Elastic Aggregation for Federated Optimization

Dengsheng Chen[1*], Jie Hu[2,3*], Vince Junkai Tan[4], Xiaoming Wei[1], Enhua Wu[2,3,5†]
[1] Meituan [2] State Key Laboratory of Computer Science, ISCAS
[3] University of Chinese Academy of Sciences [4] Bytedance Inc. [5] University of Macau
{chendengsheng,weixiaoming}@meituan.com, hujie@ios.ac.cn, vince.tan.jun.kai@gmail.com, ehwu@um.edu.mo

Abstract

Federated learning enables the privacy-preserving training of neural network models using real-world data across distributed clients. FedAvg has become the preferred optimizer for federated learning because of its simplicity and effectiveness. FedAvg uses naïve aggregation to update the server model, interpolating client models based on the number of instances used in their training. However, naïve aggregation suffers from client drift when the data is heterogenous (non-IID), leading to unstable and slow convergence. In this work, we propose a novel aggregation approach, elastic aggregation, to overcome these issues. Elastic aggregation interpolates client models adaptively according to parameter sensitivity, which is measured by computing how much the overall prediction function output changes when each parameter is changed. This measurement is performed in an unsupervised and online manner. Elastic aggregation reduces the magnitudes of updates to the more sensitive parameters so as to prevent the server model from drifting to any one client distribution, and conversely boosts updates to the less sensitive parameters to better explore different client distributions. Empirical results on real and synthetic data as well as analytical results show that elastic aggregation leads to efficient training in both convex and non-convex settings while being fully agnostic to client heterogeneity and robust to large numbers of clients, partial participation, and imbalanced data. Finally, elastic aggregation works well with other federated optimizers and achieves significant improvements across the board.

1. Introduction

Unlike traditional centralized learning in which models are trained using large datasets stored in a central server [15], federated learning - first proposed in [40] - leverages data spread across many clients to learn classification tasks distributively without explicitly sharing data [22, 26, 27, 42], thereby ensuring a basic level of privacy. Federated learning is characterized by four key features:

- *Unreliable links*: The links connecting the server and clients can be unreliable, and only a small subset of clients may be active at any given time.
- *Massive distribution*: The number of clients is typically high, but the amount of data per client is relatively small.
- *Substantial heterogeneity*: Client data is heterogeneous and non-IID [26], meaning that data across different clients can be sampled from varying regions of the sampling space.
- *Imbalanced data*: There can be significant imbalances in the amount of data available per client.

The most popular algorithm for federated learning is FedAvg [40], which tackles the communication bottleneck by performing multiple local updates on the available clients before communicating the overall change to the server. FedAvg uses naïve aggregation to interpolate client models and has shown success in certain applications. However, its performance on heterogeneous data is still an active area of research [16,25,33]. According to [25], *training models on local data that minimize local empirical loss appears to be meaningful, but yet, doing so is fundamentally inconsistent with minimizing the global empirical loss*. Client updates drive the server model away from the ideal distribution, a phenomenon known as 'client drift'. Naïve aggregation [40] is efficient in aggregating client models but does not account for distribution inconsistencies across client data or the consequent objective inconsistency. In other words, with naïve aggregation, the server model risks converging to a stationary point of a mismatched objective function which can be arbitrarily different from the true objective [53].

Prior works [24,40,48] attempt to overcome this issue by running fewer epochs or iterations of SGD on the devices or by stabilizing server-side updates so that the resulting models correspond to inexact minimizations and keep globally desirable properties. In this work, we propose a novel aggregation

[*]Equal contribution.
[†]Corresponding author. This work is supported in part by NSFC Grants (62072449).

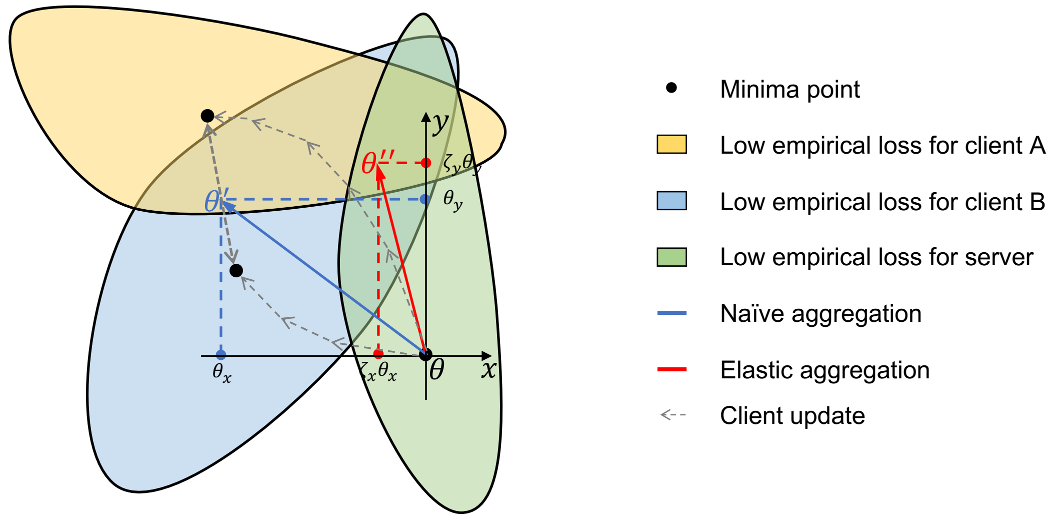
y
θ''
ζ_yθ_y
θ'
θ_y
θ_x
ζ_xθ_x
θ
x
Minima point
Low empirical loss for client A
Low empirical loss for client B
Low empirical loss for server
Naïve aggregation
Elastic aggregation
Client update

Figure 1. Illustration of naïve aggregation and elastic aggregation. The local updates of client A and client B drive the server model θ towards their individual minima (black dots in plot). Naïve aggregation simply averages the received model from clients A and B, yielding θ' as the new server model. Although θ' minimizes the local empirical loss of clients A and B, θ' drifts from ideal distribution for the server model. Elastic aggregation adjusts gradient with respect to parameter sensitivity. Parameter θ_x is more sensitive (has a larger gradient norm), and is restricted with $\zeta_x < 1$ to reduce the magnitude of its update. Parameter θ_y is less sensitive (has a smaller gradient norm), and is correspondingly boosted with $\zeta_y > 1$ to better explore the parameter space. This minimizes the loss for clients A and B, while not causing the server model to drift from its ideal distribution. Hence, elastic aggregation results in a better update θ''.

approach, elastic aggregation, to overcome client drift. We measure parameter sensitivity using unlabeled samples of client data, by computing the changes to the overall function output for a given change to the parameter in question, without relying on the loss. This allows our method to not only avoid requiring labeled data but importantly also pre-empt complications that could otherwise arise from the loss being at a local minimum with gradients close to zero. During the aggregation of client models, updates to the more sensitive parameters can then be reduced in magnitude, preventing the server model from drifting to any client distribution. Conversely, updates to the less sensitive parameters can be boosted to better explore different client distributions.

Contributions. Elastic aggregation tackles distribution inconsistency across client data using the concept of parameter sensitivity and is simple to implement, requiring little hyperparameter tuning. Furthermore, parameter sensitivity is computed in an online and unsupervised manner, and thus better utilizes the unlabeled data generated by the client during runtime. Elastic aggregation is easily integrated into different federated optimizers, achieving substantial improvements over naïve aggregation. The empirical results on real and synthetic data and analytical results show that elastic aggregation leads to efficient training in both convex and non-convex settings, across all four federated learning scenarios (unreliable links, massive distribution, substantial heterogeneity, and imbalanced data).

2. Related work

Federated learning is a fast-evolving topic. The general setup involves server and client updates; each of these updates is associated with minimizing some local loss function. The server model then benefits from all client data and achieves superior performance, for tasks such as next word prediction [17, 56], emoji prediction [45], decoder models [8], vocabulary estimation [7], low latency vehicle-to-vehicle communication [49] and predictive models in health [6]. Nevertheless, federated learning raises several issues and has been the topic of much research effort, focusing on the issues of generalization and fairness [33, 42], the design of more efficient communication strategies [4, 24, 26, 27, 51, 52], the study of lower bounds [55], differential privacy guarantees [2], security [5], etc [22]. We focus here on relevant work that specifically addresses the four federated learning characteristics noted above - massive distribution, heterogeneity, unreliable links, and imbalanced data.

Much of earlier work in this context proposes optimizing for the local risk objective with SGD [51] over mini-batches of client data, analogous to the centralized scenario, with the server then averages the received models. FedAvg [40] is a generalization of local SGD, proposing a larger number of local SGD steps per round. In the case of identical clients, it reduces to parallel SGD for which asymptotic convergence has been proven [51, 62]; more recently, [25, 43] analyzed the same method under the name of local SGD, also for identical functions. FedAvg inexactly solves client-side optimization, requiring the tuning of the number of epochs and the learning rate hyper-parameters in order to achieve a good accuracy-communication trade-off [30, 40].

Despite the strong empirical performance of FedAvg in IID settings, performance degrades in non-IID scenarios [61]. The analysis of FedAvg for heterogeneous clients is more delicate, due to the aforementioned client drift, first empirically observed by [61]. Several analyses bound this drift by assuming bounded gradients [54, 57], viewing it as additional noise [25], or assuming that the client optima are ϵ-close [16,33]. [36] proposes using variance reduction to deal with client heterogeneity but achieved slower convergence rates than SGD. Other variants include using a decreasing learning rate [30], modifying client empirical loss dynamically [1,31], adding regularizers on local updates [23,24,50], or modifying the server side updates [21, 48]. While these works do recognize the incompatibility of local and global stationary points, their proposed fixes are based on inexact minimization. Additionally, in order to establish convergence for non-IID situations, these works impose additional constraints.

Another class of work that extend the analysis of SGD-type methods to federated learning settings specifically addresses the special case of full client participation [9, 14, 26, 34–36, 38, 44, 50, 58]. For example, FedSVRG [26] and DANE [50] need gradient information from all clients each round and are as such not directly applicable to partial federated learning settings. Furthermore, it may not be trivial to extend these works to overcome the deficiency. For example, FedDANE [32] is a version of DANE that works with partial participation. However, with partial participation, FedDANE empirically performs worse than FedAvg [33]. Similar to these works, FedPD [60] proposes distributed optimization with a different notion of 'participation'. FedPD activates either all clients or no clients per round, which again fails to satisfy the partial participation condition.

Finally, there have been earlier works aiming to decrease communication costs by compressing the transmitted models [3, 12, 41], decreasing the bit-rate of the transmission. These ideas are complementary to our work and can work together with our approach.

3. Method

In this section, we focus on the introduction of parameter sensitivity in process of federated learning, which is our main innovation. *The proof of convergence is provided in the technical part of supplementary materials.*

3.1. Federated Learning with Naïve Aggregation

In federated learning, we solve an optimization problem of the form:

$$arg \min_{\theta \in \mathbb{R}^n} [\ell(\theta) \triangleq \frac{1}{m} \sum_{k=1}^{m} L_k(\theta)], \tag{1}$$

where $L_k(\theta) \triangleq \mathbb{E}_{(x,y) \backsim \mathcal{P}_k}[\ell_k(\theta; (x, y))]$ is the empirical loss of the k^{th} client, $\mathcal{P}_k$ is the data distribution for the k^{th} client across K clients.

FedAvg. A common approach to solving Eq. 1 in federated settings is FedAvg [40]. At each round, a subset of clients is selected (typically randomly) and the server broadcasts its model to each client. In parallel, the clients run SGD on their own loss function ℓ_k and then send their updated models to the server. The server then updates its model to be the average of these client models.

Suppose that at the r^{th} round, the server has model θ and samples a client set $\mathcal{S}$. Here we use a standard gradient descent form to update parameters:

$$\theta \leftarrow \theta - \eta\Delta, \tag{2}$$

where Δ is the aggregated clients' gradients and η is the learning rate of the server, which is typically 1.0. FedAvg uses naïve aggregation to compute Δ.

Naïve aggregation. We can write naïve aggregation as:

$$\Delta = \sum_{k \in \mathcal{S}} (w_k \cdot \Delta_k), \tag{3}$$

where $\Delta_k = \theta - \theta_k$ is the accumulated gradients within a training round of the k^{th} client, and $w_k = |D_k| / \sum_{k \in \mathcal{S}} |D_k|$ is the aggregated weight of the k^{th} client across the activated clients $\mathcal{S}$. D_k, θ_k are the training dataset and trained parameters on the k^{th} client, respectively.

3.2. Exploring parameter sensitivity

Due to the non-IID-ness (heterogeneity) across different clients, $\mathcal{P}$ can vary substantially across different clients. Thus, the simple and efficient way to aggregate client models described in Eq. 3 cannot solve client drift and cause oscillation as well as slow convergence. We explore parameter sensitivity as an approach to overcoming these issues.

The main idea we are proposing is that *less sensitive parameters can be freely updated to minimize the loss for individual clients in $\mathcal{S}$ without causing the server to drift; by the same reasoning, parameters that are more sensitive should not be updated as much.* This is illustrated in Fig. 1.

At the r^{th} round, the active clients receive the server model θ_r, which parameterizes the approximation function

Algorithm 1: Elastic aggregation within a single layer

```
A variable with a superscript $i$ indicates the $i^{th}$ element of the variable. A variable with a subscript $k$ indicates the variable from $k^{th}$ client. $\eta, \eta'$ are learning rates of server and clients respectively. $\mu, \tau$ are the hyper-parameters. $\theta, \theta_k \in \mathbb{R}^n$ are the server's and the $k^{th}$ client's parameters respectively. $\Omega \in \mathbb{R}^n$ is the aggregated parameter sensitivity. $\Omega_k \in \mathbb{R}^n$ is the parameter sensitivity on the $k^{th}$ client.
Initialize $\theta$
$B_k \leftarrow$ Sample a subset of training data $D_k$.
$D_k \leftarrow$ Drop the samples of $B_k$ from $D_k$.
for each round do
    for each activated client $k$ do
        Initialize $\Omega_k$ as zeros.
        for each batch data $x \in B_k$ do
            $g = \nabla||F(\theta; x)||_2^2$
            for $i \in [1, \cdots, n]$ do
                $\Omega_k^i \leftarrow \mu\Omega_k^i + (1-\mu)|g^i|$
        $\theta_k \leftarrow \theta$
        for each epoch do
            for each batch data $x \in D_k$ do
                $\theta_k \leftarrow \theta_k - \eta'\nabla\ell_k(F(\theta_k; x))$
        $\Delta_k = \theta_k - \theta$
    $w_k \leftarrow |D_k| / \sum_k |D_k|$; $\Omega = \sum_k (w_k \cdot \Omega_k)$;
    $\Omega' = \max(\Omega)$
    for $i \in [1, \cdots, n]$ do
        $\zeta^i = 1 + \tau - \Omega^i/\Omega'$
        $\Delta^i = \zeta^i \cdot \sum_k (w_k \cdot \Delta_k^i)$
        $\theta^i \leftarrow \theta^i - \eta \cdot \Delta^i$
```

F of the true function $\bar{F}$. F maps the input x to the output y, and our goal is to largely maintain the same output y for the observed x while learning additional training instances $(x, y) \backsim P_k$. To this end, we measure how sensitive the output of function F is to changes in the network parameters. To reduce confusion, we ignore the symbols denoting the round r and the client k for all variables and only discuss parameter sensitivity locally.

For a given data point x, the output of the network is $F(\theta; x)$. A small perturbation δ in the parameters θ results in a change in the function output that can be approximated by:

$$F(\theta + \delta; x) - F(\theta; x) \approx g(\theta; x)\delta, \tag{4}$$

where $g(\theta; x) = \frac{\partial F(\theta; x)}{\partial \theta}$ is the gradient of the learned function with respect to the parameter θ evaluated at the data point x. Our goal is to maintain the output of the network (the learned function F) for each observed data point and thus we reduce the magnitudes of the updates to the parameters that are more sensitive for the data point. Conversely, we boost the updates to the less sensitive parameters to better-fit client distributions.

Based on equation 4, and assuming a small constant change δ, we can measure the sensitivity of a parameter by the magnitude of the gradient $g(\theta; x)$, i.e., how much does a small perturbation to that parameter change the output of the learned function for data point x. There are many ways to measure the sensitivity of a parameter by the magnitude of the gradient, such as a fishing matrix. However, in federated learning, we have to take the computation ability of each client into account, which is only with limited computation ability. Therefore, we use the exponentially-decayed moving average[1] to obtain sensitivity Ω^i for i^{th} parameter θ^i:

$$\Omega^i \leftarrow \mu\Omega^i + (1-\mu)|g(\theta^i; x)|, \tag{5}$$

where μ is momentum and we empirically set it to 0.9.

When the output function F is multi-dimensional, as is the case for most neural networks, equation 5 involves computing the gradients for each output, which requires as many backward passes as the dimensionality of the output. As a more efficient alternative, we propose to use the gradients of the squared L^2 norm of the learned function output, i.e. $g(\theta; x) = \nabla||F(\theta; x)||_2^2$. Hence, we only need to compute one backward pass in order to estimate the parameters' sensitivity.

3.3. Elastic aggregation with respect to parameter sensitivity

The elastic aggregation for i^{th} parameter can be written as:

$$\Delta^i = \zeta^i \cdot \sum_{k \in \mathcal{S}} (w_k \cdot \Delta_k^i), \tag{6}$$

where ζ^i is the adaptive coefficient with respect to the parameter sensitivity. The definition of ζ^i is:

$$\zeta^i = 1 + \tau - \Omega^i/\Omega', \tag{7}$$

where $\Omega' = \max(\Omega)$ is the maximum of parameter sensitivities in each layer, and $\Omega^i = \sum_{k \in \mathcal{S}} (w_k \cdot \Omega_k^i)$ is the aggregated sensitivity of i^{th} parameter across activated clients. The server is able to boost updates ($\zeta^i > 1.0$) to parameters that have low sensitivity (low Ω^i). Meanwhile, sensitive parameters (high Ω^i) have their updates restricted with a penalty ($\zeta^i < 1.0$). τ is a hyper-parameter determining the ratio of boosted or restricted parameters. FedAvg with elastic aggregation is formalized in Algorithm 1.

[1]While a Polyak-style average has more theoretical guarantees for accumulating Ω over batches, our results match the claim that "an exponentially-decayed moving average typically works much better in practice" [39]. Thus, we use the exponentially-decayed moving average to accumulate Ω over batches. Refer to supplementary materials for more discussion.

Discussion on τ. In Eq. 6, $\zeta^i \in (1, 1+\tau]$ is applied to the less sensitive parameters, and $\zeta^i \in [\tau, 1)$ is applied to the more sensitive parameters. Thus, τ determines the portion of the parameter to be restricted or boosted. Generally, a larger τ indicates that a larger proportion of parameters will be boosted, as shown in Tab. 1, and vice versa.

If a large proportion of parameters is restricted (τ is small, such as 0.1), the server model will not be updated much, and exiting local minima becomes difficult. However, with a large τ such as 0.9, optimization of the server model becomes unstable, and this also causes performance degradation. It is therefore critical to maintain a balance between these extremes with a suitable τ and we find that $\tau = 0.5$ generalizes well to various experiments. At this value of τ, roughly three-quarters of the parameters are boosted.

4. Experiments settings

Inconsistent usage of models, datasets, and non-IID partition methods makes it difficult to compare the performances of federated learning algorithms fairly. To enforce fair comparison, we explicitly specify the combinations of datasets, models, and non-IID partition methods to be used for experiments, which mostly consist of those used in existing work published at top-tier machine learning venues [18]. Note that although the reported precisions are state-of-the-art for the models we have chosen, higher-capacity models can achieve higher performances on these datasets. Therefore, we compare the relative performance of elastic aggregation against naïve aggregation for various federated learning scenarios.

4.1. Implementation

We measure our algorithm against the most comprehensive and representative suite of federated datasets and modeling tasks to date. For this purpose, we carry out simulations on five diverse and representative experiments on several datasets. In order to facilitate the experiments, we use the cosine annealing scheduler [37] for the client learning rate η'_r at the r^{th} round, which is given by $\eta'_r = (1+\cos(\frac{r}{R}\pi))\eta'$. For all tasks, we measure the performance on the entire test set throughout training. Clients are sampled uniformly at random, without replacement within a given round, but with replacement across rounds. There are a variety of federated optimizers that improve the performance of federated learning. Here, we use the most basic and widely used federated optimization algorithm, i.e., FedAvg [40], as the default baseline in our experiments. With reference to [46], we implement all experiments with two important characteristics. First, instead of doing K training steps per client, we do E epochs of training over each client's data. Second, to account for varying numbers of gradient steps per client, we weigh the average of the client outputs Δ_t^k by the number of training samples. This also follows the approach of [40] and often outperforms uniform weighting [47]. We set $\mu = 0.95$ for the accumulation of the parameter sensitivity over batches in each client. $\tau = 0.5, \eta = 1.0$ is used for all experiments. While computing Eq. 7, we normalize the parameter sensitivity layer by layer, which means that the parameter sensitivity is divided by the maximum value within the same layer. Aside from experiments 5.1, we set $E = 1$ for each client for all experiments.

4.2. Generate synthetic federated datasets.

Differences in distribution have an enormous influence on the performance of federated optimization. The Dirichlet distribution is used on the label ratios to ensure uneven label distributions among clients for non-IID splits, as in [59]. To generate imbalanced data, we sample the number of data points from a log-normal distribution. The degree of imbalance depends on the variance of the log-normal distribution, which is controlled by the hyper-parameter α. A smaller α indicates a stronger non-IID-ness in partitions, as shown in Fig. 2.

5. Experiments

Under all experiments, the samples used for computing parameter sensitivities at each client will be randomly sampled training data, and these sampled data will not be used for supervised training anymore. For a fair comparison, these sampled data will not be used in the baseline methods either.

5.1. Comparison with naive aggregation

MNIST. As our baseline, we conduct a convex optimization experiment on MNIST [29] with logistic regression [31, 33, 40]. In this experiment, we generate a synthetic federated dataset of MNIST with $\alpha = 100.0$ and split the dataset into 1000 partitions. At each round of training, we randomly choose 100 clients without repeating the partition. As shown in Tab. 2, the elastic aggregation has a large advantage over naïve aggregation even with a large number of participating clients.

Federated EMNIST. Due to resource constraints of edge devices, existing works commonly use shallow neural networks for experiments. Similarly, we include Federated EMNIST [46] with a shallow neural network consisting of two convolution layers followed by two linear layers [46]. Notably, Fed-EMNIST [46] has a natural client partitioning that is highly representative of real-world federated learning problems. It contains 3,400 users (clients) and 62 label classes. The dataset is split into 671,585 training examples and 77,483 testing examples. Rather than holding out specific users, we split each user's examples across the train

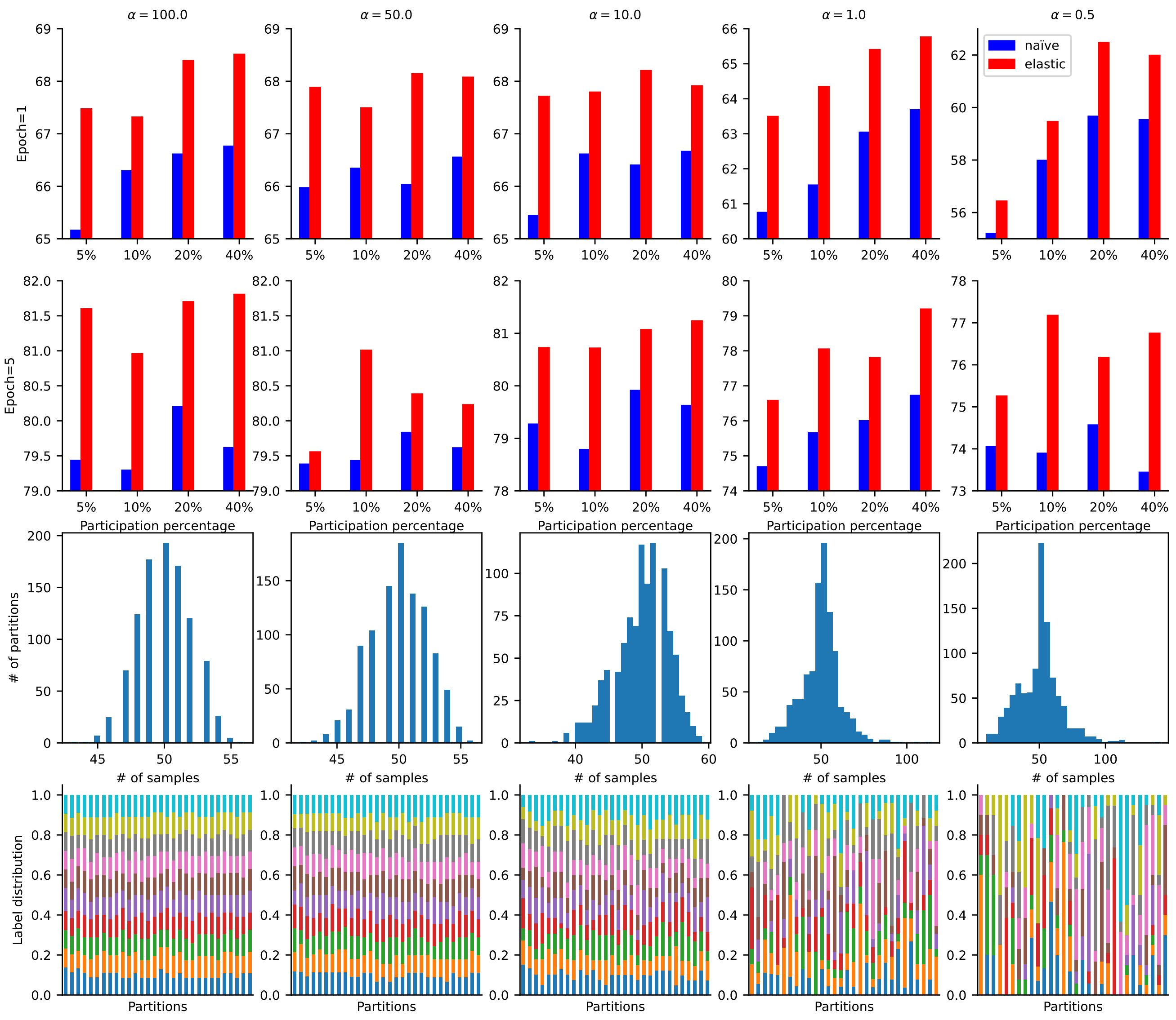

Figure 2. Performances across different partitioning distributions, participation rates and numbers of local epochs. Compared with naïve aggregation, elastic aggregation achieves significant improvements across these settings.

τ	Naïve	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Train Acc.(%)	58.36	57.21	58.33	58.76	60.66	60.47	**61.44**	60.66	60.55	61.01	59.27	58.96
Test Acc.(%)	60.74	58.30	59.76	61.17	61.91	**63.39**	62.74	**63.39**	63.23	63.22	60.83	61.37
Boosted(%)	-	0.0	11.45	27.24	43.75	59.74	74.43	84.57	89.93	93.53	96.26	97.38

Table 1. The effect of τ. Using an unsuitable τ for elastic aggregation with cause it to even under-perform naïve aggregation. The last row shows the percentage of parameters that are boosted for different τ.

and test so that all users have at least one example in the train and one example in the test. Users with fewer than 2 examples are excluded from the data set. As shown in Tab. 2, elastic aggregation generalizes well to real-world federated learning problems.

CIFAR-100. In this experiment, we conduct a balanced but non-IID partition on Fed-CIFAR100 [2], which was introduced by TensorFlow Federated [20] with ResNet-20 [19]

[2]https://github.com/tensorflow/federated

Dataset	Rounds(Epochs)	Total	Sampled	Batch	Init. LR	Model	Naïve(%)	Elastic(%)
Balanced data across clients								
CIFAR-100	4000(∼80)	500	10	10	0.05	ResNet-20	32.31	**56.64**
Unbalanced data across clients								
MNIST	20(∼2)	1000	100	100	0.1	Logistic Regression	70.14	**73.64**
EMNIST	1000(∼2.5)	3400	10	100	0.1	2Conv+2Linear	88.71	**89.82**
CIFAR-10	4000(∼40)	1000	10	10	0.05	ResNet-20	66.84	**68.74**
CINIC-10	200(∼20)	1000	100	10	0.05	ResNet-20	35.81	**36.29**
CINIC-10	4000(∼40)	1000	10	10	0.05	ResNet-20	68.68	**69.25**

Table 2. Test accuracies for various datasets. Along with rounds, we list a more intuitive measure, i.e., epochs, to better reflect the total training computation.

Optimizer	FedAvg	FedAvgM	FedProx	SCAFFOLD	AdaOpt	PFNM	Per-FedAvg	pFedMe
w/o Elastic.	49.34	62.52	51.52	72.10	78.00	71.23	50.44	83.20
w/ Elastic.	52.37	64.48	53.64	75.42	80.34	73.64	53.24	85.63
Improvement.	**3.03**	**1.96**	**2.12**	**3.32**	**2.34**	**2.41**	**2.80**	**2.43**

Table 3. Elastic aggregation can be easily integrated with different federated optimizers, achieving performance improvements. (Notes that the comparison across different federated optimizers is meaningless since they are under different experiment settings.)

as the baseline model. As shown in Tab. 2, elastic aggregation significantly speeds up training. Furthermore, the final accuracy is improved by over 20% compared with the naïve baseline, which demonstrates the effectiveness of elastic aggregation.

CIFAR-10 and CINIC-10. To further demonstrate the performance improvement resulting from elastic aggregation, we generate two synthetic federated datasets for CIFAR-10 [20] and CINIC-10 [10] with $\alpha = 100.0$. CINIC-10 extends CIFAR-10 with the addition of down-sampled ImageNet images, making for a more challenging dataset. This also enables measuring the performance of models trained on CIFAR images on ImageNet images for the same classes. As shown in Tab. 2, elastic aggregation achieves superior performances on both the CIFAR-10 and the more difficult CINIC-10 datasets, confirming that our method generalizes well to large scale datasets.

Natural Language Processing on Shakespeare dataset. We maintain the same experimental setup as [40] on the most commonly used natural language processing tasks: Shakespeare LSTM. The epoch in each round is 1, the batch size is 10 and both elastic and naive aggregation are trained for 2k rounds. We get the final accuracy as 62.14% for naive aggregation and 64.73% for elastic aggregation. The results show that elastic aggregation not only has a significant improvement in CV related tasks, but also has an positive effect on NLP related tasks.

Imbalanced and non-IID. We are also interested in understanding how parameter sensitivity can help improve convergence, particularly in cross-device and non-IID scenarios. Given the resource constraints of edge devices, large DNN models are usually trained under the cross-organization (also known as cross-silo) federated learning settings. Across the following experiments, we use CIFAR-10 [28] with ResNet-20 [19]. We study three different parameters here: non-IID-ness, participation rate, and number of epochs. We explore the effects of varying non-IID-ness with $\alpha = 100.0, 50.0, 10.0, 1.0, 0.5$ across 1000 partitions. For participation rate, we test sampling $5\%, 10\%, 20\%, 40\%$ of the clients at each round. Finally, we test having $1, 5$ epochs of local updates, to better show the robustness of elastic aggregation. As shown in Fig. 2, elastic aggregation is especially superior for situations with massive distribution, substantial heterogeneity, and imbalanced data.

5.2. Integration with modern optimizer

Federated optimizer with elastic aggregation. To illustrate the generalizability of elastic aggregation, we integrate elastic aggregation into FedAvg [40], FedAvgM [46] and FedProx [31]. FedAvgM is an enhancement of FedAvg on the server-side, making use of the momentum strategy to update the server model. FedProx is an enhancement of FedAvg on the client-side, adding a penalty to the client updates. We have presented the details of FedAvg with elastic aggregation in Algorithm 1. Please refer to supplementary materials for more details about the other federated optimizers with elastic aggregation. As shown in the left part of Tab. 3, elastic aggregation works well with other com-

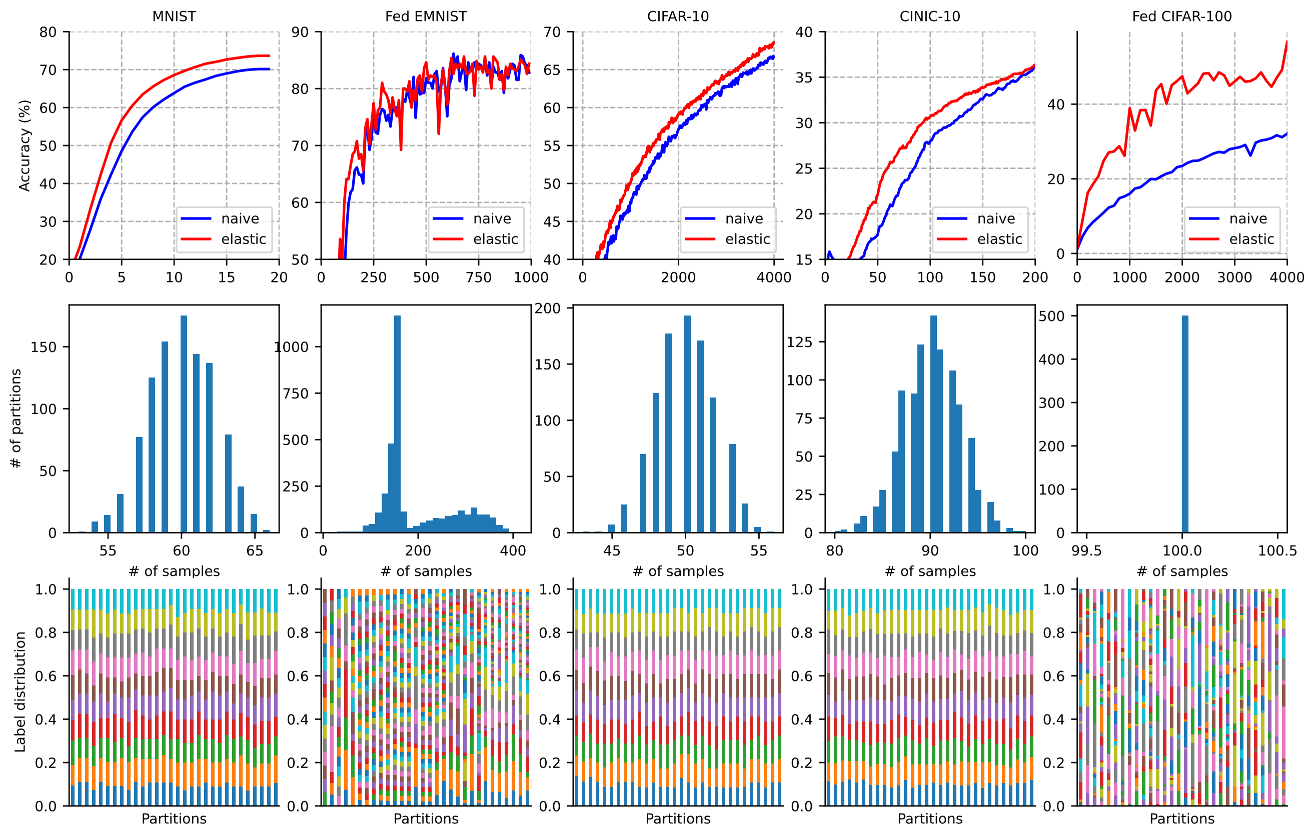

Figure 3. Convergence speeds on different datasets.

plementary approaches whether designed for client-side or server-side. In order to further verify the generality of elastic aggregation, we did further experiments on the stat-of-the-art methods in FL, i.e. SCAFFOLD [23], AdaOpt [46] and PFNM [59]. Since these methods target on different aspect of federated learning, we conducted experiments based on their original experimental settings. As shown in middle part of Tab. 3, elastic aggregation plays a positive effect in most federated optimizers.

Influence on personalization. Personalization in federated learning is also a topic of great concern. Specifically, we equipped Per-FedAvg [13], pFedMe [11] – both focus on the model personalization (local model performance) in federated learning with elastic aggregation and found that elastic aggregation is also do a favour in learning a better personalized model, as shown in the right part of Tab. 3.

5.3. Limitations

In Fig. 3, we compare testing curves on various datasets and observe that elastic aggregation outperforms naïve aggregation by converging faster. Nevertheless, using elastic aggregation results in additional computational burden and communication costs, which may affect the overall performance of federated learning scenarios. For a more detailed analysis of the computational overhead, please refer to the supplementary materials. Despite these challenges, we firmly believe that the significant performance improvements associated with elastic aggregation warrant its use over naïve aggregation.

6. Conclusion

Our work delves into the impact of heterogeneity on optimization methods for federated learning. We proposed a novel aggregation method, elastic aggregation, which utilizes parameter sensitivity to overcome gradient dissimilarity. To the best of our knowledge, we are the pioneers in utilizing unlabeled client data to enhance federated learning performances. Empirical evaluations across various federated datasets validate the theoretical analysis and reveal that elastic aggregation can significantly enhance the convergence behavior of federated learning in realistic heterogeneous scenarios. Future directions may include expanding the elastic aggregation approach to adaptive optimization methods or gossip-based training techniques.

References

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

[2] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H Brendan Mcmahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *arXiv preprint arXiv:1805.10559*, 2018.

[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*, 30:1709–1720, 2017.

[4] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification, and local computations. *arXiv preprint arXiv:1906.02367*, 2019.

[5] 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.

[6] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[7] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[8] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. *arXiv preprint arXiv:1910.03432*, 2019.

[9] Laurent Condat, Grigory Malinovsky, and Peter Richtárik. Distributed proximal splitting algorithms with rates and acceleration. *arXiv preprint arXiv:2010.00952*, 2020.

[10] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[11] C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with moreau envelopes. 2020.

[12] Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 04, pages 3817–3824, 2020.

[13] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. 2020.

[14] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020.

[15] 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.

[16] Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

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

[18] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[25] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

[26] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[27] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

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

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

[31] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[32] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smithy. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231. IEEE, 2019.

[33] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[34] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. *arXiv preprint arXiv:2002.11364*, 2020.

[35] Zhize Li and Peter Richtárik. A unified analysis of stochastic gradient methods for nonconvex federated optimization. *arXiv preprint arXiv:2006.07013*, 2020.

[36] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

[37] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[38] Ali Makhdoumi and Asuman Ozdaglar. Convergence rate of distributed admm over networks. *IEEE Transactions on Automatic Control*, 62(10):5082–5095, 2017.

[39] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

[40] 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.

[41] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

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

[43] Kumar Kshitij Patel and Aymeric Dieuleveut. Communication trade-offs for synchronized distributed sgd with large step size. *arXiv preprint arXiv:1904.11325*, 2019.

[44] Reese Pathak and Martin J Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*, 2020.

[45] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*, 2019.

[46] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[47] S Reddi, Manzil Zaheer, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Proceeding of 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

[48] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[49] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Merouane Debbah. Federated learning for ultra-reliable low-latency v2v communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[50] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

[51] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

[52] 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.

[53] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

[54] 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.

[55] Blake Woodworth, Jialei Wang, Adam Smith, Brendan McMahan, and Nathan Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. *arXiv preprint arXiv:1805.10222*, 2018.

[56] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[57] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 01, pages 5693–5700, 2019.

[58] Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. *arXiv preprint arXiv:2006.08950*, 2020.

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

[60] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *arXiv preprint arXiv:2005.11418*, 2020.

[61] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[62] Martin Zinkevich, Markus Weimer, Alexander J Smola, and Lihong Li. Parallelized stochastic gradient descent. In *NIPS*, number 1, page 4. Citeseer, 2010.