Accelerated Federated Learning with Decoupled Adaptive Optimization

Jiayin Jin[*1] Jiaxiang Ren[*1] Yang Zhou[1] Lingjuan Lyu[2] Ji Liu[3] Dejing Dou[3 4]

Abstract

The federated learning (FL) framework enables edge clients to collaboratively learn a shared inference model while keeping privacy of training data on clients. Recently, many heuristics efforts have been made to generalize centralized adaptive optimization methods, such as SGDM, Adam, AdaGrad, etc., to federated settings for improving convergence and accuracy. However, there is still a paucity of theoretical principles on where to and how to design and utilize adaptive optimization methods in federated settings. This work aims to develop novel adaptive optimization methods for FL from the perspective of dynamics of ordinary differential equations (ODEs). First, an analytic framework is established to build a connection between federated optimization methods and decompositions of ODEs of corresponding centralized optimizers. Second, based on this analytic framework, a momentum decoupling adaptive optimization method, FEDDA, is developed to fully utilize the global momentum on each local iteration and accelerate the training convergence. Last but not least, full batch gradients are utilized to mimic centralized optimization in the end of the training process to ensure the convergence and overcome the possible inconsistency caused by adaptive optimization methods.

1. Introduction

Recent advances in federated learning (FL) present a promising machine learning (ML) paradigm that enables collaborative training of shared ML models over multiple distributed devices without the need for data sharing, while mitigating the data isolation as well as protecting the data privacy (Konečný et al., 2016b;a; McMahan et al., 2016; 2017a; Kairouz et al., 2021). In a FL model, end clients (e.g., Android phones) use their local data to train a local ML model, while keeping the local data decentralized. The end clients send the parameter updates of local models (e.g., Android phone updates) rather than raw data to the central server (e.g., Android cloud). The server produces a shared global ML model by aggregating the local updates.

[*]Equal contribution [1]Auburn University, USA [2]Sony AI, Japan [3]Baidu Research, China [4]University of Oregon, USA. Correspondence to: Yang Zhou <yangzhou@auburn.edu>, Lingjuan Lyu <lingjuan.lv@sony.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

One of critical challenges in the FL paradigm is expensive communication cost between the clients and server (Konečný et al., 2016b; Caldas et al., 2018a; Liu et al., 2019; Hamer et al., 2020a; He et al., 2020). Traditional federated optimization methods, such as FedAvg and its variants (Konečný et al., 2016b;a; McMahan et al., 2017a; Kairouz et al., 2021), use local client updates, where the clients perform multiple epochs of stochastic gradient descent (SGD) on their local datasets to update their models before communicating to the server. This can dramatically reduce the communication required to train a FL model.

Despite achieving remarkable performance, FL techniques often suffer from two key challenging issues: (1) Client drift. Too many SGD epochs on the same clients may result in overfitting to their local datasets, such that the local models are far away from globally optimal models and the FL training achieves slower convergence (Karimireddy et al., 2020d; Woodworth et al., 2020; Reddi et al., 2021a; Karimireddy et al., 2021); and (2) lack of adaptivity. Standard SGD optimization methods used in most FL models may be unsuitable for federated settings and result in high communication costs (Zhang et al., 2019; Reddi et al., 2021a).

In centralized ML models, adaptive optimization methods, such as SGD with Momentum (SGDM) (Rumelhart et al., 1986; Qian, 1999; Sutskever et al., 2013), Adaptive Moment Estimation (Adam) (Kingma & Ba, 2015), Adaptive Gradient (AdaGrad) (McMahan & Streeter, 2010; Duchi et al., 2011b), etc., have achieved superior success in speeding up the training convergence. Adaptive optimization methods are designed to control possible deviations of mini-batch gradients in centralized models. Several recent studies try to adapt centralized optimization algorithms to the federated learning settings for achieving faster convergence and higher test accuracy (Xie et al., 2019; Reddi et al., 2021a; Karimireddy et al., 2021; Wang et al., 2021b).

The above adaptive optimization methods can be broadly classified into two categories: (1) Server adaptive methods. FedOpt is a generalization of FedAvg that allows the clients and the server to choose different optimization methods (Reddi et al., 2021a). Although FedOPT is a generic framework that can use adaptive optimizers as client or server optimizers, the core section (Section 3) of the FedOPT paper only considers the setting that ClientOPT is SGD, which fails to make full use of the strength of the adaptive methods for improving the convergence (Wang et al., 2021b); (2) Client adaptive techniques. Local AdaAlter proposes a novel SGD variant based on AdaGrad, and adopt the concept of local SGD to reduce the communication (Xie et al., 2019). Mime uses a combination of control-variates and server-level optimizer state (e.g. momentum) at every client-update step to ensure that each local update mimics that of the centralized method run on i.i.d. data (Karimireddy et al., 2021). However, Local AdaAlter and Mime use the same optimizer states (momentums and pre-conditioners) on all clients, which is similar to server adaptive methods in FedOpt and does not exploit local adaptivity (Wang et al., 2021b). FedLocal is the first real client adaptive approach that employs adaptive optimization methods for local updates at clients (Wang et al., 2021b). It restarts the update of client optimizer states (i.e., momentums and pre-conditioners) at the beginning of each round. This restart operation fails to inherit and aggregate the states from previous rounds and may loss the strength of adaptive optimization methods in centralized models, which aggregate the states from previous rounds to speed up the convergence. For the standard FL models, applying adaptive optimization methods to local updates may come with the cost of a non-vanishing solution bias: instead of using current gradient to update local parameters in the standard SGD, the adaptive methods utilize the aggregation of client optimizer states (i.e., momentum aggregates previous and past gradients) in multiple client-update steps to update local parameters. In this case, the heterogeneity property in federated settings results in large deviations among uploaded local gradients as well as makes the converge point far away from the global minimizer and slow down the convergence, compared with the standard SGD. FedLocal proposes correction techniques to overcome this bias issue and to complement the local adaptive methods for FL. However, the correction techniques that are essentially heuristic post-processing methods without knowledge of previous client optimizer states on the server are hard to solve this problem fundamentally for stochastic optimizers, such as SGDM, Adam, AdaGrad, etc. Therefore, the above techniques often follow manually-crafted heuristics to generalize centralized adaptive optimization methods to the federated settings. There is still a paucity of theoretical principles on where to and how to design and utilize adaptive optimization methods in federated settings.

This work aims to develop novel adaptive optimization methods for FL from the perspective of the decomposition of ODEs of centralized optimizers. We establish an analytic framework to connect the federated optimization methods with the decompositions of ODEs of corresponding centralized optimizers. The centralized gradient descent (CGD) is utilized as a warm-up example to briefly illustrate our underlying idea. The CGD reads $W(t+1) = W(t) - \sum_{i=1}^{M} \frac{N^i}{N} L^i(W(t)) * \eta$, where $W(t)$ is the global parameter, M is the number of clients, N^i is the number of training samples on i^{th} device, $N = \sum_{i}^{M} N^i$ is the total number of training samples, L^i is the loss function on i^{th} client. It is straightforward to check that $\sum_{i=1}^{M} \frac{N^i}{N} L^i(W(t))$ is the total loss of centralized training. Therefore, the CGD is the numerical solution of an ODE system $\frac{d}{d\tau} W(\tau) = -\sum_{i=1}^{M} \frac{N^i}{N} L^i(W(\tau))$. One way to decompose the ODE system is as follows: $\bar{W}(\tau) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(\tau)$, where $W^i(\tau)$ solves $\frac{d}{d\tau} W^i(\tau) = -L^i(W^i(\tau))$, $i = 1, \cdots M$. Thus, $\bar{W}(\tau)$ is an approximate solution to the above ODE system. The numerical solution of the decomposed system is $W^i(t+1) = W^i(t) - \nabla L^i(W^i(t)) * \eta$, and then $\bar{W}(t+1) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(t)$ is an approximate numerical solution to the ODE system of CGD. This scheme can be extended to the SGD case by replacing full batch gradients with mini-batch gradients, which is exactly the update rule of FedAvg (McMahan et al., 2017a). This example illustrates how to derive federated optimization methods based on the decomposition of ODEs of centralized optimizers. Moreover, this example also demonstrates the rationality of FedAvg, since FedAvg is an approximation of the above ODE system of CGD and the convergence of the CGD guarantees the convergence of FedAvg if the approximation error is well controlled. In the same spirit, to design adaptive optimization methods for FL, it is crucial to discover suitable decompositions of ODEs of centralized adaptive optimizers.

Based on the above analytic framework, we develop a momentum decoupling adaptive optimization method for FL. By decoupling the global momentum from local updates, the equation of global momentum becomes a linear equation. Consequently, we can decompose the equation of global momentum exactly and distribute the update of global momentum to local devices. The aggregation of the portions of the global momentum on local devices is exactly equal to the global momentum without any deviation. Particularly, in our FedDA method: (1) the global momentums are updated with local gradients in each local iteration; (2) all global momentums updated through local iterations will attend global training to update global model. This is an analogy to centralized training, which fully utilizes the global momentum. Notice that momentums can provide fast convergence and high accuracy for centralized training. The global momentum in our FedDA method makes the best effort to mimic the role of momentum in centralized training, which can accelerate the convergence of FL training. We theoretically demonstrate that (1) local momentum deviates from the centralized one at exponential rate; (2) global momentum in FedDA deviates from the centralized one at algebraic rate.

Adaptive optimization methods for FL are often faced with convergence inconsistency in training (Wang et al., 2021b). We propose to utilize full batch gradients of clients to mimic centralized optimization in the end of the training process to ensure the convergence. Based on our proposed framework of decomposing ODEs, if local devices only iterate once with full batch gradients in a training round, then our FedDA method is in agreement with centralized training in that round. Take the advantage of this, by reducing the iteration number to 1 in the end of training, our FedDA method is able to mimic centralized training, which ensures the convergence and overcome the inconsistency.

Empirical evaluation on real federated tasks and datasets demonstrates the superior performance of our momentum decoupling adaptive optimization model against several state-of-the-art regular federated learning and federated optimization approaches. In addition, more experiments, implementation details, and hyperparameter selection and setting are presented in Appendices A.4-A.5.

2. Preliminaries and Notations

Federated learning aims to solve the following optimization problem.

$$\begin{aligned} \min_{W\in\mathbb{R}^d} \mathcal{L}(W) &= \sum_{i=1}^{M} \frac{N_i}{N} L^i(W) \\ \text{where } L^i(W) &= \frac{1}{N_i} \sum_{k\in\mathcal{P}_i} l_k(W) \end{aligned} \tag{1}$$

where $l_k(W) = l(x_k, y_k; W)$ denotes the loss of the prediction on example (x_k, y_k) made with model parameters W. There are M clients over which the data is partitioned, with $\mathcal{P}_i$ the set of indexes of data points on client S^i, with $N_i = |\mathcal{P}_i|$. W^i, L^i, G^i, g^i, and N^i denote the local model parameter, loss function, full batch gradient, mini-batch gradient, number of training data on client S^i respectively. N is the total number of training data, i.e., $N = N^1 + \cdots + N^M$ and η represents the learning rate. In each round, there are K clients participating in the training ($K \leq M$).

This work aims to derive the theoretical principle on where to and how to design and utilize adaptive optimization methods in federated settings. Based on the theoretical principle, it tries to propose an effective and efficient framework to generalize adaptive optimization methods in centralized settings to FL with fast convergence and high accuracy.

3. Decomposition of ODEs and FL

In this section, we establish the connection between the decomposition of ODEs of centralized optimizers and FL. As a warm up, we start with the most simple centralized optimizer Gradient Descent. In particular, we demonstrate that relation between the decomposition of the ODE for GD and FedAvg, and explain why FedAvg works from ODE theory.

Centralized training and FL share the same goal, that is to minimize the total loss $\mathcal{L}$.

$$\mathcal{L}(W) = \sum_{i=1}^{M} \frac{N^i}{N} L^i(W). \tag{2}$$

The most classical centralized optimization method is Gradient Descent (GD):

$$W(t+1) = W(t) - \nabla\mathcal{L}(W(t)) * \eta. \tag{3}$$

A more popular optimization is Stochastic Gradient Descent (SGD). It updates the model with the gradient of the loss of a mini-batch at each step, which accelerates the training process comparing to GD. Denote the loss of mini-batch at step t by $\mathcal{L}^t$, then the training process of SGD can be expressed by

$$W(t+1) = W(t) - \nabla\mathcal{L}^t(W(t)) * \eta. \tag{4}$$

From the point of view of ODEs, the training process of GD in Eq.(3) is the numerical solution of the autonomous ODE

$$\frac{d}{d\tau} W(\tau) = -\nabla\mathcal{L}\big(W(\tau)\big) \leq 0. \tag{5}$$

The training process of SGD in Eq.(4) is as the numerical solution of the non-autonomous ODE

$$\frac{d}{d\tau} W(\tau) = -\nabla\mathcal{L}^{\tau}\big(W(\tau)\big). \tag{6}$$

In fact, Eq. (5) is a gradient system, i.e., the vector field of the equation $-\nabla\mathcal{L}\big(W(\tau)\big)$ is in the gradient form. As a gradient system, it is typical that the loss $\mathcal{L}$ descends along the solutions until it reaches local minimum. This is because

$$\frac{d}{d\tau}\mathcal{L}\big(W(\tau)\big) = -|\nabla\mathcal{L}\big(W(\tau)\big)|^2. \tag{7}$$

The decentralization of the training process onto local devices in FL is an analogy to the decomposition of Eq.(5) into a system of ODEs. Theoretically, a precise decomposition of Eq.(5) is $W(\tau) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(\tau)$, where $W^i(\tau)$ satisfies

$$\frac{d}{d\tau} W^i(\tau) = -\nabla L^i\big(W(\tau)\big),\ i = 1, \cdots, M. \tag{8}$$

Numerical solutions to the above decomposed ODE system is

$$W^i(t+1) = W^i(t) - \nabla L^i\big(W(t)\big) * \eta,\ i = 1, \cdots, M. \tag{9a}$$

$$W(t+1) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(t). \tag{9b}$$

Eq.(9a) is the local update rule and Eq.(9b) is the global aggregation rule. These update rules guarantees the training process is equivalent to centralized GD. When local devices are updated with mini-batch gradients, i.e.,

$$W^i(t+1) = W^i(t) - g^i\big(W(t)\big) * \eta, \tag{10}$$

then this framework agrees with centralized SGD. Moreover, if only part of the devices participate the training (either full batch or mini-batch) each round and aggregate after one iteration as above, then it is also indistinguishable to centralized SGD. These optimization methods are essentially identical to centralized optimization though they are in FL framework, which theoretically are able to produce a global model as good as centralized one. However, in Eq.(9a) ∇L^i must be evaluate at $W(t)$, i.e, global parameters $W(t)$ are necessary for the local update $W^i(t+1)$ at any step t. Therefore, local models have to aggregate after each iteration, which will cause expensive communication costs and poor efficiency. A redemption is to trade some accuracy for efficiency. To reduce communication cost, more local iterations have to be operated each round. Accordingly, Eq. (9a) is modified to following approximate decomposition of Eq.(5) so that more local iterations are admited.

$$\begin{aligned} &\frac{d}{d\tau} W^i(\tau) = -\nabla L^i\big(W^i(\tau)\big), \\ &W^i(0) = W(0), i = 1, \cdots, M. \end{aligned} \tag{11}$$

Let T denote local iteration number. The corresponding numerical solution of Eq.(11) is

$$W^i(t+1) = W^i(t) - \nabla L^i(W^i(t)) * \eta, t = 0, 1, \cdots, T-1, \tag{12}$$

which is exactly GD for S^i. To further accelerate local training, one may perform SGD for local devices, i.e.,

$$W^i(t+1) = W^i(t) - g^i(W^i(t)) * \eta, t = 0, 1, \cdots, T-1. \tag{13}$$

The cost of this method is that the aggregation of W^i in Eq.(11) is not the solution to Eq.(5). Let $\bar{W}(\tau) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(\tau)$ be the aggregation of the solution to Eq.(11). It is straightforward to check that

$$\frac{d}{d\tau} \bar{W}(\tau) = -\sum_{i=1}^{M} \frac{N^i}{N} \nabla L^i(W^i(\tau)). \tag{14}$$

Comparing the above equation with Eq.(5), one has

$$\begin{aligned} \frac{d}{d\tau}\big(W(\tau) - \bar{W}(\tau)\big) = -\sum_{i=1}^{M} \frac{N^i}{N} \big(&\nabla L^i(W(\tau)) \\ &- \nabla L^i(W^i(\tau))\big). \end{aligned} \tag{15}$$

It is clear $W(0) = \bar{W}(0)$. Integrating Eq.(15) from 0 to $t\eta$, it arrives

$$\begin{aligned} &W(t\eta) - \bar{W}(t\eta) \\ &= -\int_0^{t\eta} \sum_{i=1}^{M} \frac{N^i}{N} \big(\nabla L^i(W(\tau)) - \nabla L^i(W^i(\tau))\big) d\tau. \end{aligned} \tag{16}$$

By a very rough estimate, we have that for $t = 1, \cdots, T$

$$|W(t\eta) - \bar{W}(t\eta)| \le 2 \sup_i \|\nabla L^i\|_{L^\infty} t\eta, \tag{17}$$

which yields that $\bar{W}(\tau)$ is a good approximation of $W(\tau)$ for $\tau \le T\eta$ when $T\eta$ is small. Therefore, FL framework has to aggregate local models periodically and reset all local and global parameters to be identical. By doing so, the aggregations of local models are always stay close to the centralized one. Since the centralized GD and SGD converge to local minimum of the total loss function, so does the one for FL. This method that performs SGD or GD on local devices and aggregates periodically is exactly the FedAvg. The above arguments also yields the underlying mechanism of FedAvg, which agrees with the essential idea of the proof of convergence of FedAvg.

Notice that the purpose of Eqs.(16) and (17) is to illustrate the idea of how to connect decomposition of ODEs to FL optimizations. We only used rough estimates instead of sharp ones to illustrate the underlying ideas more straightforwardly. Our FedDA method has never used the estimate in Eqs.(16) and (17) for attending any computations.

The superb performance of adaptive optimization methods, such as SGDM, Adam, AdaGrad, have been demonstrated for centralized training. Naturally, successful extensions of these adaptive methods to FL are mostly desired.

4. Momentum Decoupling Adaptive Optimization: FedDA+SGDM

Recall the centralized SGDM.

$$\begin{aligned} &m(t+1) = \beta * m(t) + (1-\beta) * g(W(t)), \\ &W(t+1) = W(t) - \eta * m(t+1), \end{aligned} \tag{18}$$

where m is momentum, W is the parameter vector, g is the mini-batch gradient and η is the learning rate . The corresponding ODE is a slow-fast system

$$\begin{aligned} &\eta \frac{d}{d\tau} m(\tau) = -(1-\beta) m(\tau) + (1-\beta) g(W(\tau)), \\ &\frac{d}{d\tau} W(\tau) = -m(\tau). \end{aligned} \tag{19}$$

m is a fast variable cause its derivative $\frac{d}{d\tau}m(\tau)$ is $O(1/\eta)$ and η is small. Therefore the rate of change of m is much faster than that of W. It is easy to check Eq.(18) is the numerical solution of Eq.(19) using Euler's scheme.

Similar to GD case, the numerical solution of a precise decomposition of SGDM is

$$\begin{aligned} &m^i(t+1) = \beta * m^i(t) + (1-\beta) * g^i(W(t)),\\ &W^i(t+1) = W^i(t) - \eta * m^i(t+1),\\ &m(t+1) = \sum_{i=1}^{M} \frac{N^i}{N} m^i(t),\\ &W(t+1) = \sum_{i=1}^{M} \frac{N^i}{N} W^i(t). \end{aligned} \tag{20}$$

Again, the drawback is the local iteration number must be 1. To allow more local iterations, the most naive approach is to perform SGDM on local devices directly

$$\begin{aligned} &m^i(t+1) = \beta * m^i(t) + (1-\beta) * g^i(W^i(t)),\\ &W^i(t+1) = W^i(t) - \eta * m^i(t+1). \end{aligned} \tag{21}$$

Then aggregate the above local updates periodically. This method is indeed an approximation of centralized SGDM for the same reason as in FedAvg case, and therefore this naive method may converge. However, the performance of this naive method is not satisfying in experiments. The reasons are twofold. First, the momentum is used to buff the deviation of gradients. Decomposing the global momentum m onto local devices as in Eq.(21) will incapacitate it because $\sum_{i=1}^{M} \frac{N^i}{N} m^i(t)$ is deviated from the momentum of centralized optimizer. Second, since the aggregated momentum is not accurate enough, it may also harm the local training if it is inherited by the local devices. This is why the restart local adaptive method works better (Wang et al., 2021b). Therefore, though the Eq.(21) could be an approximation of centralized SGDM, it may need a rather small learning rate to ensure the quality of approximation, which results in unsatisfactory training performance. Local training generates local gradients deviating from the global ones, therefore a global momentum is favorable to control the deviations of the gradients uploaded by local devices. FedOpt updates global momentum one time after the server receiving local gradients generated by several local iterations each round. Though it achieves great performance, it does not fully take the advantage of global momentum. In FedOpt the global momentum buffs the total gradients of multiple iterations. However, for centralized SGDM, the momentum buffs the gradient in each iteration. To let the global momentum fully plays its role and accelerate the training of the global model, we propose a decoupling adaptive technique (FedDA) to approximately mimic centralized SGDM. In our FedDA method, though the global momentum does not participate local training directly, it buffers local gradients generated in each local iteration. We propose the following decoupled decomposition of Eq.(19).

$$\begin{aligned} &\frac{d}{d\tau} W^i(\tau) = -g^i(W^i(\tau)),\\ &\eta \frac{d}{d\tau} m(\tau) = -(1-\beta) m(\tau) + (1-\beta) \sum_{i=1}^{M} \frac{N^i}{N} g^i(W^i(\tau))\\ &\frac{d}{d\tau} W(\tau) = -\alpha * m(\tau), \end{aligned} \tag{22}$$

where W^i is the parameters of local devices, m is the global momentum, W is the parameters of global model and α is a parameter to concede that local and global employ different learning rates. Note that in Eq.(22), the local update of W^i is independent of the global momentum m, this is why we name this method decoupling method. The crucial point of our decoupling method is that since the the equations of $W^i(\tau)$ are independent of the global momentum m, the equation of m is totally linear and can be precisely decomposed as $m(\tau) = \sum_{i=1}^{M} \frac{N^i}{N} m^i(\tau)$, where $m^i\tau$ solves

$$\eta \frac{d}{d\tau} m^i(\tau) = -(1-\beta) m^i(\tau) + (1-\beta)\nabla g^i(W^i(\tau)).$$

Therefore, the system in Eq.(22) is exactly equivalent to

$$\begin{aligned} &\frac{d}{d\tau} W^i(\tau) = -g^i(W^i(\tau)),\\ &\eta \frac{d}{d\tau} m^i(\tau) = -(1-\beta) m^i(\tau) + (1-\beta) g^i(W^i(\tau)),\\ &\frac{d}{d\tau} W(\tau) = -\alpha * m(\tau). \end{aligned} \tag{23}$$

That is we can calculate the global momentum m by precisely decomposing it to local devices and the aggregation of the portion of the global momentum on local devices m^i will be exactly the global momentum. The numerical solution to Eq.(23) is

$$W^i(t+1) = W^i(t) - g^i(W^i(t)) * \eta, \tag{24}$$

$$m^i(t+1) = \beta * m^i(t) + (1-\beta) * g^i(W^i(t))\eta, \tag{25}$$

$$m(t+1) = \sum_{i=1}^{M} \frac{N^i}{N} m^i(t), \tag{26}$$

$$W(t+1) = W(t) - m(t+1) * \alpha * \eta. \tag{27}$$

This numerical solution demonstrates our underlying ideas better. Eq.(24) yields that local update does not depend on the global momentum m. However, the global momentum buffs local gradients in each local iteration as shown in

Table 1: Statistics of the Datasets

Dataset	#Training Clients	#Test Clients	#Training Samples	#Test Samples
CIFAR-100	500	100	50,000	10,000
EMNIST-62	3,400	3,400	671,585	77,483
Stack Overflow	342,477	204,088	135,818,730	16,586,035

Eq.(26). Based on the numerical solution Eqs.(24)-(27), the algorithm of our FedDA+SGDM method is as follows.

At round E, pick participating clients $S^1, \cdots, S^k$. For each S^i, $i = 1, \cdots, K$, initialize $P^i = 0$, $W^i(0) = W(E)$, $m^i(0) = m(E)$. For $t = 0, T-1$,

$$\begin{aligned} &W^i(t+1) = W^i(t) - g^i(W^i(t)) * \eta, \\ &m^i(t+1) = \beta m^i(t) + (1-\beta) g^i(W^i(t)), \\ &P^i = P^i + m^i(t+1). \end{aligned} \tag{28}$$

The global update rule is:

$$\begin{aligned} &P = \text{aggregation of } \ P^i, \\ &m(E+1) = \text{aggregation of } m^i(T), \\ &W(E+1) = W(E) - P * \alpha * \eta. \end{aligned} \tag{29}$$

Notice that P is the summation of global momentums updated during local iterations, and global parameter W is updated by P. This mimics the centralized SGDM. We theoretically demonstrate that (1) local momentum deviates from the centralized one at exponential rate $O(e^{\lambda t})$; and (2) global momentum in FedDA deviates from the centralized one at algebraic rate $O(t^2)$. Please refer to Appendix A.3 for detailed proof.

Full batch gradient stabilization. Adaptive optimization methods for FL are often faced with inconsistency of convergence in training (Wang et al., 2021b). When the training is almost finished, i.e, W is close to the local minimum point of loss function, the gradient has to be more accurate to ensure the convergence to local minimum point. Not sufficiently accurate gradients will lead to unstable convergence and inconsistency. Based on our framework, if local devices only iterate once with full batch gradients in a training round, then our FedDA method is in agreement with centralized training in that round. Therefore, by reducing the iteration number to 1 in the end of training, our FedDA method mimics centralized training, which ensures the convergence and overcomes the inconsistency of training. Last but not least, since this full batch method is utilized only in the end of the training when the parameter W is fairly close to local minimum point, the training converges and stabilized rather quickly. Therefore, it is a bargain to employ this full batch method in the end of the training to guarantee the convergence of the training to local minimum point.

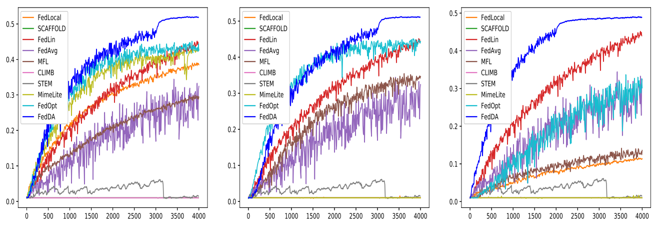
Our proposed momentum decoupling adaptive optimization method has potential be extended to other federated learning algorithms, e.g., FedProx, etc. Due to space limit, we only use FedDA+SGDM as an example to show how to apply the FedDA optimization to FedProx. At round E, pick participating clients $S^1, \cdots, S^k$. For each S^i, $i = 1, \cdots, K$, initialize $P^i = 0$, $W^i(0) = W(E)$, $m^i(0) = m(E)$. For $t = 0, \cdots, T-1$, $W^i(t+1) = W^i(t) - \big(g^i(W^i(t)) + \mu * (W^i(t) - W(E)\big) * \eta, m^i(t+1) = \beta * m^i(t) + (1-\beta) * \big(g^i(W^i(t)) + \mu * (W^i(t) - W(E)\big), P^i = P^i + m^i(t+1)$. The global update rule is the same as the one in Eq.(29). By following the similar strategy, other two versions of FedDA (FedDA+ADAM and FedDA+AdaGrad) can be added to FedProx and other regular FL algorithms, which demonstrates the applicability and generality of FedDA.

5. Experiments

In this section, we have evaluated the performance of our FedDA model and other comparison methods in serval representative federated datasets and learning tasks to date. We show that FedDA with decoupling and full batch gradient techniques is able to achieve faster convergence and higher test accuracy in cross-device settings against several state-of-the-art federated optimization methods. The experiments exactly follow the same settings described by a recent federated optimization method, FedOpt (Reddi et al., 2021a).

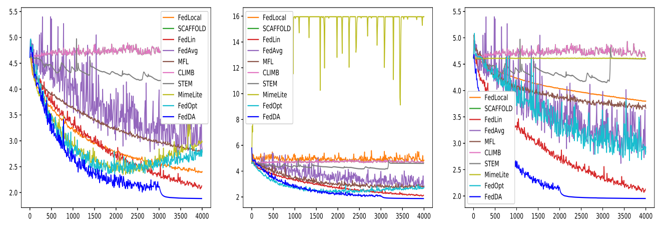
Datasets. We focus on three popular computer vision and natural language processing tasks over three representative benchmark datasets respectively: (1) image classification over CIFAR-100 (Krizhevsky, 2009). We train ResNet-18 by replacing batch norm with group norm (Hsieh et al., 2020); (2) image classification over EMNIST (Hsieh et al., 2020). We train a CNN for character recognition; and (3) text classification over Stack Overflow (TensorFlow, 2019). We perform tag prediction using logistic regression on bag-of-words vectors. We select the 10,000 most frequently used words, the 500 most frequent tags and a one-versus-rest classification strategy, by following the same strategy in FedOpt (Reddi et al., 2021a). The detailed descriptions of the federated datasets and learning tasks are presented in Appendix A.5.

Baselines. We compare the FedDA model with nine state-of-the-art federated learning models, including five regular federated learning and four federated optimization ap-



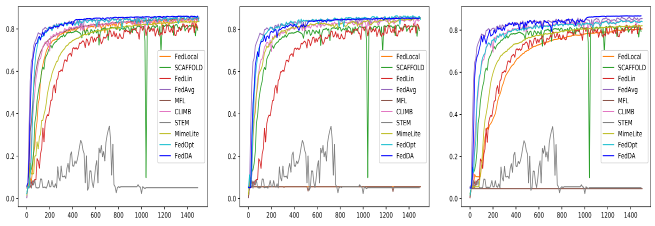
(a) SGDM (b) Adam (c) AdaGrad

Figure 1: Convergence on CIFAR-100 with Three Optimizers



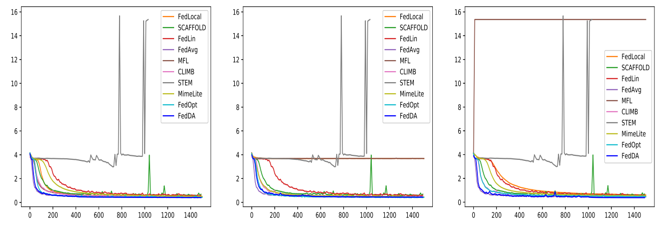
(a) SGDM (b) Adam (c) AdaGrad

Figure 2: Loss on CIFAR-100 with Three Optimizers



(a) SGDM (b) Adam (c) AdaGrad

Figure 3: Convergence on EMNIST with Three Optimizers



(a) SGDM (b) Adam (c) AdaGrad

Figure 4: Loss on EMNIST with Three Optimizers

proaches. **FedAvg** is a classical as well as practical method for the federated learning of deep networks based on iterative model averaging (McMahan et al., 2017a). **SCAFFOLD** is a algorithm which uses control variates to correct for the client-drift in its local updates (Karimireddy et al., 2020d). **FedLin** is an algorithmic framework to tackle the key challenges of objective heterogeneity, systems heterogeneity, and imprecise communication in FL (Mitra et al., 2021a). **STEM** is a stochastic two-sided momentum algorithm, that utilizes certain momentum-assisted stochastic gradient directions for both the client and server updates (Mitra et al., 2021a). **CLIMB** is an agnostic constrained learning formulation to tackle the class imbalance problem in FL without requiring further information beyond the standard FL objective (**?**). **MFL** performs momentum gradient descent in local update step of FL system to solve the distributed machine learning problem (Liu et al., 2020a). **FedOpt** is a flexible algorithmic framework that allows the clients and the server to choose different optimization methods more general than stochastic gradient descent (SGD) in FedAvg (Reddi et al., 2021a). **MimeLite** uses a combination of control-variates and server-level optimizer state at every client-update step to ensure that each local update mimics that of the centralized method run on i.i.d. data (Karimireddy et al., 2021). **Local Adaptivity (FedLocal)** proposes techniques that enable the use of adaptive optimization methods for local updates at clients in federated learning (Wang et al., 2021b). To our best knowledge, this work is the first to certify the group fairness of classifiers with theoretical input-agnostic guarantees, while there is no need to know the shift between training and deployment datasets with respect to sensitive attributes. All nine baselines used in our experiments either do not use momentum, or use momentum without momentum aggregation, or use momentum with aggregation but restart momentum at each FL round (FedLocal). Our FedDA method keeps the momentum aggregation in the entire FL process, which results in faster convergence but larger oscillation.

Evaluation metrics. We use two popular measures in federated learning and plot the measure curves with increasing training rounds to verify the convergence of different methods: **Accuracy** and **Loss Function Values (Loss)** (Karimireddy et al., 2020d; Mitra et al., 2021a; Liu et al., 2020a; Reddi et al., 2021a; Karimireddy et al., 2021; Wang et al., 2021b). A larger Accuracy or a smaller Loss score indicates a better federated learning result. We run 1,500 rounds of training on the EMNIST and Stack Overflow datasets, and 4,000 rounds over the CIFAR-100 dataset. In addition, we use final Accuracy to evaluate the quality of the federated learning algorithms.

Table 2: Final Accuracy on CIFAR-100

Optimizer	SGDM	Adam	AdaGrad
FedLocal	0.384	0.009	0.113
SCAFFOLD	0.010	0.010	0.010
FedLin	0.440	0.440	0.440
FedAvg	0.324	0.324	0.324
MFL	0.293	0.346	0.135
CLIMB	0.010	0.010	0.010
STEM	0.014	0.014	0.014
MimeLite	0.427	0.009	0.009
FedOpt	0.425	0.443	0.301
FedDA	**0.518**	**0.510**	**0.488**

Table 3: Final Accuracy on EMNIST

Optimizer	SGDM	Adam	AdaGrad
FedLocal	0.834	0.055	0.806
SCAFFOLD	0.794	0.794	0.794
FedLin	0.805	0.805	0.805
FedAvg	0.850	0.850	0.850
MFL	0.848	0.055	0.047
CLIMB	0.843	0.843	0.843
STEM	0.051	0.051	0.051
MimeLite	0.835	0.851	0.821
FedOpt	0.838	0.847	0.840
FedDA	**0.860**	**0.853**	**0.868**

Table 4: Final Mean Squared Error on EMNIST for Autoencoder

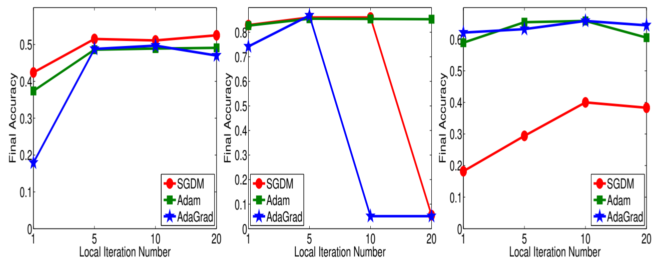
Optimizer	SGDM	Adam	AdaGrad
FedLocal	0.0169	0.0289	0.0168
FedAvg	0.0171	0.0171	0.0171
MFL	0.0168	0.0290	0.0291
MimeLite	0.0183	0.0307	0.0287
FedOpt	0.0175	0.0173	0.0145
FedDA	**0.0167**	**0.0166**	**0.0132**

Convergence on CIFAR-100 and EMNIST. Figures 1 and 3 exhibit the $Accuracy$ curves of ten federated learning models for image classification over CIFAR-100 and character recognition on EMNIST respectively. It is obvious that the performance curves by federated learning algorithms initially keep increasing with training rounds and remains relatively stable when the curves are beyond convergence points, i.e., turning points from a sharp $Accuracy$ increase to a flat curve. This phenomenon indicates that most federated learning algorithms are able to converge to the invariant solutions after enough training rounds. However, among five regular federated learning and five federated optimization approaches, our FedDA federated optimization method can significantly speedup the convergence on two datasets in most experiments, showing the superior performance of FedDA in federated settings. Compared to the learning results by other federated learning models, based on training rounds at convergence points, FedDA, on average, achieves 34.3% and 22.6% convergence improvement on two datasets respectively.

Loss on CIFAR-100 and EMNIST. Figures 2 and 4 present the $Loss$ curves achieved by ten federated learning models on two datasets respectively. We have observed obvious that the reverse trends, in comparison with the $Accuracy$ curves. In most experiments, our FedDA federated optimization method is able to achieve the fastest convergence, especially, FedDA can converge within less than 200 training rounds and then always keep stable on the EMNIST dataset. A reasonable explanation is that FedDA fully utilizes the global momentum on each local iteration as well as employ full batch gradients to mimic centralized optimization in the end of the training process for accelerating the training convergence.

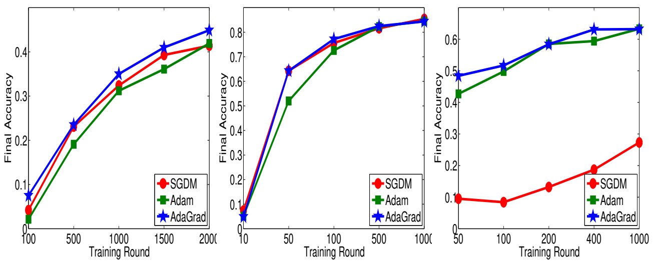
Final $Accuracy$ on CIFAR-100 and EMNIST. Tables 2-4 show the quality of ten federated learning algorithms over CIFAR-100 and EMNIST respectively. Notice that the performance achieved by five regular federated learning algorithms, including FedAvg, SCAFFOLD, FedLin, STEM, and CLIMB, keep unchanged when using different optimizers, such as SGDM, Adam, and AdaGrad, while five federated optimization approaches, including MFL, FedOpt, Mime, FedLocal, and our FedDA model obtain different performance. We have observed that our FedDA federated optimization solution outperforms the competitor methods in most experiments. FedDA achieves the highest $Accuracy$ values ($>$ 0.488 over CIFAR-100 and $>$ 0.853 on EMNIST respectively), which are better than other nine baseline methods in all tests. A reasonable explanation is that the combination of decoupling and full batch gradient techniques is able to achieve faster convergence as well as higher test accuracy in cross-device settings. In addition, the promising performance of FedDA over both datasets implies that FedDA has great potential as a general federated optimization solution to learning tasks over federated datasets, which is desirable in practice.

Impact of local iteration numbers. Figure 5 shows the impact of the numbers of local iterations in our FedDA model with the adaptive optimizers over the datasets of CIFAR-100, EMNIST, and Stack Overflow respectively. The performance curves initially raise when the local iteration number



(a) CIFAR-100 (b) EMNIST (c) Stack Overflow

Figure 5: Final Accuracy with Varying #Local Iterations



(a) CIFAR-100 (b) EMNIST (c) Stack Overflow

Figure 6: Final Accuracy with Varying Training Rounds

increases and then keep relatively stable or even drop if the local iteration number keeps increasing. This demonstrates that there must exist a suitable local iteration number for the FL training. A too large number may make the clients overfit to their local datasets, such that the local models are far away from globally optimal models and the FL training achieves slower convergence. On the other hand, a too small number may result in slow convergence of local training and also increase the difficulty of convergence of global training. Thus, it is important to choose the appropriate numbers for well balancing the local training and global training. Notice that the final accuracy of AdaGrad and SGDM is closed to 0 on the EMNIST dataset when the local iteration number is larger than 10. A reasonable explanation is EMNIST is a simple dataset and a large local iteration number makes local models converge to their local minimum, which may be distant from the minimum of global model. This leads to lower accuracy of global model.

Impact of training round numbers. Figures 6 (a)-(c) present the performance achieved by our FedDA method with varying the numbers of training rounds from 100 to 2,000, from 10 to 1,000, and from 50 to 1,000 on three datasets. It is obvious that the performance curves with each optimizer keep increasing with the increased number of training rounds. This phenomenon indicates that the accuracy in the federated settings are sensitive to training rounds. This is because the special data and system heterogeneity issues in the FL increase the difficulty in converging in a short time and the FL models need more training rounds to obtain the desired learning results. However, as shown in the above experiments of convergence in Figures 1-4, our FedDA method presents superior convergence performance, compared with other FL algorithms, including both regular federated learning and federated optimization approaches.

6. Conclusions

This work presents a theoretical principle on where to and how to design and utilize adaptive optimization methods in federated settings. First, we establish a connection between federated optimization methods and decompositions of ODEs of corresponding centralized optimizers. Second, we develop a momentum decoupling adaptive optimization method to well balance fast convergence and high accuracy in FL. Finally, we utilize full batch gradient to mimic centralized optimization in the end of the training process.

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *Int. Conf. on Learning Representations (ICLR)*, pp. 1–36, 2021.

Afonin, A. and Karimireddy, S. P. Towards model agnostic federated learning using knowledge distillation. *arXiv preprint arXiv:2110.15210*, 2021.

Anonymous. Acceleration of federated learning with alleviated forgetting in local training. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–19, 2022a. under review.

Anonymous. An agnostic approach to federated learning with class imbalance. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–12, 2022b. under review.

Anonymous. AQUILA: Communication efficient federated learning with adaptive quantization of lazily-aggregated gradients. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–23, 2022c. under review.

Anonymous. Hybrid local SGD for federated learning with heterogeneous communications. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–41, 2022d. under review.

Anonymous. Improving federated learning face recognition via privacy-agnostic clusters. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–21, 2022e. under review.

Anonymous. Privacy-preserving task-agnostic vision transformer for image processing. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–28, 2022f. under review.

Anonymous. Recycling model updates in federated learning: Are gradient subspaces low-rank? In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–71, 2022g. under review.

Anonymous. Unsupervised federated learning is possible. In *Submitted to Int. Conf. on Learning Representations (ICLR)*, pp. 1–22, 2022h. under review.

Balunović, M., Dimitrov, D. I., Staab, R., and Vechev, M. Bayesian framework for gradient leakage. *arXiv preprint arXiv:2111.04706*, 2021.

Bao, X., Liu, L., Xiao, N., Zhou, Y., and Zhang, Q. Policy-driven autonomic configuration management for nosql. In *Proceedings of the 2015 IEEE International Conference on Cloud Computing (CLOUD'15)*, pp. 245–252, New York, NY, June 27-July 2 2015.

Caldas, S., Konečný, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *CoRR*, abs/1812.07210, 2018a.

Caldas, S., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018b.

Chen, H.-Y. and Chao, W.-L. On bridging generic and personalized federated learning. *arXiv preprint arXiv:2107.00778*, 2021.

Chen, M., Poor, H. V., Saad, W., and Cui, S. Convergence time minimization of federated learning over wireless networks. In *IEEE Int. Conf. on Communications (ICC)*, pp. 1–6, 2020.

Dai, Z., Low, B. K. H., and Jaillet, P. Federated bayesian optimization via thompson sampling. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 1–13, 2020.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011a.

Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011b.

Fowl, L., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

Gao, H., Xu, A., and Huang, H. On the convergence of communication-efficient local sgd for federated learning. In *AAAI Conf. on Artificial Intelligence*, volume 35, pp. 7510–7518, 2021.

Goswami, S., Pokhrel, A., Lee, K., Liu, L., Zhang, Q., and Zhou, Y. Graphmap: Scalable iterative graph processing using nosql. *The Journal of Supercomputing (TJSC)*, 76 (9):6619–6647, 2020.

Guimu Guo, Da Yan, L. Y. J. K. C. L. Z. J. and Zhou, Y. Maximal directed quasi-clique mining. In *Proceedings of the 38th IEEE International Conference on Data Engineering (ICDE'22)*, Kuala Lumpur, Malaysia, May 9-12 2022.

Hamer, J., Mohri, M., and Suresh, A. T. Fedboost: A communication-efficient algorithm for federated learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pp. 3973–3983, 2020a.

Hamer, J., Mohri, M., and Suresh, A. T. FedBoost: A communication-efficient algorithm for federated learning. In *Int. Conf. on Machine Learning (ICML)*, volume 119, pp. 3973–3983, 2020b.

He, C., Annavaram, M., and Avestimehr, S. Group knowledge transfer: Federated learning of large cnns at the edge. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Hong, J., Wang, H., Wang, Z., and Zhou, J. Federated robustness propagation: Sharing adversarial robustness in federated learning. *arXiv preprint arXiv:2106.10196*, 2021.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. The non-iid data quagmire of decentralized machine learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4387–4398. PMLR, 2020.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

Ingerman, A. and Ostrowski, K. Introducing tensorflow federated. `https://medium.com/tensorflow/introducing-tensorflow-federated`, 2019.

Jiang, Y., Perng, C.-S., Sailer, A., Silva-Lepe, I., Zhou, Y., and Li, T. Csm: A cloud service marketplace for complex service acquisition. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–25, 2016.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

Karimireddy, S. P., He, L., and Jaggi, M. Byzantine-robust learning on heterogeneous datasets via bucketing. *arXiv preprint arXiv:2006.09365*, 2020a.

Karimireddy, S. P., Jaggi, M., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020b.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *Int. Conf. on Machine Learning (ICML)*, pp. 5132–=–5143, 2020c.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 2020d.

Karimireddy, S. P., Jaggi, M., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Mime: Mimicking centralized stochastic algorithms in federated learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Khanduri, P., Sharma, P., Yang, H., Hong, M., Liu, J., Rajawat, K., and Varshney, P. K. Stem: A stochastic two-sided momentum algorithm achieving near-optimal sample and communication complexities for federated learning. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 1–12, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016a.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016b.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Lee, K., Liu, L., Tang, Y., Zhang, Q., and Zhou, Y. Efficient and customizable data partitioning framework for distributed big rdf data processing in the cloud. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing (CLOUD'13)*, pp. 327–334, Santa Clara, CA, June 27-July 2 2013.

Lee, K., Liu, L., Schwan, K., Pu, C., Zhang, Q., Zhou, Y., Yigitoglu, E., and Yuan, P. Scaling iterative graph computations with graphmap. In *Proceedings of the 27th IEEE international conference for High Performance Computing, Networking, Storage and Analysis (SC'15)*, pp. 57:1–57:12, Austin, TX, November 15-20 2015.

Lee, K., Liu, L., Ganti, R. L., Srivatsa, M., Zhang, Q., Zhou, Y., and Wang, Q. Lightwieight indexing and querying services for big spatial data. *IEEE Transactions on Services Computing (TSC)*, 12(3):343–355, 2019.

Leroy, D., Coucke, A., Lavril, T., Gisselbrecht, T., and Dureau, J. Federated learning for keyword spotting. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6341–6345, 2019.

Li, W. and McCallum, A. Pachinko allocation: Dag-structured mixture models of topic correlations. In Cohen, W. W. and Moore, A. W. (eds.), *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pp. 577–584. ACM, 2006.

Li, Z., Kovalev, D., Qian, X., and Richtarik, P. Acceleration for compressed gradient descent in distributed and federated optimization. In *Int. Conf. on Machine Learning (ICML)*, volume 119, pp. 5895–5904, 2020.

Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., and Dou, D. From distributed machine learning to federated learning: A survey. *arXiv preprint arXiv:2104.14362*, 2021.

Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., and Dou, D. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems (KAIS)*, 64(4):885–917, 2022.

Liu, W., Chen, L., Chen, Y., and Zhang, W. Accelerating federated learning via momentum gradient descent. *IEEE Trans. Parallel Distributed Syst.*, 31(8):1754–1766, 2020a.

Liu, W., Chen, L., Chen, Y., and Zhang, W. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 31(8):1754–1766, 2020b.

Liu, Y., Kang, Y., Zhang, X., Li, L., Cheng, Y., Chen, T., Hong, M., and Yang, Q. A communication efficient vertical federated learning framework. *CoRR*, abs/1912.11187, 2019.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Singh, A. and Zhu, X. J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017a.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017b.

McMahan, H. B. and Streeter, M. J. Adaptive bound optimization for online convex optimization. In Kalai, A. T. and Mohri, M. (eds.), *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pp. 244–256. Omnipress, 2010.

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

Mills, J., Hu, J., and Min, G. Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 7(7):5986–5994, 2019.

Mills, J., Hu, J., Min, G., Jin, R., Zheng, S., and Wang, J. Accelerating federated learning with a global biased optimiser. *arXiv preprint arXiv:2108.09134*, 2021.

Mitra, A., Jaafar, R., Pappas, G., and Hassani, H. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In *The 35th Conference on Neural Information Processing Systems, (NeurIPS'21)*, Online, December 6-14 2021a.

Mitra, A., Jaafar, R., Pappas, G. J., and Hassani, H. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 1–14, 2021b.

Oh, J., Kim, S., and Yun, S.-Y. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

Ozfatura, E., Ozfatura, K., and Gündüz, D. Fedadc: Accelerated federated learning with drift control. In *IEEE Int. Symposium on Information Theory (ISIT)*, pp. 467–472, 2021.

Palanisamy, B., Liu, L., Lee, K., Meng, S., Tang, Y., and Zhou, Y. Anonymizing continuous queries with delay-tolerant mix-zones over road networks. *Distributed and Parallel Databases (DAPD)*, 32(1):91–118, 2014.

Palanisamy, B., Liu, L., Zhou, Y., and Wang, Q. Privacy-preserving publishing of multilevel utility-controlled graph datasets. *ACM Transactions on Internet Technology (TOIT)*, 18(2):24:1–24:21, 2018.

Pathak, R. and Wainwright, M. J. Fedsplit: an algorithmic framework for fast federated optimization. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7057–7066, 2020.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

Reddi, S., Zaheer, M., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 1–17, 2018.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021a.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *Int. Conf. on Learning Representations (ICLR)*, 2021b.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd:

Communication-efficient federated learning with sketching. In *Int. Conf. on Machine Learning (ICML)*, pp. 8253–8265, 2020.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing*. MIT Press, 1986.

Sharma, A., Chen, W., Zhao, J., Qiu, Q., Chaterji, S., and Bagchi, S. Tesseract: Gradient flip score to secure federated learning against model poisoning attacks. *arXiv preprint arXiv:2110.10108*, 2021.

Su, Z., Liu, L., Li, M., Fan, X., and Zhou, Y. Servicetrust: Trust management in service provision networks. In *Proceedings of the 10th IEEE International Conference on Services Computing (SCC'13)*, pp. 272–279, Santa Clara, CA, June 27-July 2 2013.

Su, Z., Liu, L., Li, M., Fan, X., and Zhou, Y. Reliable and resilient trust management in distributed service provision networks. *ACM Transactions on the Web (TWEB)*, 9(3): 1–37, 2015.

Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1139–1147, 2013.

TensorFlow. Tensorflow federated stack overflow dataset. https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data, 2019.

Triastcyn, A., Reisser, M., and Louizos, C. Dp-rec: Private & communication-efficient federated learning. *arXiv preprint arXiv:2111.05454*, 2021.

Wang, H.-P., Stich, S. U., He, Y., and Fritz, M. Progfed: Effective, communication, and computation efficient federated learning by progressive training. *arXiv preprint arXiv:2110.05323*, 2021a.

Wang, J., Tantia, V., Ballas, N., and Rabbat, M. Slowmo: Improving communication-efficient distributed sgd with slow momentum. In *Int. Conf. on Learning Representations (ICLR)*, pp. 1–27, 2020.

Wang, J., Xu, Z., Garrett, Z., Charles, Z., Liu, L., and Joshi, G. Local adaptivity in federated learning: Convergence and consistency. In *The International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021, (FL-ICML'21)*, Online, December 6-14 2021b.

Wang, J., Xu, Z., Garrett, Z., Charles, Z., Liu, L., and Joshi, G. Local adaptivity in federated learning: Convergence and consistency. *arXiv preprint arXiv:2106.02305*, 2021c.

Woodworth, B. E., Patel, K. K., and Srebro, N. Minibatch vs local SGD for heterogeneous distributed learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Wu, H. and Wang, P. Fast-convergent federated learning with adaptive weighting. *IEEE Transactions on Cognitive Communications and Networking*, 2021.

Wu, W., He, L., Lin, W., and Mao, R. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 32(7):1539–1551, 2020.

Wu, Y. and He, K. Group normalization. *Int. J. Comput. Vis.*, 128(3):742–755, 2020.

Xia, S., Zhu, J., Yang, Y., Zhou, Y., Shi, Y., and Chen, W. Fast convergence algorithm for analog federated learning. In *IEEE Int. Conf. on Communications (ICC)*, pp. 1–6, 2021.

Xie, C., Koyejo, O., Gupta, I., and Lin, H. Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates. *CoRR*, abs/1911.09030, 2019.

Yan, D., Qu, W., Guo, G., Wang, X., and Zhou, Y. Prefixfpm: A parallel framework for general-purpose mining of frequent and closed patterns. *The VLDB Journal (VLDBJ)*, 31(2):253–286, 2022a.

Yan, D., Zhou, Y., Guo, G., and Liu, H. Parallel graph processing. In Sherif Sakr, A. Y. Z. and Taheri, J. (eds.), *Encyclopedia of Big Data Technologies*. Springer, 2022b.

Yang, H. H-fl: A hierarchical communication-efficient and privacy-protected architecture for federated learning. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 479–485, 2021.

Yapp, A. Z. H., Koh, H. S. N., Lai, Y. T., Kang, J., Li, X., Ng, J. S., Jiang, H., Lim, W. Y. B., Xiong, Z., and Niyato1, D. Communication-efficient and scalable decentralized federated edge learning. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 5032–5035, 2021.

Yuan, H., Morningstar, W., Ning, L., and Singhal, K. What do we mean by generalization in federated learning? *arXiv preprint arXiv:2110.14216*, 2021a.

Yuan, H., Zaheer, M., and Reddi, S. Federated composite optimization. In *Int. Conf. on Machine Learning (ICML)*, volume 139, pp. 12253–12266, 2021b.

Yun, C., Rajput, S., and Sra, S. Minibatch vs local sgd with shuffling: Tight convergence bounds and beyond. *arXiv preprint arXiv:2110.10342*, 2021.

Zhang, H., Liu, J., Jia, J., Zhou, Y., Dai, H., and Dou, D. Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*, Messe Wien, Vienna, Austria, July 23-29 2022.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., and Sra, S. Why ADAM beats SGD for attention models. *CoRR*, abs/1912.03194, 2019.

Zhang, M., Sapra, K., Fidler, S., Yeung, S., and Alvarez, J. M. Personalized federated learning with first order model optimization. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

Zhang, Q., Liu, L., Ren, Y., Lee, K., Tang, Y., Zhao, X., and Zhou, Y. Residency aware inter-vm communication in virtualized cloud: Performance measurement and analysis. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing (CLOUD'13)*, pp. 204–211, Santa Clara, CA, June 27-July 2 2013.

Zhang, Q., Liu, L., Lee, K., Zhou, Y., Singh, A., Mandagere, N., Gopisetty, S., and Alatorre, G. Improving hadoop service provisioning in a geographically distributed cloud. In *Proceedings of the 2014 IEEE International Conference on Cloud Computing (CLOUD'14)*, pp. 432–439, Anchorage, AK, June 27-July 2 2014.

Zhang, Z., Jin, J., Zhang, Z., Zhou, Y., Zhao, X., Ren, J., Liu, J., Wu, L., Jin, R., and Dou, D. Validating the lottery ticket hypothesis with inertial manifold theory. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021 (NeurIPS'21)*, Virtual.

Zhou, C., Liu, J., Jia, J., Zhou, J., Zhou, Y., Dai, H., and Dou, D. Efficient device scheduling with multi-job federated learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, Vancouver, Canada, February 22-March 1 2022a.

Zhou, C., Liu, J., Jia, J., Zhou, J., Zhou, Y., Dai, H., and Dou, D. Efficient device scheduling with multi-job federated learning. In *AAAI Conf. on Artificial Intelligence*, pp. 1–14, 2022b.

Zhou, Y. *Innovative Mining, Processing, and Application of Big Graphs*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2017.

Zhou, Y. and Liu, L. Approximate deep network embedding for mining large-scale graphs. In *Proceedings of the 2019 IEEE International Conference on Cognitive Machine Intelligence (CogMI'19)*, pp. 53–60, Los Angeles, CA, December 12-14 2019.

Zhou, Y., Liu, L., Lee, K., Pu, C., and Zhang, Q. Fast iterative graph computation with resource aware graph parallel abstractions. In *Proceedings of the 24th ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC'15)*, pp. 179–190, Portland, OR, June 15-19 2015a.

Zhou, Y., Liu, L., Lee, K., and Zhang, Q. Graphtwist: Fast iterative graph computation with two-tier optimizations. *Proceedings of the VLDB Endowment (PVLDB)*, 8(11): 1262–1273, 2015b.

Zhou, Y., Amimeur, A., Jiang, C., Dou, D., Jin, R., and Wang, P. Density-aware local siamese autoencoder network embedding with autoencoder graph clustering. In *Proceedings of the 2018 IEEE International Conference on Big Data (BigData'18)*, pp. 1162–1167, Seattle, WA, December 10-13 2018a.

Zhou, Y., Wu, S., Jiang, C., Zhang, Z., Dou, D., Jin, R., and Wang, P. Density-adaptive local edge representation learning with generative adversarial network multi-label edge classification. In *Proceedings of the 18th IEEE International Conference on Data Mining (ICDM'18)*, pp. 1464–1469, Singapore, November 17-20 2018b.

Zhou, Y., Ye, Q., and Lv, J. Communication-efficient federated learning with compensated overlap-fedavg. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 33(1):192–205, 2022c.

A. Appendix

A.1. Related Work

Parallel, distributed, and federated learning have attracted active research in the last decade (Zhang et al., 2013; Lee et al., 2013; Su et al., 2013; Palanisamy et al., 2014; Zhang et al., 2014; Su et al., 2015; Zhou et al., 2015a; Bao et al., 2015; Zhou et al., 2015b; Lee et al., 2015; Jiang et al., 2016; Zhou, 2017; Zhou et al., 2018b;a; Palanisamy et al., 2018; Lee et al., 2019; Zhou & Liu, 2019; Goswami et al., 2020; Zhang et al.; Zhou et al., 2022a; Yan et al., 2022b;a; Liu et al., 2022; Guimu Guo & Zhou, 2022; Zhang et al., 2022). While it achieves much advancement in federated learning, FedAvg (McMahan et al., 2017b) does not consider the adaptive optimization when aggregating the weights or gradients from devices, which makes the training process inefficient and makes it difficult to tune or to achieve desired accuracy. SCAFFOLD (Karimireddy et al., 2020c) exploits control variate to reduce the client drift problem, i.e., "over-fitting" to local device data, in order to reduce the influence of the heterogeneity of non-IID data and to improve the performance (accuracy). The control parameters may result in stateful devices (Karimireddy et al., 2020c; Acar et al., 2021), which is not compatible with cross-device Federated Learning (FL) because of full participation of devices. The cross-device FL simultaneously deals with large amounts of devices while each round only samples a fraction of the available devices for the training process in order to mitigate the straggler problem (Kairouz et al., 2019; Liu et al., 2021). Although the contribution of the devices can be exploited to dynamically update the global model (Wu & Wang, 2021), the contribution-based dynamic update cannot well address the client-drift problem. The contribution can calculated based on the angle between the local gradient and the global gradient.

Adaptive optimization can be applied either at the server side to improve the performance (Reddi et al., 2021b) and to reduce the communication and computation costs (Mills et al., 2021), using multiple adaptive optimization method, e.g., AdaGrad (Duchi et al., 2011a), Yogi (Reddi et al., 2018), Adam (Kingma & Ba, 2014; Mills et al., 2019) and momentum (Rothchild et al., 2020; Mills et al., 2021), or at the client side (Yuan et al., 2021b) to reduce the number of rounds (Mills et al., 2019) utilizing momentum (Liu et al., 2020b; Gao et al., 2021; Wang et al., 2020), Adam (Mills et al., 2019). In contrast, the single side application of existing adaptive optimization, i.e., the server side or the device side, may lead to insufficient performance, due to incomplete adaptation of the adaptive optimization methods. While correction techniques can be utilized to mitigate the bias of the convergence point for the application of AdaGrad at the device side (Wang et al., 2021c), the application at both the server side and the device should be addressed at the same time to achieve better performance. In addition, the application of momentum at the device side (Liu et al., 2020b) may incur severe client-drift because of heterogeneity of non-IID data and law participation rate of devices. Moreover, the synchronization of the global momentum parameters may incur extra communication costs (Wang et al., 2020). Momentum (Karimireddy et al., 2020b; Ozfatura et al., 2021; Khanduri et al., 2021) and Adam (Leroy et al., 2019) can be utilized on both the server and the devices to achieve fast training speed and high convergence accuracy, while the simple application may correspond to limited improvement.

The adaptive optimization methods can be combined with compression mechanism (Mills et al., 2019; Gao et al., 2021; Rothchild et al., 2020; Li et al., 2020) to further reduce the communication costs. In addition, device selection methods are utilized to achieve faster convergence (Xia et al., 2021) and higher accuracy (Chen et al., 2020), even with multiple jobs (Zhou et al., 2022b). Furthermore, overlapping the local training process and the data transfer can improve the communication efficiency (Zhou et al., 2022c). Sparsification (Mitra et al., 2021b) and quantization (Anonymous, 2022c) can be exploited in FL to improve the communication efficiency, wherein the update at the device side relies on the gradients of the last round of all the devices and more communication is required to configure the sparsification parameters (Mitra et al., 2021b). Moreover, the combination of gossip protocol-based decentralized model aggregation, which is performed at devices, and the centralized model aggregation, which is performed at the server, helps to speed up the training process of FL (Anonymous, 2022d). Leading principal components (Anonymous, 2022g), low-rank hadamard product (Hyeon-Woo et al., 2021), and progressive training (Wang et al., 2021a), i.e., the model progressively grows along with the training process, can be exploited as well to reduce the communication costs and improve the performance. Some other methods, e.g., communication-efficient ensemble algorithms (Hamer et al., 2020b), hierarchical architecture (Yang, 2021; Wu et al., 2020), blockchain-based mechanism (Yapp et al., 2021), federated optimization (Pathak & Wainwright, 2020; Dai et al., 2020), dual averaging procedure for non-smooth regularizer (Yuan et al., 2021b), agnostic constrained learning formulation for class imbalance problems (Anonymous, 2022b), separation of unseen client data and unseen client distributions (Yuan et al., 2021a), knowledge distillation for heterogeneous model structures (Afonin & Karimireddy, 2021), minibatch and local random reshuffling (Yun et al., 2021), unlabeled data transformation for unsupervised LF (Anonymous, 2022h), multi-task FL for image processing (Anonymous, 2022f), and personalization (Zhang et al., 2021; Oh et al., 2021; Hyeon-Woo et al., 2021) or the trade-off between personalization and generalization (Chen & Chao, 2021), are proposed to further improve the performance of FL. However, the above works are orthogonal to our approach and out of the scope of this paper.

As private data may be recovered with small malicious modifications of the shared model (Fowl et al., 2021) or sensitive information may still be leaked from the gradients (Balunović et al., 2021), differential privacy (Anonymous, 2022e; Triastcyn et al., 2021) and compression techniques (Triastcyn et al., 2021) are combined to improve the privacy and communication efficiency of FL, while the encoding of gradient knowledge with the regularization of locally trained parameters (Anonymous, 2022a) helps improve the performance and robustness at the same time. While gradient flips are indicative of attacks, reputation-based weighted aggregation can help to improve the robustness of FL (Sharma et al., 2021). Batch-normalization (Hong et al., 2021) and Bayes optimal method (Balunović et al., 2021) can be exploited to deal with adversarial training attack, while bucketing methods can alleviate the impact of byzantine attacks (Karimireddy et al., 2020a). These methods can be combined with our approach to improve the robustness and the privacy of FL.

This work presents a theoretical principle on where to and how to design and utilize adaptive optimization methods in federated settings. We theoretically analyze the connection between federated optimization methods and decompositions of ODEs of corresponding centralized optimizers. We develop a momentum decoupling adaptive optimization method to make full use of the strength of fast convergence and high accuracy of the global momentum. We also utilize the full batch gradients to mimic centralized optimization for ensuring the convergence and overcoming the possible inconsistency caused by adaptive optimization methods.

A.2. Momentum Decoupling Adaptive Optimization: FedDA+Adam and FedDA+AdaGrad

Recall the centralized Adam optimizer.

$$\begin{aligned} &m(t+1) = \beta_1 * m(t) + (1-\beta_1) * g(W(t)),\\ &v(t+1) = \beta_2 * v(t) + (1-\beta_2) * (g(W(t))^2,\\ &\hat{m}(t+1) = m(t+1)/(1-\beta_1^t),\\ &\hat{v}(t+1) = v(t+1)/(1-\beta_2^t),\\ &W(t+1) = W(t) - \hat{m}(t+1)/(\sqrt{\hat{v}(t+1)}+\epsilon) * \eta. \end{aligned} \tag{30}$$

where $g(W) = \sum_{i=1}^{M} \frac{N^i}{N} g^i(W)$. The Adam optimizer is the numerical solution to the ODE system

$$\begin{aligned} &\eta \frac{d}{d\tau} m(\tau) = -(1-\beta_1) m(\tau) + (1-\beta_1) * g(W(\tau)),\\ &\eta \frac{d}{d\tau} v(\tau) = -(1-\beta_2) v(\tau) + (1-\beta_2) * (g(W(\tau))^2,\\ &\hat{m}(\tau) = m(\tau)/(1-\beta_1^\tau),\\ &\hat{v}(\tau) = v(\tau)/(1-\beta_2^\tau),\\ &\frac{d}{d\tau} W(\tau) = -\hat{m}(\tau)/(\sqrt{\hat{v}(\tau)}+\epsilon) * \eta. \end{aligned} \tag{31}$$

The equilibria of the above is system are $(m, v, W) = (0, 0, W^*)$, where W^* is local minimum point of the loss function, i.e., $g(W^*) = 0$. Therefore, the Adam optimizer trains the model to converge to local minimum point of the loss function. Applying our decoupling method, we first decouple the global momentum and velocity m, v with local training, i.e.,

$$\begin{aligned} &\frac{d}{d\tau} W^i(\tau) = -g^i(W^i(\tau)),\\ &\eta \frac{d}{d\tau} m(\tau) = -(1-\beta_1) m(\tau) + (1-\beta_1) * g(W(\tau)),\\ &\eta \frac{d}{d\tau} v(\tau) = -(1-\beta_2) v(\tau) + (1-\beta_2) * (g(W(\tau))^2. \end{aligned} \tag{32}$$

Similar to the SGDM case, the equation of $m(\tau)$ is totally linear and can be decomposed precisely as

$$\eta \frac{d}{d\tau} m^i(\tau) = -(1-\beta_1) m^i(\tau) + (1-\beta_1) g^i(W^i(\tau)). \tag{33}$$

Unlike the equation of m which is linear, the equation of v is nonlinear due to the presence of the nonlinear term $(g(W))^2 = \big(\sum_{i=1}^{M} \frac{N^i}{N} g^i(W)\big)^2$. It is apparent that

$$\big(\sum_{i=1}^{M} \frac{N^i}{N} g^i(W)\big)^2 \neq \sum_{i=1}^{M} \frac{N^i}{N} (g^i(W))^2.$$

Therefore, it is not possible to decompose the equation of v precisely. Moreover, $\sum_{i=1}^{M} \frac{N^i}{N} (g^i(W))^2$ is not even a close approximation of $\big(\sum_{i=1}^{M} \frac{N^i}{N} g^i(W)\big)^2$. An immediate counter example is that $(1-1)^2 \neq 1^2 + (-1)^2$. For centralized Adam, the expectation of each component of $\hat{m}/\sqrt{\hat{v}}$ is approximately ± 1 , which is a crucial point of Adam optimizer. Thus, when attempting to generalize centralized Adam optimizer to FL, the key is that m, v must be highly synchronized as in centralized Adam. There are two possible resolutions. First, as in previous arguments of precise decomposition, by reducing local iteration number to 1, one obtains a precise decomposition of Eq.(31). Consequently, the FL training matches centralized training. However, the communication cost is too expensive to be affordable for this approach. Another possible way is to utilize Adam optimizer on server, i.e., local models are trained with SGDM and then server aggregates total gradients uploaded by local devices and input the aggregation into a global Adam. This is the same as FedOpt method. However, on second thought, though v cannot be precisely decomposed, the global momentum m can be decomposed precisely as mentioned before and it still can be utilized fully. Therefore, we propose our FedDA+Adam method as follows. At round E, pick participating clients $S^1, \cdots, S^k$. For each S^i, $i = 1, \cdots, K$, initialize $P^i = 0$, $W^i(0) = W(E)$, $m^i(0) = m(E)$. For $t = 0, T-1$,

$$\begin{aligned}
&W^i(t+1) = W^i(t) - g^i(W^i(t)) * \eta \\
&m^i(t+1) = \beta_1 m^i(t) + (1-\beta_1) g^i(W^i(t)) \\
&P^i = P^i + m^i(t+1)
\end{aligned} \tag{34}$$

The global update rule is:

$$\begin{aligned}
&P = \text{aggregation of } \ P^i, \\
&m(E+1) = \text{aggregation of } m^i(T) \\
&\mathcal{G} = \big(P - \beta_1 * m(E)\big)/(1-\beta_1) \\
&\hat{m}(E+1) = \big(\beta_1 * m(E) + (1-\beta_1) * \mathcal{G}\big)/(1-\beta_1^E) \\
&V(E+1) = \beta_2 * V(E) + (1-\beta_2) * \mathcal{G}^2 \\
&\hat{V}(E+1) = V(E)/(1-\beta_2^E) \\
&W(E+1) = W(E) - \hat{m}(E+1)/(\sqrt{\hat{V}(E+1)} + \epsilon) * \eta * \alpha
\end{aligned} \tag{35}$$

The idea is to treat P as the momentum of the global Adam updated with a global gradient $\mathcal{G}$, i.e., $P = \beta_1 + (1-\beta_1) * \mathcal{G}$. Simple algebra gives the global gradient $\mathcal{G} = \big(P - \beta_1 * m(E)\big)/(1-\beta_1)$. Then the global velocity v is updated with $\mathcal{G}$ followed by the update of W.

Recall the AdaGrad optimizer

$$\begin{aligned}
&V(t) = V(t-1) + g(W(t))^2, \\
&W(t+1) = W(t) - g(W(t))/\big(\sqrt{V(t)} + \epsilon\big) * \eta.
\end{aligned} \tag{36}$$

The corresponding ODE system is

$$\begin{aligned}
&\eta * \frac{d}{d\tau} V(\tau) = g(W(\tau))^2, \\
&\frac{d}{d\tau} W(\tau) = -g(W(\tau))/\big(\sqrt{V(\tau)} + \epsilon\big).
\end{aligned} \tag{37}$$

For the same reason, i.e., the nonlinearity of $\big(\sum_{i=1}^{M} \frac{N^i}{N} g^i(W)\big)^2$, AdaGrad optimizer is not suitable for decomposing onto local devices as well. As in FedDA+Adam, we propose to utilize the decoupled global momentum to generate a global gradient $\mathcal{G}$ and update the global AdaGrad optimizer with input $\mathcal{G}$.

$$\begin{aligned} &W^i(t+1) = W^i(t) - g^i(W^i(t)) * \eta \\ &m^i(t+1) = \beta_1 m^i(t) + (1-\beta_1) g^i(W^i(t)) \\ &P^i = P^i + m^i(t+1) \end{aligned} \tag{38}$$

The global update rule is:

$$\begin{aligned} &P = \text{aggregation of } \ P^i, \\ &m(E+1) = \text{aggregation of } m^i(T) \\ &\mathcal{G} = \big(P - \beta_1 * m(E)\big)/(1-\beta_1) \\ &V(E+1) = V(E) + \mathcal{G}^2 \\ &W(E+1) = W(E) - \mathcal{G}/(\sqrt{\hat{V}(E+1)} + \epsilon) * \eta \end{aligned} \tag{39}$$

By assembling all the pieces in Sections 4 and A.2, we provide the pseudo code of our FedDA algorithm with the adaptive optimization of SGDM, Adam, and AdaGrad in Algorithms 1 and 2 respectively.

Algorithm 1 FedDA+SGDM

Input: W_0, m_0

```
 1: for $r = 0, \cdots, R-1$ do
 2:   Sample a subset $S$ of clients
 3:   for each client $S^i \in S$ in parallel do
 4:     $P^i_{r,0} = 0, W^i_{r,0} = W_r, m^i_{r,0} = m_r$
 5:     for $t = 0, \cdots, T-1$ do
 6:       $W^i_{r,t+1} = W^i_{r,t} - g^i_{r,t}(W^i_{r,t}) * \eta$
 7:       $m^i_{r,t+1} = \beta * m^i_{r,t} + (1-\beta) * g^i_{r,t}(W^i_{r,t})$
 8:       $P^i_{r,t+1} = P^i_{r,t} + m^i_{r,t+1}$
 9:   $P_r =$ aggregation of $P^i_{r,T}$
10:   $m_{r+1} =$ aggregation of $m^i_{r,T}$
11:   $W_{r+1} = W_r - P_r * \eta * \alpha$
```

Algorithm 2 FedDA+ADAM& FedDA+AdaGrad

Input: W_0, m_0, V_0

```
 1: for $r = 0, \cdots, R-1$ do
 2:   Sample a subset $S$ of clients
 3:   for each client $S^i \in S$ in parallel do
 4:     $P^i_{r,0} = 0, W^i_{r,0} = W_r, m^i_{r,0} = m_r$
 5:     for $t = 0, \cdots, T-1$ do
 6:       $W^i_{r,t+1} = W^i_{r,t} - g^i_{r,t}(W^i_{r,t}) * \eta$
 7:       $m^i_{r,t+1} = \beta_1 * m^i_{r,t} + (1-\beta_1) * g^i_{r,t}(W^i_{r,t})$
 8:       $P^i_{r,t+1} = P^i_{r,t} + m^i_{r,t+1}$
 9:   $P_r =$ aggregation of $P^i_{r,T}$
10:   $m_{r+1} =$ aggregation of $m^i_{r,T}$
11:   $\mathcal{G}_r = (P_r - \beta_1 * m_r)/(1-\beta_1)$
12:   ADAM
13:     $\hat{m}_{r+1} = (\beta_1 * m_r + (1-\beta_1) * \mathcal{G}_r)/(1-\beta_1^r)$
14:     $V_{r+1} = (\beta_2 * V_r + (1-\beta_2) * \mathcal{G}_r^2)$
15:     $\hat{V}_{r+1} = V_{r+1}/(1-\beta_2^r)$
16:     $W_{r+1} = W_r - \hat{m}_{r+1}/(\sqrt{V_{r+1}} + \epsilon) * \eta * \alpha$
17:   AdaGrad
18:     $V_{r+1} = V_r + \mathcal{G}_r^2$
19:     $W_{r+1} = W_r - \mathcal{G}_r/(\sqrt{V_{r+1}} + \epsilon) * \eta * \alpha$
```

A.3. Advantage of Global Momentum in FedDA

Assume that

1. (Lipschitz Gradient). There exists a constant L_g such that $\|g^i(W_1) - g^i(W_2)\| \leq L_g\|W_1 - W_2\|$ for any W_1, W_2 and $i = 1, \cdots, m$.
2. (Bounded Gradient) $\|g^i\|_{L^\infty} < \infty$ for any $i = 1, 2, \cdots, m$.

Under these assumptions, we theoretically demonstrate that (1) local momentum deviates from the centralized one at exponential rate $O(e^{\lambda t})$; and (2) global momentum in FedDA deviates from the centralized one at algebraic rate $O(t^2)$.

Local momentum deviates from the centralized one at exponential rate. Let $(m^i(\tau), W^i(\tau))$ be the solution to the decomposed system

$$\begin{aligned} &\eta\frac{d}{d\tau}m^i(\tau) = -(1-\beta)m^i(\tau) + (1-\beta)g^i(W^i(\tau)), \\ &\frac{d}{d\tau}W^i(\tau) = -m^i(\tau). \end{aligned} \tag{40}$$

Let $(m(\tau), W(\tau))$ be the solution to Eq.(19) with $(m(0), W(0)) = (m^i(0), W^i(0))$, i.e., centralized SGDm optimizer with the same initialization as the decomposed optimizer. By direct calculations, we have

$$\begin{aligned} &\eta\frac{d}{d\tau}(m^i(\tau) - m(\tau)) = (1-\beta)\Big(-(m^i(\tau) - m(\tau)) + \big(g^i(W^i(\tau)) - g^i(W(\tau))\big) + R(\tau)\Big), \\ &\frac{d}{d\tau}(W^i(\tau) - W(\tau)) = -(m^i(\tau) - m(\tau)), \end{aligned} \tag{41}$$

where $R(\tau) = \sum_{j\neq i}\frac{N_j}{N}\big(g^j(W(\tau)) - g^i(W(\tau))\big)$. One can check that $\nabla\|x\| = \frac{x}{\|x\|}$ for nay $x \in \mathbb{R}^n$. Therefore, it holds that $\|\nabla\|x\|\| \leq 1$ and $x\cdot\nabla\|x\| = \|x\|$, where $\cdot$ means dot product. Taking the inner product of the first equation in Eq.(41) with $\nabla\|m^i(\tau) - m(\tau)\|$, we obtain

$$\eta\frac{d}{d\tau}\|m^i(\tau) - m(\tau)\| \leq (1-\beta)\Big(-\|m^i(\tau) - m(\tau)\| + L_g\|W^i(\tau) - W(\tau)\| + \|R(\tau)\Big). \tag{42}$$

Similarly, take the inner produce of the second equation in Eq.(41) with $\nabla\|W^i - W\|$, we obtain

$$\frac{d}{d\tau}\|W^i(\tau) - W(\tau)\| \leq \|m^i(\tau) - m(\tau)\| \tag{43}$$

The system of Eq.(42) and Eq.(43) can be written as the following matrix form.

$$\frac{d}{d\tau}\begin{pmatrix}\|m^i(\tau) - m(\tau)\| \\ \|W^i(\tau) - W(\tau)\|\end{pmatrix} \leq A\begin{pmatrix}\|m^i(\tau) - m(\tau)\| \\ \|W^i(\tau) - W(\tau)\|\end{pmatrix} + (1-\beta)\begin{pmatrix}\|R(\tau)\|/\eta \\ 0\end{pmatrix} \tag{44}$$

where $A = \begin{pmatrix}-(1-\beta)/\eta & (1-\beta)L_g/\eta \\ 1 & 0\end{pmatrix}$. By direct computations, the eigenvalue of the matrix A is $\lambda^\pm = \frac{-(1-\beta)\pm\sqrt{(1-\beta)^2+4(1-\beta)L_g}}{2\eta}$. Therefore $\|e^{At}\| \leq C_A e^{\lambda^+ t}$ for any $t \geq 0$ for some constant C_A. Note that $\lambda^+ > 0$. Applying variation of constants formula to Eq.(44), one has

$$\begin{pmatrix}\|m^i(t\eta) - m(t\eta)\| \\ \|W^i(t\eta) - W(t\eta)\|\end{pmatrix} \leq e^{At\eta}\begin{pmatrix}\|m^i(0) - m(0)\| \\ \|W^i(0) - W(0)\|\end{pmatrix} + (1-\beta)\int_0^{t\eta} e^{A(t\eta-\tau)}\begin{pmatrix}\|R(\tau)\|/\eta \\ 0\end{pmatrix}d\tau. \tag{45}$$

Note that $m^i(0) - m(0) = 0$ and $W^i(0) - W(0)$. Therefore, for any t, it holds

$$\begin{pmatrix}\|m^i(t\eta) - m(t\eta)\| \\ \|W^i(t\eta) - W(t\eta)\|\end{pmatrix} \leq (1-\beta)\int_0^{t\eta} e^{A(t\eta-\tau)}\begin{pmatrix}\|R(\tau)\|/\eta \\ 0\end{pmatrix}d\tau. \tag{46}$$

Utilizing $\|e^{At}\| \le C_A e^{\lambda^+ t}$ for $t \ge 0$ in the above inequality , we obtain

$$\begin{aligned}&\|m^i(t\eta) - m(t\eta)\| + \|W^i(t\eta) - W(t\eta)\| \\ &\le \int_0^{t\eta} e^{\lambda^+(t\eta-\tau)}\|R(\tau)\|/\eta d\tau = e^{\lambda^+ t\eta}(1-\beta)\int_0^{t\eta} e^{-\lambda^+\tau}\left(\|R(\tau)\|/\eta\right)d\tau,\end{aligned} \tag{47}$$

which implies that $\|m^i(t\eta) - m(t\eta)\|$ is exponentially growing. Essentially, the term $\big(g^i(W^i(\tau)) - g^i(W(\tau))$ in Eq. (41) causes such exponential growth. More precisely, since $g^i(W^i) - g^i(W) = \nabla g^i(W)(W^i - W) + O(|W^i - W|^2)$, it contributes a linear term $\nabla g^i(W)(W^i - W)$ to Eq.(41). Consequently, the matrix A has a positive eigenvalue, which implies that $\|e^{At}\|$ is exponentially growing. Thus, the difference of momentum $\|m^i(t\eta) - m(t\eta)\|$ is exponentially growing. However, in our FedDA method, since momentum is decoupled with local training, there is no such a term as $g^i(W^i) - g^i(W)$. Therefore, in our FedDA framework, the matrix analogous to matrix A is $\begin{pmatrix} -(1-\beta)/\eta & 0 \\ 1 & 0\end{pmatrix}$, whose eigenvalues are $0, -(1-\beta)$. Therefore, the momentum deviation in our FedDA method is only algebraic growing.

Global momentum in FedDA deviates from the centralized one at algebraic rate. Let $(\bar{W}^i(\tau), \bar{m}(\tau), \bar{W}(\tau))$ be the solution to Eq.(22) with $\alpha = 1$. Let $(m(\tau), W(\tau))$ be the solution to Eq.(19) with $(m(0), W(0)) = (\bar{m}(0), \bar{W}(0))$. It is straightforward to compute

$$\begin{aligned}&\eta\frac{d}{d\tau}(\bar{m}(\tau) - m(\tau)) = -(1-\beta)\Big((\bar{m}(\tau) - m(\tau)) + \sum_i^M \frac{N_i}{N}\big(g^i(\bar{W}^i(\tau)) - g^i(W(\tau))\big)\Big) \\ &\frac{d}{d\tau}(\bar{W}(\tau) - W(\tau)) = -(\bar{m}(\tau) - m(\tau)),\end{aligned} \tag{48}$$

By calculations similar to the above ones, we first have

$$\frac{d}{d\tau}\begin{pmatrix}\|\bar{m}(\tau) - m(\tau)\| \\ \|\bar{W}(\tau) - W(\tau)\|\end{pmatrix} \le \bar{A}\begin{pmatrix}\|\bar{m}(\tau) - m(\tau)\| \\ \|\bar{W}(\tau) - W(\tau)\|\end{pmatrix} + (1-\beta)\begin{pmatrix}\|\sum_i^M \frac{N_i}{N}\big(g^i(\bar{W}^i(\tau)) - g^i(W(\tau))\big)\|/\eta \\ 0\end{pmatrix} \tag{49}$$

where $\bar{A} = \begin{pmatrix} -(1-\beta)/\eta & 0 \\ 1 & 0\end{pmatrix}$. One can check that the eigenvalues of $\bar{A}$ are $0, -(1-\beta)/\eta$, which implies that there exists a constant $C_{\bar{A}}$ such that $\|e^{\bar{A}t}\| \le C_{\bar{A}}$ for any $t \ge 0$, i.e., $\|e^{\bar{A}t}\|$ is uniformly bounded without growth. Clearly,

$$\|g^i(\bar{W}^i(\tau)) - g^i(W(\tau))\| \le L_g\|\bar{W}^i(\tau) - W(\tau)\|. \tag{50}$$

Moreover, utilizing $\bar{W}^i(0) = W(0)$, we have

$$\bar{W}^i(\tau) - W(\tau) = \int_0^\tau -g^i(\bar{W}^i(s)) + m(s)ds. \tag{51}$$

It follows that

$$\|\bar{W}^i(\tau) - W(\tau)\| \le \big(\|g^i\|_{L^\infty} + \sup_{s\le\tau}\|m(s)\|\big)\tau. \tag{52}$$

By variation of constants formula, we have

$$\|m(\tau)\| \le e^{-(1-\beta)\tau/\eta}\|m(0)\| + \frac{1}{\eta}\int_0^\tau e^{-(1-\beta)(\tau-s)/\eta}(1-\beta)\|g(W(s))\|ds, \tag{53}$$

which implies that $|m(\tau)| \le |m(0)| + \|g\|_{L^\infty}$ for any $\tau \ge 0$. Therefore, we have

$$\|\bar{W}^i(\tau) - W(\tau)\| = \big(\|g^i\|_{L^\infty} + \|m(0)\| + \|g\|_{L^\infty}\big)\tau. \tag{54}$$

Combining Eq.(50) and Eq.(54), we have

$$\|g^i(\bar{W}^i(\tau)) - g^i(W(\tau))\| \le \|\nabla g^i\|_{L^\infty}\big(\|g^i\|_{L^\infty} + \|m(0)\| + \|g\|_{L^\infty}\big)\tau. \tag{55}$$

Applying the variation of constants formula to Eq.(49), we have

$$\begin{pmatrix} \|\bar{m}(t\eta) - m(t\eta)\| \\ \|\bar{W}(t\eta) - W(t\eta)\| \end{pmatrix} \leq \int_0^{t\eta} e^{\bar{A}(t\eta-\tau)} \begin{pmatrix} \| \sum_i^M \frac{N_i}{N} \left(g^i(\bar{W}^i(\tau)) - g^i(W(\tau)) \right) \| / \eta \\ 0 \end{pmatrix} d\tau. \tag{56}$$

Utilizing $\|e^{\bar{A}t}\| \leq C_{\bar{A}}$ and Eq.(55), it holds

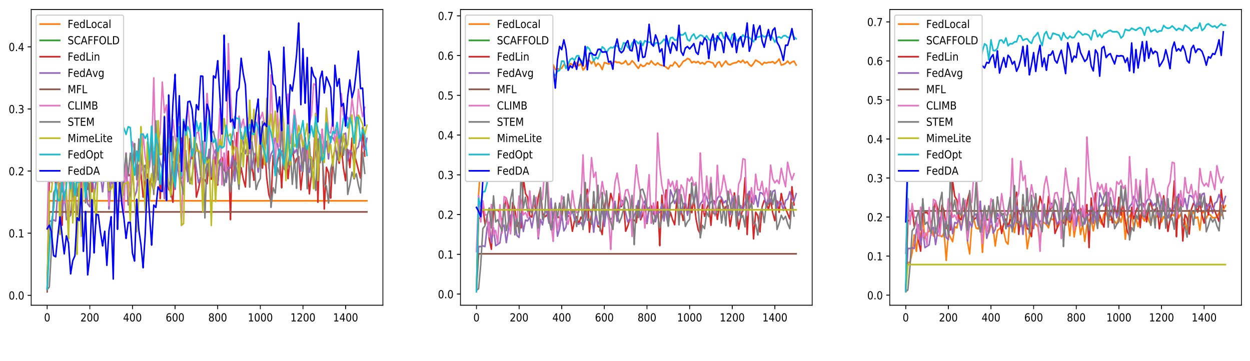
$$\|\bar{m}(t\eta) - m(t\eta)| + \|\bar{W}(t\eta) - W(t\eta)\| \leq \int_0^{t\eta} C^* \tau/\eta d\tau = \frac{C^*\eta}{2} t^2, \tag{57}$$

where $C^* = \sup_i \|\nabla g^i\|_{L^\infty} \left(\|g^i\|_{L^\infty} + \|m(0)\| + \|g\|_{L^\infty} \right)$. Therefore, the momentum in our FedDA method deviates from the centralized one in algebraic rate $O(t^2)$, which is much slower than that of local momentum.

A.4. Additional Experiments

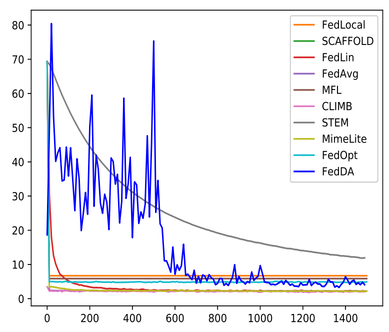
In this section, we conduct more experiments to validate the accuracy and convergence of our proposed FedDAmethod and evaluate the sensitivity of client and server learning rates in our momentum decoupling adaptive optimization method for the FL task.

Convergence and loss over Stack Overflow. Figures 7-8 present the *Accuracy* and *Loss* curves of ten federated learning algorithms on Stack Overflow. Similar trends are observed for the performance comparison: FedDA achieves the 75.4% convergence improvement, which are obviously better than other methods in most experiments. This demonstrates that the full batch gradient techniques that mimic centralized optimization in the end of the training process are able to ensure the convergence and overcome the possible inconsistency caused by adaptive optimization methods.

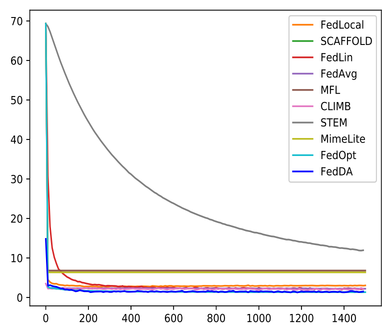


(a) SGDM (b) Adam (c) AdaGrad

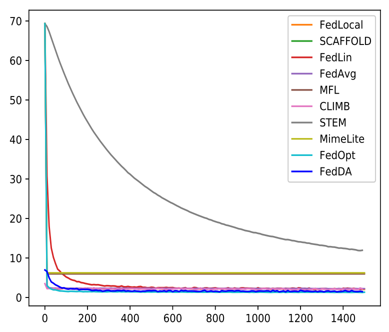
Figure 7: Convergence on Stack Overflow with Three Optimizers



(a) SGDM



(b) Adam



(c) AdaGrad

Figure 8: Loss on Stack Overflow with Three Optimizers

Final *Accuracy* over Stack Overflow. Table 5 presents the final *Accuracy* scores of ten federated learning algorithms on Stack Overflow. We have observed similar trends: the accuracy achieved by our FedDA is the highest in most tests. Especially, as shown the experiment with SGDM as the optimizer, compared to the best competitors among ten federated learning algorithms, the final *Accuracy* scores achieved by FedDA averagely achieves 22.3% improvement. A rational guess is that the global momentum in our FedDA method makes the best effort to mimic the role of momentum in centralized training, which can accelerate the convergence of FL training.

Table 5: Final Accuracy on Stack Overflow

Optimizer	SGDM	Adam	AdaGrad
FedLocal	0.152	0.576	0.229
SCAFFOLD	0.250	0.250	0.250
FedLin	0.224	0.224	0.224
FedAvg	0.252	0.252	0.252
MFL	0.134	0.101	0.215
CLIMB	0.302	0.302	0.302
STEM	0.196	0.196	0.196
MimeLite	0.271	0.211	0.078
FedOpt	0.225	**0.642**	**0.691**
FedDA	**0.273**	**0.642**	0.674

Impact of client and server learning rates. Tables 6-11 report how the test *Accuracy* changes with server and client learning rates on three datasets by fixing the server learning rates and changing the client learning rates, or by utilizing the reverse settings. We have observed that the *Accuracy* scores oscillate within the range of 0.002 and 0.868 when changing the client learning rates, while the *Accuracy* values fluctuate between 0.010 and 0.861. This demonstrates that it is crucial to choose the optimal learning rates for the training on the clients and server to achieve the competitive performance. Please refer to Table 12 for the implementation details of the server and client learning rates used in our current experiments.

Table 6: Final Accuracy with SGDM Optimizer and Varying Client Learning Rate

Dataset	Server Learning Rate	Accuracy / Client Learning Rate			
CIFAR-100	0.1	0.521 / 1	0.448 / 3.3	0.236 / 10	0.099 / 33
EMNIST	0.1	0.778 / 1	0.819 / 3.3	0.783 / 10	0.052 / 33
Stack Overflow	0.1	0.174 / 1	0.262 / 10	0.250 / 100	0.002 / 1,000

Table 7: Final Accuracy with Adam Optimizer and Varying Client Learning Rate

Dataset	**Server Learning Rate**	**Accuracy / Client Learning Rate**			
CIFAR-100	0.1	0.179 / 1	0.317 / 3.3	0.269 / 10	0.179 / 33
EMNIST	0.03	0.488 / 0.01	0.836 / 0.1	0.842 / 0.3	0.804 / 1
Stack Overflow	0.3	0.409 / 0.1	0.492 / 1	0.473 / 10	0.459 / 100

Table 8: Final Accuracy with AdaGrad Optimizer and Varying Client Learning Rate

Dataset	**Server Learning Rate**	**Accuracy / Client Learning Rate**			
CIFAR-100	0.1	0.462 / 0.03	0.488 / 0.1	0.410 / 0.3	0.259 / 1
EMNIST	0.1	0.055 / 0.00001	0.055 / 0.001	0.055 / 0.01	0.868 / 0.1
Stack Overflow	10	0.576 /1	0.651 / 10	0.629 / 30	0.632 / 100

Table 9: Final Accuracy with SGDM Optimizer and Varying Server Learning Rate

Dataset	**Client Learning Rate**	**Accuracy / Server Learning Rate**			
CIFAR-100	0.03	0.374 / 1	0.522 / 3.3	0.356 / 10	0.273 / 33
EMNIST	0.1	0.859 / 0.3	0.861 / 1	0.778 / 3.3	0.783 / 10
Stack Overflow	100	0.191 / 0.001	0.256 / 0.003	0.204 / 0.01	0.292 / 0.1

Table 10: Final Accuracy with Adam Optimizer and Varying Server Learning Rate

Dataset	**Client Learning Rate**	**Accuracy / Server Learning Rate**			
CIFAR-100	0.03	0.510 / 0.33	0.183 / 3.3	0.010 / 10	0.010 / 33
EMNIST	0.03	0.546 / 0.1	0.803 / 1	0.051 / 3.3	0.051 / 10
Stack Overflow	100	0.425 / 0.1	0.522 / 0.3	0.641 / 1	0.633 / 10

Table 11: Final Accuracy with AdaGrad Optimizer and Varying Server Learning Rate

Dataset	**Client Learning Rate**	**Accuracy / Server Learning Rate**			
CIFAR-100	0.03	0.352 / 0.03	0.462 / 0.1	0.466 / 0.3	0.312 / 1
EMNIST	0.03	0.806 / 0.1	0.055 / 3.3	0.055 / 10	0.055 / 33
Stack Overflow	100	0.227 / 0.1	0.306 / 0.3	0.397 / 1	0.632 / 10

A.5. Experimental Details

Environment. Our experiments were conducted on a compute server running on Red Hat Enterprise Linux 7.2 with 2 CPUs of Intel Xeon E5-2650 v4 (at 2.66 GHz) and 8 GPUs of NVIDIA GeForce GTX 2080 Ti (with 11GB of GDDR6 on a 352-bit memory bus and memory bandwidth in the neighborhood of 620GB/s), 256GB of RAM, and 1TB of HDD. Overall, our experiments took about 2 days in a shared resource setting. We expect that a consumer-grade single-GPU machine (e.g., with a 1080 Ti GPU) could complete our full set of experiments in around tens of hours, if its full resources were dedicated.

All the codes were implemented based on the Tensorflow Federated (TFF) package (Ingerman & Ostrowski, 2019). Clients are sampled uniformly at random, without replacement in a given round, but with replacement across rounds. Our implementation follows the same settings in the approaches of FedAvg (McMahan et al., 2017a) and FedOpt (Reddi et al., 2021a). First, instead of doing K training steps per client, we do E epochs of training over each client's dataset. Second, to account for varying numbers of gradient steps per client, we weight the average of the client outputs by each client's number of training samples. Since the datasets used are all public datasets and the hyperparameter settings are explicitly described, our experiments can be easily reproduced on top of a GPU server. We promise to release our open-source codes on GitHub and maintain a project website with detailed documentation for long-term access by other researchers and end-users after the paper is accepted.

Datasets. We following the same strategy in FedOpt (Reddi et al., 2021a) to create a federated version of CIFAR-100 by randomly partitioning the training data among 500 clients, with each client receiving 100 examples. We randomly partition the data to reflect the coarse and fine label structure of CIFAR-100 by using the Pachinko Allocation Method (Li & McCallum, 2006). In the derived client datasets, the label distributions better resemble practical heterogeneous client datasets. We train a modified ResNet-18, where the batch normalization layers are replaced by group normalization layers (Wu & He, 2020). We use two groups in each group normalization layer. Group normalization can lead to significant gains in accuracy over batch normalization in federated settings (Hsieh et al., 2020).

The federated version of EMNIST partitions the digits by their author (Caldas et al., 2018b). The dataset has natural heterogeneity stemming from the writing style of each person. We use a convolutional network for character recognition. The network has two convolutional layers with 3×3 kernels, max pooling, and dropout, followed by a 128 unit dense layer.

Stack Overflow is a language modeling dataset consisting of question and answers from the question and answer site (TensorFlow, 2019). The questions and answers also have associated metadata, including tags. The dataset contains 342,477 unique users which we use as clients. We choose the 10,000 most frequently used words, the 500 most frequent tags and adopt a one-versus-rest classification strategy, where each question/answer is represented as a bag-of-words vector.

Implementation. For four regular federated learning models of FedAvg [1], SCAFFOLD [2], STEM [3], and CLIMB [4], we used the open-source implementation and default parameter settings by the original authors or the Google Research for our experiments. For three federated optimization approaches of MimeLite [5], FedOpt [6], and FedLocal [7], we also utilized the same model architecture as the official implementation provided by the Google Research and used the same datasets to validate the performance of these federated optimization models in all experiments. For other regular federated learning or federated optimization approaches, including MFL and FedLin, to our best knowledge, there are no publicly available open-source implementations on the Internet. All hyperparameters are standard values from the reference works. The above open-source codes from the GitHub are licensed under the MIT License, which only requires preservation of copyright and license notices and includes the permissions of commercial use, modification, distribution, and private use.

For our proposed decoupled adaptive optimization algorithm, we performed hyperparameter selection by performing a parameter sweep on training rounds $\in \{1,500, 2,000, 2,500, 3,000, 3,500, 4,000\}$, momentum parameter $\beta_1 \in \{0.84, 0.86, 0.88, 0.9, 0.92, 0.94\}$, second moment parameter $\beta_2 \in \{0.984, 0.986, 0.988, 0.99, 0.992, 0.994\}$, fuzz factor $\epsilon \in \{0.00001, 0.0001, 0.001, 0.01, 0.1\}$, local iteration number $\in \{1, 2, 5, 10, 20\}$, and learning rate $\eta \in \{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3.3, 10, 33, 100, 333, 1,000\}$. The above search process is often done using validation data in centralized settings. However, such data is often inaccessible in federated settings, especially cross-device settings. Therefore, we tune by selecting the parameters that minimize the average training loss over the last 100 rounds of training. We run 1,500 rounds of training on the EMNIST and Stack Overflow, and 4,000 rounds over the CIFAR-100.

Hyperparameter settings.

Unless otherwise explicitly stated, we used the following default parameter settings in the experiments, as shown in Table 12.

[1]https://github.com/google-research/federated/tree/780767fdf68f2f11814d41bbbfe708274eb6d8b3/optimization

[2]https://github.com/google-research/public-data-in-dpfl

[3]https://papers.neurips.cc/paper/2021/hash/3016a447172f3045b65f5fc83e04b554-Abstract.html

[4]https://openreview.net/forum?id=Xo0lbDt975

[5]https://github.com/google-research/public-data-in-dpfl

[6]https://github.com/google-research/federated/tree/master/optimization

[7]https://github.com/google-research/federated/tree/master/local_adaptivity

Table 12: Hyperparameter Settings

Parameter	**Value**
Training rounds for EMNIST and Stack Overflow	1,500
Training rounds for CIFAR-100	4,000
Momentum parameter β_1	0.9
Second moment parameter β_2	0.99
Client learning rate with SGDM on CIFAR-100	0.03
Client learning rate with Adam on CIFAR-100	0.03
Client learning rate with AdaGrad on CIFAR-100	0.1
Server learning rate with SGDM on CIFAR-100	3.3
Server learning rate with Adam on CIFAR-100	0.3
Server learning rate with AdaGrad on CIFAR-100	0.1
Local iteration number with SGDM on CIFAR-100	5
Local iteration number with Adam on CIFAR-100	5
Local iteration number with AdaGrad on CIFAR-100	5
Fuzz factor with Adam on CIFAR-100	0.1
Fuzz factor with AdaGrad on CIFAR-100	0.1
Client learning rate with SGDM on EMNIST	0.1
Client learning rate with Adam on EMNIST	0.1
Client learning rate with AdaGrad on EMNIST	0.1
Server learning rate with SGDM on EMNIST	1
Server learning rate with Adam on EMNIST	0.1
Server learning rate with AdaGrad on EMNIST	0.1
Local iteration number with SGDM on EMNIST	10
Local iteration number with Adam on EMNIST	10
Local iteration number with AdaGrad on EMNIST	5
Fuzz factor with Adam on EMNIST	0.1
Fuzz factor with AdaGrad on EMNIST	0.1
Client learning rate with SGDM on Stack Overflow	100
Client learning rate with Adam on Stack Overflow	100
Client learning rate with AdaGrad on Stack Overflow	100
Server learning rate with SGDM on Stack Overflow	10
Server learning rate with Adam on Stack Overflow	1
Server learning rate with AdaGrad on Stack Overflow	10
Local iteration number with SGDM on Stack Overflow	5
Local iteration number with Adam on Stack Overflow	1
Local iteration number with AdaGrad on Stack Overflow	5
Fuzz factor with Adam on Stack Overflow	0.00001
Fuzz factor with AdaGrad on Stack Overflow	0.00001