Breaking the centralized barrier for cross-device federated learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated learning (FL) is a challenging setting for optimization due to the heterogeneity of the data across different clients which can cause a *client drift* phenomenon. In fact, designing an algorithm for FL that is uniformly better than simple centralized training has been a major open problem thus far. In this work, we propose a general algorithmic framework, MIME, which i) mitigates client drift and ii) adapts an arbitrary centralized optimization algorithm such as momentum and Adam to the cross-device federated learning setting. 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. We prove a reduction result showing that MIME can translate the convergence of a generic algorithm in the centralized setting into convergence in the federated setting. Moreover, we show that, when combined with momentum-based variance reduction, MIME is provably *faster than any centralized method*–the first such result. We also perform a thorough experimental exploration of MIME's performance on real world datasets.

1 Introduction

Federated learning (FL) is an increasingly important large-scale learning framework where the training data remains distributed over a large number of clients, which may be mobile phones or network sensors [35, 34, 40, 41, 26]. A server then orchestrates the clients to train a single model, here referred to as a *server model*, without ever transmitting client data over the network, thereby providing some basic levels of data privacy and security.

Two important settings are distinguished in FL [26, Table 1]: the *cross-device* and the *cross-silo* settings. The cross-silo setting corresponds to a relatively small number of reliable clients, typically organizations, such as medical or financial institutions. In contrast, in the *cross-device* federated learning setting, the number of clients may be extremely large and include, for example, all 3.5 billion active android phones [23]. Thus, in that setting, we may never make even a single pass over the entire clients' data during training. The cross-device setting is further characterized by resource-poor clients communicating over a highly unreliable network. Together, the essential features of this setting give rise to unique challenges not present in the cross-silo setting. In this work, we are interested in the more challenging cross-device setting, for which we will formalize and study stochastic optimization algorithms. Importantly, recent advances in FL optimization, such as SCAFFOLD [29] or FedDyn [1], are *not anymore applicable* since they are designed for the cross-silo setting.

The problem. The de facto standard algorithm for the cross-device setting is FEDAVG [40], which performs multiple SGD updates on the available clients before communicating to the server. While this approach can reduce the *frequency* of communication required, performing multiple steps on the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

same client can lead to 'over-fitting' to its atypical local data, a phenomenon known as *client drift* [29]. This in turn leads to slower convergence and can, somewhat counter-intuitively, require *larger total communication* [63]. Despite significant attention received from the optimization community, the communication complexity of heterogeneous cross-device has not improved upon that of simple centralized methods, which take no local steps (aka SERVER-ONLY methods). Furthermore, algorithmic innovations such as momentum [53, 13], adaptivity [32, 69, 71], and clipping [65, 66, 70] are critical to the success of deep learning applications. The lack of a theoretical understanding of the impact of multiple client steps has also hindered adapting these techniques in a principled manner into the client updates, in order to replace the vanilla SGD update of FEDAVG.

To overcome such deficiencies, we propose a new framework, MIME, that mitigates client drift and can adapt an arbitrary centralized optimization algorithm, e.g. SGD with momentum or Adam, to the federated setting. In each local client update, MIME uses global optimizer state, e.g. momentum or adaptive learning rates, and an SVRG-style correction to mimic the updates of the centralized algorithm run on i.i.d. data. This optimizer state is computed only at the server level and kept fixed throughout the local steps, thereby avoiding overfitting to the atypical local data of any single client.

Contributions. We summarize our main results below.

- MIME **framework.** We formalize the cross-device federated learning problem, and propose a new framework MIME that can adapt arbitrary centralized algorithms to this setting.
- **Convergence result.** We prove a result showing that MIME successfully reduces client drift. We also prove that the convergence of any generic algorithm in the centralized setting translates convergence of its MIME version in the federated setting.
- **Speed-up over centralized methods.** By carefully tracking the bias introduced due to multiple local steps, we prove that MIME with momentum-based variance reduction (MVR) can beat a lower bound for centralized methods, thus breaking a fundamental barrier. This is the first such result in FL, and also the first general result showing asymptotic speed-up due to local steps.
- **Empirical validation.** We propose a simpler variant, MIMELITE, with an empirical performance similar to MIME. We report the results of thorough experimental analysis demonstrating that both MIME and MIMELITE indeed converge faster than FEDAVG.

Related work. *Analysis of* FEDAVG*:* Much of the recent work in federated learning has focused on analyzing FEDAVG. For identical clients, FEDAVG coincides with parallel SGD, for which [72] derived an analysis with asymptotic convergence. Sharper and more refined analyses of the same method, sometimes called local SGD, were provided by [50], and more recently by [51], [43], [31], and [64], for identical functions. Their analysis was extended to heterogeneous clients in [62, 68, 29, 31, 33]. [11] derived a tight characterization of FedAvg with quadratic functions and demonstrated the sensitivity of the algorithm to both client and server step sizes. Matching upper and lower bounds were recently given by [29] and [63] for general functions, proving that FEDAVG can be slower than even SGD for heterogeneous data, due to the *client-drift*.

Comparison to SCAFFOLD*:* For the cross-silo setting where the number of clients is relatively low, [29] proposed the SCAFFOLD algorithm, which uses control-variates (similar to SVRG) to correct for client drift. However, their algorithm crucially relies on *stateful clients* which repeatedly participate in the training process. FedDyn [1] reduces the communication requirements, but also requires persistent stateful clients. In contrast, we focus on the cross-device setting where clients may be visited only once during training and where they are *stateless* (and thus SCAFFOLD and FedDyn are inapplicable). This is akin to the difference between the finite-sum (corresponding to cross-silo) and stochastic (cross-device) settings in traditional centralized optimization [36].

Comparison to FedAvg and variants: [24] and [61] observed that using *server momentum* significantly improves over vanilla FEDAVG. This idea was generalized by [45], who replaced the server update with an arbitrary optimizer, e.g. Adam. However, these methods only modify the server update while using SGD for the client updates. MIME, on the other hand, ensures that every *local client update* resembles the optimizer e.g. MIME would apply momentum in every client update and not just at the server level. Beyond this, [37] proposed to add a regularizer to ensure client updates remain close. However, this may slow down convergence [cf. Fig. 7, 29, 60]. Other orthogonal directions which can be combined with MIME include tackling computation heterogeneity, where some clients perform many more updates than others [60], improving fairness by modifying the objective [41, 38], incorporating differential privacy [18, 2, 55], Byzantine adversaries [44, 59, 28], secure aggregation [8, 22], etc. We defer additional discussion to the extensive survey by [26].

2 Problem setup

This section formalizes the problem of cross-device federated learning [26]. Cross-device FL is characterized by a large number of client devices like mobile phones which may potentially connect to the server at most once. Due to their transient nature, it is not possible to store any state on the clients, precluding an algorithm like SCAFFOLD. Furthermore, each client has only a few samples, and there is wide heterogeneity in the samples across clients. Finally, communication is a major bottleneck and a key metric for optimization in this setting is the number of communication rounds.

Thus, our objective will be to minimize the following quantity within the fewest number of client-server communication rounds:

$$f(\boldsymbol{x}) = \mathbb{E}_{i\sim\mathcal{D}}\Big[f_i(\boldsymbol{x}) := \frac{1}{n_i}\sum_{\nu=1}^{n_i} f_i(\boldsymbol{x};\zeta_{i,\nu})\Big]\,. \tag{1}$$

Here, f_i denotes the loss function of client i and $\{\zeta_{i,1},\ldots,\zeta_{i,n_i}\}$ its local data. Since the number of clients is extremely large, while the size of each local data is rather modest, we represent the former as an expectation and the latter as a finite sum. In each round, the algorithm samples a subset of clients (of size S) and performs some updates to the server model. Due to the transient and heterogeneous nature of the clients, it is easy to see that the problem becomes intractable with arbitrarily dissimilar clients. Thus, it is **necessary** to assume bounded dissimilarity across clients.

(A1) G^2**-BGV** or bounded inter-client gradient variance: there exists $G \geq 0$ such that

$$\mathbb{E}_{i\sim\mathcal{D}}[\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2] \leq G^2\,,\ \forall \boldsymbol{x}\,.$$

Next, we also characterize the variance in the Hessians.

(A2) δ**-BHV** or bounded Hessian variance: Almost surely, the loss function of any client i satisfies

$$\|\nabla^2 f_i(\boldsymbol{x};\zeta) - \nabla^2 f(\boldsymbol{x})\| \leq \delta\,,\ \forall \boldsymbol{x}\,.$$

This is in contrast to the usual smoothness assumption that can be stated as:

(A2*) L**-smooth**: $\|\nabla^2 f_i(\boldsymbol{x};\zeta)\| \leq L\,,\ \forall \boldsymbol{x}$, a.s. for any i.

Note that if $f_i(\boldsymbol{x};\zeta)$ is L-smooth then (A2) is satisfied with $\delta \leq 2L$, and hence (A2) is *weaker* than (A2*). In realistic examples we expect the clients to be similar and hence that $\boldsymbol{\delta} \ll \boldsymbol{L}$. In addition, we assume that $f(\boldsymbol{x})$ is bounded from below by $f^\star$ and is L-smooth, as is standard.

3 Mime framework

In this section we describe how to adapt an arbitrary centralized optimizer (referred to as the "base" algorithm) which may have internal state (e.g. momentum in SGD) to the federated learning problem (1) while ensuring there is no client-drift. Algorithm 1 describes our framework. We develop two variants, MIME and MIMELITE, which consist of three components i) a base algorithm we are seeking to mimic, ii) how we compute the global (server) optimizer state, and iii) the local client updates.

Base algorithm. We assume the centralized base algorithm we are imitating can be decomposed into two steps: an *update step* $\mathcal{U}$ which updates the parameters $\boldsymbol{x}$, and a *optimizer state update step* $\mathcal{V}(\cdot)$ which keeps track of global optimizer state $\boldsymbol{s}$. Each step of the base algorithm $\mathcal{B} = (\mathcal{U}, \mathcal{V})$ uses a gradient $\boldsymbol{g}$ to update the parameter $\boldsymbol{x}$ and the optimizer state $\boldsymbol{s}$ as follows:

$$\begin{aligned} \boldsymbol{x} &\leftarrow \boldsymbol{x} - \eta\,\mathcal{U}(\boldsymbol{g},\boldsymbol{s})\,, \\ \boldsymbol{s} &\leftarrow \mathcal{V}(\boldsymbol{g},\boldsymbol{s})\,. \end{aligned} \tag{BASEALG}$$

As an example, consider SGD with momentum (SGDm). The state in SGDm is the momentum $\boldsymbol{m}_t$. SGDm uses the following update steps:

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{x}_{t-1} - \eta\left((1-\beta)\nabla f_i(\boldsymbol{x}_{t-1}) + \beta \boldsymbol{m}_{t-1}\right), \\ \boldsymbol{m}_t &= (1-\beta)\nabla f_i(\boldsymbol{x}_{t-1}) + \beta \boldsymbol{m}_{t-1}\,. \end{aligned} \tag{2}$$

Thus, SGDm can be represented in the above generic form with $\mathcal{U}(\boldsymbol{g},\boldsymbol{s}) = (1-\beta)\boldsymbol{g} + \beta\boldsymbol{s}$ and $\mathcal{V}(\boldsymbol{g},\boldsymbol{s}) = (1-\beta)\boldsymbol{g} + \beta\boldsymbol{s}$. Table 4 in Appendix shows how other algorithms like Adam, Adagrad, etc. can be represented in this manner. We keep the update $\mathcal{U}$ to be linear in the gradient $\boldsymbol{g}$, whereas $\mathcal{V}$ can be more complicated. This implies that while the parameter update step $\mathcal{U}$ is relatively resilient to receiving a biased gradient $\boldsymbol{g}$ while $\mathcal{V}$ can be much more sensitive.

Algorithm 1 Mime and MimeLite

input: initial $\boldsymbol{x}$ and $\boldsymbol{s}$, learning rate η and base algorithm $\mathcal{B} = (\mathcal{U}, \mathcal{V})$
for each round $t = 1, \cdots, T$ **do**
  sample subset $\mathcal{S}$ of clients
  communicate $(\boldsymbol{x}, \boldsymbol{s})$ to all clients $i \in \mathcal{S}$
  communicate $\boldsymbol{c} \leftarrow \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \nabla f_j(\boldsymbol{x})$ (only Mime)
  on client $i \in \mathcal{S}$ **in parallel do**
    initialize local model $\boldsymbol{y}_i \leftarrow \boldsymbol{x}$
    for $k = 1, \cdots, K$ **do**
      sample mini-batch ζ from local data
      $\boldsymbol{g}_i \leftarrow \nabla f_i(\boldsymbol{y}_i;\zeta) - \nabla f_i(\boldsymbol{x};\zeta) + \boldsymbol{c}$ (**Mime**)
      $\boldsymbol{g}_i \leftarrow \nabla f_i(\boldsymbol{y}_i;\zeta)$ (**MimeLite**)
      update $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta\mathcal{U}(\boldsymbol{g}_i, \boldsymbol{s})$
    end for
    compute full local-batch gradient $\nabla f_i(\boldsymbol{x})$
    communicate $(\boldsymbol{y}_i, \nabla f_i(\boldsymbol{x}))$
  end on client
  $\boldsymbol{s} \leftarrow \mathcal{V}\Big(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla f_i(\boldsymbol{x}),\ \boldsymbol{s}\Big)$ (update optimizer state)
  $\boldsymbol{x} \leftarrow \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \boldsymbol{y}_i$ (update server parameters)
end for

Compute optimizer state globally, apply locally. When updating the optimizer state of the base algorithm, we use only the gradient computed at the server parameters. Further, they remain fixed throughout the local updates of the clients. This ensures that these optimizer state remain unbiased and representative of the global function $f(\cdot)$. At the end of the round, the server performs

$$\boldsymbol{s} \leftarrow \mathcal{V}\Big(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla f_i(\boldsymbol{x}),\ \boldsymbol{s}\Big)\,,$$
$$\nabla f_i(\boldsymbol{x}) = \frac{1}{n_i}\sum_{\nu=1}^{n_i} \nabla f_i(\boldsymbol{x};\zeta_{i,\nu})\,. \qquad \text{(OptState)}$$

Note that we use full-batch gradients computed at the server parameters $\boldsymbol{x}$, not client parameters $\boldsymbol{y}_i$.

Local client updates. Each client $i \in \mathcal{S}$ performs K updates using $\mathcal{U}$ of the base algorithm and a minibatch gradient. There are two variants possible corresponding to Mime and MimeLite differentiated using colored boxes. Starting from $\boldsymbol{y}_i \leftarrow \boldsymbol{x}$, repeat the following K times

$$\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta\mathcal{U}(\boldsymbol{g}_i, \boldsymbol{s}) \qquad \text{(CltStep)}$$

where $\boldsymbol{g}_i \leftarrow \nabla f_i(\boldsymbol{y}_i;\zeta)$ for MimeLite, and $\boldsymbol{g}_i \leftarrow \nabla f_i(\boldsymbol{y}_i;\zeta) - \nabla f_i(\boldsymbol{x};\zeta) + \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \nabla f_j(\boldsymbol{x})$ for Mime. MimeLite simply uses the local minibatch gradient whereas Mime uses an SVRG style correction [25]. This is done to reduce the noise from sampling a local mini-batch. While this correction yields faster rates in theory (and in practice for convex problems), in deep learning applications we found that MimeLite closely matches the performance of Mime.

Finally, there are two modifications made in practical FL: we weight all averages across the clients by the number of datapoints n_i [40], and we perform K epochs instead of K steps [60].

4 Theoretical analysis of Mime

Table 1 summarizes the rates of Mime (highlighted in blue) and MimeLite (highlighted in green) and compares them to Server-Only methods when using SGD, Adam and momentum methods as the base algorithms. We will first examine the convergence of Mime and MimeLite with a generic base optimizer and show that its properties are preserved in the federated setting. We then examine a specific momentum based base optimizer, and prove that Mime and MimeLite can be asymptotically faster than the best server-only method. This is the first result to prove the usefulness of local steps and demonstrate asymptotic speed-ups.

4.1 Convergence with a generic base optimizer

We will prove a generic reduction result demonstrating that if the underlying base algorithm converges, and is robust to slight perturbations, then Mime and MimeLite also preserve the convergence of the algorithm when applied to the federated setting with additinoal local steps.

Theorem I. *Suppose that we have G^2 inter-client gradient variance (A1), L-smooth $\{f_i\}$ (A2*), and σ^2 intra-client gradient variance (A3). Further, suppose that the updater $\mathcal{U}$ of our base-optimizer $\mathcal{B} = (\mathcal{U}, \mathcal{V})$ satisfies i) linearity: $\mathcal{U}(\boldsymbol{g}_1 + \boldsymbol{g}_2) = \mathcal{U}(\boldsymbol{g}_1) + \mathcal{U}(\boldsymbol{g}_2)$, and ii) Lipschitzness: $\|\mathcal{U}(\boldsymbol{g})\| \leq B\|\boldsymbol{g}\|$ for some $B \geq 0$. Then, running* Mime *or* MimeLite *with K local updates*

Table 1: Number of communication rounds required to reach $\|\nabla f(\boldsymbol{x})\|^2 \le \epsilon$ (log factors are ignored) with S clients sampled each round. All analyses except SCAFFOLD assume G^2 bounded gradient dissimilarity (A1). All analyses assume L-smooth losses, except MimeLiteMVR and MimeMVR, which only assume δ bounded Hessian dissimilarity (A2). Convergence of SCAFFOLD depends on the total number of clients N which is potentially infinite. FEDAVG and MIMELITE are slightly slower than the server-only methods due to additional drift terms in most cases. MIME is the fastest and either matches or improves upon the optimal statistical rates (first term in the rates). In fact, MimeMVR and MimeLiteMVR beat lower bounds for any server-only method when $\delta \ll L$.

Algorithm	Non-convex	μ-PL inequality
SCAFFOLD[a] [29]	$\left(\frac{N}{S}\right)^{\frac{2}{3}}\frac{L}{\epsilon}$	$\frac{N}{S}+\frac{L}{\mu}$
SGD		
SERVER-ONLY [19]	$\frac{LG^2}{S\epsilon^2}+\frac{L}{\epsilon}$	$\frac{G^2}{\mu S\epsilon}+\frac{L}{\mu}$
MimeLiteSGD≡FedSGD [c]	$\frac{LG^2}{S\epsilon^2}+\frac{L^2G}{\epsilon^{3/2}}+\frac{L}{\epsilon}$	$\frac{G^2}{\mu S\epsilon}+\frac{LG}{\mu\sqrt{\epsilon}}+\frac{L}{\mu}$
MimeSGD	$\frac{LG^2}{S\epsilon^2}+\frac{L}{\epsilon}$	$\frac{G^2}{\mu S\epsilon}+\frac{L}{\mu}$
ADAM		
SERVER-ONLY [69][b]	$\frac{L}{\epsilon-G^2/S}$	–
MimeLiteAdam[bc]	$\frac{L\sqrt{S}}{\epsilon-G^2/S}$	–
MimeAdam[b]	$\frac{L}{\epsilon-G^2/S}$	–
Momentum Variance Reduction (MVR)		
SERVER-ONLY [13]	$\frac{LG}{\sqrt{S}\epsilon^{3/2}}+\frac{L}{\epsilon}$	–
MimeLiteMVR[d]	$\frac{\delta(G+\sigma)}{\epsilon^{3/2}}+\frac{G^2+\sigma^2}{\epsilon}+\frac{\delta}{\epsilon}$	–
MimeMVR[d]	$\frac{\delta G}{\sqrt{S}\epsilon^{3/2}}+\frac{G^2}{S\epsilon}+\frac{\delta}{\epsilon}$	–
SERVER-ONLY lower bound [5]	$\Omega\left(\frac{LG}{\sqrt{S}\epsilon^{3/2}}+\frac{G^2}{S\epsilon}+\frac{L}{\epsilon}\right)$	$\Omega\left(\frac{G^2}{S\epsilon}\right)$

[a] Num. clients (N) can be same order as num. total rounds or even ∞, making the bounds vacuous.
[b] Adam requires large batch-size $S \ge G^2/\epsilon$ to converge [46, 69]. Convergence of FedAdam with client sampling is unknown ([45] only analyze with full client participation).
[c] Requires $K \ge \sigma^2/G^2$ number of local updates. Typically, intra-client variance is small ($\sigma^2 \lesssim G^2$).
[d] Requires $K \ge L/\delta$ number of local updates. Faster than the lower bound (and hence any SERVER-ONLY algorithm) when $\delta \ll L$ i.e. our methods can take advantage of Hessian similarity, whereas SERVER-ONLY methods cannot. In worst case, $\delta \approx L$ and all methods are comparable.

*and step-size η is equivalent to running a **centralized** algorithm with step-size $\tilde{\eta} := K\eta \le \frac{1}{2LB}$, and updates*

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \tilde{\eta}\,\mathcal{U}(\boldsymbol{g}_t + \boldsymbol{e}_t, \boldsymbol{s}_{t-1}), \text{ and}$$
$$\boldsymbol{s}_t \leftarrow \mathcal{V}(\boldsymbol{g}_t, \boldsymbol{s}_{t-1}), \text{ where we have}$$

$\mathbb{E}_t[\boldsymbol{g}_t] = \nabla f(\boldsymbol{x}_{t-1})$, $\mathbb{E}_t\|\boldsymbol{g}_t - \nabla f(\boldsymbol{x}_{t-1})\|^2 \le G^2/S$, *and*

$$\frac{1}{B^2L^2\tilde{\eta}^2}\,\mathbb{E}_t\|\boldsymbol{e}_t\|^2 \le \begin{cases}\mathbb{E}_t\|\boldsymbol{g}_t\|^2 & \text{MIME},\\ \mathbb{E}_t\|\boldsymbol{g}_t\|^2 + G^2 + \frac{\sigma^2}{K} & \text{MIMELITE}.\end{cases}$$

Here, we have proven that MIME and MIMELITE truly mimic the centralized base algorithm with very small perturbations—the magnitude of $\boldsymbol{e}_t$ is $\mathcal{O}(\tilde{\eta}^2)$. The key to the result is the linearity of the parameter update step $\mathcal{U}(\cdot)$. By separating the base optimizer into a very simple parameter step $\mathcal{U}$ and a more complicated optimizer state update step $\mathcal{V}$, we can ensure that commonly used algorithms such as momentum, Adam, Adagrad, and others all satisfy this property. Armed with this general reduction, we can easily obtain specific convergence results.

Corollary II ((Mime/MimeLite) with SGD)**.** *Given that the conditions in Theorem I are satisfied, let us run T rounds with K local steps using SGD as the base optimizer and output $\boldsymbol{x}^{out}$. This output satisfies $\mathbb{E}\|\nabla f(\boldsymbol{x}^{out})\|^2 \leq \epsilon$ for $F := f(\boldsymbol{x}_0) - f^\star$, $\tilde{G}^2 := G^2 + \sigma^2/K$ and*

- ***μ-PL inequality:*** *$\eta = \tilde{\mathcal{O}}\big(\frac{1}{\mu KT}\big)$, and*

$$T = \begin{cases} \tilde{\mathcal{O}}\Big(\frac{LG^2}{\mu S\epsilon} + \frac{LF}{\mu}\log\big(\frac{1}{\epsilon}\big)\Big) & \text{MIME}, \\ \tilde{\mathcal{O}}\Big(\frac{L\tilde{G}^2}{\mu S\epsilon} + \frac{L\tilde{G}}{\mu\sqrt{\epsilon}} + \frac{LF}{\mu}\log\big(\frac{1}{\epsilon}\big)\Big) & \text{MIMELITE}. \end{cases}$$

- ***Non-convex:*** *for $\eta = \mathcal{O}\big(\sqrt{\frac{FS}{L\tilde{G}^2TK^2}}\big)$, and*

$$T = \begin{cases} \mathcal{O}\Big(\frac{LG^2F}{S\epsilon^2} + \frac{LF}{\epsilon}\Big) & \text{MIME}, \\ \mathcal{O}\Big(\frac{L\tilde{G}^2F}{S\epsilon^2} + \frac{L^2\tilde{G}F}{\epsilon^{3/2}} + \frac{LF}{\epsilon}\Big) & \text{MIMELITE}. \end{cases}$$

If we take a sufficient number of local steps $K \geq G^2/\sigma^2$, then we have $\tilde{G} = \mathcal{O}(G)$ in the above rates. On comparing with the rates in Table 1 for SERVER-ONLY SGD, we see that MIME exactly matches its rates. MIMELITE matches the asymptotic term but has a few higher order terms. Note that when using SGD as the base optimizer, MIMELITE becomes exactly the same as FEDAVG and hence has the same rate of convergence.

Corollary III ((Mime/MimeLite) with Adam)**.** *Suppose that the conditions in Theorem I are satisfied, and further $|\nabla_j f_i(\boldsymbol{x})| \leq H$ for any coordinate $j \in [d]$. Then let us run T rounds using Adam as the base optimizer with K local steps, $\beta_1 = 0$, $\varepsilon_0 > 0$, $\eta \leq \varepsilon_0^2/KL(H+\varepsilon_0)$, and any $\beta_2 \in [0, 1)$. Output $\boldsymbol{x}^{out}$ chosen randomly from $\{\boldsymbol{x}_1, \ldots \boldsymbol{x}_T\}$ satisfies $\mathbb{E}\|\nabla f(\boldsymbol{x}^{out})\|^2 \leq \epsilon$ for*

$$T = \begin{cases} \mathcal{O}\Big(\frac{LF(H+\varepsilon_0)^2}{\varepsilon_0^2(\epsilon - \tilde{G}^2/S)}\Big) & \text{MIME Adam}, \\ \mathcal{O}\Big(\frac{LF(H+\varepsilon_0)^2\sqrt{S}}{\varepsilon_0^2(\epsilon - \tilde{G}^2/S)}\Big) & \text{MIMELITE Adam}. \end{cases}$$

where $F := f(\boldsymbol{x}_0) - f^\star$, $\tilde{G}^2 := G^2 + \sigma^2/K$.

Note that here ε_0 represents a small positive parameter used in Adam for regularization, and is different from the accuracy ϵ. Similar to the SERVER-ONLY analysis of Adam [69], we assume $\beta_1 = 0$ and that batch size is large enough such that $S \geq G^2/\epsilon$. A similar analysis can also be carried out for AdaGrad, and other novel variants of Adam [39].

4.2 Circumventing server-only lower bounds

The rates obtained above, while providing a safety-check, do not beat those of the SERVER-ONLY approach. The previous best rates for cross-device FL correspond to MimeLiteSGD which is $\mathcal{O}(\frac{LG^2}{S\epsilon^2} + \frac{L^2G}{\epsilon^{3/2}})$ [31, 33, 63]. While, using a separate server-learning rate can remove the effect of the second term [30], this at best matches the rate of SERVER-ONLY SGD $\mathcal{O}(\frac{LG^2}{S\epsilon^2})$. This is significantly slower than simply using momentum based variance reduction (MVR) as in in the FL setting (SERVER-ONLY MVR) which has a communication complexity of $\mathcal{O}(\frac{LG}{\sqrt{S}\epsilon^{3/2}})$ [13]. Thus, even though the main reason for studying local-step methods was to improve the communication complexity, none thus far show such improvement. The above difficulty of beating SERVER-ONLY may not be surprising given the two sets of strong lower bounds known.

Necessity of local steps. Firstly, [5] show a gradient oracle lower bound of $\Omega(\frac{LG}{\sqrt{S}\epsilon^{3/2}})$. This matches the complexity of MVR, and hence at first glance it seems that SERVER-ONLY MVR is optimal. However, the lower bound is really only on the number of gradients computed and not on the number of clients sampled (sample complexity) [15], or number of rounds of communication required. In particular, multiple local updates which increases number of gradients computed *without needing additional communication* offers us a potential way to side-step such lower bounds. A careful analysis of the bias introduced as a result of such local steps is a key part of our analysis.

Necessity of δ-BHD. A second set of lower bounds directly study the number of communication rounds required in heterogeneous optimization [6, 63]. These results prove that there exist settings

where local steps provide no advantage and SERVER-ONLY methods are optimal. This however contradicts real world experimental evidence [40]. As before, the disparity arises due to the contrived settings considered by the lower bounds. For distributed optimization (with full client participation) and convex quadratic objectives, δ-BHD (A2) was shown to be a sufficient [48, 47] and *necessary* [6] condition to circumvent these lower bounds and yield highly performant methods. We similarly leverage δ-BHD (A2) to design novel methods which significantly extend prior results to i) all smooth non-convex functions (not just quadratics), and ii) cross-device FL with client sampling.

We now state our convergence results with momentum based variance reduction (MVR) as the base-algorithm since it is known to be optimal in the SERVER-ONLY setting.

Theorem IV. *For L-smooth f with G^2 gradient dissimilarity (A1), δ Hessian dissimilarity (A2) and $F := (f(\boldsymbol{x}^0) - f^\star)$, let us run MVR as the base algorithm for T rounds with $K \geq L/\delta$ local steps and generate an output $\boldsymbol{x}^{out}$. This output satisfies $\mathbb{E}\|\nabla f(\boldsymbol{x}^{out})\|^2 \leq \epsilon$ for*

- ***MimeMVR***: $\eta = \mathcal{O}\Big(\min\big(\frac{1}{\delta K}, (\frac{SF}{G^2TK^3})^{1/3}\big)\Big)$, *momentum* $\beta = 1 - \mathcal{O}(\frac{\delta^2 S^{2/3}}{(TG^2)^{2/3}})$, *and*
$$T = \mathcal{O}\big(\frac{\delta GF}{\sqrt{S}\epsilon^{3/2}} + \frac{G^2}{S\epsilon} + \frac{\delta F}{\epsilon}\big)\,.$$
- ***MimeLiteMVR***: $\eta = \mathcal{O}\Big(\min\big(\frac{1}{\delta K}, (\frac{F}{\hat{G}^2TK^3})^{1/3}\big)\Big)$, *momentum* $\beta = 1 - \mathcal{O}(\frac{\delta^2}{(T\hat{G}^2)^{2/3}})$, *and*
$$T = \mathcal{O}\big(\frac{\delta \hat{G}F}{\epsilon^{3/2}} + \frac{\hat{G}^2}{\epsilon} + \frac{\delta F}{\epsilon}\big)\,.$$

Here, we define $\hat{G}^2 := G^2 + \sigma^2$ and the expectation in $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 \leq \epsilon$ is taken both over the sampling of the clients during the running of the algorithm, the sampling of the mini-batches in local updates, and the choice of $\boldsymbol{x}^{\text{out}}$ (which is chosen randomly from the client iterates $\boldsymbol{y}_i$).

Remarkably, the rates of our methods are independent of L and only depend on δ. Thus, when $\delta \leq L$ and $\delta \leq L/S$ for MimeMVR and MimeLiteMVR, the rates beat the server only lower bound of $\Omega(\frac{LG}{\sqrt{S}\epsilon^{3/2}})$. In fact, if the Hessian variance is small and $\delta \approx 0$, our methods only need $\mathcal{O}(1/\epsilon)$ rounds to communicate. Intuitively, our results show that local steps are very useful when heterogeneity (represented by δ) is smaller than optimization difficulty (captured by smoothness constant L).

MimeMVR uses a momentum parameter β of the order of $(1 - \mathcal{O}(TG^2)^{-2/3})$ i.e. as T increases, β asymptotically approaches 1. In contrast, previous analyses of distributed momentum (e.g. [67]) prove rates of the form $\frac{G^2}{S(1-\beta)\epsilon^2}$, which are worse than that of standard SGD by a factor of $\frac{1}{1-\beta}$. Thus, ours is also the first result which theoretically showcases the usefulness of using large momentum in distributed and federated learning.

Our analysis is highly non-trivial and involves two crucial ingredients: i) computing the momentum at the server level to ensure that it remains unbiased and then applying it locally during every client update to reduce variance, and ii) carefully keeping track of the bias introduced via additional local steps. Our experiments (Sec. 5) verify our theoretical insights are indeed applicable in deep learning settings as well. See App. D for a proof sketch and App. F–G detailed proofs.

5 Experimental analysis on real world datasets

We run experiments on *natively* federated datasets to confirm our theory and accurately measure real world performance. Our main findings are i) MIME and MIMELITE consistently outperform FEDAVG, and ii) momentum and adaptivity significantly improves performance.

5.1 Setup

Algorithms. We consider three (meta) algorithms: FEDAVG, MIME, and MIMELITE. Each of these adapt four base optimizers: SGD, momentum, Adam, and Adagrad.
FEDAVG follows [45] who run multiple epochs of SGD on each client sampled, and then aggregate the net client updates. This aggregated update is used as a pseudo-gradient in the base optimizer (called server optimizer). The learning rate for the server optimizer is fixed to 1 as in [61]. This is done to ensure all algorithms have the same number of hyper-parameters.
MIME and MIMELITE follow Algorithm 1 and also run a fixed number of epochs on the client. However, note that this requires communicating both the full local-batch gradient as well as the parameter updates doubling the communication required to be sent by the client. For a fairer comparison, we split the sampled clients in MIME and MIMELITE into two groups–the first communicates only full

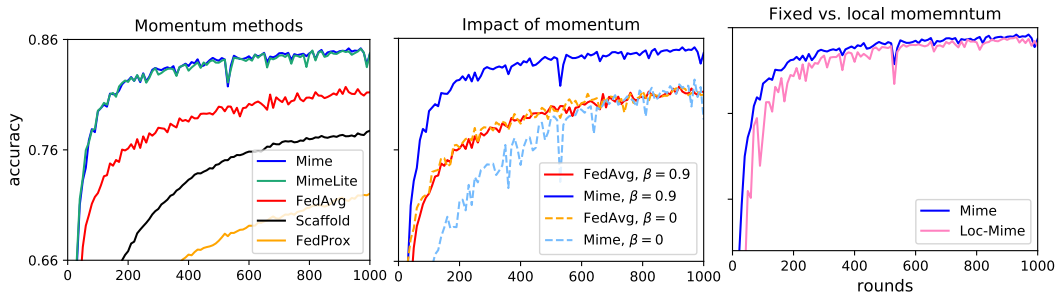

Figure 1: **Mime**, **MimeLite**, **FedAvg**, **Scaffold**, **FedProx**, and **Loc-Mime** with SGD+momentum using 10 local epochs, run on EMNIST62 and a 2 hidden layer (300u-100) MLP. (Left) Mime and MimeLite are nearly identical and outperform the rest ($7\times$ faster). (Center) Mime makes better use of momentum than FedAvg, with a large increase in performance. (Right) Locally adapting momentum slows down convergence and makes it more unstable.

local-batch gradient and the latter communicates only parameter updates. Thus, all methods have **equal client communication** to the server. This variant retains the convergence guarantees up to constants (details in the Appendix). We also run Loc-MIME where instead of keeping the global optimizer state fixed, we update it locally within the client. The optimizer state is reset after the round finishes. In all methods, aggregation is weighted by the number of samples on the clients.

Datasets and models. We run five simulations on three real-world federated datasets: EMNIST62 [10] with i) a linear classifier, ii) an MLP, and iii) a CNN, iv) a charRNN on Shakespeare [40], and v) an LSTM for next word prediction on StackOverflow [14], all accessed through Tensorflow Federated [54]. The learning rates were individually tuned and other optimizer hyperparameters such as β for momentum, β_1, β_2, ε_0 for Adam etc., were left to their default values, unless explicitly stated otherwise. We refer to Appendix B for additional setup details.

5.2 Ablation and comparative study

In order to study the different algorithms, we train a 2 hidden layer (300μ-100) MLP on EMNIST62 with 10 local epochs for 1k rounds and use SGD+momentum (with tuned β) as the base optimizer.

Mime $\approx$ MimeLite $>$ FedAvg $>$ SCAFFOLD $>$ FedProx. Fig. 1 (left) shows MIME and MIMELITE have nearly identical performance, and are about $7\times$ faster than FedAvg. This implies our strategy of applying momentum to client updates is faster than simply using server momentum. FedProx [37] uses an additional regularizer μ tuned over $[0.1, 0.5, 1]$ ($\mu = 0$ is the same as FedAvg). Regularization does not seem to reduce client drift but still slows down convergence [60]. SCAFFOLD [29] is also slower than Mime and FedAvg in this setup. This is because in cross-device setting with a large number of clients ($N = 3.4k$) means that each client is visited less than 6 times during the entire training (20 clients per round for 1k rounds). Hence, the client control variate stored is quite stale (from about 200 rounds ago) which slows down the convergence.

With momentum $>$ without momentum. Fig. 1 (center) examines the impact of momentum on FedAvg and Mime. Momentum slightly improves the performance of FedAvg, whereas it has a significant impact on the performance of Mime. This is also in line with our theory and confirms that Mime's strategy of applying it locally at every client update makes better use of momentum.

Fixed $>$ locally updated optimizer state. Finally, we check how the performance of Mime changes if instead of keeping the momentum fixed throughout a round, we let it change. The latter is a way to combine global and local momentum. The momentum is reset at the end of the round ignoring the changes the clients make to it. Fig. 1 (right) shows that this *worsens* the performance, confirming that it is better to keep the global optimizer state fixed as predicted by our theory.

Together, the above observations validate all aspects of Mime (and MimeLite) design: compute statistics at the server level, and apply them unchanged at every client update.

5.3 Large scale comparison with equal server and client communication

We perform a larger scale study closely matching the setup of [45]. For both MIME and MIMELITE, only half the clients compute and transmit the updated parameters, and other half transmit the full local-batch gradients. Hence, client to server communication cost is the same for all methods for all clients. However, MIME and MIMELITE require sending additional optimization state to the clients. Hence, we also reduce the number of clients sampled in each round to ensure *sum total* of

Table 2: Validation % accuracies after training for 1000 rounds. Best results for each dataset is underlined and the best within each base optimizer is bolded. The number of clients sampled per round has been reduced for MIME and MIMELITE to ensure all methods have **equal client and server communication**. Final accuracies obtained by MIME and MIMELITE are competitive with FEDAVG, especially with adaptive base optimizers. FEDAVG seems unstable with Adam.

		EMNIST logistic	EMNIST CNN	Shakespeare	StackOverflow
SGD	FedAvgSGD	66.8	**85.8**	**56.7**	**23.8**
	MimeLiteSGD	66.8	**85.8**	**56.7**	**23.8**
	MimeSGD	**67.4**	85.3	56.1	12.5
MOMENTUM	FedAvgMom	67.4	85.7	**55.4**	**22.2**
	MimeLiteMom	67.4	**86.0**	49.8	19.9
	MimeMom	**67.5**	85.9	53.6	19.3
ADAM	FedAvgAdam	67.3	85.9	18.5	3.2
	MimeLiteAdam	**68.0**	86.4	54.0	21.5
	MimeAdam	**68.0**	**86.6**	**54.1**	**22.8**
ADAGRAD	FedAvgAdagrad	**67.6**	86.3	55.5	**24.2**
	MimeLiteAdagrad	66.6	85.5	56.8	23.8
	MimeAdagrad	67.4	**86.3**	**57.1**	14.7

communication at each round is $40\times$ model size for EMNIST and Shakespeare experiments, and $100\times$ model size for the StackOverflow next word prediction experiment.

Since we only perform 1 local epoch, the hyper-parameters (e.g. epsilon for adaptive methods) are more carefully chosen following [45], and MIME and MIMELITE use significantly fewer clients per round, the difference between FEDAVG and MIME is smaller here. Table 2 summarizes the results.

For the image classification tasks of EMNIST62 logistic and EMNIST62 CNN, Mime and MimeLite with Adam achieve the best performance. Using momentum (both with SGDm, and in Adam) significantly improves their performance. In contrast, FedAvgAdam is more unstable with worse performance. This is because FedAvg is excessively sensitive to hyperparameters (cf. App. C).

We next consider the character prediction task on Shakespeare dataset, and next word prediction on StackOverflow. Here, the momentum based methods (SGDm and Adam) are slower than their non-momentum counterparts (SGD and AdaGrad). This is because the mini-batch gradients in these tasks are *sparse*, with the gradients corresponding to tokens not in the mini-batch being zero. This sparsity structure is however destroyed when using momentum or Adam. For the same reason, Mime which uses an SVRG correction also significantly increases the gradient density.

Discussion. For traditional deep learning tasks such as image classification, we observe that Mime outperforms MimeLite which in turn outperforms FedAvg. These methods are able to successfully leverage momentum to improve performance. For tasks where the client gradients are sparse, the SVRG correction used by Mime hinders performance. Adapting our techniques to work with sparse gradients (à la Yogi [69]) could lead to further improvements. Also, note that we reduce communication by naïvely reducing the number of participating clients per round. More sophisticated approaches to save on client communication including quantization or sparsification [52, 3], or even novel algorithmic innovations [1] could be explored. Further, server communication could be reduced using memory efficient optimizers e.g. AdaFactor [49] or SM3 [4].

6 Conclusion

Our work initiated a formal study of the cross-device federated learning problem and provided theoretically justified algorithms. We introduced a new framework MIME which overcomes the natural client-heterogeneity in such a setting, and can adapt arbitrary centralized algorithms such as Adam without additional hyper-parameters. We demonstrated the superiority of MIME via strong convergence guarantees and empirical evaluations. Further, we proved that a particular instance of our method, MimeMVR, beat centralized lower-bounds, demonstrating that additional local steps can yield asymptotic improvements for the first time. We believe our analysis will be of independent interest beyond the federated setting for understanding the sample complexity of non-convex optimization, and for yielding improved analysis of decentralized optimization algorithms.

References

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

[2] Naman Agarwal, Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Proceedings of NeurIPS*, pages 7575–7586, 2018.

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

[4] Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization. *arXiv preprint arXiv:1901.11150*, 2019.

[5] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

[6] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems*, pages 1756–1764, 2015.

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

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

[9] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[10] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[11] Zachary Charles and Jakub Konečnỳ. On the outsized importance of learning rates in local update methods. *arXiv preprint arXiv:2007.00878*, 2020.

[12] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. *arXiv preprint arXiv:2002.03305*, 2020.

[13] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. In *Advances in Neural Information Processing Systems*, pages 15210–15219, 2019.

[14] Stack Exchange. *Stack Exchange Data Dump*, 2021.

[15] Dylan J Foster, Ayush Sekhari, Ohad Shamir, Nathan Srebro, Karthik Sridharan, and Blake Woodworth. The complexity of making the gradient small in stochastic convex optimization. In *Conference on Learning Theory*, pages 1319–1345. PMLR, 2019.

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

[17] Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 2018.

[18] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[19] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[20] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. FedBoost: Communication-efficient algorithms for federated learning. In *37th International Conference on Machine Learning (ICML)*, 2020.

[21] Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.

[22] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. Secure byzantine-robust machine learning. *arXiv preprint arXiv:2006.04747*, 2020.

[23] Arne Holst. Smartphone users worldwide 2016-2021, 2019.

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

[25] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

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

[27] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[28] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *38th International Conference on Machine Learning (ICML)*, 2021.

[29] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. In *37th International Conference on Machine Learning (ICML)*, 2020.

[30] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *36th International Conference on Machine Learning (ICML)*, 2019.

[31] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local SGD on indentical and heterogeneous data. In *Proceedings of AISTATS*, 2020.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U Stich. A unified theory of decentralized SGD with changing topology and local updates. In *37th International Conference on Machine Learning (ICML)*, 2020.

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

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

[36] Lihua Lei and Michael Jordan. Less than a single pass: Stochastically controlled stochastic gradient. In *AISTATS*, pages 148–156, 2017.

[37] Tian Li, Anit Kumar Sahu, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

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

[39] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, pages 1273–1282, 2017.

[41] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[42] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[43] Kumar Kshitij Patel and Aymeric Dieuleveut. Communication trade-offs for synchronized distributed SGD with large step size. In *33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[44] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

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

[46] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *International Conference on Learning Representations (ICLR)*, 2018.

[47] Sashank J. Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

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

[49] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[50] Sebastian U. Stich. Local SGD converges fast and communicates little. *International Conference on Learning Representations (ICLR)*, 2019.

[51] Sebastian U. Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

[52] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3329–3337. JMLR. org, 2017.

[53] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[54] TFF. *Tensorflow Federated Datasets*, 2020.

[55] Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Françoise Beaufays. Understanding unintended memorization in federated learning. *arXiv preprint arXiv:2006.07490*, 2020.

[56] Quoc Tran-Dinh, Nhan H. Pham, Dzung T. Phan, and Lam M. Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

[57] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. *arXiv preprint arXiv:1810.07288*, 2018.

[58] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[59] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

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

[61] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed sgd with slow momentum. *International Conference on Learning Representations (ICLR)*, 2020.

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

[63] Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local SGD for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020.

[64] Blake Woodworth, Kumar Kshitij Patel, Sebastian U Stich, Zhen Dai, Brian Bullins, H Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local SGD better than minibatch SGD? In *37th International Conference on Machine Learning (ICML)*, 2020.

[65] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[66] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

[67] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

[68] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[69] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pages 9793–9803, 2018.

[70] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[71] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, and Suvrit Sra. Why ADAM beats SGD for attention models. *arXiv preprint arXiv:1912.03194*, 2019.

[72] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Section 4.1 discusses our generic reduction results, and 4.2 discusses how we break centralized (server-only) lower bounds.
   (b) Did you describe the limitations of your work? [Yes] See discussion in Section 5.3.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We only propose a new algorithm for an already existing approach (federated learning)
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] I have read them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Main paper has theorem statements with an emphasis on readability, appendices have more general versions of some of the theorem statements.
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs, and proof sketches for intuition are provided in the appendices.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Code is still in the process of being open sourced.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We use standard federated datasets accessed through Tensorflow federated. We specify our grid search of all hyperparameters.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Experiments were run only once due to time constraints. We are confident results will hold across repetitions due to low levels of noise in the final convergence curves.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Did not keep track of resources used - experiments were conducted across a broad range of setups.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]