BooNTK: Convexifying Federated Learning using Bootstrapped Neural Tangent Kernels

Anonymous Author(s)
Affiliation
Address
email

Abstract

State-of-the-art federated learning methods can perform far worse than their centralized counterparts when clients have dissimilar data distributions. For neural networks, even when centralized SGD easily finds a solution that is simultaneously performant for all clients, current federated optimization methods fail to converge to a comparable solution. We show that this performance disparity can largely be attributed to optimization challenges presented by *non-convexity*. Specifically, we find that the early layers of the network do learn useful features, but the final layers fail to make use of them. That is, federated optimization applied to this non-convex problem distorts the learning of the final layers. Leveraging this observation, we propose a simple, two-stage training procedure *Bootstrapped eNTK* (BooNTK)[1] to sidestep this issue: first, learn features using off-the-shelf methods (e.g. FedAvg); then, optimize a *convexified* problem obtained from the network's empirical neural tangent kernel approximation. Our technique yields accuracy improvements of up to $+36\%$ on FMNIST and $+37\%$ on CIFAR10 when clients have dissimilar data.

1 Introduction

Federated learning is a newly emerging paradigm for machine learning where multiple data holders (clients) collaborate to train a model on their combined dataset. Clients only share partially trained models and other statistics about their dataset, keeping their raw data local and private [46, 32]. By obviating the need for a third party to collect and store clients' data, federated learning has several advantages over the classical, centralized paradigm [13, 27, 22]: it ensures clients' consent is tied to the specific task at hand by requiring active participation of the clients in training, confers some basic level of privacy, and has the potential to make machine learning more participatory in general [38, 31]. Further, widespread legislation of data portability and privacy requirements (such as GDPR and CCPA) might even make federated learning a necessity [51].

Collaboration among clients is most attractive when clients have very different subsets of the combined dataset (*data heterogeneity*). For example, different autonomous driving companies may only be able to collect data in weather conditions specific to their location, whereas their vehicles would need to function under all conditions. In such a scenario, it would be mutually beneficial for companies in geographically diverse locations to collaborate and share data with each other. Further, in such settings, clients are physically separated and connected by ad-hoc networks with large latencies and limited bandwidth. This is especially true when clients are edge devices such as mobile phones, IoT sensors, etc. Thus, *communication efficiency* is crucial for practical federated learning. However, it is precisely under such circumstances (large data heterogeneity and low communication) that current algorithms fail dramatically [25, 43, 34, 53, 63, 1, 41, 2, 64, etc.]. This motivates our central question: *Why do current federated methods fail in the face of data heterogeneity—and how can we fix them?*

[1] Pronounced *boon-T-K*.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Our solution. We make two main observations: (i) We show that, even with data heterogeneity, linear models can be trained in a federated manner through gradient correction techniques such as SCAFFOLD [34]. While this observation is promising, it alone remains limited, as linear models are not rich enough to solve practical problems of interest (e.g. those that require feature learning). (ii) We shed light on why current federated algorithms struggle to train deep, non-convex models. We observe that the failure of existing methods for neural networks is not uniform across the layers. The early layers of the network do in fact learn useful features, but the final layers fail to make use of them. Specifically, federated optimization applied to this non-convex problem results in distorted final layers.

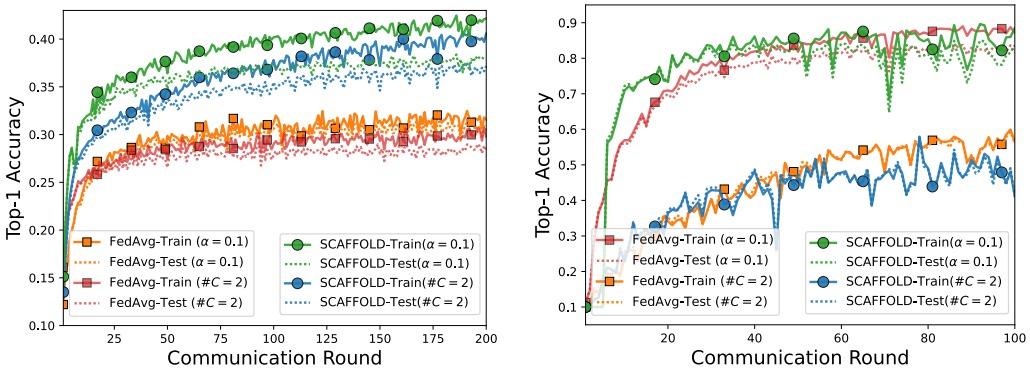
These observations suggest a two-stage federated training algorithm, which we call *BooNTK*: first, use any off-the-shelf federated algorithm (such as FedAvg [46]) to train a deep model to extract useful features; then, compute a convex approximation of the deep model using its empirical Neural Tangent Kernel (eNTK) [29, 39, 20, 66], and use gradient correction methods such as SCAFFOLD [34] to train the final model. Effectively, the second stage features freezes the features learnt in the first stage and learns a linear model over them. We show that this simple strategy is highly performant on a variety of tasks and models—we see accuracy gains up to 36% points on FMNIST with a CNN, 37% points on CIFAR10 with ResNet18-GN, and 16% points on CIFAR100 with ResNet18-GN. Further, its convergence remains unaffected even by extreme data heterogeneity.

2 Related Work

Federated learning. There are two main motivating scenarios for federated learning (FL). The first is where internet service companies (e.g. Google, Facebook, Apple, etc.) want to train machine learning models over their users' data, but do not want to transmit raw personalizable data away from user devices [52, 7]. This is the setting of *cross-device* federated learning and is characterized by an extremely large number of unreliable clients, each of whom has very little but homogeneous data [32, 9, 33, 7]. The second motivating scenario is when valuable data is split across different organizations, each of whom is either protected by privacy regulation or is simply unwilling to share their raw data. Such "data islands" are common among hospital networks, financial institutions, autonomous vehicle companies, etc. This is known as *cross-silo* federated learning and is characterized by a few highly reliable clients, who potentially have extremely diverse data. In this work, we focus on the latter.

Metrics in FL. FL research considers numerous metrics, such as fairness across users [48, 42, 54], formal security and privacy guarantees [8, 52, 21, 49], robustness to corrupted agents and data during training [6, 56, 19, 35], preventing backdoors at test time [5, 58, 61, 45], etc. While these concerns are important, the main goal of FL (and our work) is to achieve high accuracy with minimal communication [46]. Clients are typically geographically separated yet need to communicate large deep learning models over unoptimized ad-hoc networks [32]. Furthermore, we focus on the setting where all users are interested in training the same model over the combined dataset. This is in contrast to model-agnostic protocols [44, 50, 2] or personalized federated learning [15, 18, 69, 11, 37]. Finally, we focus on minimizing the number of rounds required. Our approach can be combined with communication compression, which reduces bits sent per round [59, 3, 23, 57].

Federated optimization. Algorithms for FL proceed in rounds. In each round, the server sends a model to the clients, who partially train this model using their local compute and data. The clients send these partially trained models back to the server who then aggregates them, finishing a round. FedAvg [46], which is the de-facto standard FL algorithm, uses SGD to perform local updates on the clients and aggregates the client models by simply averaging their parameters. However, FedAvg was observed to perform terribly when faced with data heterogeneity across the clients [25, 43, 34, 53, 63, 1, 41, 2, 64, etc.]. Theoretical investigations of this phenomenon [34, 67] hypothesized that this was a result of *gradient heterogeneity* across the clients. Consider FedAvg initialized with the globally optimal model. If this model is not also optimal for each of the clients as well, the local updates will push it away from the global optimum. Thus, convergence would require a careful tuning of hyper-parameters. To overcome this issue, SCAFFOLD [34] and FedDyn [1] propose to use control variates to correct for the biases of the individual clients akin to variance reduction [30, 14]. This *gradient correction* is applied in every local update by the client and provably nullifies the effect of gradient heterogeneity [34, 47]. However, as we show here, such methods seem insufficient to overcome high data heterogeneity especially for deep learning. Other, more ad-hoc approaches to combat gradient heterogeneity include using a regularizer FedProx [43] and sophisticated server



(a) Using linear model.

(b) Using ResNet-18.

Figure 1: Performance of FedAvg and SCAFFOLD on CIFAR10 split among 10 clients in two ways (#C=2 and α=0.1). The #C=2 split is more non-i.i.d. than the α=0.1 split. For convex problems (left), gradient correction methods such as SCAFFOLD are relatively unaffected by data heterogeneity, and consistently outperform FedAvg. However, for non-convex problems (right), FedAvg and SCAFFOLD perform very similarly and both are strongly negatively affected by data heterogeneity.

aggregation strategies such as momentum [26, 62] or adaptivity [53, 10]. A second line of work pins the blame on performance loss due to averaging non-convex models. To overcome this, [55, 71] propose to learn a mapping between the weights of the client models before averaging, [2] advocates a functional perspective and replaces the averaging step with knowledge distillation, and [65, 41, 60] attempt to align the internal representations of the client models. However, averaging is unlikely to be the only culprit since FedAvg does succeed under low heterogeneity, and averaging non-convex models can lead to improved performance [28, 68].

3 The Effect of Non-convexity

In this section, we empirically investigate the poor performance of FedAvg [46] and SCAFFOLD [34] when applied to deep neural networks, focusing on image classification with a ResNet-18. To construct our federated learning setup, we split the CIFAR-10 dataset in a highly heterogeneous manner among 10 clients. We either assign each client two classes (denoted by #C=2) or distribute samples according to a Dirichlet distribution with $\alpha = 0.1$ (denoted by α=0.1). For more details, see Section 5.1.

Insufficiency of gradient correction methods. Prior theoretical work (e.g. [34, 53, 63, 1, 64]) attributes the slowdown from data heterogeneity to the individual clients having wildly varying local optima. If no single model is simultaneously optimal for all clients, then the updates of different clients can compete with and distort each other, leading to slow convergence. This tension is captured by the variance of the updates across the clients (a.k.a. client gradient heterogeneity [64]). Gradient correction methods such as SCAFFOLD [34] and FedDyn [1] explicitly correct for this and are provably unaffected by gradient heterogeneity for both convex and non-convex losses.

These theoretical predictions are perfectly aligned with the results of Figure 1(a), where the loss landscape is convex: SCAFFOLD is relatively unaffected by the level of heterogeneity and consistently outperforms FedAvg. In particular, performance is largely dictated by the algorithm and not the data distributions. This shows that client gradient heterogeneity captures the difficulty of the problem well. On the other hand, when training a ResNet-18 model with non-convex loss landscape, Figure 1(b) shows that both FedAvg and SCAFFOLD suffer from data heterogeneity. This is despite the theory of gradient correction applying to both convex and non-convex losses [34]. Further, the train and test accuracies in Figure 1(b) match quite closely, suggesting that the failure lies in optimization (not fitting the training data) rather than generalization. The difference between theoretical predictions and practical reality in the non-convex setting suggests black-box notions such as gradient heterogeneity are insufficient for capturing the difficulty of training deep models.

Table 1: Feature learning by FedAvg. We report the test accuracy of a ResNet-18 after (centralized) retraining of the last ℓ layers on CIFAR10. The earlier $(7-\ell)$ layers are frozen to either random initialization or to the weights of a FedAvg-trained model. The difference measures utility of the $(7-\ell)$ layers learnt by FedAvg. The baseline FedAvg model without additional training gets 56.9% accuracy. We see that all layers of the FedAvg model contain useful information.

Layers retrained	Accuracy (%) Random init	Accuracy (%) FedAvg init	Improvement (%) (FedAvg - Random)
1/7 last layer	35.37	77.93	42.56
2/7 last layers	67.33	87.04	19.71
3/7 last layers	80.18	89.28	9.10
4/7 last layers	88.03	90.57	2.54
5/7 last layers	91.34	91.61	0.27
6/7 last layers	91.78	91.91	0.13

Ease of feature learning. We now dive into how a ResNet-18 trained with FedAvg (56.9% accuracy) differs from the centralized baseline (91.9% accuracy). We first apply linear probing to the FedAvg model (i.e. retraining with all but the output layer frozen). Note that this is equivalent to (convex) logistic regression over the last layer activations. This simple procedure produces a striking jump from 56.9% to 77.9% accuracy. Thus, of the 35% gap in accuracy between the FedAvg and centralized models, 21% may be attributed to a failure to optimize the linear output layer. We next extend this experiment towards probing the information content of other layers.

Given a FedAvg-trained model, we can use centralized training to retrain only the last ℓ layers while keeping the rest of the $(7-\ell)$ layers (or ResNet blocks) frozen. We can also perform this procedure starting from a randomly initialized model. The performance difference between these two models can be attributed to the information content of the frozen $(7-\ell)$ layers of the FedAvg model. Table 1 summarizes the results of this experiment. The large difference in accuracy (up to 42.6%) indicates the initial layers of the FedAvg model have learnt useful features. There continues to be a gap between the FedAvg features and random features in the earlier layers as well[2], meaning that all layers of the FedAvg model learn useful features. We conjecture this is because from the perspective of earlier layers which perform simple edge detection, the tasks are independent of labels and the clients are i.i.d. However, the higher layers are more specialized and the effect of the heterogeneity is stronger.

4 Method

Based on our observations in Section 3, we propose *Bootstrapped eNTK* as a method for overcoming data heterogeneity when training deep models in a federated setting. Our high-level intuition is that we want to leverage both the features learned from applying FedAvg to neural networks and the effectiveness of *convex* federated optimization. We thus perform several rounds of "bootstrap" FedAvg to learn features before solving a convexified version of the original optimization problem.

4.1 Computing the Empirical Neural Tangent Kernel

To sidestep the challenges presented by non-convexity, we describe how we approximate a neural network by its "linearization." Given a neural network $f(\cdot;\theta_0)$ with weights $\theta_0 \in \mathbb{R}^P$ mapping inputs $x \in \mathbb{R}^D$ to $\mathbb{R}^C$, we replace it by its *empirical neural tangent kernel (eNTK)* approximation at θ_0 given by

$$f(x;\theta) \approx f(x;\theta_0) + (\theta-\theta_0)^\top \frac{\partial}{\partial\theta} f(x;\theta_0)$$

at each $x \in \mathbb{R}^D$. Under this approximation, $f(x;\theta)$ is a linear function of the "feature vector" $(f(x;\theta_0), \frac{\partial}{\partial\theta} f(x;\theta_0))$ and the original non-convex optimization problem becomes (convex) linear regression with respect to these features.[3]

[2]The significant decrease in the gap as we go down the layers may be because of the skip connections in the lower ResNet blocks which allow the random frozen layers to be side-stepped. This underestimates the true utility and information content in the earlier FedAvg layers.

[3]For classification problems, we one-hot encode labels and fit a linear model using squared loss.

To reduce the computational burden of working with the eNTK approximation, we make two further approximations: First, we randomly reinitialize the last layer of θ_0 and only consider $\frac{\partial}{\partial\theta} f(x;\theta_0)$ with respect to a single output logit. Over the randomness of this reinitialization, $\mathbb{E}[f(x;\theta_0)] = 0$. And given the random reinitialization, all the output logits of $f(x;\theta_0)$ are symmetric. These observations mean each data point x can be represented by a P-dimensional feature vector $\frac{\partial}{\partial\theta} f_1(x;\theta_0)$, where $f_1(\,\cdot\,;\theta_0)$ refers to the first output logit. Then, we apply a dimensionality reduction by subsampling p random coordinates from this P-dimensional featurization.[4] In our setting, this sub-sampling has the added benefit of reducing the number of bits communicated per round.

In summary, we transform our original (non-convex) optimization problem over a neural network initialized at θ_0 into a convex optimization problem in three steps: (i) reinitialize the last layer of θ_0; (ii) for each data point x, compute the gradient $\phi_{\mathrm{eNTK}}(x;\theta_0) := \frac{\partial}{\partial\theta} f_1(x;\theta_0)$; (iii) subsample the coordinates of $\phi_{\mathrm{eNTK}}(x;\theta_0)$ for each x to obtain a reduced-dimensionality eNTK representation. Let $\mathcal{S}\colon \mathbb{R}^P \to \mathbb{R}^p$ denote this subsampling operation. Finally, we solve the resulting linear regression problem over these eNTK representations.[5]

4.2 Convexifying Federated Learning via eNTK Representations

The eNTK approximation lets us convexify the neural net optimization problem: following Section 4.1, we may extract (from a model trained with FedAvg) eNTK representations of inputs from each client. It remains to fit an overparameterized linear model using these eNTK features in a federated manner. For ease of presentation, we denote the subsampled eNTK representation of input x by $z \in \mathbb{R}^p$, where p is the eNTK feature dimension after subsampling. We use z_i^k to represent the eNTK feature of the i-th sample from the k-th client. Then, for K the number of clients, Y_i^k the one-hot encoded labels, n_k the number of data points of the k-th client, $n := \sum_{k\in[K]} n_k$ the number of data points across all clients, and $p_k := n_k/n$, we can approximate the non-convex neural net optimization problem by the convex linear regression problem

$$\min_W L(W) := \sum_{k=1}^{K} p_k \cdot L_k(W), \qquad \text{where} \quad L_k(W) := \frac{1}{n_k}\sum_{i=1}^{n_k} \|W^\top z_i^k - Y_i^k\|_2^2. \tag{1}$$

To obtain the eNTK representation z of an input x, we take θ_0 in Section 4.1 to be the weights of a model trained with FedAvg. As we will show in Section 5, the convex reformulation in Eq. (1) significantly reduce the number of communication rounds needed to find an optimal solution.

4.3 Bootstrapped eNTK (BooNTK)

We now present our algorithm *Bootstrapped eNTK* (BooNTK) for federated optimization.

Bootstrapped eNTK (BooNTK)

- **Stage 1:** *Extracting eNTK features from a FedAvg-trained model.* FedAvg is first used to train the model for T_1 communication rounds. Let θ_{T_1} the model weights after these T_1 rounds. Then, each client locally computes subsampled eNTK features, i.e., $z_i^k = \mathcal{S}(\phi_{\mathrm{eNTK}}(x_i^k;\theta_{T_1}))$ for $k \in [K]$ and $i \in [n_k]$.
- **Stage 2:** *Decentralized linear regression with gradient correction.* Given samples $\{(z_i^k, Y_i^k)\}_{i=1}^{n_k}$ on each client k, first normalize the eNTK inputs of all clients with a single communication round.[a] Then, solve the linear regression problem defined in Eq. (1) by SCAFFOLD with local learning rate η and local steps M.[b]

[a] For every feature in the eNTK representation, subtract the mean and scale to unit variance.

[b] The detailed description of SCAFFOLD for solving linear regression problems can be found in Algorithm 1, Appendix A. It has the same communication and computation cost as FedAvg.

[4] That such representations empirically have low effective dimension due to fast eigenvalue decay (see e.g. [66]) means that such a random projection approximately preserves the geometry of the data points [4]. For all of our experiments, we set $p = 100,000$.

[5] Given a fitted linear model with weights $W \in \mathbb{R}^{p\times C}$, the prediction at x is $\arg\max_j [W^\top \mathcal{S}(\phi_{\mathrm{eNTK}}(x))]_j$.

To motivate BooNTK, recall that in Section 3 we found that FedAvg learns "useful" features despite its poor performance, especially in the earlier layers. By taking an eNTK approximation, BooNTK optimizes a convex approximation while using information from *all* layers of the model. Empirically, we find that these extracted eNTK features significantly reduce the number of communication rounds needed to learn a performant model, even with data heterogeneity.

5 Experiments

We now study the performance of BooNTK for the decentralized training of deep neural networks in the presence of data heterogeneity. We compare BooNTK to state-of-the-art federated learning algorithms on three benchmark tasks in federated learning. For each task, we apply these algorithms on client data distributions with varying degrees of data heterogeneity. We find that our proposed approach significantly outperforms existing algorithms when clients have highly heterogeneous data across all tasks. For additional experimental results and implementation details, see Appendix B.

5.1 Experimental Setup

Datasets and degrees of data heterogeneity. We assess the performance of federated learning algorithms on the image classification tasks FMNIST [70], CIFAR10, and CIFAR100 [36]. FMNIST and CIFAR10 each consist of 10 classes, while CIFAR100 includes images from 100 classes. There are 60,000 training images in FMNIST, and 50,000 training images in CIFAR10/100.

To vary the degree of data heterogeneity, we follow the setup of Li et al. [40]. We consider two types of non-i.i.d. data distribution: *(i) Data heterogeneity sampled from a symmetric Dirichlet distribution with parameter α [44, 63].* That is, we sample $p_c \sim \mathrm{Dir}_K(\alpha)$ from a K-dimensional symmetric Dirichlet distribution and assign a p_c^k-fraction of the class c samples to client k. (Smaller α corresponds to more heterogeneity.) *(ii) Clients get samples from a fixed subset of classes [46].* That is, each client is allocated a subset of classes; then, the samples of each class are split into non-overlapping subsets and assigned to clients that were allocated this class. We use `#C` to denote the number of classes allocated to each client. For example, `#C=2` means each client has samples from 2 classes. To allow for consistent comparisons, all of our experiments are run with 10 clients.

Models. For FMNIST, we use a convolutional neural network with ReLU activations consisting of two convolutional layers with max pooling followed by two fully connected layers (SimpleCNN). For CIFAR10 and CIFAR100, we mainly consider an 18-layer residual network [24] with 4 basic residual blocks (ResNet-18). In Appendix B.2, we present experimental results for other architectures.

Algorithms and training schemes. We compare BooNTK to state-of-the-art federated learning algorithms, focusing on the widely-used algorithms FedAvg [46], FedProx [43], and SCAFFOLD [34]. (For comparisons to additional algorithms, see Appendix B.1.) Each client uses SGD with weight decay 10^{-5} and batch size 64 by default. For each baseline method, we run it for 200 total communication rounds using 5 local training epochs with local learning rate selected from $\{0.1, 0.01, 0.001\}$ by grid search. For BooNTK, we run 100 rounds of FedAvg in Stage 1 following the above and use 100 communication rounds in Stage 2 with $M = 500$ local steps and local learning rate $\eta = 5 \cdot 10^{-5}$.

5.2 Main Results

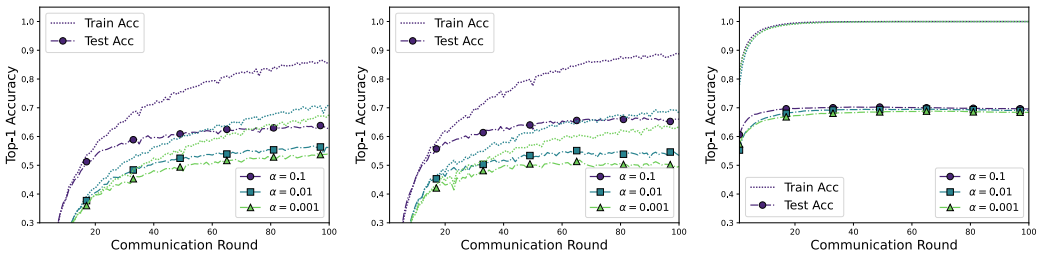
Table 2 displays the top-1 accuracy of all algorithm on the three tasks with varying degrees of data heterogeneity. We evaluated each algorithms on each task under four degrees of data heterogeneity. Smaller `#C` and α in Table 2 correspond to higher heterogeneity.

We find that the existing federated algorithms all suffer when data heterogeneity is high across all three tasks. For example, the top-1 accuracy of FedAvg on CIFAR-10 is 56.86% when `#C=2`, which is much worse than the 90.43% achieved in a more homogeneous setting (e.g. $\alpha = 0.5$). In contrast, BooNTK achieves consistently strong performance, even in the face of high data heterogeneity. More specifically, BooNTK achieves the best top-1 accuracy performance across all settings except CIFAR-100 with $\alpha = 0.5$, where BooNTK does only slightly worse than SCAFFOLD.

In absolute terms, we find that BooNTK is not affected much by data heterogeneity, with performance dropping by less than 1.5% on CIFAR100 as α goes from 0.5 to 0.001. Moreover, our algorithm improves over existing methods by at least 15% in the challenging cases, including FMNIST with

Table 2: The top-1 accuracy (%) of our algorithm (BooNTK) vs. state-of-the-art federated learning algorithms evaluated on FMNIST, CIFAR10, and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\mathrm{Dir}_K(\alpha)$ and the `#C` parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**.

Datasets	**Architectures**	**Methods**	**Non-i.i.d. degree**			
			`#C` $= 1$	`#C` $= 2$	$\alpha = 0.1$	$\alpha = 0.5$
FMNIST	SimpleCNN	FedAvg	35.10%	85.18%	86.18%	90.09%
		FedProx	50.04%	84.91%	86.31%	89.77%
		SCAFFOLD	12.80%	42.80%	83.87%	89.40%
		BooNTK	**86.32%**	**90.33%**	**90.78%**	**91.13%**
			`#C` $= 1$	`#C` $= 2$	$\alpha = 0.1$	$\alpha = 0.5$
CIFAR-10	ResNet-18	FedAvg	11.27%	56.86%	82.60%	90.43%
		FedProx	12.30%	56.87%	83.31%	90.68%
		SCAFFOLD	10.00%	46.75%	80.46%	90.72%
		BooNTK	**49.92%**	**83.02%**	**89.21%**	**91.10%**
			$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$
CIFAR-100	ResNet-18	FedAvg	53.89%	54.22%	63.49%	67.65%
		FedProx	52.87%	54.32%	63.47%	67.54%
		SCAFFOLD	49.86%	54.07%	65.67%	**71.07%**
		BooNTK	**68.42%**	**69.07%**	**69.66%**	69.68%



(a) FedAvg. (b) SCAFFOLD. (c) BooNTK.

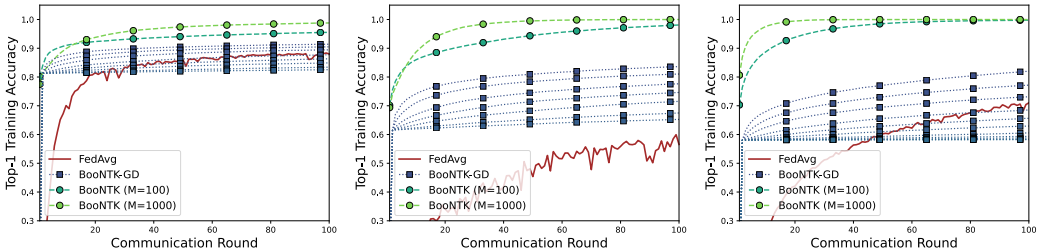
Figure 2: Training/test accuracy vs. communication round for FedAvg (left), SCAFFOLD (middle), and our algorithm BooNTK (right) on the CIFAR100 dataset with various degrees of non-iid-ness ($\mathrm{Dir}_K(\alpha)$ with $\alpha \in \{0.1, 0.01, 0.001\}$). Dotted lines represent the training accuracy, and dashdot lines with markers represent the test accuracy.

`#C=1`, CIFAR-10 with `#C=1` and `#C=2`, and CIFAR-100 with $\alpha = 0.01$ and $\alpha = 0.001$. And, perhaps surprisingly, our algorithm still performs relatively well in the extreme non-i.i.d. setting where each client sees only a single class.

Figure 2 compares the performances of FedAvg, SCAFFOLD, and BooNTK in more detail on CIFAR100 dataset with different degrees of data heterogeneity. We consider the Dirichlet distribution with parameter $\alpha \in \{0.1, 0.01, 0.001\}$ and compare the training and test accuracy of these three algorithms. As shown in Figures 2(a) and 2(b), both FedAvg and SCAFFOLD struggle when data heterogeneity is high: for both algorithms, test accuracy drops significantly when α decreases. In contrast, we see from Figure 2(c) that BooNTK maintains almost the same test accuracy for different α. Furthermore, the same set of default parameters for our algorithm, including local learning rate and the number of local steps, is relatively robust to different levels of data heterogeneity.

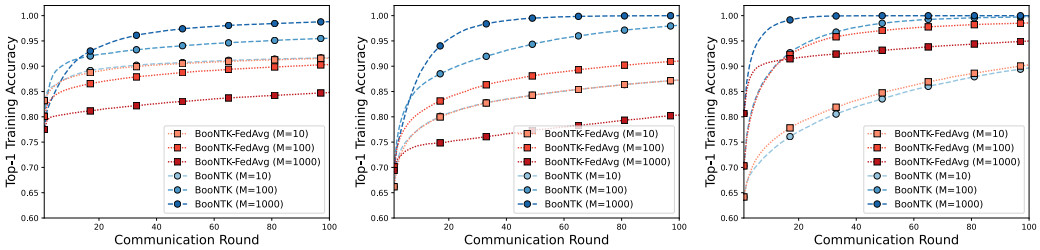
5.3 Communication Efficiency

To understand the effectiveness of the local steps in our algorithm, we compare SCAFFOLD (used in BooNTK-Stage 2) to full batch gradient descent (GD) applied to the overparameterized linear



(a) FMNIST. (b) CIFAR10. (c) CIFAR100.

Figure 3: Training accuracy vs. communication round for full batch gradient descent (GD) and BooNTK on FMNIST-[`#C=2`] **(a)**, CIFAR10-[`#C=2`] **(b)**, and CIFAR100-[$\alpha = 0.01$] **(c)**. Each dotted line with square markers represents the training accuracy of GD with some learning rate. Dashed lines with circle markers represent the training accuracy of BooNTK with different numbers of local steps. We also include the training accuracy results of FedAvg with learning rate $\eta = 0.1$. We use BooNTK-GD to denote the variant of BooNTK which replaces SCAFFOLD with GD in Stage 2.



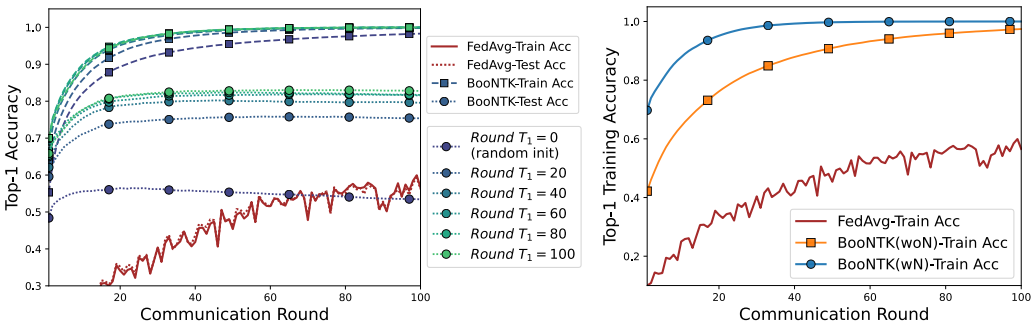
(a) FMNIST. (b) CIFAR10. (c) CIFAR100.

Figure 4: Comparing BooNTK to BooNTK-FedAvg for solving the overparameterized linear regression problem on **(a)** FMNIST-[`#C=2`], **(b)** CIFAR10-[`#C=2`], and **(c)** CIFAR100-[$\alpha = 0.01$]. We use BooNTK-FedAvg to denote a variant of BooNTK that uses FedAvg instead of SCAFFOLD to perform linear regression in BooNTK-Stage 2. Dotted red lines with square markers represent the training accuracy of BooNTK-FedAvg with different numbers of local steps. Dashed blue red lines with circle markers represent the training accuracy of BooNTK with different numbers of local steps. A darker color means more local steps.

regression problem in Stage 2 of BooNTK on these datasets. For our algorithm, we set local steps $M \in \{10^2, 10^3\}$ and use the default local learning rate. For full batch GD, we vary the learning rate from 10^{-5} to 10^{-1} and visualize the ones that do not diverge. The results are summarized in Figure 3. Each dotted line with square markers in Figure 3 corresponds to full batch GD with some learning rate. Across all three datasets, our proposed algorithm consistently outperforms full batch GD. Meanwhile, we find that more local steps for our algorithms lead to faster convergence across all settings. In particular, our algorithm converges within 20 communication rounds on CIFAR100 (as shown in Figure 3(c)). These results suggest that our proposed algorithm can largely leverage the local computation and improve communication efficiency.

5.4 Ablations

Gradient correction. We investigate the role of gradient correction when solving overparameterized linear regression with eNTK features in BooNTK. We compare SCAFFOLD (used in BooNTK) to FedAvg on solving the regression problems and summarize the results in Figure 4. We use the default local learning rate and consider three different numbers of local steps for both algorithms, i.e., $M \in \{10, 100, 1000\}$. As shown in Figure 4, our approach largely outperforms FedAvg when the number of local steps is large ($M \geq 100$) across three datasets. We also find that the performance of FedAvg can even degrade when the number of local steps increases. For example, FedAvg with $M = 1000$ performs the worst across all three datasets. In contrast to FedAvg, SCAFFOLD converges faster when the number of local steps increases. These observations highlight the importance of gradient correction in our algorithm.



(a) Effect of FedAvg communication rounds in Stage 1.

(b) Effect of normalization.

Figure 5: **(a).** We evaluate BooNTK on using checkpoints save at different communication rounds T_1 in Stage 1. $T_1 = 0$ corresponds to the randmon initialized model weights scenario (without FedAvg training). Dash lines with square markers represent the training accuracy, and dotted lines with circle makers represent the test accuracy. **(b).** We study the effect of pre-conditioning on BooNTK. BooNTK(wN) corresponds to the setting where eNTK features are normalized, and BooNTK(woN) corresponds to the without normalization step setting.

Model weights for computing eNTK features. To understand the impact of the model weights trained in Stage 1 of BooNTK, we evaluate BooNTK run with different T_1 parameters. We consider $T_1 \in \{0, 20, 40, 60, 80, 100\}$, where $T_1 = 0$ corresponds to randomly initialized weights. From Figure 5(a), we find that weights after FedAvg training are much more effective than weights at random initialization. Specifically, without FedAvg training, the eNTK (at random initialization) performs worse than standard FedAvg. In contrast, BooNTK significantly outperforms FedAvg by a large margin (by ~20% in test accuracy) when eNTK features are extracted from a FedAvg-trained model. Also, we find that BooNTK is stable with respect to the choice of communication rounds T_1 in Stage 1. For example, models trained by BooNTK with $T_1 \geq 60$ achieve similar performance.

Effect of normalization. In Figure 5(b), we investigate the role of normalization on BooNTK by comparing BooNTK run with normalized and unnormalized eNTK features. The same number of local steps ($M = 500$) is applied for both settings. We tune the learning rate η for each setting and plot the run that performs best (as measured in training accuracy). The results in Figure 5(b) suggest that the normalization step in BooNTK significantly improves the communication efficiency by increasing convergence speed. In particular, BooNTK with normalization converges to ~100% training accuracy in ~40 communication rounds, which is much faster than BooNTK without normalization.

6 Conclusion

Through experiments, we showed that non-convexity poses a significant challenge for federated learning algorithms. We found that a neural network trained in such a manner does learn some useful features, but fails to use them and thus has poor overall accuracy. To side-step this issue, we proposed a two-step training procedure: first, train the neural network using FedAvg; then, optimize (using SCAFFOLD) a convex approximation of the model obtained using its empirical neural tangent kernel. We showed that the first stage extracts meaningful features, whereas the second stage learns to utilize these features to obtain a highly performant model. Our algorithm obtains much higher test accuracies, is significantly faster, and more stable to hyper-parameters than existing federated learning approaches.

In this work, we leveraged the idea of *convexifying* a deep model to mainly improve federated training. The same idea could potentially also improve differential privacy guarantees, since convex losses have tighter bounds on privacy leakage [16]. More generally, we believe our approach can be widely used to develop principled solutions for FL problems where directly analyzing deep models is intractable, e.g. generalizing to unseen clients [72], continual learning [17], personalization [12], etc.

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. URL https://openreview.net/forum?id=B7v4QMR6Z9w.

[2] Andrei Afonin and Sai Praneeth Karimireddy. Towards model agnostic federated learning using knowledge distillation. *arXiv preprint arXiv:2110.15210*, 2021.

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30, 2017.

[4] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Sharper bounds for regularized data fitting. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques,* .

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[7] Kallista Bonawitz, Peter Kairouz, Brendan McMahan, and Daniel Ramage. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue*, 19(5):87–114, 2021.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[9] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1: 374–388, 2019.

[10] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. On large-cohort training for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pages 2089–2099. PMLR, 2021.

[12] Liam Collins, Aryan Mokhtari, Sewoong Oh, and Sanjay Shakkottai. Maml and anil provably learn representations. *arXiv preprint arXiv:2202.03483*, 2022.

[13] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[15] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[16] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[17] Hubert Eichner, Tomer Koren, Brendan McMahan, Nathan Srebro, and Kunal Talwar. Semi-cyclic stochastic gradient descent. In *International Conference on Machine Learning*, pages 1764–1773. PMLR, 2019.

[18] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[19] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[20] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems*, volume 33, pages 5850–5861, 2020.

[21] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[22] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[23] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 2350–2358. PMLR, 2021.

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

[25] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[26] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[27] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2592–2600, 2016.

[28] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[29] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[30] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

[31] Charles I Jones and Christopher Tonetti. Nonrivalry and the economics of data. *American Economic Review*, 110(9):2819–58, 2020.

[32] Peter Kairouz, H. Brendan McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[33] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

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

[35] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via bucketing. In *International Conference on Learning Representations*, 2021.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[37] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[38] Bogdan Kulynych, David Madras, Smitha Milli, Inioluwa Deborah Raji, Angela Zhou, and Richard Zemel. Participatory approaches to machine learning. International Conference on Machine Learning Workshop, 2020.

[39] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

[40] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[41] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[42] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[43] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

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

[45] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[46] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[47] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtárik. Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally! *arXiv preprint arXiv:2202.09357*, 2022.

[48] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[49] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[50] Kaan Ozkara, Navjot Singh, Deepesh Data, and Suhas Diggavi. Quped: Quantized personalization via distillation with applications to federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Alex Pentland, Alexander Lipton, and Thomas Hardjono. *Building the New Economy: Data as Capital*. MIT Press, 2021.

[52] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*, 2020.

[53] 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. URL https://openreview.net/forum?id=LkFG3lB13U5.

[54] Yuxin Shi, Han Yu, and Cyril Leung. A survey of fairness-aware federated learning. *arXiv preprint arXiv:2111.01872*, 2021.

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

[56] Jinhyun So, Başak Güler, and A Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2020.

[57] Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates. *Journal of Machine Learning Research*, 21:1–36, 2020.

[58] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[59] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pages 3329–3337. PMLR, 2017.

[60] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning over heterogeneous devices. *arXiv preprint arXiv:2105.00243*, 2021.

[61] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33: 16070–16084, 2020.

[62] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *arXiv preprint arXiv:1910.00643*, 2019.

[63] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[64] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[65] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[66] Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. *arXiv preprint arXiv:2203.06176*, 2022.

[67] Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

[68] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

[69] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 1: 35–44, 2020.

[70] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[71] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2066–2074, 2021.

[72] Honglin Yuan, Warren Richard Morningstar, Lin Ning, and Karan Singhal. What do we mean by generalization in federated learning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=VimqQq-i_Q.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our work furthers the goal of federated learning which aims to make ML more participatory. We see this overall as a positive outcome. However, it can potentially also be used to circumvent existing data portability and privacy regulations. E.g. hospitals with previously confidential patient health records could circumvent privacy regulations by using FL. They can sell their information to an insurance company, without directly revealing any raw patient data. We believe preventing such bad outcomes requires direct regulatory measures (e.g. prohibiting insurance companies from discriminating among their consumers) are necessary rather than on relying on vague notions of privacy. More generally, laws and mechanisms need to be designed to ensure that companies don't capture all the benefits arising from the access to data through FL, and instead these benefits are passed on to all sections of society.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.1 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]