On the Role of Server Momentum in Federated Learning

Jianhui Sun[1*], **Xidong Wu**[2*], **Heng Huang**[3], **Aidong Zhang**[1]

[1]Computer Science, University of Virginia, VA, USA
[2]Electrical and Computer Engineering, University of Pittsburgh, PA, USA
[3]Computer Science, University of Maryland College Park, MD, USA
js9gu@virginia.edu, xidong_wu@outlook.com, heng@umd.edu, aidong@virginia.edu

Abstract

Federated Averaging (FedAvg) is known to experience convergence issues when encountering significant clients system heterogeneity and data heterogeneity. Server momentum has been proposed as an effective mitigation. However, existing server momentum works are restrictive in the momentum formulation, do not properly schedule hyperparameters and focus only on system homogeneous settings, which leaves the role of server momentum still an under-explored problem. In this paper, we propose a general framework for server momentum, that (a) covers a large class of momentum schemes that are unexplored in federated learning (FL), (b) enables a popular stagewise hyperparameter scheduler, (c) allows heterogeneous and asynchronous local computing. We provide rigorous convergence analysis for the proposed framework. To our best knowledge, this is the first work that thoroughly analyzes the performances of server momentum with a hyperparameter scheduler and system heterogeneity. Extensive experiments validate the effectiveness of our proposed framework. Due to page limit, we leave all proofs to the full version https://arxiv.org/abs/2312.12670.

1 Introduction

Federated Averaging (FedAvg) (McMahan et al. 2017), which runs multiple epochs of Stochastic Gradient Descent (SGD) locally in each client and then averages the local model updates once in a while on the server, is probably the most popular algorithm to solve many federated learning (FL) problems, mainly due to its low communication cost and appealing convergence property.

Though it has seen great empirical success, vanilla FedAvg experiences an unstable and slow convergence when encountering *client drift*, i.e., the local client models move away from globally optimal models due to client heterogeneity (Karimireddy et al. 2020). On the server side, FedAvg is in spirit similar to an SGD with a constant learning rate one and updates the global model relying only on the averaged model update from the current round, thus extremely vulnerable to client drift. Note that in non-FL settings, SGD in its vanilla form has long been replaced by some momentum scheme (e.g. heavy ball momentum (SHB) and Nesterov's accelerated gradient (NAG)) in many tasks, as momentum schemes achieve an impressive training time saving and generalization performance boosting compared to competing optimizers (Sutskever et al. 2013; Wilson et al. 2017), which promises a great potential to apply momentum in FL settings as well. Incorporating server momentum essentially integrates historical aggregates into the current update, which could conceptually stabilize the global update against dramatic local drifts.

Though various efforts have been made to understand the role of server momentum in FL, e.g. (Hsu, Qi, and Brown 2019; Rothchild et al. 2020), it is still largely an under-explored problem due to the following reasons:

(1) Lack of diversity in momentum schemes. Most existing server momentum works only focus on SHB (e.g. FedAvgM (Hsu, Qi, and Brown 2019)). It is unclear whether many momentum schemes that outperformed SHB in non-FL settings can also perform better in FL, and there is no unified analysis for momentum schemes other than SHB.

(2) No hyperparameter schedule. Properly scheduling hyperparameters is key to train deep models more efficiently and an appropriate selection of server learning rate η_t is also important in obtaining optimal convergence rate (Yang, Fang, and Liu 2021). Existing works either still employ a constant server learning rate one or consider a η_t schedule that is uncommonly used in practice, e.g., polynomially decay (i.e., $\eta_t \propto \frac{1}{t^\alpha}$) (Khanduri et al. 2021). Moreover, it is known that increasing momentum factor β is also a critical technique in deep model training (Sutskever et al. 2013; Smith, Kindermans, and Le 2018), while to our best knowledge, there is no prior work considering time-varying β in FL.

(3) Ignoring client system heterogeneity. Existing works make unrealistic assumptions on system homogeneity and client synchrony, e.g., clients are sampled uniformly at random, all participating clients synchronize at each round t, and all clients run identical number of local epochs, none of which holds in most cross device FL deployments (Kairouz et al. 2021). System heterogeneity (i.e., the violation of above assumptions), alongside with data heterogeneity, is also a main source client drift (Karimireddy et al. 2020). Thus, ignoring it would provide an incomplete understanding of the role of server momentum.

To address the above limitations, we propose a novel formulation which we refer to as Federated General Momentum (FedGM). FedGM includes the following hyperparameters, learning rate η, momentum factor β, and instant discount fac-

*These authors contributed equally.

tor ν. With different specifications of (η, β, ν), FedGM subsumes the FL version of many popular momentum schemes, most of which have never been explored in FL yet.

We further incorporate a widely used hyperparameter scheduler "constant and drop" (a.k.a. "step decay") in FedGM. We refer to this framework as multistage FedGM. Specifically, with a prespecified set of hyperparameters $\{\eta_s, \beta_s, \nu_s\}_{s=1}^S$ and training lengths $\{T_s\}_{s=1}^S$, the training process is divided into S stages, and at stage s, FedGM with $\{\eta_s, \beta_s, \nu_s\}$ is applied for T_s rounds. Compared to many unrealistic schedule in existing works, "constant and drop" is the de-facto scheduler in most model training (Sutskever et al. 2013; He et al. 2016; Huang et al. 2017). Multistage FedGM is extremely flexible, as it allows the momentum factor to vary stagewise as well, and subsumes single-stage training as a special case. We provide the convergence analysis of multistage FedGM. Our theoretical results reveal why stagewise training can provide empirically faster convergence.

Furthermore, in order to understand how server momentum behaves in the presence of system heterogeneity, we propose a framework that we refer to as Autonomous Multistage FedGM, in which clients can do heterogeneous and asynchronous computing. Specifically, we allow each client to (a) update local models based on an asynchronous view of the global model, (b) run a time-varying, client-dependent number of local epochs, and (c) participate at will. We provide convergence analysis of Autonomous Multistage FedGM. Autonomous Multistage FedGM is a more realistic characterization of real-world cross-device FL applications.

Finally, we conduct extensive experiments that validate, (a) FedGM is a much more capable momentum scheme compared with existing FedAvgM in both with and without system heterogeneity settings; and (b) multistage hyperparameter scheduler further improves FedGM effectively.

Our main contributions can be summarized as follow,

- We propose FedGM, which is a general framework for server momentum and covers a large class of momentum schemes that are unexplored in FL. We further propose Multistage FedGM, which incorporates a popular hyperparameter scheduler to FedGM.
- We show the convergence of multistage FedGM in both full and partial participation settings. We also empirically validate the superiority of multistage FedGM. To our best knowledge, this is the first work that provides convergence analysis of server-side hyperparameter scheduler.
- We propose Autonomous Multistage FedGM, which requires much less coordination between server and workers than most existing algorithms, and theoretically analyze its convergence. Our work is the first to study the interplay between server momentum and system heterogeneity.

The rest of the paper is organized as follows. In Section 2, we formally introduce federated optimization. In Section 3, we introduce Federated General Momentum (FedGM), followed by multistage FedGM and its convergence analysis in Section 4. In Section 5, we introduce Autonomous multistage FedGM and provide its convergence analysis. Section 6 presents the experimental results. Due to the page limit, we leave related work, all proofs, and additional experimental results to extended version https://arxiv.org/abs/2312.12670.

Algorithm 1: FedOPT (Reddi et al. 2020): A Generic Formulation of Federated Optimization

Input: Number of clients n, objective function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$, initialization x_0, Number of communication rounds T, **local** learning rate η_l, **local** number of updates K, **global** hyperparameters $\mathbb{H}$;

for $t \in \{1, ..., T\}$ **do**
  Randomly sample a subset $\mathcal{S}_t$ of clients
  Server sends x_t to subset $\mathcal{S}_t$ of clients
  for *each client* $i \in \mathcal{S}_t$ **do**
    $\Delta_t^i = \textbf{LocalOPT}\,(i, \eta_l, K, x_t)$
  end
  Server aggregates $\Delta_t = \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\Delta_t^i$
  $x_{t+1} = \textbf{ServerOPT}\,(x_t, \Delta_t, \mathbb{H})$
end
return x_T

Algorithm 2: LocalOPT (i, η_l, K, x_t)

Input: client index i, data distribution $\mathcal{D}_i$, **local** learning rate η_l, **local** updating number K, round t, **global** model x_t;

Initialize $x_{t,0}^i \leftarrow x_t$
for $k \in \{0, 1, ..., K-1\}$ **do**
  Randomly sample a batch $\xi_{t,k}^i$ from $\mathcal{D}_i$
  Compute $g_{t,k}^i = \nabla f_i(x_{t,k}^i, \xi_{t,k}^i)$
  Update $x_{t,k+1}^i = x_{t,k}^i - \eta_l g_{t,k}^i$
end
$\Delta_t^i = x_t - x_{t,K}^i$ return Δ_t^i

2 Background: FedOPT and FedAvg

Many FL tasks can be formulated as solving the following optimization problems,

$$\min_{x\in\mathbb{R}^d} f(x) \triangleq \frac{1}{n}\sum_{i=1}^n f_i(x), \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i} f_i(x, \xi). \tag{1}$$

where n is the total number of clients, x is the model parameter with d dimension. Each client i has a local data distribution $\mathcal{D}_i$ and a local objective function $f_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i} f_i(x, \xi)$. The global objective function is the averaged objective among all clients. $\mathcal{D}_i$ can be very different from $\mathcal{D}_j$ when $i \neq j$.

FedAvg (McMahan et al. 2017) and its variants are a special case of a more flexible formulation, **FedOPT** (Reddi et al. 2020), which is formalized in Algorithm 1. Suppose the total number of rounds is T, and the global model parameter is $\{x_t\}_{t=1}^T$. At each round t, the server randomly samples a subset of clients $\mathcal{S}_t$ and sends the global model x_t to them. Upon receiving x_t, each participating client would do **LocalOPT** (Algorithm 2). Specifically, each client i would initialize their

local model at x_t, run K steps of local SGD with local η_l and updated locally to $x^i_{t,K}$. The client then sends the local model update $\Delta^i_t = x_t - x^i_{t,K}$ back to the server. The server aggregates by averaging, i.e. $\Delta_t = \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\Delta^i_t$, and then triggers server-side optimization **ServerOPT**, which takes x_t, Δ_t, and a hyperparameter set $\mathbb{H}$ as input, and outputs the next round's global model x_{t+1}.

In FedAvg, ServerOPT is simply $x_{t+1} = x_t - \Delta_t$, which is in spirit identical to SGD with a constant learning rate one if viewing Δ_t as a pseudo gradient.

3 FedGM: Federated Learning with General Momentum Acceleration

Partially due to its equivalence of constant learning rate SGD, FedAvg has two main limitations, (a) it is extremely vulnerable to client drift, as FedAvg relies entirely on its current aggregate Δ_t and ignores historical directions; (b) FedAvg may not be the best option in many applications, e.g. training large-scale vision or language models (Devlin et al. 2018; Dosovitskiy et al. 2021) where its counterpart SGD is known to be inferior to momentum or adaptive optimizers in non-FL settings (Wilson et al. 2017; Zhang et al. 2020).

Note that in FedOPT, ServerOPT could in principle be any type of gradient-based optimizers. In non-FL settings, the momentum scheme is known to not only exhibit convincing acceleration in training, it has also achieved better generalizability in many tasks than adaptive optimizers like Adam (Wilson et al. 2017; Cutkosky and Mehta 2020), which provides a strong motivation to incorporate server momentum.

Moreover, server-side momentum integrates historical aggregates into the current update and thus could potentially make the global model more robust to drastic local drifts.

Existing server momentum works mostly focus on one specific type of momentum, i.e. stochastic heavy ball momentum (SHB) (Hsu, Qi, and Brown 2019; Rothchild et al. 2020; Khanduri et al. 2021), while ignoring many other momentum schemes that outperform SHB in many non-FL settings.

In order to systematically understand the role of server momentum schemes in FL, we propose a new algorithm which we refer to as Federated General Momentum (FedGM). FedGM replaces the ServerOPT $x_{t+1} = x_t - \Delta_t$ in FedAvg with the following,

$$d_{t+1} = (1-\beta)\Delta_t + \beta d_t, \quad h_{t+1} = (1-\nu)\Delta_t + \nu d_{t+1}, \\ x_{t+1} = x_t - \eta h_{t+1}. \tag{2}$$

where the hyperparameter set $\mathbb{H} = \{\eta, \beta, \nu\}$. η is server learning rate, β and ν are two hyperparameters which we call momentum factor and instant discount factor.

By setting ν as 0, FedGM becomes FedAvg with two-sided learning rates (Yang, Fang, and Liu 2021), i.e., choices of η other than 1 is allowed, which we refer to as FedSGD.

By setting $\nu = 1$, FedGM becomes FedAvgM (Hsu, Qi, and Brown 2019) (or FedSHB), which essentially applies server SHB, i.e. we update the model by a "momentum buffer" d_{t+1}. β controls how slowly the momentum buffer is updated. FedGM could be interpreted as a ν-weighted average of the FedAvgM update step and the plain FedAvg update step. ν is thus referred to as instant discount factor.

Algorithm 3: Multistage FedGM

Input:
Initialization x_0, number of rounds T, **local** learning rate η_l, **local** updating number K;
Number of stages S, stage lengths $\{T_s\}_{s=1}^S$;
Stagewise hyperparameters $\{\eta_s, \beta_s, \nu_s\}_{s=1}^S$;

for $s \in \{1, ..., S\}$ **do**
  for t *in stage* s **do**
    Randomly sample a subset $\mathcal{S}_t$ of clients
    Server sends x_t to subset $\mathcal{S}_t$ of clients
    for *each client* $i \in \mathcal{S}_t$ **do**
      $\Delta^i_t = \text{LocalOPT}\,(i, \eta_l, K, x_t)$
    end
    Server aggregates $\Delta_t = \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\Delta^i_t$
    $d_{t+1} = (1-\beta_s)\Delta_t + \beta_s d_t$
    $h_{t+1} = (1-\nu_s)\Delta_t + \nu_s d_{t+1}$
    Update $x_{t+1} = x_t - \eta_s h_{t+1}$
  end
end
return x_T

FedGM leverages the general formulation of QHM (Ma and Yarats 2019; Sun et al. 2021, 2022) and is much more general than just FedAvg and FedAvgM. It subsumes many other momentum variants. For example, if $\nu = \beta$, FedGM becomes a new algorithm which can be naturally referred to as FedNAG, i.e. application of the popular optimizer Nesterov's accelerated gradient (NAG) to FL. Specifically, we update model by $x_{t+1} = x_t - \eta\left[(1-\beta)\Delta_t + \beta d_{t+1}\right]$, where d_{t+1} is the momentum buffer. FedGM could further recover the FL version of many other momentum schemes, e.g., SNV (Lessard, Recht, and Packard 2014), PID (An et al. 2018), ASGD (Kidambi et al. 2018), and Triple Momentum (Van Scoy, Freeman, and Lynch 2018), with different η, β, ν. Therefore, FedGM describes a family of momentum schemes, most of which have never been studied in FL.

4 Multistage FedGM and Convergence

4.1 Proposed Algorithm: Multistage FedGM

One major limitation in FedGM (2) is that all server-side hyperparameters are held constant, which are inconsistent with common practice. Adaptively adjusting hyperparameters throughout the training is key to the success of many optimizers. Learning rate scheduling has been thoroughly studied in non-FL settings, e.g., (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Goyal et al. 2017; Smith 2017). Scheduling other hyperparameters (e.g. momentum factor and batch size) is also shown to be very effective in many settings. For example, (Sutskever et al. 2013; Smith and Le 2018; Smith, Kindermans, and Le 2018) show a slowly increasing schedule for the momentum factor β is crucial in training deep models faster.

We focus on a simple yet effective hyperparameter sched-

uler, "constant and drop" (a.k.a. "step decay"). In its non-FL SGD version (a.k.a. multistage SGD), with a prespecified set of learning rates $\{\eta_s\}_{s=1}^S$ and training lengths $\{T_s\}_{s=1}^S$ (measured by number of iterations/epochs), the training process is divided into S stages, and SGD with η_s is applied for T_s iterations/epochs at s-th stage, where $\{\eta_s\}_{s=1}^S$ is usually a non-increasing sequence [1]. We concentrate on "constant and drop" as it is the de-facto learning rate scheduler in most large-scale neural networks (Krizhevsky, Sutskever, and Hinton 2012; Sutskever et al. 2013; He et al. 2016; Huang et al. 2017), and has been theoretically shown to achieve near-optimal rate in non-FL settings (Ge et al. 2019; Wang, Magnússon, and Johansson 2021).

The intuition behind "constant and drop" is straightforward: a large learning rate is held constant for a reasonably long period to take advantage of faster convergence until it saturates, and then the learning rate is dropped by a constant factor for more refined training.

We extend "constant and drop" to FedGM in Algorithm 3, which we refer to as Multistage FedGM. In Multistage FedGM (Algorithm 3), each stage has length T_s ($T = \sum_{s=1}^S T_s$), and has its triplet of stagewise hyperparameters $\{\eta_s, \beta_s, \nu_s\}_{s=1}^S$. The convergence analysis in Sec 4.2 also applies to single-stage FedGM by $S = 1$.

To our best knowledge, there is no prior work giving definitive theoretical guarantee or empirical performances of any hyperparameter schedule in FL, especially considering multistage FedGM is an extremely flexible framework that allows both learning rate and momentum factor to evolve.

4.2 Convergence Analysis of Multistage FedGM

We now analyze the convergence of Algorithm 3 under both full and partial participation settings.

We aim to optimize objective (1). Each local loss f_i (and therefore f) may be general nonconvex function. We study the general *non-i.i.d.* setting, i.e. $\mathcal{D}_i \neq \mathcal{D}_j$ when $i \neq j$. We state the assumptions that are needed in the analysis.

Assumption 1 (Smoothness)**.** Each local loss $f_i(x)$ is differentiable and has L-Lipschitz continuous gradients, i.e., $\forall x, x' \in \mathbb{R}^d$, we have $\|\nabla f_i(x) - \nabla f_i(x')\| \leq L \|x - x'\|$. And $f^* \triangleq \min_x f(x)$ exists, i.e., $f^* > -\infty$.

Assumption 2 (Bounded Local Variance)**.** $\forall t, i$, LocalOPT can access an unbiased estimator $g_{t,k}^i = \nabla f_i(x_{t,k}^i, \xi_{t,k}^i)$ of true gradient $\nabla f_i(x_{t,k}^i)$, where $g_{t,k}^i$ is the stochastic gradient estimated with minibatch $\xi_{t,k}^i$. And each stochastic gradient on the i-th client has a bounded local variance, i.e., we have $\mathbb{E}\left[\left\|g_{t,k}^i - \nabla f_i(x_{t,k}^i)\right\|^2\right] \leq \sigma_l^2$.

Assumption 3 (Bounded Global Variance)**.** The local loss $\{f_i(x)\}$ across all clients have bounded global variance, i.e., $\forall x$, we have $\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \sigma_g^2$.

Assumption 1-3 are standard assumptions in nonconvex optimization and FL research, and have been universally adopted in most existing works (Reddi, Kale, and Kumar 2018; Li et al. 2020b; Reddi et al. 2020; Yang, Fang, and Liu 2021; Wang, Lin, and Chen 2022; Wu et al. 2023a,b). $\sigma_g^2 = 0$ in Assumption 3 corresponds to the *i.i.d.* setting. And we do not require the restrictive bounded gradient assumption (Reddi, Kale, and Kumar 2018; Avdiukhin and Kasiviswanathan 2021; Wang, Lin, and Chen 2022).

Recall $T = \sum_{s=1}^S T_s$ is the number of rounds. Denote the expected gradient square as $\{\mathcal{G}_t \triangleq \mathbb{E}\left[\|\nabla f(x_t)\|^2\right]\}_{t \leq T}$. Define the average expected gradient square at s-th stage as $\bar{\mathcal{G}}_s \triangleq \frac{1}{T_s}\sum_{t=T_1+\cdots+T_{s-1}+1}^{T_1+\cdots+T_s} \mathcal{G}_t$ and the average expected gradient square across S stages as $\bar{\mathcal{G}} \triangleq \frac{1}{S}\sum_{s=1}^S \bar{\mathcal{G}}_s$. Bounding $\bar{\mathcal{G}}$ generalizes from bounding $\frac{1}{T}\sum_{t=1}^T \mathbb{E}\left[\|\nabla f(x_t)\|^2\right]$ in single-stage to multistage setting.

To reflect the common hyperparameter scheduling practice that is adopted by existing works e.g. (Sutskever et al. 2013; Smith, Kindermans, and Le 2018; Liu, Gao, and Yin 2020), We request the stagewise hyperparameters fulfill the following constraints,

$$\eta_S \leq \eta_{S-1} \leq \cdots \leq \eta_1 \quad \beta_1 \leq \beta_2 \leq \cdots \leq \beta_S < 1$$
$$W_1 \equiv \frac{\eta_s \beta_s \nu_s}{1-\beta_s} \quad \text{and} \quad W_2 \equiv T_s \eta_s \tag{3}$$

where W_1 and W_2 are two constants. Constraint (3) essentially requires learning rate to be non-increasing and momentum factor to be non-decreasing at a similar rate, which is consistent with common practice, e.g. for SHB and NAG, (Sutskever et al. 2013; Smith, Kindermans, and Le 2018; Liu, Gao, and Yin 2020) propose a scheduler for β to increase and close to 1 for faster convergence. And it is also natural for (3) to require $T_s\eta_s$ as a constant. As the learning rate is decaying, more rounds in later stages are necessary for sufficient refined training.

We now state the convergence guarantee of the multistage training regime in FL framework.

Full Participation If all clients are required to participate in each round, i.e. $\mathcal{S}_t = \{1, 2, \ldots, n\}$, we have,

Theorem 4.1. *We optimize $f(x)$ with Algorithm 3 (Full Participation) under Assumptions 1-3. Denote $\bar{\eta} \triangleq \frac{1}{S}\sum_{s=1}^S \eta_s$ as the average server learning rate and $C_\eta \triangleq \frac{\eta_1}{\eta_S}$. Under the condition [2] $\eta_l \leq \min\left\{\frac{1}{8KL}, \frac{1}{KSC_\eta\left(L\bar{\eta}+1+L^2W_1^2C_\eta\right)}\right\}$, we would have:*

$$\bar{\mathcal{G}} \triangleq \frac{1}{S}\sum_{s=1}^S \frac{1}{T_s} \sum_{t=T_0+\cdots+T_{s-1}}^{T_0+\cdots+T_s-1} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right]$$
$$\leq \frac{64}{17}\frac{f(x_0)-f^*}{SW_2\eta_l K} + \Psi_l\sigma_l^2 + \Psi_g\sigma_g^2$$

where $\Psi_l \triangleq \frac{32}{17}\frac{L^2W_1^2T\bar{\eta}\eta_l}{nW_2} + \frac{32}{17}\frac{L\bar{\eta}\eta_l}{n} + \frac{32}{17}\frac{\eta_l}{n} + \frac{160}{17}\eta_l^2L^2K$, and $\Psi_g \triangleq \frac{960}{17}\eta_l^2L^2K^2$.

[1] The name "constant and drop" refers to learning rate is dropped by some constant factor after each stage.

[2] The condition could be fulfilled by typical value assignment, and would recover the typical $\eta_l \leq \min\left\{\frac{1}{8KL}, \frac{1}{KL\eta}\right\}$ constraint in FedAvg analysis (Yang, Fang, and Liu 2021), by setting $S = 1$.

Corollary 4.2 (Convergence Rate of Multistage FedAvg)**.** *Suppose $\nu_s = 0$, i.e., the FedAvg algorithm that allows learning rate vary across S stages. By setting $\bar{\eta} = \Theta\left(\sqrt{nK}\right)$ and $\eta_l = \Theta\left(\frac{1}{\sqrt{TK}}\right)$, $W_2 = \Theta\left(\frac{T\sqrt{nK}}{S}\right)$, i.e. $T\bar{\eta}$ equally divided into S stages. $W_1 = 0$ as $\nu_s = 0$. Suppose T is sufficiently large, i.e. $T \geq nK$, we have a $\mathcal{O}\left(\frac{1}{\sqrt{TKn}}\right)$ convergence rate.*

Remark 4.3 (Why Multistage Helps?). Corollary 4.2 indicates multistage FedAvg recovers the best-known rate for general FL nonconvex optimization approaches, e.g. SCAFFOLD (Karimireddy et al. 2020) and FedAdam (Reddi et al. 2020). Note single-stage FedAvg with two-sided learning rates also achieves the same rate (Yang, Fang, and Liu 2021). However, we do observe multistage FedAvg empirically converges much better than single-stage. We can obtain insights from Theorem 4.1 why multistage helps. We note that Ψ_l is only related to average learning rate $\bar{\eta}$ (instead of initial learning rate η_1). At initial rounds, the first term with $f(x_0) - f^*$ dominates, and thus we could select a relatively large η_1 to ensure a more dramatic decay of this term. At later rounds, when $f(x_t) - f^*$ plateaus, we could enable smaller learning rate to control $\bar{\eta}$. Thus, Theorem 4.1 indicates a less stringent reliance on η_1, which enables us to flexibly select suitable η depending on which training stage we are in, that can still guarantee convergence.

Corollary 4.4 (Convergence Rate of Multistage FedGM)**.** *Suppose $S > 1$, i.e. the multistage regime, by setting $\bar{\eta} = \Theta\left(\sqrt{nK}\right)$, $\eta_l = \Theta\left(\frac{1}{\sqrt{TK}}\right)$, $W_2 = \Theta\left(\frac{T\sqrt{nK}}{S}\right)$. Let $W_1^2 = \mathcal{O}\left(\frac{\sqrt{nK}}{S}\right)$ [3]. When $T \geq Kn$, we have a $\mathcal{O}\left(\frac{1}{\sqrt{TKn}}\right)$ convergence rate.*

Remark 4.5 (Why Momentum Helps?). We attribute the empirically superior performances of momentum to two reasons. (a) When clients are dynamically heterogeneous, historical gradient information has regularization effect to avoid the search direction from going wild. (b) Server learning rate η acts like a multiplier to client learning rate η_l in FedAvg, i.e. $\eta > 1$ effectively enhances the reliance on current round gradient. Due to the same reason as in (a), such reliance can harm convergence. In contrast, in FedGM, β and ν act as a buffer that could to some extent absorb the impact from a large η. We empirically observe in Appendix, with same η_l, FedGM could sustain a much larger η, while FedAvg diverges very easily with a moderately large η.

Partial Participation Full participation rarely holds in reality, thus we further analyze multistage FedGM in partial participation setting [4].

Theorem 4.6. *We optimize $f(x)$ with Algorithm 3 (Partial Participation) under Assumptions 1-3. Denote $\bar{\eta}$ and C_η as in Theorem 4.1. Under the condition $\eta_l \leq \frac{1}{8KL}$, and $\eta_l\left(C_\eta + L\bar{\eta}C_\eta + L^2W_1^2C_\eta\right)SK \leq \min\left\{\frac{m(n-1)}{n(m-1)}, \frac{17m}{282}\right\}$, we would have:*

$$\bar{\mathcal{G}} \leq \frac{64}{17}\frac{f(z_0) - f^*}{SW_2\eta_l K} + \Psi_l\sigma_l^2 + \Psi_g\sigma_g^2$$

where $\Psi_l \triangleq \frac{\eta_l}{m}\Phi + \frac{15(n-m)K^2L^3\eta_l^3}{m(n-1)}\Phi + \frac{160}{17}\eta_l^2L^2K$, $\Psi_g \triangleq \frac{90(n-m)K^3L^3\eta_l^3}{m(n-1)}\Phi + \frac{3\eta_l(n-m)K}{m(n-1)}\Phi + \frac{960}{17}\eta_l^2L^2K^2$, and $\Phi \triangleq \frac{32T\bar{\eta}+32LT\hat{\eta}^2+32L^2W_1^2T\bar{\eta}}{17W_2}$.

Corollary 4.7 (Convergence Rate of Multistage FedGM)**.** *Suppose $S > 1$, i.e. the multistage regime, by setting $\bar{\eta} = \Theta\left(\sqrt{mK}\right)$, $\hat{\eta}^2 = \Theta\left(mK\right)$, $\eta_l = \Theta\left(\frac{1}{\sqrt{TK}}\right)$, $W_2 = \Theta\left(\frac{T\sqrt{mK}}{S}\right)$ and $W_1^2 = \mathcal{O}\left(\sqrt{mK}\right)$, we have convergence rate as $\mathcal{O}\left(\sqrt{\frac{K}{Tm}}\right)$.*

Remark 4.8. $\mathcal{O}\left(\sqrt{\frac{K}{Tm}}\right)$ recovers the best known convergence rate for FL nonconvex optimization (Yang, Fang, and Liu 2021). Similar to Remark 4.3, Theorem 4.6 shows an reliance on average learning rate, which explains why multistage scheme helps empirically. $\mathcal{O}\left(\sqrt{\frac{K}{Tm}}\right)$ indicates a slowdown effect from more local computation, which is supported by some existing works (Li et al. 2020b), while others observe a different effect of K (Lin et al. 2020). The exact impact of K on convergence warrants further investigation.

5 Momentum with System Heterogeneity

5.1 Autonomous Multistage FedGM

For a simplified abstraction of real world settings, most FL algorithms make the assumption that, all clients synchronize with the same global model and they conduct identical number of local updates at any given round. Though the assumption has been adopted in most existing works (McMahan et al. 2017; Hsu, Qi, and Brown 2019; Li et al. 2020a; Karimireddy et al. 2020; Reddi et al. 2020; Wang, Lin, and Chen 2022), it rarely holds in reality.

In light of the limitations of existing works, we propose a general framework called Autonomous Multistage FedGM that enables the following three features, i.e. **heterogeneous local computing**, **asynchronous aggregation**, and **flexible client participation**, which is formalized in Algorithm 4.

Autonomous Multistage FedGM could effectively mitigate straggler effect and poor convergence issue in highly heterogeneous cross-device deployments. We leave a more detailed discussion of Algorithm 4 to Appendix due to space limit.

Specifically, in Autonomous Multistage FedGM, the client decides when to participate in the training, and idling between rounds or even completely unavailable are both allowed. Once it decides to participate at round t, it retrieves current global model x_μ from the server and conduct $K_{t,i}$ local steps to update to $x^i_{\mu,K_{t,i}}$. Note in vanilla FedAvg, $K_{t,i} = K$ for any i and t. In contrast, we allow $K_{t,i}$ to be time-varying and

[3] It holds by setting an infinitesimal β or ν at early stages when η is large, but β or ν can go to 1 when η is reduced to $o\left(\frac{\sqrt[4]{nK}}{\sqrt{S}}\right)$.

[4] In each round t, the server samples a subset of clients $\mathcal{S}_t$ (suppose $|\mathcal{S}_t| = m < n$) uniformly at random without replacement, i.e. $\mathbb{P}\{i \in \mathcal{S}_t\} = \frac{m}{n}$ and $\mathbb{P}\{i, j \in \mathcal{S}_t\} = \frac{m(m-1)}{n(n-1)}$.

Algorithm 4: Autonomous Multistage FedGM

Input: Same as Algorithm 3

1 **for** $s \in \{1, ..., S\}$ **do**
2 **for** t *in stage* s **do**
3 **At Each Client (Concurrently)**
4 Once decided to participate in the training, retrieve x_μ from the server and its timestamp, set $x^i_{\mu,0} = x_\mu$.
5 Select a number of local steps $K_{t,i}$, which is time-varying and device-dependent.
6 $\Delta^i_\mu = \textbf{LocalOPT}\,(i, \eta_l, K_{t,i}, x_\mu)$
7 Normalize and send $\Delta^i_\mu = \frac{\Delta^i_\mu}{K_{t,i}}$
8 **At Server (Concurrently)**
9 Collect m local updates $\{\Delta^i_{t-\tau_{t,i}}, i \in \mathcal{S}_t\}$ returned from the clients to form set $\mathcal{S}_t$, where $\tau_{t,i}$ is the random delay of the client i's local update, $i \in \mathcal{S}_t$
10 Aggregate $\Delta_t = \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t} \Delta^i_{t-\tau_{t,i}}$
11 $d_{t+1} = (1-\beta_s)\Delta_t + \beta_s d_t$
12 $h_{t+1} = (1-\nu_s)\Delta_t + \nu_s d_{t+1}$
13 Update $x_{t+1} = x_t - \eta_s h_{t+1}$
14 **end**
15 **end**
16 return x_T

device-dependent. The client then normalizes the model update by $K_{t,i}$ to avoid model biased towards clients with more local updates. Concurrently, the server collects the model updates from the clients. As every client may participate in training at a different round, the collected model update $\Delta^i_{t-\tau_{t,i}}$ may be from a historic timestamp, i.e. $\tau_{t,i}$ away from current time t. The server triggers global update whenever it collects m model updates and we denote the set of m responsive clients as $\mathcal{S}_t$. The global update is same as multistage FedGM (i.e. Lines 11-13). Note that server optimization is concurrent with clients, i.e., the global update happens whenever m model updates are collected, regardless of whether there are still some clients conducting local computation, thus ensuring there is no straggler.

Autonomous multistage FedGM, i.e. Algorithm 4, will recover multistage FedGM, i.e. Algorithm 3, if we set $K_{t,i} = K$ and $\tau_{t,i} = 0$ for $\forall t, i$. Please note that varying $K_{t,i}$ and nonzero $\tau_{t,i}$ bring nontrivial extra complexity to the theoretical analysis as can be seen in our proof.

5.2 Convergence Analysis

We state the convergence guarantee of autonomous multistage FedGM as follows,

Theorem 5.1. *We optimize $f(x)$ with Algorithm 4 under assumptions 1-3. Suppose the maximum delay is bounded, i.e. $\tau_{t,i} \le \tau < \infty$ for any $i \in \mathcal{S}_t$ and $t \in \{0, 1, \ldots, T-1\}$. Under the condition $\eta_l \le \min\left\{\frac{1}{8K_{t,max}L}, \sqrt{\frac{1}{120L^2C_\eta\tau K^2_{t,max}}}\right\}$, where $K_{t,max} = \max_{i\in\mathcal{S}_t} K_{t,i}$. And further assume each client is included in $\mathcal{S}_t$ with probability $\frac{m}{n}$ uniformly and independently. With necessary abbreviation for ease of notation [5], we would have:*

$$\bar{\mathcal{G}} \le \frac{4\left(f(x_0) - f^*\right)}{SW_2\eta_l} + \Phi_l\sigma_l^2 + \Phi_g\sigma_g^2$$

$\Phi_l \triangleq \frac{20\eta_l^2L^2T\bar{\eta}}{W_2}\phi_1 + \frac{4L^2\tau^2\hat{\eta}^3\eta_l^2T}{mW_2}\phi_3 + \frac{2L^2W_1^2\bar{\eta}\eta_lT}{mW_2}\phi_3 + \frac{2\bar{\eta}\eta_lT}{mW_2}\phi_3 + \frac{2L\hat{\eta}^2\eta_l}{mW_2}\phi_3$, *and* $\Phi_g \triangleq \frac{120\eta_l^2L^2T\bar{\eta}\phi_2}{W_2}$.

Corollary 5.2 (Convergence Rate)**.** *Suppose an identical K for all t and i. By appropriately setting $\bar{\eta}$, η_l, W_1, W_2, we have the convergence rate as, $\mathcal{O}\left(\frac{1}{\sqrt{mKT}}\right) + \mathcal{O}\left(\frac{\tau^2}{T}\right) + \mathcal{O}\left(\frac{K^2}{T}\right)$.*

Remark 5.3. Corollary 5.2 indicates τ brings a slowdown in convergence. Fortunately, with a sufficiently large T (e.g. $T \ge mK^5$) and a manageable τ (e.g. $\tau \le \frac{T^{\frac{1}{4}}}{(mK)^{\frac{1}{4}}}$), autonomous multistage FedGM obtains a $\mathcal{O}\left(\frac{1}{\sqrt{mKT}}\right)$ rate. Note that we make an additional assumption that each client is included in $\mathcal{S}_t$ with probability $\frac{m}{n}$ uniformly and independently, which is necessary as the following Corollary 5.4 indicates if without such assumption, the rate has a non-

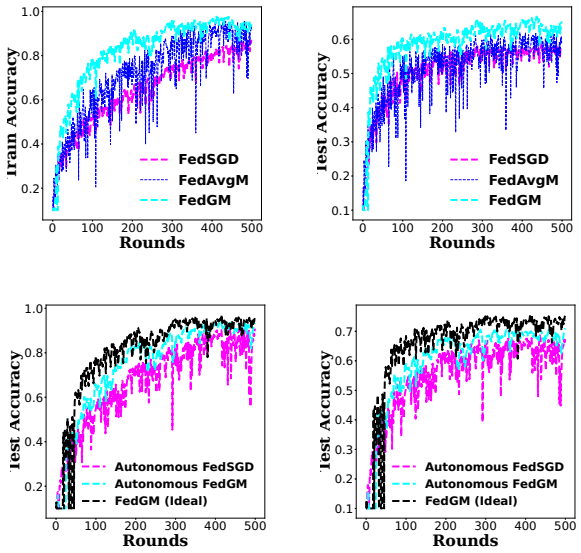

Figure 1: 1(a) Training and 1(b) Testing Curves for FedGM (ResNet on CIFAR-10). FedGM outperforms FedAvg/FedAvgM. 1(c) Training and 1(d) Testing for Autonomous FedGM (ResNet on CIFAR-10).

[5] We denote $\bar{\eta} \triangleq \frac{1}{S}\sum_{s=0}^{S-1}\eta_s$ (average server learning rate), $\hat{\eta}^2 \triangleq \frac{1}{S}\sum_{s=0}^{S-1}\eta_s^2$, $\hat{\eta}^3 \triangleq \frac{1}{S}\sum_{s=0}^{S-1}\eta_s^3$, $\frac{1}{K_t} = \frac{1}{m}\sum_{i\in\mathcal{S}_t}\frac{1}{K_{t,i}}$, $\bar{K}_t \triangleq \frac{1}{m}\sum_{i\in\mathcal{S}_t}K_{t,i}$, $\hat{K}_t^2 \triangleq \frac{1}{m}\sum_{i\in\mathcal{S}_t}K_{t,i}^2$, $\phi_1 \triangleq \frac{1}{T}\sum_{t=0}^{T-1}\bar{K}_t$, $\phi_2 \triangleq \frac{1}{T}\sum_{t=0}^{T-1}\hat{K}_t^2$, and $\phi_3 \triangleq \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{K_t}$, for ease of notation.

convergent $\mathcal{O}\left(\sigma_g^2\right)$ term that we cannot avoid (the lower bound is $\Omega\left(\sigma_g^2\right)$).

Corollary 5.4 (Convergence Rate w/o Uniform Sampling Assumption)**.** *Suppose an identical K for all t and i. By appropriately setting $\bar{\eta}$, η_l, W_1, W_2, we have the convergence rate as, $\mathcal{O}\left(\frac{1}{\sqrt{mKT}}\right) + \mathcal{O}\left(\frac{\tau^2}{T}\right) + \mathcal{O}\left(\frac{K^2}{T}\right) + \mathcal{O}\left(\sigma_g^2\right)$, and the non-vanishing $\mathcal{O}\left(\sigma_g^2\right)$ is unavoidable.* [6]

6 Experimental Results

In this section, we present empirical evidence to verify our theoretical findings. We train ResNet (He et al. 2016) and VGG (Simonyan and Zisserman 2015) on CIFAR10 (Krizhevsky 2009). To simulate data heterogeneity in CIFAR-10, we impose label imbalance across clients, i.e. each client is allocated a proportion of the samples of each label according to a Dirichlet distribution (Hsu, Qi, and Brown 2019; Yurochkin et al. 2019). The concentration parameter $\alpha > 0$ indicates the level of *non-i.i.d.*, with smaller α implies higher heterogeneity, and $\alpha \to \infty$ implies *i.i.d.* setting. Unless specified otherwise, we have 100 clients in all experiments, and the partial participation ratio is 0.05, i.e., 5 out of 100 clients are picked in each round, *non-i.i.d.* is $\alpha = 0.5$, and local epoch is 3. We defer many more results and details of hyperparameter settings to Appendix.

6.1 Results on FedGM

Figure 1 shows the results for ResNet on CIFAR-10 with FedGM, FedAvgM, and FedAvg. We perform grid search over $\eta \in \{0.5, 1.0, 1.5, \ldots, 5.0\}$, $\beta \in \{0.7, 0.9, 0.95\}$, and $\nu \in \{0.7, 0.9, 0.95\}$. We report their respective best results in Figure 1. We observe that though FedAvgM converges faster than FedAvg, it is only marginally better in terms of testing. FedGM, in contrast, outperforms FedAvgM and FedAvg in both measures. Therefore, a general momentum, instead of only SHB, is critical empirically. We analyze possible reasons and leave more results with VGG and different heterogeneity levels α to Appendix.

6.2 Results on Multistage FedGM

Figure 2 shows the results for ResNet on CIFAR-10 with multistage vs. single-stage FedGM. The two black vertical lines at round 143 and 429 mark the end of 1st/2nd stage. For multistage FedGM, $(\eta_1 = 2.0, \eta_2 = 1.0, \eta_3 = 0.5)$, the β also changes according to Eq. 3. From Figure 2, we observe multistage FedGM is better than single-stage FedGM, no matter what constant η it takes. Specifically, at first stage, $\eta_1 = 2.0$ makes the training curve fluctuate dramatically, but later into 2nd/3rd stage, the training stabilizes with smaller η_2 and η_3. Multistage FedGM achieves a balance between early exploration and late exploitation. Multistage is also superior to its counterpart in testing. We leave more experiments to Appendix.

[6] We informally state Corollary 5.4 due to page limit, please refer to Appendix for a formal statement.

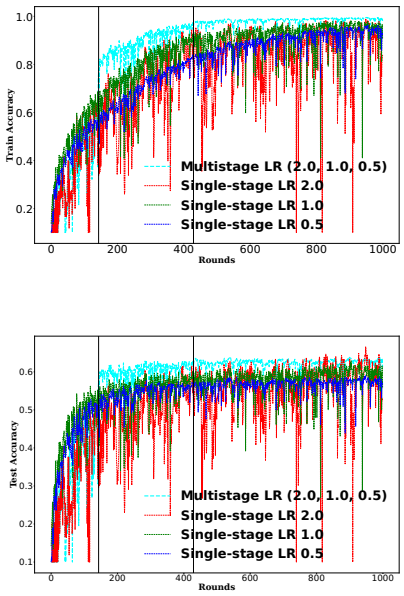

Figure 2: 2(a) Training and 2(b) Testing Curves for Multistage FedGM vs. Single-stage FedGM.

6.3 Results on Autonomous FedGM

Figure 1 shows the results for ResNet on CIFAR-10 with Autonomous FedGM (& FedAvg). Please refer to Appendix for detailed settings. We perform a grid search as in Section 6.1. We report their respective best curves. We plot an ideal FedGM (i.e. synchronous and identical local epochs) as reference line. We could observe Autonomous FedGM outperforms Autonomous FedAvg with system heterogeneity. Though Autonomous FedGM suffers a slowdown compared to the ideal FedGM, it is within a small margin, which supports our theory in Corollary 5.2 and validates the effectiveness of Autonomous FedGM. We leave more experiments to Appendix.

7 Conclusion

This paper systematically studied how the server momentum could help alleviate client drift that arises from both data heterogeneity and system heterogeneity. We demonstrated the critical role of momentum schemes and proper hyperparameter schedule by providing a rigorous convergence analysis and extensive empirical evidence, which pave a way for more widely and disciplined use of server momentum in the federated learning research community.

Acknowledgments

This work was partially supported by NSF 2217071, 2213700, 2106913, 2008208, 1955151 at UVA. This work was partially supported by NSF IIS 2347592, 2348169, 2348159, 2347604, CNS 2347617, CCF 2348306, DBI 2405416 at Pitt and UMD.

References

An, W.; Wang, H.; Sun, Q.; Xu, J.; Dai, Q.; and Zhang, L. 2018. A PID Controller Approach for Stochastic Optimization of Deep Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8522–8531.

Avdiukhin, D.; and Kasiviswanathan, S. 2021. Federated Learning under Arbitrary Communication Patterns. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 425–435. PMLR.

Cutkosky, A.; and Mehta, H. 2020. Momentum Improves Normalized SGD. In *International Conference on Machine Learning*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Ge, R.; Kakade, S. M.; Kidambi, R.; and Netrapalli, P. 2019. *The Step Decay Schedule: A near Optimal, Geometrically Decaying Learning Rate Procedure for Least Squares*. Red Hook, NY, USA: Curran Associates Inc.

Goyal, P.; Dollár, P.; Girshick, R. B.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR*, abs/1706.02677.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hsu, T.-M. H.; Qi; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *ArXiv*, abs/1909.06335.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

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.; Konecný, 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. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*, 14(1-2): 1–210.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 5132–5143. PMLR.

Khanduri, P.; Sharma, P.; Yang, H.; Hong, M.; Liu, J.; Rajawat, K.; and Varshney, P. 2021. Stem: A stochastic two-sided momentum algorithm achieving near-optimal sample and communication complexities for federated learning. *Advances in Neural Information Processing Systems*, 34.

Kidambi, R.; Netrapalli, P.; Jain, P.; and Kakade, S. M. 2018. On the insufficiency of existing momentum schemes for Stochastic Optimization. *CoRR*, abs/1803.05591.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. 1097–1105.

Lessard, L.; Recht, B.; and Packard, A. 2014. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. *SIAM Journal on Optimization*, 26.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020a. Federated Optimization in Heterogeneous Networks. In Dhillon, I.; Papailiopoulos, D.; and Sze, V., eds., *Proceedings of Machine Learning and Systems*, volume 2, 429–450.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. In *International Conference on Learning Representations*.

Lin, T.; Stich, S. U.; Patel, K. K.; and Jaggi, M. 2020. Don't Use Large Mini-batches, Use Local SGD. In *ICLR - International Conference on Learning Representations*.

Liu, Y.; Gao, Y.; and Yin, W. 2020. An Improved Analysis of Stochastic Gradient Descent with Momentum. arXiv:2007.07989.

Ma, J.; and Yarats, D. 2019. Quasi-hyperbolic momentum and Adam for deep learning. In *International Conference on Learning Representations*.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*.

Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; and McMahan, H. B. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Reddi, S. J.; Kale, S.; and Kumar, S. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

Rothchild, D.; Panda, A.; Ullah, E.; Ivkin, N.; Stoica, I.; Braverman, V.; Gonzalez, J.; and Arora, R. 2020. FetchSGD: Communication-Efficient Federated Learning with Sketching. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Smith, L. N. 2017. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 464–472.

Smith, S.; and Le, Q. V. 2018. A Bayesian Perspective on Generalization and Stochastic Gradient Descent.

Smith, S. L.; Kindermans, P.-J.; and Le, Q. V. 2018. Don't Decay the Learning Rate, Increase the Batch Size. In *International Conference on Learning Representations*.

Sun, J.; Huai, M.; Jha, K.; and Zhang, A. 2022. Demystify Hyperparameters for Stochastic Optimization with Transferable Representations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 1706–1716. New York, NY, USA: Association for Computing Machinery. ISBN 9781450393850.

Sun, J.; Yang, Y.; Xun, G.; and Zhang, A. 2021. A Stagewise Hyperparameter Scheduler to Improve Generalization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, 1530–1540. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383325.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, III–1139–III–1147.

Van Scoy, B.; Freeman, R. A.; and Lynch, K. M. 2018. The Fastest Known Globally Convergent First-Order Method for Minimizing Strongly Convex Functions. *IEEE Control Systems Letters*, 2(1): 49–54.

Wang, X.; Magnússon, S.; and Johansson, M. 2021. On the Convergence of Step Decay Step-Size for Stochastic Optimization. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Wang, Y.; Lin, L.; and Chen, J. 2022. Communication-Efficient Adaptive Federated Learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 22802–22838. PMLR.

Wilson, A. C.; Roelofs, R.; Stern, M.; Srebro, N.; and Recht, B. 2017. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In *Advances in Neural Information Processing Systems 30*, 4148–4158. Curran Associates, Inc.

Wu, X.; Sun, J.; Hu, Z.; Li, J.; Zhang, A.; and Huang, H. 2023a. Federated Conditional Stochastic Optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wu, X.; Sun, J.; Hu, Z.; Zhang, A.; and Huang, H. 2023b. Solving a Class of Non-Convex Minimax Optimization in Federated Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yang, H.; Fang, M.; and Liu, J. 2021. Achieving Linear Speedup with Partial Worker Participation in Non-IID Federated Learning. In *International Conference on Learning Representations*.

Yurochkin, M.; Agarwal, M.; Ghosh, S. S.; Greenewald, K. H.; Hoang, T. N.; and Khazaeni, Y. 2019. Bayesian Nonparametric Federated Learning of Neural Networks. In *International Conference on Machine Learning*.

Zhang, J.; Karimireddy, S. P.; Veit, A.; Kim, S.; Reddi, S.; Kumar, S.; and Sra, S. 2020. Why are Adaptive Methods Good for Attention Models? In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 15383–15393. Curran Associates, Inc.