Fed-ensemble: Ensemble Models in Federated Learning for Improved Generalization and Uncertainty Quantification

Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury, *Member, IEEE*

***Abstract*— The increase in the computational power of edge devices has opened up the possibility of processing some of the data at the edge and distributing model learning. This paradigm is often called federated learning (FL), where edge devices exploit their local computational resources to train models collaboratively. Though FL has seen recent success, it is unclear how to characterize uncertainties in FL predictions. In this paper, we propose *Fed-ensemble*: a simple approach that brings model ensembling to 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 compared with single-model methods. 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. Also, by carefully choosing client-dependent weights in the inference stage, *Fed-ensemble* becomes personalized and yields even better performance. 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 and justifies the uncertainty quantification capability. We also illustrate that *Fed-ensemble* has an elegant Bayesian interpretation.**

***Note to Practitioners*—*Fed-ensemble* provides an algorithm that extracts a set of K solutions without imposing any additional communication overhead in FL. Given multiple solutions, *Fed-ensemble* can be exploited to personalize inference as well as quantify uncertainty. Such capabilities may be beneficial within multiple practical systems that require uncertainty-aware decision-making. Further, *Fed-ensemble* may be useful for model validation and hypothesis testing.**

***Index Terms*— Federated learning, ensemble learning, kernel method, uncertainty quantification.**

Manuscript received 4 January 2023; revised 28 March 2023; accepted 8 April 2023. This article was recommended for publication by Associate Editor C. Yang and Editor L. Moench upon evaluation of the reviewers' comments. The work of Naichen Shi was supported by NSF CAREER Award under Grant 2144147. *(Corresponding author: Raed Al Kontar.)*

Naichen Shi and Raed Al Kontar are with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: alkontar@umich.edu).

Fan Lai and Mosharaf Chowdhury are with the Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48105 USA.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2023.3269639.

Digital Object Identifier 10.1109/TASE.2023.3269639

I. Introduction

SMART and connected Internet of Things (IoT) systems are rapidly infiltrating the industry. The conventional way of analyzing the data collected by IoT devices is to upload them to a cloud/data center, where complex models are learned on the aggregated data. As the size of datasets scales, such an approach leads to high communication and storage costs. Sharing data with others also raises concerns about data privacy.

Rapid computational power growth 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 models with the orchestration of a central server while keeping their personal data stored locally. Only model parameters are shared. This paradigm shift reduces privacy concerns and brings many intrinsic advantages, including cost efficiency, diversity, and reduced communication, amongst many others [1], [2].

The earliest and perhaps most popular FL algorithm is FederatedAveraging (*Fedavg*) [3]. 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. The server then takes a weighted average of the resulting local models to update the global model. This process iterates over multiple training rounds to maximize the performance of all devices. *Fedavg* has seen prevailing empirical successes in many real-world applications [4], [5]. 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 affects their participation in the training process [6], [7]. 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 [6], [7], [8], [9], [10].

Another critical challenge in FL is uncertainty quantification. Due to the randomness in the model initialization and training process, it is essential to characterize the variance in the model prediction. However, very little literature exists on uncertainty quantification in FL.

In this paper, we adopt the idea of ensemble training to FL and propose *Fed-ensemble*, which iteratively updates an ensemble of models. Using an ensemble in FL brings several benefits. Firstly, *Fed-ensemble* can characterize uncertainty in predictions. With an ensemble of K models, we can naturally use variance in the prediction as a measure of knowledge uncertainty. Secondly, *Fed-ensemble* improves generalization. We show, both theoretically and empirically, that ensembling is efficient in reducing variance and achieving better generalization performance in FL. Thirdly, *Fed-ensemble* provides a means for personalization to different clients as highlighted in Sec. V-D.

It is worth noting that our approach does not increase any additional communication cost though we train an ensemble of K models. What's more, *Fed-ensemble* is orthogonal to current efforts in FL aimed at reducing communication cost, alleviating heterogeneity, or finding better fixed points, as such approaches can be directly integrated into our ensembling framework. Our contributions are summarized below:

- **Model:** We propose an ensemble treatment to FL that updates K models over local datasets. Point predictions are obtained by model averaging. We also show that *Fed-ensemble* excels at uncertainty quantification (UQ), which is still an under-investigated topic in FL. Our approach does not impose an 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 the bias-variance decomposition. Using neural tangent kernels (NTK), we 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 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 a general multilayer neural network in FL in the kernel regime and the first justification for using model ensembling in FL. Our proof also offers standalone value as it extends NTK results to FL settings.
- **Personalization:** We propose an approach through which clients can personalize their *Fed-ensemble* predictions by taking a locally optimized weighted average over the K models. Compared to current literature, this approach provides a new viewpoint on how personalization can be achieved in FL.
- **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 [11]. Our results highlight the unique advantages of ensembling in FL in terms of generalization, uncertainty quantification, and personalization.

II. Related Work

A. Single-Model FL

Many approaches have been proposed to tackle the aforementioned FL challenges. Below we list a few, yet this is by no means an extensive list. *Fedavg* [3] allows multiple steps of local updating to balance communication vs. computation on resource-limited edge devices. *Fedavg* reports decent performance on mitigating performance bias on skewed client data distributions. *Fedprox* [12] attempts to solve the heterogeneity issue by adding a proxy term to control the shift of model weights between the global and local client updates. This proxy term can be viewed as a Gaussian prior on model weights. Several influential works aim at expediting convergence, like *FedAcc* [13], *FedAdam* and *FedYogi* [14], reducing communication cost, like *DIANA* [15], *DORE* [16], or finding exact optimal solutions, like *FedSplit* [17], *FedPD* [18], *FedDyn* [19]. As aforementioned, although single model approaches are useful in some applications, they are not capable of uncertainty quantification. Also, several techniques introduced in these efforts can be integrated into our ensembling framework. An example of marrying *Fed-ensemble* with *FedDyn* is shown in Sec. V-D.

B. Personalized FL

To tackle excessive heterogeneity, personalized FL allows each client to retain their own model while borrowing strength from each other. Current literature handles personalization from two perspectives. The first is specific to deep neural networks, where layers are split into common and personalized layers (often the last few layers), then all clients exploit the global modeling approaches discussed above to learn the common layer parameters [20], [21]. In contrast, the other perspective follows train-then-personalize thinking (often iteratively), where a global model is first learned, and then each client fine-tunes this model using their own data. Popular approaches encourage the weights of personalized models to stay in a small region in the parameter space of the global model to balance each client's shared knowledge and unique characteristics [22], [23], [24].

C. Ensemble of Deep Neural Nets

Recently, ensembling methods in conventional, non-federated deep learning have seen great success. Amongst them, [25] analyzes the loss surface of deep neural networks and uses cyclic learning rate to learn multiple models and form ensembles. [26] visually demonstrates the diversity of randomly initialized neural networks and empirically shows the stability of ensembled solutions. Also, [27] connects ensembling to Bayesian deep neural networks and highlights the benefits of ensembling. These works mainly show the benefits of ensembles in the centralized regime. It is not clear how to extend these results to the federated settings, where communication cost is a key bottleneck.

D. Bayesian Methods in FL and UQ

There are also some recent attempts to exploit Bayesian philosophies in FL. Very recently, *Fedbe* was proposed [28] to

couple Bayesian model averaging with knowledge distillation on the server. *Fedbe* fits a Gaussian or Dirichlet distribution for 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, Bayesian non-parametrics have been investigated for advanced model aggregation through matching and re-permuting neurons using neuron matching algorithms [29], [30]. Despite the adopted Bayesian philosophy above, these methods above do not provide predictive UQ. *Indeed, UQ in FL is yet to be fully investigated*.

E. Industrial Applications of FL

Recently, FL has been applied to several industrial applications [31], [32], [33], [34], [35]. Among them, [31] and [32] use FL to improve the predictive performance on several prognosis tasks, [35] applies FL to distributed manufacturing, and [36] discusses the applications of FL in healthcare. We also use *Fed-ensemble* on distributed 3D printers for improved vibration predictive performance.

For a comprehensive review of recent FL literature, please refer to Kontar et al. 2021 [34].

III. Fed-Ensemble

In this section, we elaborate on the details of the training and inference procedures of *Fed-ensemble*. We carefully design our training algorithm so that an ensemble of K models is trained efficiently without additional communication costs.

A. Parameter Updates Through Random Permutations

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 parametrized by weight vector $\boldsymbol{w}$.

1) Design Principle: To train the ensemble efficiently, our goal is to have 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 of neural networks, which is usually Gaussian with zero mean and width-dependent variance, [37], [38]. In each training round, the server sends one of the K models to every client to train on the local data. The server then aggregates the updated models received from the clients. All K models eventually learn from the entire dataset. Hereon, we use model or weight to refer to $\boldsymbol{w}_k$ and model to refer to the corresponding $f_{\boldsymbol{w}_k}$.

2) Objective of Ensemble Training: Since we aim to learn K models, 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 $\mathcal{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K\}$, 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 criterion such as cross entropy.

Algorithm 1 *Fed-ensemble* training

Input: Client datasets $\{\boldsymbol{D}_i\}_{i=1}^{N}$, T, K, N, initialization for $\{\boldsymbol{w}_k\}_{k=1}^{K}$
for $t = 0, 2, \cdots T$ **do**
  for $i = 1, 2, \cdots N$ **do**
    if $t = 0(\text{mod } K)$ **then**
      Index Permutation: $\boldsymbol{P}_{i,\cdot} \leftarrow$ *shuffle_list*$[1, 2, \cdots K]$
    end if
    Server sends model $\boldsymbol{w}_{\boldsymbol{P}_{i,t \bmod K}}$ to client i
    $\boldsymbol{u}^{(i)} \leftarrow$ *local_training* $[\boldsymbol{w}_{\boldsymbol{P}_{i,t \bmod K}}, \boldsymbol{D}_i]$
    Client i sends $\boldsymbol{u}^{(i)}$ to server
  end for
  $\{\boldsymbol{w}_k\}_{k=1}^{K} \leftarrow$*server_update* $[\boldsymbol{u}^{(1)}, .., \boldsymbol{u}^{(N)}]$.
end for

3) 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). Let $t \in [1, .., T]$ denote the communication rounds. Every K communication rounds, at the beginning of the t-th round where $t = 0(\text{mod } K)$, each client decides how K models will be trained in the following K communication rounds. To do so, we can define a permutation matrix $\boldsymbol{P}$ of size $N \times K$ such that at each round t, each row of $\boldsymbol{P}$ is an independent random shuffle of $\{1, 2, \cdots, K\}$. Then in the t-th communication round, client i gets assigned model $\boldsymbol{w}_{\boldsymbol{P}_{i,t \bmod K}}$ as its initialization for $\boldsymbol{w}$ and then performs a training procedure on the local data, denoted as $\boldsymbol{u}^{(i)} = local\_training[\boldsymbol{w} = \boldsymbol{w}_{\boldsymbol{P}_{i,t \bmod K}}, \boldsymbol{D}_i]$, where $\boldsymbol{u}^{(i)}$ is the updated model weight. The $local\_training$ can be any local optimization method like SGD or Adam. Note that the use of the random permutation matrix $\boldsymbol{P}$ ensures that every model is downloaded and trained on all clients in K consecutive communication rounds. Thus models are trained on diverse clients. Upon receiving updated models from all N clients, the server activates *server_update* to calculate the new set of weights $\{\boldsymbol{w}_1, .., \boldsymbol{w}_K\}$.

Remark 1: The simplest form of the server_update function is to average models from the same initialization: $\boldsymbol{w}_k \leftarrow \frac{\sum_{i=1}^{N} \delta_{ik} \boldsymbol{u}^{(i)}}{\sum_{i=1}^{N} \delta_{ik}}$ $\forall k$, *where* $\delta_{ik} = 1$ *if client* i *downloads model* k *at communication round* t*, 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 models from the same initialization. Similarly, different local training schemes can be used within local_training.*

Remark 2: The use of multiple models does not increase computation or communication overhead on clients compared with single-model FL algorithms, as for every round, only one model is sent to and trained on each client. However, after carefully selecting models by random shuffling, models trained by Fed-ensemble can efficiently learn statistical patterns on all clients as proven in the convergence and generalization results in Sec. IV.

B. Model Prediction With Fed-ensemble

We differentiate the model inference procedures for existing and new clients. After the training process is complete, all K

models are sent to clients. For a new client, we do not have any additional information on their data. Thus a reasonable prediction is simply achieved via a uniform averaging of predictions at a new input point $x^\star$.

$$f_{\mathcal{W}}(x^\star) = \frac{1}{K}\sum_{k=1}^{K} f_{\boldsymbol{w}_k}(x^\star) \tag{2}$$

While for clients that participated in model training, one may take an additional step that leverages the information on the local training dataset. Specifically, client i uses a weighted average of predictions:

$$f_i(x^\star) = \sum_{k=1}^{K} \hat{\boldsymbol{\alpha}}_{ik} f_{\boldsymbol{w}_k}(x^\star) \tag{3}$$

where $\hat{\boldsymbol{\alpha}}_i$ is the model weighting coefficients in $\mathbb{R}^K$ determined by solving the following problem on client i:

$$\begin{aligned} \hat{\boldsymbol{\alpha}}_i = \arg\min_{\boldsymbol{\alpha}_i} & \sum_{k=1}^{K} \boldsymbol{\alpha}_{ik}\ell_i(\boldsymbol{w}_k) + \gamma\mathcal{H}(\boldsymbol{\alpha}_i) \\ \text{subject to } & \sum_{k=1}^{K} \boldsymbol{\alpha}_{ik} = 1\,, \end{aligned} \tag{4}$$

where $\mathcal{H}(\boldsymbol{\alpha}_i)$ is the entropy of $\boldsymbol{\alpha}_i$ defined as $\mathcal{H}(\boldsymbol{\alpha}_i) = -\sum_{k=1}^{K} \boldsymbol{\alpha}_{ik}\log\boldsymbol{\alpha}_{ik}$. The rationale of objective (4) is simple; we want to choose $\boldsymbol{\alpha}_i$ to minimize the weighted empirical loss while penalizing a radical $\boldsymbol{\alpha}_i$ by adding an entropy regularization term. The closed-form solution to problem (4) is $\hat{\boldsymbol{\alpha}}_{ik} = \exp(-\gamma\ell_i(\boldsymbol{w}_k))/\sum_{k=1}^{K}\exp(-\gamma\ell_i(\boldsymbol{w}_k))$. The hyperparameter γ thus controls the variation of entries in $\hat{\boldsymbol{\alpha}}$. Small γ encourages $\hat{\boldsymbol{\alpha}}$ to put more weights on low loss models, while large γ drives $\hat{\boldsymbol{\alpha}}$ close to uniform weighting.

In the following, we simply call (2) uniform weighting and (3) personalized weighting. Here we should highlight that personalized weighting offers a new perspective on current personalization techniques. Instead of fine-tuning weights or splitting a network into shared and individual weights, *Fed-ensemble* can naturally personalize by simply taking a personalized weighting scheme of the ensemble solutions. In the numerical sections, we will show that besides uncertainty quantification, this approach allows improved personalization compared to state-of-the-art techniques like *Ditto* [22].

C. 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:

$$\begin{aligned} &\min_{q} D_{KL}\big[q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\big] \\ &\qquad = \mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}\big[ln\, q(\boldsymbol{w}) - ln\, p(\boldsymbol{w}|\boldsymbol{D})\big], \end{aligned} \tag{5}$$

Now, if we take $q(\boldsymbol{w})$ as a mixture of isotropic Gaussians, $q(\boldsymbol{w}) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}\big(\boldsymbol{w}_k, \sigma^2 I\big)$, where $\boldsymbol{w}_k$'s are the variational parameters to be optimized.

Given (5), $D_{KL}\big[q(\boldsymbol{w}) \,||\, p(\boldsymbol{w}|\boldsymbol{D})\big]$ can be decomposed by conditional expectation law:

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

where *Uniform(K)* denotes a uniform distribution on $\{1, \cdots, K\}$. To estimate the expectation, we take the assumption that the variance parameter σ is very small, then different Gaussian centers $(\boldsymbol{w}_k)$ are well-separated such that $q(\boldsymbol{w})$ is close to zero outside small vicinity of $\boldsymbol{w}_k$'s. Thus,

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\big[\ln\, q(\boldsymbol{w})\big] \\ &\quad \approx \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{w}_k,\sigma^2\boldsymbol{I})}\left[-\frac{1}{2\sigma^2}\|\boldsymbol{w}-\boldsymbol{w}_k\|^2 - d\ln\sigma\right] + C \\ &\quad = -\frac{d}{2} - d\ln\ \sigma + C \end{aligned} \tag{6}$$

where C is a constant independent of k. As a result, the entire expectation (6) is independent of $\boldsymbol{w}_k$. Now taking a first-order Taylor expansion of $\ln\, p(\boldsymbol{w}|\boldsymbol{D})$ around $\boldsymbol{w}_k$, we have

$$\begin{aligned} \ln\, p(\boldsymbol{w}|\boldsymbol{D}) = \ln\, p(\boldsymbol{w}_k|\boldsymbol{D}) + (\boldsymbol{w}-\boldsymbol{w}_k)^T\nabla\ln\, p(\boldsymbol{w}_k|\boldsymbol{D}) \\ + h.o.t. \end{aligned} \tag{7}$$

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

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

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 $\boldsymbol{w} = \boldsymbol{w}_k$.

Notice that data on different clients are usually independent. Therefore the log-likelihood factorizes as $\ln\, p(\boldsymbol{D}|\boldsymbol{w}_k) = \sum_i \ln\, p(\boldsymbol{D}_i|\boldsymbol{w})$. 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) from (8). The $p(\boldsymbol{w}_k)$ term acts as a regularizer on negative log-likelihood loss $p(\boldsymbol{D}|\boldsymbol{w}_k)$. This Bayesian view highlights the ensemble diversity as the K models can be viewed as models of a mixture of Gaussian.

IV. Convergence and Limiting Behavior of Sufficiently Over Parameterized Neural Networks

In this section, we present theoretical results on the training and prediction of *Fed-ensemble*. We analyze sufficiently over-parametrized networks trained by *Fed-ensemble* and explain their advantageous generalization performance through the lens of bias-variance decomposition. **Technically, 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, and UQ can be established using *Fed-ensemble*.**

A. Convergence and Variance Reduction Using Neural Tangent Kernels

Inspired by recent work on neural tangent kernels [39], [40], we analyze *Fed-ensemble* in sufficiently overparametrized neural networks. We focus on regression tasks and define the local empirical loss in (1) as:

$$\ell_i(\boldsymbol{w}_k) = \frac{1}{2n_i} \sum_{(x_{ij}, y_{ij}) \in \boldsymbol{D}_i} \left(f_{\boldsymbol{w}_k}(x_{ij}) - y_{ij}\right)^2 \tag{9}$$

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 that $f_{\boldsymbol{w}_k}(x)$ 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 Appendix 1. 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. The local epoch of training t is defined as the product between the number of communication rounds and the number of local updates in each communication round. We also let p_{init} denote the initialization distribution for weights $\boldsymbol{w}$. Conditions for p_{init} are found in Appendix 1. 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, [39] 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 [39] and extend the analysis to FL settings.

Below we state an informal statement of our convergence result.

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

1) Θ *is full rank i.e.,* $\lambda_{min}(\Theta) > 0$
2) *The norm of every input* x *is smaller than 1:* $||x|| \leq 1$
3) *The stationary points of all local losses coincide:* $\nabla_{\boldsymbol{w}} \mathbb{E}_{(x,y)\sim \mathbf{D}} |f_{\boldsymbol{w}}(x) - y|^2 = 0$ *leads to* $\nabla_{\boldsymbol{w}} \mathbb{E}_{(x,y)\sim \mathbf{D}_i} |f_{\boldsymbol{w}}(x) - y|^2 = 0$ *for all clients*
4) *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 model decreases exponentially*

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

if the learning rate η *is smaller than a threshold.*

Remark 3: Assumptions (i) and (ii) are standard for theoretical development in NTK. Assumption (iii) can be derived directly from B-local dissimilarity condition in [12]. 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 [18], [19] has tried to propose 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 step size according to the number of data points in each round.

We would like to note that after writing this paper, we find that [41] shows the 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, their work is mainly concerned with the theoretical understanding of the optimization of FL under NTK regimes, while our overarching goal is to propose an algorithm aimed at ensembling, *Fed-ensemble*, and motivating 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 2: (Informal) If Algorithm 1 is used to train $\{\boldsymbol{w}_k\}_{k=1,...K}$ *for the regression task (1) and (9), then after sufficient communication rounds, functions* $f_{\boldsymbol{w}_k}(x)$ *can be regarded as independent samples from a* $\mathcal{GP}\big(m(x), k(x, x')\big) + o(\eta_0^2)$, *with mean and variance defined as* $m(x) = \Theta(x, X)\Theta^{-1}(X, X)Y$ *and* $k(x, x') = \mathcal{K}\big(x, x'\big) + \Theta(x, X)\Theta^{-1}\mathcal{K}\Theta^{-1}\Theta\big(X, x'\big) - \big(\Theta(x, X)\Theta^{-1}\mathcal{K}\big(X, x'\big) + \Theta\big(x', X\big)\Theta^{-1}\mathcal{K}(X, x)\big)$, *and* (X, Y) *represents the entire dataset* $\boldsymbol{D}$.

Remark 4: The result in Theorem 2 is illustrated in Fig. 1. The central result is that training K models 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 can obtain multiple samples from the posterior predictive distribution. As the variance in posterior Gaussian distribution naturally represents the knowledge uncertainty,* ***this evidences the ability of Fed-ensemble to quantify uncertainty****. Our result is similar to the simple sample-then-optimize method by [42], 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.*

To see the implications of theorem 2 in predictive performance, we introduce some notations in bias-variance decomposition to explain the variance reduction effect in ensemble learning. We use $\boldsymbol{\theta}$ to parametrize a general hypothesis class $h_{\boldsymbol{\theta}}$, and $\overline{h_{\boldsymbol{\theta}}}(x)$ to denote the average of $h_{\boldsymbol{\theta}}(x)$ under some distribution $q(\boldsymbol{\theta})$, i.e. $\overline{h_{\boldsymbol{\theta}}}(x) = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}[h_{\boldsymbol{\theta}}(x)]$. Similarly $\overline{y}(x)$

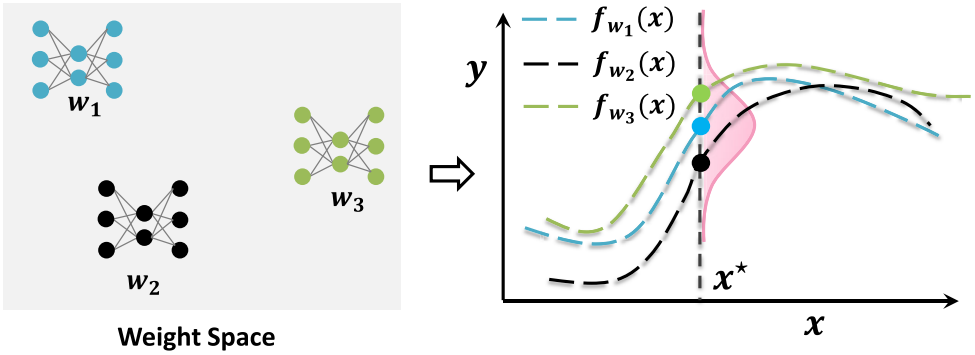


Fig. 1. An illustration of an ensemble of 3 samples from the posterior distribution.

is defined as the conditional expectation $\overline{y}(x) = \mathbb{E}_y[y|x]$, then:

$$\mathbb{E}_{y,x,\boldsymbol{\theta}}\left[(y - h_{\boldsymbol{\theta}}(x))^2\right] = \mathbb{E}_{y,x,\boldsymbol{\theta}}\left[(y - \overline{y}(x))^2\right] + \mathbb{E}_{x,\boldsymbol{\theta}}\left[\left(\overline{y}(x) - \overline{h_{\boldsymbol{\theta}}}(x)\right)^2\right] + \mathbb{E}_{x,\boldsymbol{\theta}}\left[\left(\overline{h_{\boldsymbol{\theta}}}(x) - h_{\boldsymbol{\theta}}(x)\right)^2\right] \quad (10)$$

where the expectation over x is taken under some input distribution $p(x)$, and that over $\boldsymbol{\theta}$ is taken under $q(\boldsymbol{\theta})$. On the right-hand side of (10), the first term represents the intrinsic noise in the data, or 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 [43]. In FL, this variance is often large due to data heterogeneity, partial participation, etc. However, we will show that a direct consequence of the theorem 2 indicates that knowledge uncertainty decreases through model ensembling. In *Fed-ensemble*, the prediction is given by $h_{\boldsymbol{\theta}}(x) = \frac{1}{K}\sum_{k=1}^{K} f_{\boldsymbol{w}_k}(x)$, and the expected prediction is simply $\overline{h_{\boldsymbol{\theta}}}(x) = m(x)$, where $m(x)$ is the *maximum a posteriori* solution of the limiting Gaussain Process obtained in Theorem 2. From theorem 2, $f_{\boldsymbol{w}_k}$ are K i.i.d. samples, therefore the variance shrinks as the K grows. More formally, we have the corollary 1:

Corollary 1: Let $\boldsymbol{\theta} = \{\boldsymbol{w}_k\}_{k=1,\cdots,K}$ be the parameters of all models in the ensemble. If the assumptions in theorem 1 are satisfied, when we train with Fed-ensemble with K models, after sufficient iterations, for a given test input $x^\star$, we have that:

$$\mathbb{E}_{\boldsymbol{\theta}}\left(\left|\frac{1}{K}\sum_{k=1}^{K} f_{\boldsymbol{w}_k}(x^\star) - m(x^\star)\right|^2\right) = \frac{k(x^\star, x^\star)}{K} + o(\eta_0^2) \quad (11)$$

where k is the posterior covariance function in theorem 2.

This corollary shows that the variance decreases at the rate of $\frac{1}{K}$. Therefore, averaging over multiple models is capable of getting closer to $m(x)$. If we take expectation over x^* on (11), it becomes clear that knowledge uncertainty also decreases at the rate of $\frac{1}{K}$.

V. 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. At the end of this section, we benchmark *Fed-ensemble* on a 3D printing dataset. We note that since ensembling is an approach yet to be fully investigated in FL, we dedicate many experiments to a proof of concept.

TABLE I
Bias-Variance Decomposition on the Toy Regression Model

	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

A. A Simple Toy Example With Kernels

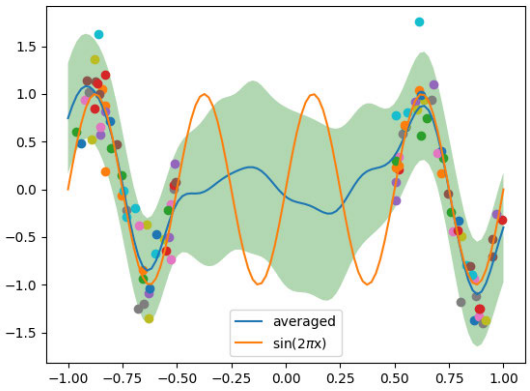
We start with a toy example of kernel methods that illustrate the benefits of using multiple models and reinforce the key conclusions from Theorem 2. We create 50 clients and generate the data of each client following a noisy sine function $y = a\sin(2\pi x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.2^2)$, 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 $f(x; \mathcal{W}) = \sum_{p=1}^{100} w_p \varphi_p(x)$, where $\varphi_p(x)$ is a radial basis kernel defined as: $\phi_p(x) = \exp\left(-\frac{(x-v_p)^2}{2b^2}\right)$. In this basis function, $b = 0.08$ and v_p's are 100 uniformly randomly sampled parameters from $[-1, 1]$ that remain constant during training. Note that the expectation of the generated function is $\mathbb{E}(y|x) = \sin(2\pi x)$.

We report our predictive results in Table I.

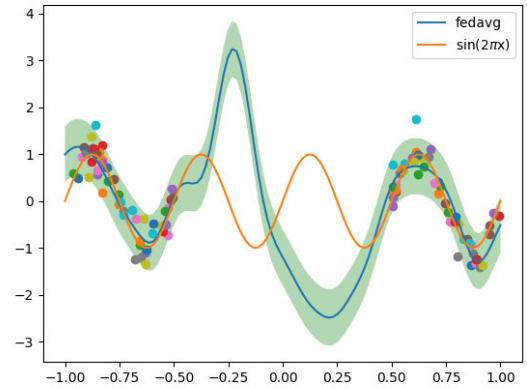
In Table I we vary K and calculate the bias-variance decomposition in (10) in each case. For each choice of K, we run 100 experiments from different random initializations and calculate the predictive variance for models trained from different initializations. As seen in the table, the variance of *Fed-ensemble* can be efficiently reduced compared with a single model approach such as *Fedavg*.

B. Uncertainty Quantification (UQ)

As discussed before, UQ is important in understanding the reliability of predictions given by FL models. An ideal model should acknowledge the level of confidence in its



(a) Prediction of *Fed-ensemble*.



(b) Prediction of *Fedavg*.

Fig. 2. Linear model on a toy dataset. Dots represent data points from a subset of clients. The green area denotes 95% predictive interval, $f_{\boldsymbol{w}_k}(x)$. The "averaged" reports the final prediction obtained after model averaging in *Fed-ensemble*.

prediction. From (10), the total average error is decomposed as the summation of data uncertainty, bias, and knowledge uncertainty. From a single model, one cannot estimate knowledge uncertainty. While for an ensemble of models, we can naturally estimate knowledge uncertainty by sample variance: $\frac{1}{K}\sum_{k=1}^{K}\left(f_{\boldsymbol{w}_k}(x)-\frac{1}{K}\sum_{k'=1}^{K}f_{\boldsymbol{w}_{k'}}(x)\right)^2$. To show the difference between the two models in uncertainty quantification, we examine their performance again on sine function regression. The training data are uniformly generated from the domain $[-1,-0.5]\bigcup[0.5,1]$. We train single and ensemble models, respectively, then calculate point predictions and uncertainty over the entire domain. Since single-model algorithms cannot output predictive variances, we instead use residual fitting error to estimate the variance and provide a confidence interval. Note that here the variance is constant over all input points.

Fig 2(a) and 2(b) show that *Fed-ensemble* can appropriately characterize knowledge uncertainty on regions without labeled data. In contrast, the *Fedavg*-trained model is overconfident about the prediction in such a region.

C. Experimental Setup

In our evaluation, we compare the performance of *Fed-ensemble* with several existing FL approaches. We experiment with six different datasets of varying sizes:

- *MNIST*: a popular image classification dataset with 55k training samples across 10 categories.
- *FEMNIST*: a federated dataset of handwritten digits and alphabets of 62 classes from over 3k writers [44].
- *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 the next work prediction.
- *OpenImage*: a real-world image dataset with 1.1M images of 600 classes from 13k image uploaders [45]. We use the realistic distribution of client data in FedScale benchmark [11].

In our experiments, we use $K=5$ models by default, except in the sensitivity analysis, where we vary K.

We compare *Fed-ensemble with uniform weighting* with *Fedavg, Fedprox, Fedbe* and *Fedbe* without knowledge distillation, where we use random sampling to replace knowledge distillation. Some entries in *Fedbe*/*Fedbe*-noKD columns are missing either because it's impractical (the dataset on the central server does not exist) or we cannot fine-tune the hyperparameters to achieve reasonable performance.

Also, we compare *Fed-ensemble with personalized weighting* with popular personalization methods, including *LG-fedavg* [20] and *Ditto* [22]. The testing results on MNIST, CIFAR10/100, Shakespeare, and OpenImage are evaluated on new clients where we do not have labeled data available, while results on FEMNIST are tested on existing clients, where we can fine-tune some parameters based on the local training set.

Note that we train a 2-layer CNN for MNIST, and ResNet18 [46] for CIFAR10, 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 first sort the images by label and then assign them to different clients to make sure that each client has images from exactly two labels. The dataset FEMNIST and OpenImage are naturally separated by contributors/users.

We now specify some hyperparameters in our experiments. For experiments on MNIST, we partition the data to 100 clients. For *local_training* in algorithm 1, we use 10 epochs of SGD with the batch size 16, learning rate 0.001, and weight decay 0.001. For experiments on CIFAR10, we also partition the data to 100 clients. We use Resnet18 to fit CIFAR10. For *local_training* in algorithm 1, we also use 10 epochs of SGD, with a learning rate 0.005, batch size 16. For CIFAR100, we use the i.i.d. partition scheme to partition the data to 10 clients. We train Resnet34 with batch size 16 and a learning rate 0.01. On Shakespeare, we partition the dataset to 10 clients by the i.i.d. partition scheme. For *local_training*, we use Adam [47] with batch size 128 and learning rate 0.001. On the OpenImage dataset, we simulate the real FL deployment in Google [48], 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. In all experiments, we decay learning rates by a constant of 0.99 after 10 communication rounds.

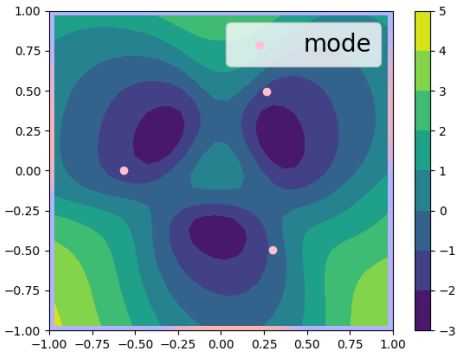


Fig. 3. Projection of the loss surface to a plane spanned by 3 models of the ensemble trained by *Fed-ensemble*. Color represents the logarithm of cross-entropy loss on the entire training set.

TABLE II
TESTING ACCURACY OF MODELS TRAINED BY DIFFERENT FL ALGORITHMS ON FIVE DATASETS

Testing acc (%)	*Fedavg*	*Fedprox*	*Fedbe*	*Fedbe*-nokd	*Fed-ensemble*
MNIST-i.i.d.	95.08 ± 0.3	95.76 ± 0.12	95.21 ± 0.11	-	**97.75**±0.04
MNIST-non-i.i.d.	90.17 ± 0.19	90.75 ± 0.34	-	-	**95.44**±0.14
CIFAR10-i.i.d.	86.43 ± 0.11	86.77 ± 0.06	83.13 ± 0.34	86.78 ± 0.4	**87.99**±0.09
CIFAR10-non-i.i.d.	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

Other algorithm-specific hyper-parameters are chosen based on the existing literature for each method.

We run *Fed-ensemble* and benchmark algorithms under the same communication cost budget, then calculate the mean and standard deviation of test accuracy for the last 3 communication rounds. Results are shown in Table II. From Table II, it is clear that *Fed-ensemble* outperforms all benchmarked single-model FL algorithms. This confirms the effectiveness of ensembling over single-model methods in improving generalization. *Fedbe* turns out to perform closely to single model algorithms eventually. We conjecture that this happens as the diversity of local models is lost in the knowledge distillation step. To show the diversity of models in *Fed-ensemble*, we plot the projection of the loss surface on a plane spanned by 3 models from the ensemble at the end of MNIST training in Fig 3. Fig 3 shows that models $\{\boldsymbol{w}_k\}$ have rich diversity: different models correspond to different local minima with a high loss barrier on the segment connecting them. In the non-i.i.d. setting, the performance gap between *Fed-ensemble* and single model 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** [27].

We also present the round-to-accuracy figure on the Shakespeare dataset in Fig.4 to better visualize the training process. The next-word-prediction model we use has 2 LSTM layers, each with 256 hidden states. Due to the page limit, we plot round-to-accuracy curves on other datasets in Appendix 2.

In Fig 4, the comparison between Fed-ensemble and single-model algorithms is conspicuous. One can see 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.

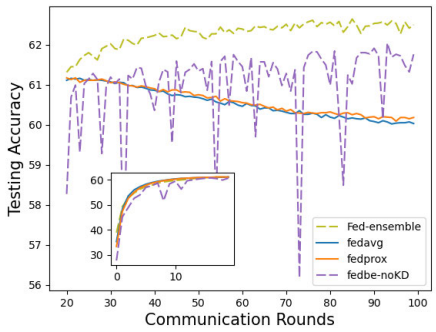


Fig. 4. Round-to-accuracy curves of shakespeare.

TABLE III
AVERAGE TESTING ACCURACY ON FEMNIST

Fedavg	*Ditto*	*LG-fedavg*	*Fed-ensemble*	*FedDyn*	*FedDyn*-ensemble
77.9±0.3	80.2±0.3	46.0±3	80.5±0.2	82.0 ±1.0	83.4±0.9

D. Fed-ensemble With Personalized Weighting

We test *Fed-ensemble* with personalized weighting on FEMNIST. As writers have different writing styles, FEMNIST is naturally a heterogeneous dataset. On FEMNIST, we choose *Fedavg*, *Ditto*, *LG-fedavg*, *FedDyn* as our baseline methods. Among them, *Ditto* and *LG-fedavg* are state-of-the-art personalized FL algorithms, and *FedDyn* [19] is a communication-efficient FL algorithm for training a global model. Intuitively, *FedDyn* adopts constrained optimization algorithms and uses dynamic regularization terms to accelerate training. To show that the proposed *Fed-ensemble* can work well with other methods, we combine *Fed-ensemble* with *FedDyn* to create a new algorithm called *FedDyn-ensemble*. In *FedDyn-ensemble*, we simply use the *FedDyn* objective as our local training objective for each model and use the corresponding server average step also for each mode. We run each experiment three times and calculate the mean and variance of the testing accuracy averaged on all clients. Results are shown in Table III. The comparison between *Fed-ensemble* and state-of-the-art personalized models confirms *Fed-ensemble*'s flexibility to fit the heterogeneous dataset. Moreover, as *FedDyn-ensemble* outperforms the vanilla *FedDyn*, our model demonstrates its ability to collaborate with other FL algorithms to produce even stronger results.

E. 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 data from fewer categories, the data distribution is more variant. The results are shown in Table IV. As expected, the performance of all algorithms degrades with such heterogeneity. Furthermore, *Fed-ensemble* outperforms its counterparts, 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.

F. Effect of Number of Models K

Since the number of models, K, in the ensemble is an important hyperparameter, we choose different values of K

TABLE IV
SENSITIVITY ANALYSIS WITH DIFFERENT ASSIGNED N.O.C ON CIFAR-10. GAP DENOTES THE DIFFERENCE IN TESTING ACCURACY BETWEEN *Fed-ensemble* AND *Fedprox*

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 V
SENSITIVITY OF NUMBER OF MODELS K ON MNIST. ACC MAX/MIN IS THE MAXIMUM/MINIMUM TESTING ACCURACY AMONG ALL MODELS IN THE ENSEMBLE. AVG ENTROPY IS THE AVERAGE OF THE ENTROPY OF THE EMPIRICAL PREDICTIVE DISTRIBUTION ACROSS ALL MODELS

	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 VI
Fed-ensemble'S TEST ACCURACY ON CIFAR10 WITH DIFFERENT NUMBER OF MODELS K

K	1	2	5
i.i.d.	86.4±0.1	88.6 ± 0.1	88.0± 0.1
non i.i.d.	66.1±0.1	68.0±0.2	71.2±0.3

and test the performance on MNIST. We vary K from 3 to 80. The results are shown in Table V. Besides testing the accuracy of ensemble predictions, we also calculate the accuracy and the entropy of the predictive distribution of each model in the ensemble. As shown in Table V, 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 the number of data points assigned to each model, significantly drops. This, in turn, decreases learning accuracy, specifically when datasets are relatively small.

To see if similar patterns arise in larger image datasets, we also study the relation between predictive performance and the number of models K in the larger dataset CIFAR10. More specifically, we randomly partition the CIFAR10 dataset to 100 clients. Then we fit *Fed-ensemble* with a different number of models K on the partitioned dataset. Results are shown in Table. VI.

From Table. VI one can see that when the number of models K increases, the test accuracy also increases. This is especially the case for the non-i.i.d. data partition. Table. VI again corroborates our theoretical analysis in Corollary IV that the prediction becomes more accurate when K is larger.

G. Ablation Study on Different Training Algorithms

In this set of ablation studies, we compare algorithm 1 with another algorithm to train the ensemble. Remember

TABLE VII
TEST ACCURACY OF *Fed-ensemble* UNDER DIFFERENT SAMPLING SCHEMES

	CIFAR10 i.i.d.	CIFAR10 non-i.i.d.
ensembling with random sampling	87.8±0.1	70.6±0.2
Fed-ensemble	88.0± 0.1	71.2±0.3

that algorithm 1 uses random shuffling to ensure that each model in the ensemble is trained on all clients every K communication rounds. To show the effect of this mechanism, we design a simplified version of algorithm 1, wherein every communication round, each client randomly receives one from K models and updates it. We compare *Fed-ensemble* with the random sample method on CIFAR10. Results are shown in Table. VII. We use "ensembling with random sampling" to denote the simplified algorithm where clients randomly receive one model to update in every communication round. Still, we calculate the mean and standard deviation of the test accuracy from the last 3 communication rounds.

It is seen that the algorithm with random sampling performs comparably to *Fed-ensemble* in the i.i.d. partition regime. However, in the non-i.i.d. regime, *Fed-ensemble* outperforms the algorithm with random sampling. The results show that numerically, the random shuffling mechanism in *Fed-ensemble* is helpful for model optimization. This highlights *Fed-ensemble*'s ability to train ensemble models reliably and efficiently.

H. Extension to Fully Decentralized Setting

In this subsection, we discuss the extension of *Fed-ensemble* in the fully decentralized setting, where multiple connected clients train models collaboratively without the orchestration of a central server. One effective algorithm in decentralized learning is the random walk. Random walk algorithms are related to incremental gradient methods [49] and are active research topics in distributed optimization [50], [51], [52]. The motivation of the random walk algorithm is very simple. Suppose that in a decentralized system, clients are connected by a graph $\mathcal{G}$, where the nodes denote clients and edges denote connectivity, i.e., client i is connected to client j if and only if there is an edge connecting node i and j in $\mathcal{G}$. An example of $\mathcal{G}$ is shown in Fig. 5. The random walk algorithm works as follows. In one round, one client updates a model on its local dataset, then randomly sends the updated model to one of its neighbors in $\mathcal{G}$. That neighbor receives the model, updates it, and again sends it to a random neighbor. It is shown [50], [53] that theoretically, such procedures converge into stationary points in expectation and have good numerical performance with carefully designed transition probability matrix on the graph $\mathcal{G}$.

We term the above process as *Fedavg* Random Walk. A natural extension of *Fed-ensemble* in the decentralized setting is *Fed-ensemble* Random Walk, where we independently train K models using a random walk and still use the ensemble average to make predictions.

To compare the performance of *Fedavg* Random Walk and *Fed-ensemble* Random Walk, we run numerical simulations

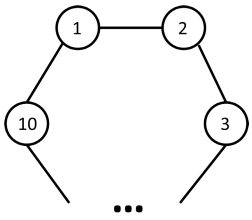


Fig. 5. Ring topology of clients connectivity graph $\mathcal{G}$. Circles denote clients, and edges denote client connectivity.

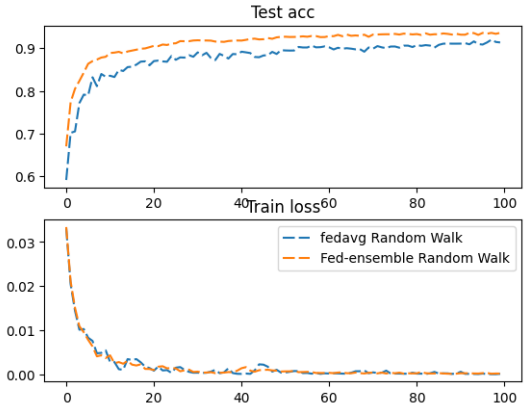


Fig. 6. Learning curve of two random walk algorithms. The x-axis is the round of random walks of individual models.

on CIFAR10. In the experiment, we use 10 clients connected in a ring graph, shown in Fig. 5. In Fig. 5, every client is connected to only 2 neighbors.

We use the i.i.d. partition to separate the CIFAR10 dataset into 10 clients. We then run *Fedavg* Random Walk and *Fed-ensemble* Random Walk on this instance and compare the testing accuracy of the two models. Results are shown in Fig. 6.

We can see that the random walk version of *Fed-ensemble* again outperforms the random walk version of *Fedavg*.

I. 3D Printing Dataset

Finally, we present the results of a case study on 3D printers. 3D printing is a popular technology in modern manufacturing. A 3D printer usually prints an object layer by layer. For material extrusion printers, the printhead moves along the predefined trajectory to stack the printing material on top of base materials.

Printing quality and time efficiency are two important factors in material extrusion 3D printing. During printing, the printhead vibrates due to various operating conditions. Such vibration degrades the quality of the printed objects and even results in model distortions or scrapped parts [54]. The printhead vibration is directly related to the speed at which the printhead moves along the trajectory. When the speed is higher, the printhead carries larger momentum and thus vibrates more. Time efficiency is another issue for material extrusion 3D printers. It can take hours for an ordinary material extrusion 3D printer to print a simple object. Increasing printing speed can decrease the time for printing the object, but printers can suffer from a higher level of vibration.

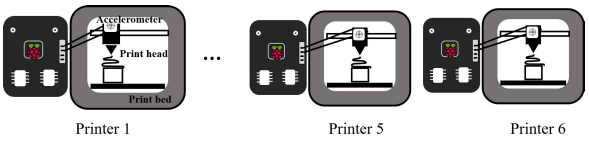


Fig. 7. An illustration of 6 smart and connected 3D printers. Vibration data are collected by sensors attached to print heads.

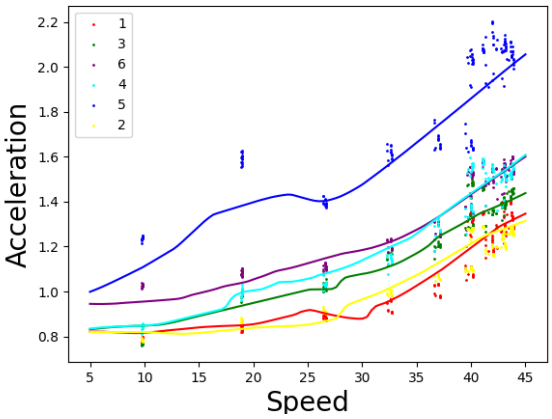


Fig. 8. The printing speed and average acceleration of 6 different 3D printers. Different colors represent different clients. Dots represent observations, and lines represent *Fed-ensemble* predictions.

To balance printing quality and time efficiency, we study the print speed to vibration curve of material extrusion 3D printers. The dataset is collected on 6 Ender 3D printers (The dataset can be found at [55]). The printers have the same type but are shipped in the package and assembled in different places. Therefore, they share similarities in mechanical design while retaining some uniqueness from the assembling process. The 6 printers are asked to print the same object at different printing speeds. Vibration is measured by accelerometers glued to the print head to measure the acceleration in the x and y direction. When the measured accelerations are higher, the vibration level is also higher. In each printing experiment, the accelerometers collect acceleration signals on the x and y axis at the frequency of 227 Hz. We calculate the root mean square of acceleration signals in x axis σ_{a_x} and in y axis σ_{a_y}, and use the mean square acceleration $\sigma_a = \sqrt{\sigma_{a_x}^2 + \sigma_{a_y}^2}$ to represent the average level of acceleration. The task is to predict the acceleration σ_a at a given print speed s for a printer. An illustration of the 3D printers is shown in Fig. 7.

We split the dataset collected on each printer into a training and a testing set. The testing set contains the data when the print speed is set to 30. And the training set contains the remaining collected data. We compare *Fed-ensemble* and *Fedavg* on the acceleration prediction task. We use a fully connected neural network as the parametric model $f_{\boldsymbol{w}}$. The input to the model $f_{\boldsymbol{w}}$ is the printing speed s, and the printer label, and the output is the acceleration.

We plot the data and predictions of the fully trained model in Fig. 8. Measured data are denoted as dots, and the predictions are denoted as solid lines. Different colors represent data and predictions on different clients. From Fig. 8, it is clear that *Fed-ensemble* is flexible to fit observations on each dataset while learning the shared information about the shape of the speed-vibration curve.

TABLE VIII
TESTING MEAN SQUARE ERROR ON THE 3D PRINTER VIBRATION DATASET. WE REPEAT EACH EXPERIMENT 5 TIMES TO ESTIMATE THE STANDARD DEVIATION

Model	Testing error
Fedavg	$(7.6 \pm 0.7) \times 10^{-3}$
Fed-ensemble	$(5.5 \pm 0.9) \times 10^{-3}$

Furthermore, the testing error is shown in Table VIII. From Table VIII, *Fed-ensemble* has a lower testing error, which indicates *Fed-ensemble* has a better predictive performance. Therefore, *Fed-ensemble* can give a more precise prediction on acceleration.

VI. CONCLUSION

In this work, we propose *Fed-ensemble*. *Fed-ensemble* provides a systematic approach to quantify uncertainty and boost the model capacity of existing FL techniques, all while retaining the same communication cost of training a single model. We provide convergence guarantees that extend FL literature to the kernel regime. The theory presents insights on the improved generalization power of *Fed-ensemble* through reducing variance in the predictions. An approach to obtain personalized predictions from *Fed-ensemble* is also proposed to further mitigate heterogeneity.

We believe *Fed-ensemble* opens up new directions in FL where clients collaboratively build more robust and flexible models without increasing communication costs. It may find value across different predictive models where uncertainty quantification is critical. A key realization from *Fed-ensemble* is its unique ability to handle heterogeneity by ensembling, which is shown to be advantageous when compared to existing techniques that add strong regularizers. An in-depth investigation of this phenomenon is an interesting direction we are currently pursuing. Also, we believe that exploiting different models in the ensemble for detecting client drift, anomalies, or fairness challenges in model training may be an area worth investigating.

VII. SUPPLEMENTARY MATERIAL

The supplementary material contains: Appendix 1) A formal proof of our main theorems. Appendix 2) Additional communication round-to-accuracy figures.

REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[2] P. Kairouz et al., "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[4] W. D. Brouwer, "The federated future is ready for shipping," doc.ai, Palo Alto, CA, USA, Tech. Rep., Accessed: Jul. 18, 2020.

[5] A. Hard et al., "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[6] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[7] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, 2012.

[8] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.

[9] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," 2019, *arXiv:1910.10252*.

[10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[11] F. Lai et al., "FedScale: Benchmarking model and system performance of federated learning at scale," 2021, *arXiv:2105.11367*.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. 3rd MLSys Conf.*, 2018, pp. 429–450.

[13] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," in *Proc. NIPS*, 2020, pp. 5332–5344.

[14] S. J. Reddi et al., "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–38. [Online]. Available: https://openreview.net/forum?id=LkFG3lB13U5

[15] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," 2019, *arXiv:1901.09269*.

[16] X. Liu, Y. Li, J. Tang, and M. Yan, "A double residual compression algorithm for efficient distributed learning," 2019, *arXiv:1910.07561*.

[17] R. Pathak and M. J. Wainwright, "FedSplit: An algorithmic framework for fast federated optimization," in *Proc. NIPS*, 2020, pp. 7057–7066.

[18] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "FedPD: A federated learning framework with optimal rates and adaptivity to non-IID data," 2020, *arXiv:2005.11418*.

[19] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–43. [Online]. Available: https://openreview.net/forum?id=B7v4QMR6Z9w

[20] P. P. Liang et al., "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv:2001.01523*.

[21] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.

[22] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," 2020, *arXiv:2012.04221*.

[23] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," 2020, *arXiv:2002.05516*.

[24] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020, pp. 21394–21405.

[25] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of DNNs," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–10.

[26] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," 2019, *arXiv:1912.02757*.

[27] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," 2020, *arXiv:2002.08791*.

[28] H.-Y. Chen and W.-L. Chao, "Fedbe: Making Bayesian model ensemble applicable to federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–21. [Online]. Available: https://openreview.net/forum?id=dgtpE6gKjHn

[29] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=BkluqlSFDS

[30] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. ICLR*, 2019, pp. 7252–7261.

[31] W. Zhang and X. Li, "Data privacy preserving federated transfer learning in machinery fault diagnostics using prior distributions," *Struct. Health Monitor.*, vol. 21, no. 4, pp. 1329–1344, Jul. 2022, doi: 10.1177/14759217211029201.

[32] W. Zhang and X. Li, "Federated transfer learning for intelligent fault diagnostics using deep adversarial networks with data privacy," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 1, pp. 430–439, Feb. 2022.

[33] M. Dhada, A. Parlikad, and A. S. Palau, "Federated learning for collaborative prognosis," Dept. Eng., Inst. Manuf., Univ. Cambridge, Cambridge, U.K., Tech. Rep. 288348890, 2020.

[34] R. Kontar et al., "The internet of federated things (IoFT)," *IEEE Access*, vol. 9, pp. 156071–156113, 2021.

[35] N. Shi and R. A. Kontar, "Personalized federated learning via domain adaptation with an application to distributed 3D printing," *Technometrics*, pp. 1–22, Jan. 2022, doi: 10.1080/00401706.2022.2157882.

[36] N. Rieke et al., "The future of digital health with federated learning," *NPJ Digit. Med.*, vol. 3, no. 1, pp. 1–7, 2020.

[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[39] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. NIPS*, 2018, pp. 1–10.

[40] J. Lee et al., "Wide neural networks of any depth evolve as linear models under gradient descent," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[41] B. Huang, X. Li, Z. Song, and X. Yang, "FL-NTK: A neural tangent kernel-based framework for federated learning convergence analysis," 2021, *arXiv:2105.05001*.

[42] A. G. D. G. Matthews, J. Hron, R. E. Turner, and Z. Ghahramani, "Sample-then-optimize posterior sampling for Bayesian linear models," in *Proc. NeurIPS Workshop Adv. Approx. Bayesian Inference*, 2017, pp. 1–5.

[43] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in gradient boosting via ensembles," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=1Jv6b0Zq3qi

[44] S. Caldas et al., "LEAF: A benchmark for federated settings," in *Proc. NIPS*, 2019, pp. 1–9.

[45] A. Kuznetsova et al., "The open images dataset v4," *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1–26, 2020.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[47] N. Shi, D. Li, M. Hong, and R. Sun, "RMSprop converges with proper hyper-parameter," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–10. [Online]. Available: https://openreview.net/forum?id=3UDSdyIcBDA

[48] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. MLSys*, 2019, pp. 374–388.

[49] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.

[50] A. Triastcyn, M. Reisser, and C. Louizos, "Decentralized learning with random walks and communication-efficient adaptive optimization," in *Proc. Workshop Federated Learn., Recent Adv. New Challenges (NIPS)*, 2022, pp. 1–30. [Online]. Available: https://openreview.net/forum?id=QwL8ZGl_QGG

[51] T. Sun, D. Li, and B. Wang, "Adaptive random walk gradient descent for decentralized optimization," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20790–20809.

[52] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. H. Sayed, and W. Yin, "Walkman: A communication-efficient random-walk algorithm for decentralized optimization," *IEEE Trans. Signal Process.*, vol. 68, pp. 2513–2528, 2020.

[53] T. Sun, Y. Sun, and W. Yin, "On Markov chain gradient descent," *Adv. neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.

[54] M. Duan, D. Yoon, and C. E. Okwudire, "A limited-preview filtered B-spline approach to tracking control–with application to vibration-induced error compensation of a 3D printer," *Mechatronics*, vol. 56, pp. 287–296, Dec. 2018.

[55] IoFT Datasets. (2021). *The Internet of Federated Things (IoFT) Data Directory*. Accessed: Jul. 18, 2021. [Online]. Available: https://ioft-data.engin.umich.edu/