol{w}(\xi K\tau+r\tau+\tau))\right\| \\
&\leq \left(1+\eta_0^2\tau^2\left\|D_4^{(\xi,r)}\right\|\right)\exp\left(-\eta_0\tau\frac{1}{\bar{n}}\lambda_{min}\left(\Theta_0\right)\right)\left\|g^{lin}(\boldsymbol{w}(\xi K\tau+r\tau)) - g(\boldsymbol{w}(\xi K\tau+r\tau))\right\| \\
&+ \kappa_4 + o(\tau^2)
\end{aligned}
$$

where κ_4 is defined as:

$$
\kappa_4 = \exp\left(\frac{\eta_0\tau C_2^2}{\bar{n}}\right)\left[\eta_0^2\tau^2 8R_0^2C_1 l^{-0.5}C_2^2 + \eta_0\tau\exp\left(\frac{\eta_0\tau}{\bar{n}C_2^2}\right)2R_0C_2^2 l^{-\frac{1}{2}}\right]
$$

From the expression above, we can see that $\kappa_4 \to 0$ when $l \to \infty$. When $\eta_0\tau$ is small enough suth that $\left(1+\eta_0^2\tau^2\left\|D_4^{(\xi,r)}\right\|\right)\exp\left(-\eta_0\tau\frac{1}{\bar{n}}\lambda_{min}\left(\Theta_0\right)\right) < 1$, we have:

$$
\begin{aligned}
&\left\|g^{lin}(\boldsymbol{w}(\xi K\tau+r\tau+\tau)) - g(\boldsymbol{w}(\xi K\tau+r\tau+\tau))\right\| \\
&\leq \left(1-\left(1+\eta_0^2\tau^2\left\|D_4^{(\xi,r)}\right\|\right)\exp\left(-\eta_0\tau\frac{1}{\bar{n}}\lambda_{min}\left(\Theta_0\right)\right)\right)^{-1}\kappa_4
\end{aligned}
$$

Thus the difference of error vector approaches zero in the limit $l \to \infty$, which indicates that g^{lin} can approximate g well.