Fed-LAMB: Layer-wise and Dimension-wise Locally Adaptive Federated Learning

Belhal Karimi, Ping Li, Xiaoyun Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St, Bellevue, WA 98004, USA
{belhal.karimi, pingli98, lixiaoyun996}@gmail.com

Abstract

In the emerging paradigm of Federated Learning (FL), large amount of clients such as mobile devices are used to train possibly high-dimensional models on their respective data. Combining (dimension-wise) adaptive gradient methods (e.g., Adam, AMSGrad) with FL has been an active direction, which is shown to outperform traditional SGD based FL in many cases. In this paper, we focus on the problem of training federated deep neural networks, and propose a novel FL framework which further introduces layer-wise adaptivity to the local model updates to accelerate the convergence of adaptive FL methods. Our framework includes two variants based on two recent locally adaptive federated learning algorithms. Theoretically, we provide a convergence analysis of our layer-wise FL methods, coined Fed-LAMB and Mime-LAMB, which match the convergence rate of state-of-the-art results in adaptive FL and exhibits linear speedup in terms of the number of workers. Experimental results on various datasets and models, under both IID and non-IID local data settings, show that both Fed-LAMB and Mime-LAMB achieve faster convergence speed and better generalization performance, compared to various recent adaptive FL methods.

1 INTRODUCTION

A growing and important task while learning models on observed data, is the ability to train over a large number of clients which could either be personal devices or distinct entities. In the paradigm of Federated Learning (FL) [Konečný et al., 2016, McMahan et al., 2017], a central server orchestrates the optimization over those clients under the constraint that the data can neither be gathered nor shared among the clients. This is computationally more efficient, since more distributed computing resources are used; also, this is a very practical scenario which allows individual data holders (e.g., mobile devices) to train a model jointly without leaking private data. In this paper, we consider the following optimization problem:

$$\min_{\theta} f(\theta) := \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi \sim X_i}[F_i(\theta; \xi)],$$

where the nonconvex function (e.g., deep networks) $f_i$ represents the average loss over the local data samples for worker $i \in [n]$, and $\theta \in \mathbb{R}^d$ the global model parameter. $X_i$ is the data distribution on each client $i$. There are two general scenarios of FL [Yang et al., 2019]: (i) cross-silo setting where $n$ is small/moderate and the clients can be, e.g., different data servers; (ii) cross-device setting, where $n$ can be large (e.g., millions) and the clients are mobile devices. While [1] reminds that of standard distributed optimization, the principle and setting of FL are different from the classical distributed paradigm. Two of the main differences are: (i) Local updates: FL allows clients to perform multiple updates on the local models before the global aggregation, which improves the computational resource efficiency and reduces the frequency of communication; (ii) Data heterogeneity: in FL, the local data distributions $X_i$ are usually different across workers, hindering the convergence of the global model. Federated learning aims at finding a global solution of [1] in fewest number of communication rounds.

One of the standard and most popular frameworks for FL is called Fed-SGD [McMahan et al., 2017]: we adopt multiple local Stochastic Gradient Descent (SGD) steps in each device, send those local models to the server that computes the average over the received local model parameters, and broadcasts it back to the devices. Moreover, momentum can be added to local SGD training for faster convergence and better learning performance [Yu et al., 2019]. On the other hand, adaptive gradient methods (e.g., Adam [Kingma and Ba, 2015], AMSGrad [Reddi et al., 2018]) have shown great success in many deep learning tasks. For instance, the
update rule of Adam at step $t$ reads as
\[ \theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t}}, \quad m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \]
\[ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \]
where $\alpha$ is the learning rate and $g_t$ is the gradient at time $t$.

We note that the effective learning rate of Adam is $\alpha / \sqrt{v_t}$, which is different across dimensions, i.e., dimension-wise adaptive. Recently, we have seen growing research efforts in the design of FL frameworks that adopt adaptive gradient methods as the protocols for local model training instead of SGD. Examples include federated AMSGrad (Fed-AMS) [Chen et al., 2020] and Mime [Karimireddy et al., 2020] with Adam updates. Specifically, in both methods, in each round the global server not only aggregates the local models, but also broadcasts to the workers a “global” second moment estimation to reconcile the dimension-wise adaptive learning rates across the clients. Therefore, this step can be regarded as a natural mitigation to data heterogeneity, which is a common and important practical scenario that affects the performance of FL algorithms [Li et al., 2020a, Liang et al., 2019, Karimireddy et al., 2019]. Adaptive-optimizer based FL have been shown to outperform many SGD based FL methods on various tasks, making it a promising direction in FL system design.

In this work, we specifically focus on further improving the convergence speed and learning performance of locally adaptive FL algorithms. Our construction is based on introducing a special learning rate schedule into the local training of FL, which has not been proposed in the literature before. For (single-machine) training of deep neural networks using Adam, [You et al., 2020] proposed a layer-wise adjusted learning rate scheme called LAMB, where in each update, the effective update $m_t / \sqrt{v_t}$ is further normalized by the weight of each layer in the deep neural network, respectively. In [You et al., 2020], the authors proved that LAMB matches the convergence rate of Adam theoretically, and demonstrated the superior performance of LAMB empirically. With this weight-dependent adjusted learning rates, LAMB allows large-batch training which could in particular speed up training large datasets and models like ImageNet [Deng et al., 2009] and BERT [Devlin et al., 2019].

**Contributions.** Despite that layer-wise learning rate has been successfully implemented in (single-machine) model learning, one question that has not been explored is: can we also use methods like LAMB in the local training in federated learning? Is it able to also speedup the global model convergence? In this paper, we propose an improved framework for locally adaptive FL algorithms, integrating both dimension-wise and layer-wise adaptive learning rates in each device’s local update. We provide theoretical and empirical justification on the efficacy of such layer-wise adaptivity in local federated training. More specifically, our contributions are summarized as follows:

- We develop Fed-LAMB and Mime-LAMB, two instances of our layer-wise adaptive optimization framework for federated learning, following a principled layer-wise adaptive strategy to accelerate the training of deep neural networks.
- We show that our algorithm converges at the rate of $O\left(\frac{1}{\sqrt{hbr}}\right)$ to a stationary point, where $h$ is the number of layers of the network, $n$ is the number of clients and $R$ is the number of communication rounds. This matches the convergence rate of LAMB, AMSGrad, as well as the state-of-the-art results in federated learning. The theoretical communication efficiency matches that of Fed-AMS [Chen et al., 2020].
- We empirically compare several recent adaptive FL methods under both homogeneous and heterogeneous data setting on various benchmark datasets. Our results confirm the accelerated empirical convergence of Fed-LAMB and Mime-LAMB over the baseline methods, including Fed-AMS and Mime. In addition, Fed-LAMB and Mime-LAMB can also reach similar, or better, test accuracy than their corresponding baselines.

## 2 BACKGROUND

We summarize some relevant work on adaptive optimization, layer-wise adaptivity and federated learning, which compose the key ingredients of our algorithm.

**Adaptive gradient methods.** Adaptive methods have proven to be the spearhead for many nonconvex optimization tasks. Gradient based optimization algorithms alleviate the possibly high nonconvexity of the objective function by adaptively updating each coordinate of their learning rate using past gradients. Common used examples include RMSProp [Tieleman and Hinton, 2012], Adadelta [Zeiler, 2012], Adam [Kingma and Ba, 2015], Nadam [Dozat, 2016] and AMSGrad [Reddi et al., 2018]. Their popularity owes to their great performance in training deep neural networks. They generally combine the idea of adaptivity from AdaGrad [Duchi et al., 2011], McMahan and Streeter, 2010, as explained above, and the idea of momentum from Nesterov’s Method [Nesterov, 2003] or Heavy ball method [Polyak, 1964] using past gradients. AdaGrad displays superiority when the gradient is sparse compared to other classical methods [Duchi et al., 2011]. Yet, when applying AdaGrad to train deep neural networks, it is observed that the learning rate might decay too fast. Consequently, [Kingma and Ba, 2015] developed Adam whose updating rule is presented in Eq. A variant, called AMSGrad [Reddi et al., 2018], forces $v_t$ to be monotone to fix the convergence issue. [Loshchilov and Hutter, 2019] proposed AdamW that combines weight decay with Adam. The convergence and generalization of adaptive methods are studied in, e.g., [Zhou et al., 2018, Chen et al., 2019, Zhou et al., 2020], among others.
Layer-wise Adaptivity. When training deep networks, in many cases the scale of the gradients differs a lot across the network layers. When we use the same learning rate for the whole network, the update might be too conservative for some specific layers (with large weights) which may slow down the convergence. Based on this observation, You et al. [2018] proposed LARS, an extension of SGD with layer-wise adjusted scaling, whose performance, however, is not consistent across tasks. Later, You et al. [2020] proposed LAMB, an analogous layer-wise adaptive variant of Adam. The update rule of LAMB for the ℓ-th layer of the network can be expressed as

$$\theta^t_\ell = \theta^{t-1}_\ell - \frac{\alpha}{\|\psi^t_\ell\|} \psi^t_\ell, \text{ with } \psi^t_\ell = m^t_\ell / \sqrt{v^t_\ell},$$

where $m^t$ and $v^t$ are defined in \(\ddagger\). Intuitively, for the ℓ-th layer, when the gradient magnitude is too small compared to the scale of the model parameter, we increase the effective learning rate to make the model move sufficiently far. Theoretically, You et al. [2020] showed that LAMB achieves the same convergence rate as Adam; empirically, LAMB can significantly accelerate the convergence of Adam, allowing the use of large mini-batch size with fewer training iterations for large datasets.

Federated learning. An extension of the classic distributed training paradigm is called Federated Learning (FL) \cite{konecny2016federated, mcmahan2017communication} which has seen many applications in various fields \cite{yang2019federated, leroy2019federated, bonawitz2019secureml, niknam2020federated, xu2021federated}. For Fed-SGD (where clients perform SGD-based updates), recent variants and theoretical analysis on the convergence can be found in \cite{yu2019pfl, karimireddy2019scaffold, xu2021federated}. \cite{li2020federated, woodworth2020replica, wang2020federated}. Many works have considered adaptive gradient methods in FL. \cite{reddi2021adaptive} proposed Adp-Fed where the central server applies Adam-type updates and the local clients perform SGD updates. \cite{li2022distributed, li2023fl} studied distributed and federated adaptive method under communication compression. \cite{chen2020federated, karimireddy2019scaffold, khaled2020fedprox, li2020federated, woodworth2020replica, wang2020federated} proposed Fed-AMS and Mime respectively, to adopt Adam/AMSGrad at the client level. Both works mitigate the influence of data heterogeneity by “sharing” the second moment $v$ which controls the effective learning rates (more details will be provided later). On many tasks, these methods outperform Fed-SGD and other popular methods like SCAFFOLD \cite{karimireddy2019scaffold} and FedProx \cite{li2020federated, yuan2022federated, charles2021federated} empirically tested a FL method where LARS (i.e., layer-wise SGD) \cite{you2018federated} is applied at the central server in local SGD, which is very different from our work in that the layer-wise adjustment happens locally in our method, and our local optimizer is the adaptive AMSGrad. That is, our local models are trained with dual adaptivity.

3 LAYER-WISE LOCALLY ADAPTIVE FEDERATED LEARNING

In this section, we introduce our proposed FL framework, admitting both dimension-wise adaptivity (of adaptive learning rate) and layer-wise adaptivity (of layer-wise scaling). We mainly consider AMSGrad \cite{reddi2018adaptive} as the prototype adaptive gradient method. We assume the loss function $f(\cdot)$ is induced by a multi-layer neural network, which includes a broad class of network architectures like MLP, CNN, ResNet and Transformers.

Notations. We denote by $\theta$ the vector of parameters taking values in $\mathbb{R}^p$. Suppose the neural network has $h$ layers, each with size $p_\ell$ (thus, $p = \sum_{\ell=1}^h p_\ell$). For each layer $\ell \in [h]$, denote $\theta^\ell$ as the sub-vector corresponding to the $\ell$-th layer. Let $R$ be the number of communication rounds and $T$ be the number of local iterations per round. Moreover, $\theta_{r,\ell,i}$ is the model parameter of layer $\ell$ at round $r$, local iteration $t$ and for worker $i$.

Algorithm. In general, our proposed algorithm can be viewed as a novel extension of LAMB to the more complicated federated learning setting. Based on the two recent works regarding locally adaptive FL mentioned above, we present the framework with two instances, Fed-LAMB and Mime-LAMB, as summarized in Algorithm 1 and depicted in Figure 1. We differentiate the steps of these two methods by blue (Fed-LAMB) and red (Mime-LAMB) boxes surrounding the text. Both methods use layer-wise adaptive LAMB for local updates (Line 13). The update in \(\ddagger\) on local workers can be expressed as

$$\theta \leftarrow \theta - \alpha \frac{\phi(\|\theta\|)}{\|\psi + \lambda \theta\|} (\psi + \lambda \theta),$$

where $\phi(\cdot) : \mathbb{R}_+ \mapsto \mathbb{R}_+$ is a scaling function (usually chosen to be the identity function in practice) and $\lambda$ is the weight decay rate. This way, the gradients are effectively normalized by the magnitude of layer weights, forcing the model move sufficiently far at every layer. Such normalization effect may accelerate the convergence of the model.

The main difference between the two variants, Fed-LAMB and Mime-LAMB, is the way the second moment $v$ is synchronized, i.e., the dimension-wise adaptive learning rate. Both methods maintain a global $\hat{v}$ at the central server:

- **Fed-LAMB (Line 20)**: at the end of each round, the $i$-th client communicates the local $v_i$; the server updates the global $\hat{v}$ by the max operation with the averaged $v$ among all clients, and sends back the global $\hat{v}$.

- **Mime-LAMB (Line 21-23)**: in each round $r$, the client computes and transmits the gradient at the global model $\hat{\theta}_r$ using full local data; the server updates the global $v$ and $\hat{v}$ in the same manner as AMSGrad.
When implementing the algorithms, note that in Mime-LAMB, the global $v$ is directly calculated using full-batches (averaged over all clients). As a result, Mime-LAMB needs to calculate the gradients twice, leading to twice the computational cost as Fed-LAMB. We also note that, the local update of Fed-LAMB (Line 13 of Algorithm 1) also incorporates the “decoupled” weight decay, which is same as the weight decay mechanism used in the AdamW algorithm [Loshchilov and Hutter, 2019].

Data Heterogeneity: Conceptually, both the two approaches aim at alleviating the impact of data heterogeneity by globally reconciling the adaptive learning rates. We call this “moment sharing”. Therefore, in some sense, Algorithm 1 is naturally capable of balancing the heterogeneity in different local data distributions. Indeed, in [Chen et al., 2020] and [Karimireddy et al., 2020], the authors have shown that Fed-AMS and Mime would perform much worse, or even diverge, without aggregating and sharing the second moment $\dot{v}$ please refer to the papers for details). Intuitively, synchronizing $\dot{v}$ makes all the clients “on the same pace” which is crucial for the convergence of locally adaptive FL methods.

Extension: skip synchronization of $\dot{v}_t$. In practice, when trained with the same number of rounds $T$, Mime, Fed-LAMB and Fed-Mime all require communicating two tensors (the local model update, and second moment $v$), while Fed-SGD [McMahan et al., 2017] and Adp-Fed [Reddi et al., 2021] only communicate one local update tensor. Hence, locally adaptive methods in general tend to require more communication. We now discuss a simple implementation trick of our algorithm that reduces this extra cost. Note that, as long as $\dot{v}_t$ is consistent across clients, we may not need to update and broadcast it in every round. To reduce the extra communication overhead of $\dot{v}_t$, one trick is to reduce the aggregation frequency of $\dot{v}$ in Algorithm 1 (e.g., we synchronize $\dot{v}$ every $Z$ rounds). It can be shown that this “skip” aggregation of the second moment does not affect the convergence rate of our Fed-LAMB (see Theorem 5). Yet it can effectively reduce the communication of $\dot{v}$ by a factor of $Z$, which to a great extent alleviates the extra communication cost of locally adaptive methods. We will also show empirical evidence of this strategy in our experiments.

4 THEORETICAL ANALYSIS

In the context of nonconvex stochastic optimization for federated learning, we will make the following standard analytical assumptions.

Assumption 1 (Smoothness). For all $i \in [n]$ and $\ell \in [h]$, the local loss function is $L_\ell$-smooth: $\|\nabla f_i(\theta^\ell) - \nabla f_i(\theta^\ell')\| \leq L_\ell \|\theta^\ell - \theta^\ell'\|$.  

Assumption 2 (Unbiased and bounded gradient). The stochastic gradient is unbiased for $\forall r, t, i$: $E[g_{r,i}^t] = \nabla f_i(\theta^t_r)$ and bounded by $\|g_{r,i}^t\| \leq M$.  

Assumption 3 (Bounded variance). The stochastic gradient admits (locally) $E[\|g_{r,i}^t - \nabla f_i(\theta^t_r)\|^2] \leq \sigma^2$, and (globally) $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(\theta_r) - \nabla f(\theta_i)\|^2 \leq G^2$.  

Figure 1: Illustration of Fed-LAMB framework (Algorithm 1), with a three-layer network and $\phi(x) = x$ as an example. For device $i$ and each local iteration in round $r$, the adaptive ratio of $j$-th layer $\psi^j_{r,i}$ is normalized according to $\|\theta^j_{r,i}\|$, and then used for updating the local model. At the end of each round $r$, client $i$ sends $\theta_{r,i} = [\theta^j_{r,i}]^h$ and $v_{r,i}$ to the central server, which transmits back aggregated $\theta$ and $\dot{v}$ to devices to complete a round of training.
Algorithm 1 Fed-LAMB and Mime-LAMB

1: Input: $0 < \beta_1, \beta_2 < 1$; learning rate $\alpha$; weight decay rate $\lambda \in [0, 1]$; frequency parameter $Z$.
2: Initialize: $\theta_{0,i} \in \Theta \subseteq \mathbb{R}^{d_i}; m_{0,i} = \tilde{v}_{0,i} = v_{0,i} = 0$,
   \hspace{.15in} \forall i \in [n]; \bar{\theta}_0 = \frac{1}{T} \sum_{t=1}^T \theta_{0,i}; \bar{v}_0 = \epsilon
3: for $r = 1$ to $R$ do
4:   Sample a set of clients $D_r$
5:   for parallel for device $i \in D_r$ do
6:     Set $\theta_{0,i} = \bar{\theta}_{r-1}$, $m_{0,i} = \tilde{v}_{r-1,i}$, $v_{0,i} = \hat{v}_{r-1,i}$
7:     for $t = 1$ to $T$ do
8:       Sample a mini-batch from the local data
9:       Compute stochastic gradient $\theta_{r,i}^t$ at $\theta_{r-1}$
10:      $m_{r,i}^t = \beta_1 m_{r-1,i} + (1 - \beta_1) g_{r,i}^t$
11:      $v_{r,i}^t = \beta_2 v_{r-1,i} + (1 - \beta_2) (g_{r,i}^t)^2$
12:      Compute the ratio $\psi_{r,i}^t = m_{r,i}^t / v_{r,i}^t$
13:      Update local model for each layer $e \in [h]$
14:      $\theta_{r,i}^t = \theta_{r-1,i} - \alpha \rho_e \left( \nabla f (\theta_{r,i}^{t-1}) + \lambda \theta_{r-1,i} \right) \psi_{r,i}^t$
15:     end for
16:     Communicate $\theta_{r,i}^T = \{ \theta_{r,i}^t \}_{t=1}^T$ to server
17:     Communicate $v_{r,i}^T$ to server
18:     Communicate $\nabla f_i (\bar{\theta}_{r-1})$ using full local data
19: end for
20: Server compute $\bar{\theta}_r = \frac{1}{|D_r|} \sum_{i \in D_r} \theta_{r,i}^T$
21: Server compute $\hat{v}_r = \max (\bar{\theta}_{r-1}, \frac{1}{|D_r|} \sum_{i \in D_r} v_{r,i}^T)$
22: Compute $\nabla f (\bar{\theta}_{r-1}) = \frac{1}{|D_r|} \sum_{i \in D_r} \nabla f_i (\bar{\theta}_{r-1})$
23: Compute $v_r = \beta_2 v_{r-1} + (1 - \beta_2) \nabla f (\theta_{r-1})$
24: Update $\hat{v}_r = \max (\bar{\theta}_{r-1}, v_r)$

Assumption 4 (Bounded scaling function). For any $a > 0$, there exist $\phi_m > 0, \phi_M > 0$ such that $\phi_m \leq \phi(a) \leq \phi_M$.

Assumption 3 can be satisfied when, for example, we let $\phi(a) = \min \{ a + \zeta, \phi_M \}$ be the identity map plus a small constant $\zeta$ with an upper clipping threshold at some $\phi_M$.

We now state our main result regarding the convergence rate of the proposed Algorithm 1

Theorem 5. Under Assumption 4 and Assumption 2, consider $\{ \bar{\theta}_r \}_{r \geq 0}$ obtained from Algorithm 1 with a constant learning rate $\alpha$. Suppose $\lambda = 0$. Then the squared gradient of the global model uniformly chosen from round 1, ..., $R$ is bounded by

\[
1 \leq \frac{1}{R} \sum_{r=1}^R \mathbb{E} \left[ \left\| \nabla f (\bar{\theta}_r) \right\|^2 \right] \\
\leq \sqrt{\frac{M^2 p \triangle}{\eta h R}} + 4\alpha^2 M^2 T^2 \phi_M^2 (1 - \beta_2) p + 4\alpha^2 \phi_M^2 \sqrt{M^2 + \sigma^2} \\
+ 4G^2 \eta \sqrt{\frac{M^2 T^2 G^2 (1 - \beta_2) p + 4\alpha \phi_M h^2 \sigma^2}{h \sqrt{n R}}} \\
= O \left( \frac{\sqrt{p} + \sqrt{\eta h R} + G^2 T^2}{\sqrt{n R}} \right)
\]  

where $\triangle = \mathbb{E}[f(\bar{\theta}_1)] - \min_{\theta \in \Theta} f(\theta)$ and $L = \frac{\sum_{t=1}^T \mathbb{L}_t}{t}$.

Remark 6. Theorem 5 applies to both Fed-LAMB and Mime-LAMB variants. Also, the manifestation of $p$ in the rate is because the variance bound is assumed on each dimension in Assumption 3. This dependency on $p$ can be removed when Assumption 3 is assumed globally, which is also common in optimization literature. Moreover, this result also holds for Algorithm 1 with skip synchronization of $\hat{v}_r$ as discussed earlier.

Using the uniform boundedness of the second moment accumulation $\| \hat{v}_r \|$ (which can be shown by Assumption 2) and by choosing a suitable decreasing learning rate, we have the following simplified statement.

Corollary 7. Under the same setting as Theorem 5 with $\alpha = O \left( \frac{1}{\sqrt{n R}} \right)$, it holds that

\[
1 \leq \frac{1}{R} \sum_{r=1}^R \mathbb{E} \left[ \left\| \nabla f (\bar{\theta}_r) \right\|^2 \right] \\
\leq O \left( \frac{\sqrt{p} + \sqrt{\eta} \sigma^2 / \sqrt{n R} + G^2 T^2 / (R h)}{\sqrt{n R}} \right)
\]

The leading two terms display a dependence of the convergence rate of Fed-LAMB on the initialization and the local variance of the stochastic gradients (Assumption 3). The last term involves the number of local updates $T$, and the global variance $G^2$ characterizing the data heterogeneity. Next, we provide detailed discussion and comparison of our result to related prior works.
that of LAMB in the sense that, the convergence criterion is given at the averaged parameters (global model) at the end of each round. In Corollary 7, our rate would match LAMB if we take number of local step \( T = 1 \). This is also true for any fixed \( T \) and \( R \) sufficiently large. In addition, the \( \mathcal{O}(\frac{1}{\sqrt{R}}) \) rate of Fed-LAMB implies a linear speedup effect that is important in distributed training: the number of iterations to reach a \( \delta \)-stationary point of Fed-LAMB decreases linearly in \( n \), which displays the merit of distributed (federated) learning.

**Fed-AMS bound in [Chen et al., 2020]:** We now compare our method theoretically with Fed-AMS, the baseline distributed adaptive method developed in [Chen et al., 2020]. Their results state that when \( T \leq \mathcal{O}(R^{1/3}) \), the convergence rate of Fed-AMS is \( \mathcal{O}(\frac{1}{\sqrt{Rn}}) \). Firstly, when the number of rounds \( R \) is sufficiently large, both our rate \( (5) \) and the rate of Fed-AMS are dominated by \( \mathcal{O}(\frac{1}{\sqrt{Rn}}) \), improving the convergence rate of the standard AMSGrad, e.g., [Zhou et al., 2018] by \( \mathcal{O}(1/\sqrt{n}) \) (i.e., linear speedup). Secondly, in (5), the last term containing the number of local updates \( T \) is small as long as \( T^4 \leq \mathcal{O}(\frac{R}{nR}) \). If we further assume \( h \simeq T \), then we get the same rate of convergence as Fed-AMS with \( T \leq \mathcal{O}(R^{1/3}) \) local iterations, identical to the condition of Fed-AMS. Under these analytic settings and conditions, the convergence rate of Fed-LAMB also matches many popular federated learning methods in nonconvex optimization, e.g., Fed-SGD [McMahan et al., 2017], Mime [Karimireddy et al., 2020] and Adp-Fed [Reddi et al., 2021]. Moreover, when \( G \) is small (less data heterogeneity), the bound on \( T \) would increase, i.e., we can conduct more local updates. This is intuitive, for example, when \( G = 0 \) in the IID data setting, \( T \) can be very large.

As a brief summary, Fed-LAMB achieves the same asymptotic convergence rate as Fed-AMS in the federated (distributed) learning setting. Our method also exhibits the favorable linear speedup property regarding the number of clients in the system. Next, we will show that Fed-LAMB and its variants provide impressive acceleration empirically in our experimental study presented next.

5 EXPERIMENTS

In this section, we conduct experiments on benchmark datasets with various network architectures to justify the effectiveness of our proposed method in practice. Our method empirically confirms its merit in terms of convergence speed. Basically, Fed-LAMB and Mime-LAMB reduce the number of rounds and thus the communication cost required to achieve a similar stationary point (or test accuracy) than the baseline methods. In many cases, Fed-LAMB also brings notable improvement in generalization over baselines.

**Methods.** We evaluate the following five FL algorithms, mainly focusing on recent federated optimization approaches based on adaptive gradient methods:

1. **Fed-SGD** [McMahan et al., 2017], standard federated averaging with local SGD updates.
2. **Adp-Fed** (Adaptive Federated Optimization, see Appendix for more details), the federated adaptive algorithm proposed by [Reddi et al., 2021]. Adp-Fed performs local SGD updates. In each round \( r \), the changes in local models, \( \Delta_i = w^r_{t,i} - w^0_{t,i}, i = 1, ..., n \), are sent to the central server for an aggregated Adam update.
3. **Fed-AMS** [Chen et al., 2020], locally adaptive AMSGrad algorithm.
4. **Mime** [Karimireddy et al., 2020] with AMSGrad, which performs adaptive local updates with central-server-guided global adaptive learning rate.
5. **Our proposed Fed-LAMB and Mime-LAMB methods** (Algorithm 1).

For all the adaptive gradient methods, we set \( \beta_1 = 0.9, \beta_2 = 0.999 \) by default [Reddi et al., 2018]. We present the results of \( n = 50 \) clients with 0.5 participation rate, i.e., we randomly pick half of the clients to be active for training in each round, and the local mini-batch size is set as 128. In each round, the training samples are allocated to the active devices, and one local epoch is completed after all the local devices run one pass over their received samples via mini-batch training. Results with more clients can be found in the Appendix, which give the same conclusions as what we will present below.

We tune the learning rate \( \alpha \) for each algorithm over a fine grid. For Adp-Fed, there are two learning rates involved (global and local), both of which are tuned. More tuning details can be found in the Appendix. For Fed-LAMB and Mime-LAMB, the weight decay rate \( \lambda \) is tuned from \{0, 0.01, 0.1\}, and \( \phi(x) = x \) is the identity mapping. For each run, we report the best test accuracy. The results are averaged over 3 runs each from a same initialization point.

**Datasets and models.** We experiment with four popular benchmark image classification datasets: MNIST [LeCun, 1998], Fashion MNIST (FMNIST) [Xiao et al., 2017], CIFAR-10 [Krizhevsky and Hinton, 2009] and TinyImageNet [Deng et al., 2009]. For MNIST, we apply 1) a simple multi-layer perceptron (MLP), which has one hidden layer containg 200 cells; 2) Convolutional Neural Network (CNN), which has two max-pooled convolutional layers followed by a dropout layer and two fully-connected layers with 320 and 50 cells respectively. This CNN is also implemented for FMNIST. For CIFAR-10 and TinyImageNet, we use ResNet-18 [He et al., 2016].
5.1 COMPARISON UNDER IID SETTINGS

In Figure 2, we report the test accuracy of MLP trained on MNIST, and of CNN trained on MNIST and FMNIST, where the data are IID allocated among the clients. We test 1 local epoch and 3 local epochs. In all the figures, we observe a clear advantage of Fed-LAMB over the competing methods in terms of the convergence speed. In particular, we can see that Fed-LAMB is able to achieve the same accuracy with fewer number of communication rounds, thus improving the model training efficiency. For instance, this can be observed as follows: on MNIST + CNN (1 local epoch), Fed-AMS requires 20 rounds to achieve 90% accuracy, while Fed-LAMB only takes 5 rounds. This implies a 75% reduction in the communication cost and training time. Moreover, on MNIST, Fed-LAMB also leads to improved generalisation performance, i.e., test accuracy. We can draw same conclusions with 3 local epochs. Also, similar comparison holds for Mime-LAMB vs. Mime. In general, the Mime-LAMB and Fed-LAMB perform similarly.

5.2 COMPARISON UNDER NON-IID SETTINGS

In Figure 3, we provide the results on MNIST and FMNIST with non-IID local data distribution. In particular, in each round of federated training, every local device only receives samples from one or two classes (out of ten). We see that for experiments with 1 local epoch, in all cases our proposed Fed-LAMB outperforms all the baseline methods. Similar to the IID data setting, Fed-LAMB provides faster convergence speed and achieves higher test accuracy than Fed-SGD and Fed-AMS. The advantage is especially significant for the CNN model, e.g., it improves the accuracy of Fed-SGD and Fed-AMS by more than 10% on FMNIST at the 50th round. The other baseline method, Adp-Fed, performs as good as our Fed-LAMB on FMNIST, but worse than other methods on MNIST. Mime-LAMB also considerably improves Mime, in all the runs, see Figure 3.

The relative comparison is basically the same for 3 local epochs, but the advantage of Fed-LAMB becomes less significant than what we observed in Figure 2 with IID data. One plausible reason is that when the local data is highly non-IID. Intuitively, with more local steps, learning the local models fast might not always do good to the global model, as local models target at different loss functions.

In Figure 4 we present the results on CIFAR-10 and TinyImageNet datasets trained by ResNet-18. When training these two models, we decrease the learning rate to 1/10 at the 30th and 70th communication round. From Figure 4, we can draw similar conclusion as before: the proposed Fed-LAMB
Adp-Fed performs better than Fed-SGD and Fed-AMS, it is most efficacious of this more efficient strategy in practice. It may converge even faster. Our results demonstrate the importance of using layer-wise adaptive learning rates. Specifically, in turn leading to reduced communication and training time by a factor of 3 or 5 rounds.

Skip synchronization. In Figure 5, we further present the results of methods with skip synchronization of $\hat{v}$, where the server updates and broadcasts $\hat{v}$ every $Z = 3, 5$ rounds, instead of in very single round. This reduces the communication cost of transmitting the second moment $v$ by a factor of $3$ or $5$. We see that, the empirical performance of skip synchronization is similar to the standard design; some times it may converge even faster. Our results demonstrate the efficacy of this more efficient strategy in practice.

is the best method in terms of both convergence speed and generalization accuracy. In particular, on TinyImageNet, we see that Fed-LAMB has a significant advantage over all the four baselines without layer-wise acceleration. Although Adp-Fed performs better than Fed-SGD and Fed-AMS, it is considerably worse than Fed-LAMB. We report the test accuracy at the end of training in Table 1. Fed-LAMB achieves the highest accuracy on both datasets. Mime-LAMB also substantially improves Mime.

5.3 SUMMARY OF EMPIRICAL FINDINGS

To sum up, we provide a brief summary of our empirical results. On all datasets, the primary comparison of most interest appears evident:


The proposed scheme (with two variants Fed-LAMB and Mime-LAMB) exhibits faster convergence and better generalization accuracy than recently proposed adaptive FL algorithms. Our results suggest that, using layer-wise acceleration in the local training can speedup the overall model performance of locally adaptive federated learning. Moreover, in practice we may adopt the skip aggregation strategy to further reduce the additional communication required for our proposed approach, without losing utility. As discussed earlier, Mime-LAMB typically requires more gradient computation than Fed-LAMB. Therefore, with similar performance as Mime-LAMB, the Fed-LAMB protocol might be more efficient and convenient in practical applications.

6 CONCLUSION

We study a doubly adaptive method in the particular framework of federated learning (FL). Built upon the acceleration effect of layer-wise learning rate scheduling and of state-of-the-art adaptive gradient methods, we derive a locally layer-wise FL framework that performs local updates using adaptive AMSGrad on each worker and periodically averages local models stored on each device. The core of our Fed-LAMB scheme, is to speedup up local training by adopting layer-wise adaptive learning rates. To our knowledge, this is the first FL algorithm in literature that possess both the dimension-wise adaptivity (by AMSGrad) and layer-wise adaptivity (by layer-wise adjusted learning rate). We provide the convergence analysis of Fed-LAMB that matches many existing methods, with a linear speedup against the number of clients. We also provide a skip aggregation trick to further reduce the communication overhead. Extensive experiments on various datasets and models, under both IID and non-IID data settings, validate that both Fed-LAMB and Mime-LAMB are able to provide faster convergence which in turn leads to reduced communication and training time to reach a certain accuracy. In many cases, our framework also improves the overall performance of federated learning over prior methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>90.75 ± 0.48</td>
<td>91.57 ± 0.38</td>
<td>90.93 ± 0.22</td>
<td>92.44 ± 0.53</td>
<td>90.94 ± 0.13</td>
<td>92.00 ± 0.21</td>
</tr>
<tr>
<td>TinyImageNet</td>
<td>67.58 ± 0.21</td>
<td>74.17 ± 0.43</td>
<td>64.86 ± 0.83</td>
<td>76.00 ± 0.26</td>
<td>67.82 ± 0.24</td>
<td>73.46 ± 0.25</td>
</tr>
</tbody>
</table>

Table 1: Test accuracy with ResNet-18 network after 100 communication rounds.
References


