# Layerwise and Dimensionwise Locally Adaptive Optimization

**Anonymous Author**
Anonymous Institution

## 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. Due to the low bandwidth of mobile devices, decentralized optimization methods need to shift the computation burden from those clients to the computation server while preserving *privacy* and reasonable *communication cost*. In this paper, we focus specifically on the training of deep neural networks, under the FL framework. We present FED-LAMB a novel Federated Learning method based on a *layerwise* and *dimensionwise* updates of the local models, alleviating the nonconvexity and the multilayeredness of the optimization task at hand. We provide detailed convergence analysis for FED-LAMB and compare it with other existing methods. Finally, we provide extensive experimental results with both iid and non-iid local data distribution to validate the theory and demonstrate that the proposed algorithm exhibits faster convergence speed, which could lead to reduced communication cost. In addition, in many cases, FED-LAMB also achieves improved generalization accuracy.

## 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 devices or distinct entities. In the paradigm of Federated Learning (FL) [10; 18], a central server orchestrates the optimization over those clients under the constraint that the data can neither be centralized nor shared among the clients. This is more computationally efficient, since more 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 the private data. Most modern machine learning tasks can be casted as a large finite-sum optimization problem written as:

$$\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)\,, \tag{1}$$

where $n$ denotes the number of workers, $f_i$ represents the average loss for worker $i$ and $\theta$ the global model parameter taking value in $\Theta$, a subset of $\mathbb{R}^p$. While this formulation recalls that of standard distributed optimization, the core principle of FL is different from the traditional distributed paradigm in that FL allows local models to perform multiple local updates before the global aggregation.

In this paper, we mainly address two important aspects of FL: communication efficiency and model performance. While local updates can effectively reduce the number of communication rounds between the central server and devices, new techniques are still necessary to tackle the challenge of communication between devices and server, due to, e.g., wireless bandwidth. Some quantization [1; 28], compression [17] and sketching [25] methods allow to decrease the number of bits communicated at each round and are efficient methods in a distributed setting. Another direction to improve FL methods is to design better algorithms to accelerate local training, such that better local models are sent to the server at each round. This could lead to reduced communication (to reach a certain accuracy) as well as possibly improved overall learning performance.

One of the most popular frameworks for FL is called Fed-Avg [18]: we adopt multiple local Stochastic Gradient Descent (SGD) steps in each device, send those local models to the server that computes the average over those received local model parameters, and broadcasts it back to the devices. As mentioned above, momentum can be added to the local SGD training for faster convergence and better learning performance [30]. In this work, we focus on an alternative framework as

Preliminary work. Under review by AISTATS 2022. Do not distribute.

proposed by Chen et al. [2], a popular adaptive optimization method, is adopted locally instead of SGD. The authors showed that the so-called "Local AMSGrad" algorithm has communication cost sublinear in $R$, that is guaranteed to converge to a stationary point with rate $\mathcal{O}(\sqrt{p/Rn})$, where $R$ is the number of communication rounds, $p$ is the model dimensionality and $n$ corresponds to the number of federated clients. Specifically, in Local AMS, each round the global server not only aggregates the local models, but also averages and broadcasts the local second moment estimations, which is a crucial ingredient in AMSGrad controlling the dimensionwise learning rates. Thus, this step can be regarded as a natural remedy to data heterogeneity (i.e., the data on local workers follows different probability distributions), which is a common scenario in practice that affects the performance of FL algorithms [15; 16; 8].

**Contributions.** Based on the recent progress in accelerating adaptive methods for efficient training You et al. [29], we propose an improved variant of Local AMSGrad [2], integrating both dimensionwise and layerwise adaptive learning rates in each device's local update. More specifically:

- We develop Fed-LAMB, a novel optimization algorithm for federated learning, following a principled layerwise adaptation strategy to accelerate training of deep neural networks. Our method is provably and empirically communication-efficient for compositional structural deep models.
- We provide theoretical analysis on the non-asymptotic convergence rate of Fed-LAMB. Our rate, $\mathcal{O}\left(\sqrt{\frac{p}{n}}\frac{1}{\sqrt{\mathsf{h}R}}\right)$, where $\mathsf{h}$ is the total number of layers and $p$ denotes the dimension, matches the state of the art methods in federated learning and reaches a sublinear convergence in the total number of communication rounds. It also improves the theoretical communication efficiency compared with the baseline Local AMSGrad.
- We demonstrate the advantages of our method to reach similar, or better, test accuracy than the baseline SGD and Local AMS methods with less number of communication rounds, on several benchmarks supervised learning methods on both homogeneous and heterogeneous settings, various benchmark datasets such as CIFAR-10 and TinyImagenet and models including Convolutional Neural Networks and ResNets.

**Roadmap.** After establishing a literature review of both realms of federated and adaptive learning in Section 2, we develop in Section 3, our method, namely Fed-LAMB, based on the computation per layer and per dimension, of the adaptive learning rate in the traditional AMSGrad. Theoretical understanding of our method's behaviour with respect to convergence towards a stationary point is developed in Section 4 in nonconvex optimization. We present numerical experiments showing the effectiveness and advantages of our method in Section 5.

## 2 Related Work

Below, we summarize some relevant works on federated learning and adaptive optimization.

**Adaptive gradient methods.** Adaptive methods have proven to be the spearhead in 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 AMSGrad [23], Adam [9], RMSprop [27], Adadelta [31], and Nadam [4]. Their popularity and efficiency are due to their great performance at training deep neural networks. They generally combine the idea of adaptivity from AdaGrad [5; 19], as explained above, and the idea of momentum from Nesterov's Method [20] or Heavy ball method [21] using past gradients. AdaGrad displays a great edge when the gradient is sparse compared to other classical methods. The anisotropic nature of this update represented a real breakthrough in the training of high dimensional and nonconvex loss functions. This adaptive learning rate helps accelerate the convergence when the gradient vector is sparse [5]. Yet, when applying AdaGrad to train deep neural networks, it is observed that the learning rate might decay too fast, see Kingma and Ba [9] for more details. Consequently, [9] develops Adam leveraging a moving average of the gradients divided by the square root of the second moment of this moving average (element-wise multiplication). A variant, called AMSGrad described in Reddi et al. [23] ought to fix Adam failures using a max operator. Beyond improving the convergence speed of optimization methods, several studies including Zhou et al. [35] also focus on improving their generalization properties. A natural extension of AMSGrad has been developed in You et al. [29] specifically for multi layered neural network where a principled layerwise adaptation strategy is employed to accelerate training of deep neural networks using large mini-batches using either SGD or adaptive method under the setting of a classical single server. In simple terms, the idea is based on the observation that in a large deep neural network, the magnitude of the gradient might be too small in comparison with the magnitude of the weight for some layers of the

model, hence slowing down the overall convergence. As a consequence, layerwise adaptive learning rate is applied, such that in each iteration the model can move sufficiently far. This method empirically speeds up the convergence significantly in classical sequential models and can be provably faster than baseline methods.

**Federated learning.** An extension of the well known parameter server framework, where a model is being trained on several servers in a distributed manner, is called federated learning (FL), see Konečnỳ et al. [10]. Here, the central server only plays the role of computing power for aggregation and global update of the model. Compared with the distributed learning paradigm, in federated learning, the data stored in each worker must not be seen by the central server – preserving privacy is key – and the nature of those workers (e.g., mobile devices), combined with their usually large amount, makes communication between the devices and the central server less appealing – communication cost needs to be controlled. Thus, while traditional distributed gradient methods [22; 14; 32] do not respect those constraints, it has been proposed in McMahan et al. [18], an algorithm called Federated Averaging – Fed-Avg – extending parallel SGD with local updates performed on each device. In Fed-Avg, each worker updates their own model parameters locally using SGD, and the local models are synchronized by periodic averaging on the central parameter server.

## 3 Layerwise and Dimensionwise Adaptive Method

**Notations.** We denote by $\theta$ the vector of parameters taking values in $\mathbb{R}^d$. For each layer $\ell \in [\![\mathsf{h}]\!]$, where $\mathsf{h}$ is the total number of layers of the neural networks, and each coordinate $j \in [\![p_\ell]\!]$ where $p_\ell$ is the dimension per layer $\ell$ ($p := \sum_{\ell=1}^{\mathsf{h}} p_\ell$ denotes the total dimension). We also note $\theta_{r,i}^{\ell,t}$ its value for layer $\ell$ at round $r$, local iteration $t$ and for worker $i$. The gradient of $f$ with respect to $\theta^\ell$ is denoted by $\nabla_\ell f(\theta)$. The smoothness per layer is denoted by $L_\ell$ for each layer $\ell \in [\![\mathsf{h}]\!]$. We note by $D^r$ for each communication $r > 0$, the set of randomly drawn devices performing local updates.

### 3.1 AMSGrad, Local AMSGrad and Periodic Averaging

Under the federated learning setting, we stress on the importance of reducing, at each round, the communication cost between the central server, used mainly for aggregation purposes, and the many clients used for gradient computation and local updates. Using Periodic Averaging after few local epochs, updating local models on each device, as developed in McMahan et al. [18] is the gold standard for achieving such communication cost reduction. Intuitively, one rather shifts the computation burden from the many clients to the central server as much as possible. This technique allows for fewer local epochs and a better global model, from a loss minimization (or model fitting) perspective. The premises of that new paradigm are SGD updates performed locally on each device then averaged periodically, see Konečnỳ et al. [10]; Zhou and Cong [34]. The heuristic efficiency of local updates using SGD and periodic averaging has been studied in Stich [26]; Yu et al. [30] and shown to reach a similar sublinear convergence rate as in the standard distributed optimization settings. Then, with the growing need of training far more complex models, e.g., deep neural networks, several efficient methods, built upon adaptive gradient algorithms, such as Local AMSGrad in Chen et al. [2], extended both empirically and theoretically, the benefits of performing local updates coupled with periodic averaging.

### 3.2 Layerwise and Dimensionwise Learning with Periodic Averaging

Recall that our original problem is the following optimization task $\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n f_i(\theta)$ where $f_i(\theta)$ is the loss function associated to the client $i \in [\![n]\!]$ and is parameterized, in our paper, by a deep neural network. The stacking and nonconvex nature of the loss function imply having recourse to particular optimization methods in order to efficiently train our model. Besides, the decentralized clients low-bandwidth constraints are strong motivations for improving existing methods for (1).

Based on the periodic averaging and local AMSGrad algorithms, presented prior, we propose a layerwise and dimensionwise local AMS algorithm which is detailed in Algorithm 1 and depicted in Figure 1. The proposed algorithm is a natural adaptation of the vanilla AMSGrad method, for *multilayer* neural networks under the *federated* setting. In particular, while Line 8 and Line 9 corresponds to the standard approximation of the first and second moments, via the smooth updates allowed by the tuning parameters $\beta_1$ and $\beta_2$ respectively and that both Line 1 and Line 1 are correct the biases of those estimates, the final local update in (2) is novel and corresponds to the specialization per layer of our federated method.

Note that a scaling factor is applied to the learning rate $\alpha_r$ at each round $r > 0$ via the quantity $\phi(\|\theta_{r,i}^{\ell,t-1}\|)$ depending on the dimensionwise and layerwise quantity computed in Line 12. This function is user designed and can be, for instance, set to the identity function. In other words, we normalize the gradient in each layer according to the magnitude of the layer's weight. The

**Algorithm 1** FED-LAMB

1: **Input**: parameter $0 < \beta_1, \beta_2 < 1$, and learning rate $\alpha_t$, weight decaying parameter $\lambda \in ]0, 1[$.
2: Init: $\theta_0 \in \Theta \subseteq \mathbb{R}^d$, as the global model and $\hat{v}_0 = v_0 = \epsilon \mathbf{1} \in \mathbb{R}^d$ and $\bar{\theta}_0 = \frac{1}{n}\sum_{i=1}^n \theta_0$.
3: **for** $r = 1$ to $R$ **do**
4:  **for** parallel for device $i \in D^r$ **do**
5:  Set $\theta_{r,i}^0 = \bar{\theta}_{r-1}$.
6:  Set $v_{r,i}^t = \hat{v}_{r-1}$.
7:  **for** $t = 1$ to $T$ **do**
8:  Compute stochastic gradient $g_{r,i}^t$ at $\theta_{r,i}^0$.
9:  $m_i^t = \beta_1 m_i^{t-1} + (1-\beta_1) g_{r,i}^t$ and $m_i^t = m_i^t / (1-\beta_1^t)$.
10:  $v_{r,i}^t = \beta_2 v_{r-1,i}^t + (1-\beta_2)(g_{r,i}^t)^2$ and $v_{r,i}^t = v_{r,i}^t / (1-\beta_2^t)$.
11:  Compute the ratio $p_{r,i}^t = m_i^t / (\sqrt{\hat{v}_r^t} + \epsilon)$.
12:  Update local model for each layer $\ell \in [\![\mathsf{h}]\!]$:

$$\theta_{r,i}^{\ell,t} = \theta_{r,i}^{\ell,t-1} - \alpha_r \phi(\|\theta_{r,i}^{\ell,t-1}\|) \frac{p_{r,i}^{\ell,t} + \lambda \theta_{r,i}^{\ell,t-1}}{\|p_{r,i}^{\ell,t} + \lambda \theta_{r,i}^{\ell,t-1}\|}. \tag{2}$$

13:  **end for**
14:  Devices send $\theta_{r,i}^T = [\theta_{r,i}^{\ell,T}]_{\ell=1}^{\mathsf{h}}$ and $v_{r,i}^T$ to server.
15:  **end for**
16:  Server computes averages of the local models $\bar{\theta}_r = [\bar{\theta}_r^{\ell,T}]_{\ell=1}^{\mathsf{h}} = [\frac{1}{n}\sum_{i=1}^n \theta_{r,i}^{\ell,T}]_{\ell=1}^{\mathsf{h}}$ and $\hat{v}_{r+1} = \max(\hat{v}_r, \frac{1}{n}\sum_{i=1}^n v_{r,i}^T)$ and send them back to the devices.
17: **end for**
18: **Output**: Vector of parameters $\bar{\theta}_R = [\bar{\theta}_R^{\ell,T}]_{\ell=1}^{\mathsf{h}}$.

adaptivity of our federated learning method is thus manifold. There occurs a normalization per dimension with respect to the square root of the second moment used in adaptive gradient methods and a layerwise normalization obtained via the final local update (Line 13).

## 4 Convergence of Fed-LAMB

In this section, we develop the theoretical analysis of Algorithm 1. Based on classical result for stochastic nonconvex optimization, we present a collection of results that aims to providing a better understanding of the convergence behavior of our distributed optimization method under the federated learning framework. The main challenges we ought to overcome are manifold: (i) The large amount of decentralized workers working solely on their own data stored locally. (ii) A periodic averaging occurs on the central server pushing each of those clients to send local models after some local iterations. (iii) Each client computes a backpropagation of the main model, i.e., the deep neural network,

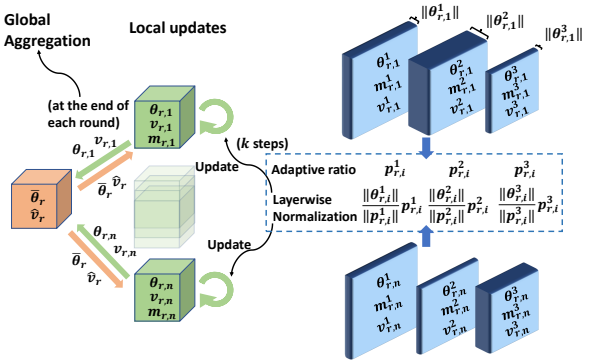


Figure 1: Illustration of FED-LAMB (Algorithm 1), with a three-layer network and $\phi(x) = x$ as an example. For device $i$ and each local iteration in round $r$, the adaptive ratio of $j$-th layer $p_{r,i}^j$ is normalized according to $\|\theta_{r,i}^j\|$, and then used for updating the local model.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

and then updates its local version of the model via an adaptive gradient method: the distinctiveness being that those updates are done *dimensionwise* and *layerwise*. Our analysis encompasses the consideration of those challenges and leads to an informative convergence rates depending on the quantities of interest in our problem: the number of layers of the DNN, the number of communications rounds and the number of clients used under our federated settings.

### 4.1 Finite Time Analysis of Fed-LAMB

In the sequel, the analysis of our scheme we provide is *global*, in the sense that it does not depend on the initialization of our algorithm, and *finite-time*, meaning that it is true for any arbitrary number of communication rounds, in particular small ones. In the particular context of nonconvex stochastic optimization for distributed clients, we assume the following:

**H1.** *(Smoothness per layer) For $i \in [\![n]\!]$ and $\ell \in [\![L]\!]$:* $\left\|\nabla f_i(\theta^\ell) - \nabla f_i(\vartheta^\ell)\right\| \leq L_\ell \left\|\theta^\ell - \vartheta^\ell\right\|$.

We add some classical assumption on the gradient of the objective function:

**H2.** *(Unbiased and Bounded gradient) The stochastic gradient is unbiased for any iteration $r > 0$: $\mathbb{E}[g_r] = \nabla f(\theta_r)$ and is bounded from above, i.e., $\|g_t\| \leq M$.*

**H3.** *(Bounded variance) The variance of the stochastic gradient is bounded for any iteration $r > 0$ and any dimension $j \in [\![d]\!]$: $\mathbb{E}[|g_r^j - \nabla f(\theta_r)^j|^2] < \sigma^2$.*

**H4.** *(Bounded Scale) For any $a \in \mathbb{R}_+^*$, there exists strictly positive constants such that $\phi_m \leq \phi(a) \leq \phi_M$.*

Two important Lemmas are required in the proof of the Theorem above. We also report the complete proof of our bound in the Appendix of this paper.

The first result gives a characterization of the gap between the averaged model, that is computed by the central server in a periodic manner, and each of the local models stored in each client $i \in [\![n]\!]$.

**Lemma 1.** *Consider $\{\overline{\theta_r}\}_{r>0}$, the sequence of parameters obtained running Algorithm 1. Then for $i \in [\![n]\!]$ and $r > 0$, the gap $\|\overline{\theta_r} - \theta_{r,i}\|^2$ satisfies:*

$$\|\overline{\theta_r} - \theta_{r,i}\|^2 \leq \alpha_r^2 M^2 \phi_M^2 \frac{(1-\beta_2)p}{v_0},$$

*where $\phi_M$ is defined in H4 and $p$ is the total number of dimensions $p = \sum_{\ell=1}^{\mathsf{h}} p_\ell$.*

The gap is provably bounded by some quantities of interest such as the total dimension of the multilayered model $p$, the learning rate and the assumed upper bound of the gradient, see H2.

Note that the end goal is to characterize how fast the gradient of the averaged/global parameter $\overline{\theta_r}$ goes to zero, but not the averaged gradient. The following Lemma allows us to convert the suboptimality condition $\left\| \frac{\overline{\nabla f(\theta_r)}}{\sqrt{v_r^t}} \right\|$ to the desired one which is $\left\| \frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}} \right\|$.

**Lemma 2.** *Consider $\{\overline{\theta_r}\}_{r>0}$, the sequence of parameters obtained running Algorithm 1. Then for $r > 0$:*

$$\left\| \frac{\overline{\nabla f(\theta_r)}}{\sqrt{v_r^t}} \right\|^2 \geq \frac{1}{2} \left\| \frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}} \right\|^2 - \overline{L} \alpha^2 M^2 \phi_M^2 \frac{(1-\beta_2)p}{v_0},$$

*where $M$ is defined in H2, $p = \sum_{\ell=1}^{\mathsf{h}} p_\ell$ and $\phi_M$ is defined in H4.*

We now state our main result regarding the non asymptotic convergence analysis of our Algorithm 1 for multiple local updates and true for any communication rounds number $R$.

**Theorem 1.** *Assume **H1-H4**. Consider $\{\overline{\theta_r}\}_{r>0}$, the sequence of parameters obtained running Algorithm 1 with a constant learning rate $\alpha$. Let the number of local epochs be $T \geq 1$ and $\lambda = 0$. Then, for any round $R > 0$, we have*

$$\begin{aligned}
&\frac{1}{R}\sum_{r=1}^{R} \mathbb{E}\left[ \left\| \frac{\nabla f(\overline{\theta_t})}{\hat{v}_t^{1/4}} \right\|^2 \right] \leq \sqrt{\frac{M^2 p}{n}} \frac{\mathbb{E}[f(\bar{\theta}_1)] - \min_{\theta \in \Theta} f(\theta)}{\mathsf{h} \alpha R} \\
&+ 4\alpha \left[ \frac{\alpha^2 L_\ell}{\sqrt{v_0}} M^2 (T-1)^2 \phi_M^2 (1-\beta_2) p + \phi_M^2 \sqrt{M^2 + p\sigma^2} \right] \\
&+ 4\alpha \frac{M^2}{\sqrt{v_0}} + \frac{\phi_M \sigma^2}{Rn} \sqrt{\frac{1-\beta_2}{M^2 p}} + 4\alpha \left[ \phi_M \frac{\mathsf{h}\sigma^2}{\sqrt{n}} \right] + cst.
\end{aligned} \tag{3}$$

By choosing a suitable learning rate, we have the following simplified result.

**Corollary 1.** *Under the same setting as Theorem 1, with $\alpha = \mathcal{O}(\frac{1}{\sqrt{\mathsf{h}R}})$, it holds that*

$$\begin{aligned}
&\frac{1}{R}\sum_{r=1}^{R} \mathbb{E}\left[ \left\| \frac{\nabla f(\overline{\theta_t})}{\hat{v}_t^{1/4}} \right\|^2 \right] \\
&\leq \mathcal{O}\left( \sqrt{\frac{p}{n}} \frac{1}{\sqrt{\mathsf{h}R}} + \frac{\sigma^2}{Rn\sqrt{p}} + \frac{(T-1)^2 p}{\mathsf{h}^3 R^{3/2}} \right).
\end{aligned} \tag{4}$$

### 4.2 Comparisons

We dedicate the following paragraph to a discussion on the bound derived above in comparison with known results most relevant to our interest in the literature.

**LAMB bound in You et al. [29]:** We first start our discussion with the comparison of convergence rate of FED-LAMB with that of LAMB, Theorem 3 in You et al. [29]. The convergence rates of FED-LAMBand LAMB differ in two ways: (i) First, note that the characterization, on the suboptimality, or convergence criterion, is given at the averaged parameters noted $\overline{\theta_r}$ due to our distributed settings. It is thus natural to consider the evolution of our objective function, precisely its gradient, evaluated at some global model values –as opposed to the outcome of a single step drift in the central server paradigm. Besides, for ease of interpretation, the LHS of (3) is summed over all rounds instead of a fictive random termination point. A simple calculation would lead to such characterization, found in several nonconvex stochastic optimization paper such as [6]. (ii) Assuming that the convergence criterion in both Theorems is of similar order (which happens for a large enough number of rounds), convergence rate of FED-LAMB displays a similar $\mathcal{O}(1/R)$ behaviour for the initialization term, meaning that, despite the distributed (federated) settings, our dimensionwise and layerwise method benefits from the double adaptivity phenomenon explained above and exhibited in the LAMB method of [29], under a central server setting.

**Local-AMS bound in Chen et al. [2]:** We now discuss the similarities and differences between local-AMS, the distributed adaptive method developed in Chen et al. [2], and our *layerwise federated* method, namely FED-LAMB. For clarity, we recall their main result under our notations:

**Theorem 2** (Theorem 5.1 in Chen et al. [2])**.** *Under some regularity conditions on the local losses and similar assumption as ours, with $\alpha = \mathcal{O}(\sqrt{n}/\sqrt{Rp})$, Local-AMS has the following convergence rate:*

$$\frac{1}{R}\sum_{r=1}^{R} \mathbb{E}\left[ \left\| \frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}} \right\|^2 \right] \leq \mathcal{O}\left( \sqrt{\frac{p}{Rn}} (1+\sigma^2) + \frac{n(T-1)^2}{R} \right)$$

*where $\epsilon$ corresponds to the initialization of $\hat{v}_0$ .*

Firstly, the leading two terms of their result and ours displays a dependence of the convergence rate on the initialization and the bounded variance assumption of the stochastic gradient, see H3. In general, when the number of rounds $R$ is sufficiently large, both rates are dominated by $\mathcal{O}(\frac{\sqrt{p}}{\sqrt{nR}})$, matching the convergence rate of AMSGrad (e.g. [33]). The acceleration of our layerwise scheme is exhibited in the $\mathcal{O}(1/(nR))$ term and the dependence on the number of layers $\mathcal{O}(1/(h^3R^{3/2}))$ term. Note that the boundedness assumption is done on each dimension in H3 and leads to a manifestation of the term $\sqrt{p}$ in both rates. This dependency on $p$ can be removed when H3 is assumed globally, which is also common in optimization literature.

In (4), the last term containing the number of local updates $T$ is small as long as $T \leq \mathcal{O}(\frac{R^{1/2}h^{5/4}}{(np)^{1/4}})$. Treating $p^{1/4}/h = \mathcal{O}(1)$, the result implies that we can get the same rate of convergence as vanilla AMSGrad, with $R/T \geq \mathcal{O}(R^{1/2}n^{1/4})$ rounds of communication. For Local-AMS, by similar argument we know that, it requires $\mathcal{O}(R^{3/4}n^{3/4})$ communication rounds. Thus, our result reduces the number of communication rounds needed to reach a $\epsilon$-stationary point compared with [2], hence improving the communication efficiency.

### 4.3 Discussions

We provide more discussion on the algorithmic and theoretical properties of Fed-LAMB from following the aspects:

**Communication and Client Efficiency:** The (sublinear) dependence on the number of communication rounds of our bound matches that of most recent methods in federated learning, e.g. SCAFFOLD Karimireddy et al. [8], a solution to the problem posed by heterogeneity of the data in each client. Yet, contrary to SCAFFOLD, our method only sends bits once per communication round while SCAFFOLD needs to send two vectors, including an additional control variate term from the clients to the central server. Our result also matches the communication bound of [24] which adapts ADAM [9] to the federated setting. However, the algorithm of [24] performs adaptive updates only at the central server, while SGD is still used for local updates. In addition, the $1/\sqrt{n}$ term in convergence rate of Fed-LAMB implies a linear speedup effect in the number of local workers, which also matches the rates of most federated learning methods.

**Data Heterogeneity:** To demonstrate the effect of non-iid data distribution theoretically, some related works pose assumptions on the global variance and the local variance of the stochastic gradients of the objective function (1) separately, such that both variances appears in the convergence rate. Our analysis can also be easily extended to incorporate the global variance term. While some works (e.g.,Karimireddy et al. [8]) target on designing specific strategies to alleviate the negative influence of data heterogeneity, we note that our Fed-LAMB are, in some sense, naturally capable of balancing the heterogeneity in different local data distributions. As mentioned before, this is largely due to the "moment averaging" step (line 16 in Algorithm 1), where the adaptive learning rates are aggregated periodically. In our experiments, we will see that the advantage of Fed-LAMB is consistent under both homogeneous and heterogeneous data setting.

## 5 Numerical Experiments

In this section, we conduct numerical experiments on various datasets and network architectures to justify the effectiveness of our proposed method in practice. Our main objective is to validate the benefit of dimensionwise adaptive learning when integrated with the locally adaptive FL method. We observe that our method empirically confirms its edge in terms of convergence speed. Basically, Fed-LAMB reduces the number of rounds and thus the communication cost required to achieve a similar stationary point (or test accuracy) than baseline methods. In many cases, Fed-LAMB could also bring significant improvement in the generalization performance.

**Settings.** In our experiment, we will evaluate three federated learning algorithms: 1) Fed-SGD [18], federated averaging of local SGD, 2) Fed-AMS [2], locally adaptive FL, and 3) our proposed Fed-LAMB (Algorithm 1), where the first two serve as the baseline methods. For adaptive methods 2) and 3), we set $\beta_1 = 0.9$, $\beta_2 = 0.999$ as the recommended default [23]. Regarding the federated learning environment, we use 50 local workers with 0.5 participation rate. That is, we randomly pick half of the workers to be active for training in each round. To best accord with real scenarios where the local training batch size is usually limited, we set the local update batch size as 32. In each round, the training samples are allocated to the active devices, and one local epoch is finished after all the local devices run one epoch over their received samples by batch training. We test different number of local epochs in our experiments.

For each dataset and number of local epochs, we tune the initial learning rate $\alpha$ for each algorithm over a fine grid. For Fed-LAMB the parameter $\lambda$ in Algorithm 1 controlling the overall scale of the layerwise gradients is tuned from $\{0, 0.01, 0.1\}$, and use the identity function for $\phi(\cdot)$. For each run, we report the test accuracy

with the best $\alpha$ and $\lambda$. The results are averaged over three independent runs, each with same initialization for every method.

**Datasets and models.** We run our experiments on the popular benchmark MNIST [13] and CIFAR10 [11] image classification datasets. The MNIST dataset contains 60000 training samples and 10000 test sample, from 10 classes. The CIFAR10 dataset has 50000 training images and 10000 test images, from 10 classes. The TinImageNet is a subset of ImageNet [3], consisting of $64 \times 64$ images from 200 classes. For MNIST, we apply 1) a simple multilayer perceptron (MLP), which has one hidden layer containing 200 cells with dropout; 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. For CIFAR10, we implement: 1) a CNN with three convolutional layers followed by two fully-connected layers, and 2) ResNet-9 proposed by [7]. Additionally, a larger ResNet-18 model is tested on CIFAR10 and TinyImageNet.

### 5.1 Comparison under iid setting

In Figure 2, we report the test accuracy of ResNet-9 trained on CIFAR10 dataset, where the data is i.i.d. allocated among clients. When we run 1 local epoch per device, we observe a clear advantage of FED-LAMB over Fed-AMS on both the test accuracy and the convergence speed. In particular, FED-LAMB uses fewest number of communication rounds to achieve a relatively stable accuracy. Increasing the number of local epochs in each device per communication round leads to similar observations: the proposed FED-LAMB converges the fastest, with similar generalization error as the baseline Fed-AMS.

### 5.2 Comparison under non-iid settings

In Figure 3, we provide the test accuracy on MNIST and CIFAR10 dataset using either a MLP, CNN or ResNet-9 architecture. We compare each method under the heterogeneous (non-iid) data distribution setting. In particular, in each round of federated training, every device only receives samples from one class (out of ten). This is known to be the scenario where federated learning is harder to generalize well McMahan et al. [18]. First of all, we observe that on MNIST, using a simple MLP model, our proposed FED-LAMB outperforms Fed-AMS and Fed-SGD with both 1 and 3 local epochs. This to a good extent illustrates the benefit of layerwise adaptivity in FED-LAMB since compared with Fed-AMS, the only critical difference is that the learning rate becomes adaptive to the scale of the single hidden layer. In addition, FED-LAMB over-

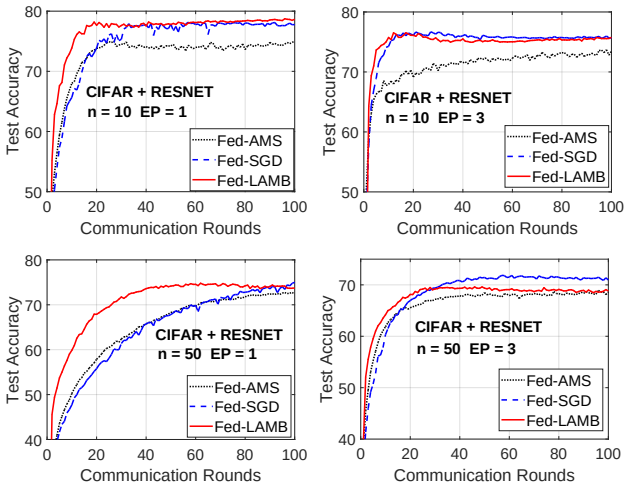


Figure 2: **i.i.d. data setting**. **Top row:** Test accuracy on CIFAR+ResNet-9, with iid data distribution for 10 clients. **Middle row:** Test accuracy on CIFAR+ResNet-9 with iid data distribution for 50 clients. **Left panel:** 1 local epoch. **Right panel:** 3 local epochs.

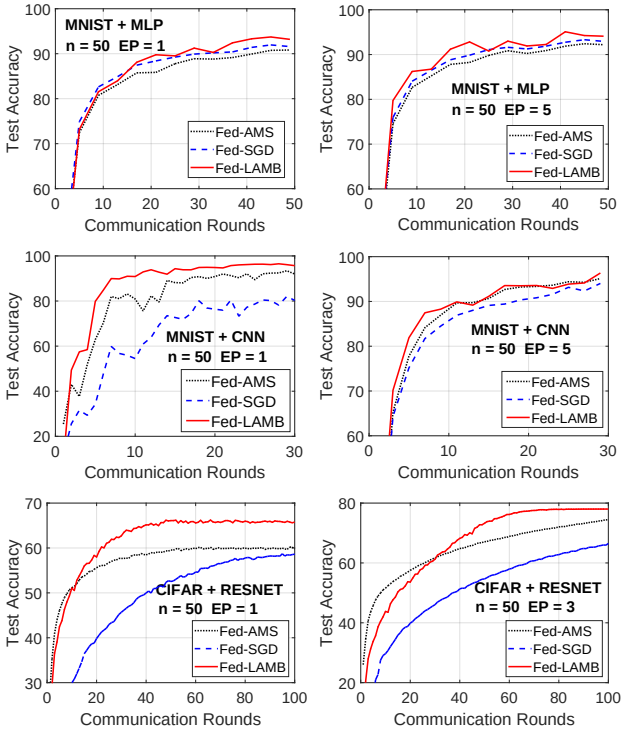


Figure 3: **non-i.i.d. data setting. Top row:** Test accuracy on MNIST+MLP. **Middle row:** Test accuracy on MNIST+CNN. **Bottom row:** Test accuracy on CIFAR+ResNet-9. **Left panel:** 1 local epoch. **Right panel:** 3 local epochs. 50 federated clients.

achieves both baseline methods on this task in terms of generalization accuracy.

For MNIST with CNN, under this non-iid data distribution, we see that FED-LAMB outperforms Fed-AMS in both cases. The advantage is especially significant

with 1 local epoch, where Fed-SGD generalizes poorly, see the left panel of Figure 3. Importantly, we again would like to address the acceleration effect of FED-LAMB in the early stage of training. FED-LAMB converges much faster than the vanilla Fed-AMS at first few communication rounds, in both cases. As a side note, the poor performance of Fed-SGD is, to some extent, consistent with prior literature and practical numerical experiments showing that the standard periodical averaging of SGD usually performs worse with highly non-iid data [2; 24]. Also, we see that FED-LAMB shows more substantial advantage with CNN than with the simple MLP, since the layerwise adaptive learning is specifically designed for multi-layer deep learning networks. This can also be expected from Corollary 1, where the convergence bound decreases with more layers $\mathsf{h}$.

We test the algorithms on CIFAR10 using a ResNet-9 model. Similarly, we observe in Figure 3 that FED-LAMB improves the performance of vanilla Fed-AMS method under various settings with different number of clients and number local epochs, as shown by its remarkable improvement in both the convergence speed and the test accuracy over both baseline methods. In all our empirical results, due to the faster convergence, FED-LAMB requires much fewer communication rounds to achieve satisfactory learning performance, validating its key advantage in practical training systems.

Table 1: Test Accuracies for larger models.

| | Fed-SGD | Fed-AMS | **Fed-LAMB** |
|---|---|---|---|
| CIFAR10 + Resnet18 | 90.75 ± 0.48 | 90.93 ± 0.22 | 92.44 ± 0.53 |
| Tiny-Imagenet + Resnet18 | 67.58 ± 0.21 | 64.86 ± 0.83 | 76.00 ± 0.26 |

Lastly, In Figure 4, we present the test accuracy on CIFAR-10 and TinyImageNet [12] dataset using a larger ResNet-18 architecture, with 50 clients with partial participation and 1 local epoch. Again, we see that FED-LAMB outperforms Fed-SGD and Fed-AMS. For reference, we report the overall test accuracy at the end of training in Table 1. FED-LAMB achieves the highest accuracy on both datasets. For example, on TinyImagenet, FED-LAMB reaches 93.50% after 100 communication rounds, against 79.63% for Fed-SGD.

## 6 Conclusion

In this paper, we study a doubly adaptive method in the particular framework of federated learning. Built upon the success of periodic averaging, and of state-of-the-art adaptive gradient methods for single server non-convex stochastic optimization, we derive FED-LAMB, a distributed AMSGrad method that performs local updates on each worker and periodically averages local models. Besides, when the trained model is a deep

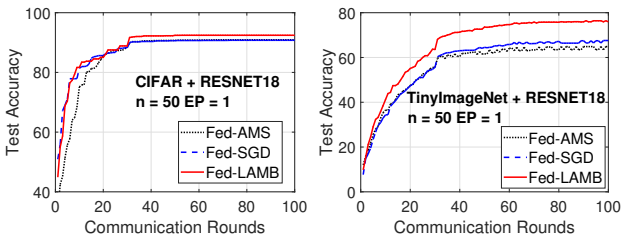


Figure 4: **non-i.i.d. data setting.** Test accuracy on CIFAR10+Resnet-18 and TinyImagenet+Resnet-18, with 1 local epoch and 50 clients.

neural network, a core component of our method, FED-LAMB, is a *layerwise* update of each local model. The main contribution of our paper is thus a federated learning optimization algorithm that leverages a double level of adaptivity: the first one stemming from a *dimensionwise* adaptivity of adaptive gradient methods, extended to their distributed (and local) counterpart, the second one is due to a *layerwise* adaptivity making use of the particular compositionality of the considered model. Proved convergence guarantees of our scheme are provided in our contribution, and exhibit a sublinear dependence on the total number of communications rounds, and a linear speedup against the number of clients. Extensive experiments on various datasets and models, under both iid and non-iid data setting, validates that FED-LAMB is able to provide faster convergence speed which in turn could lead to reduced communication cost. Moreover, in many cases, FED-LAMB can also improve the overall learning performance of federated learning over prior methods.

## References

[1] 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 (NIPS)*, pages 1709–1720, Long Beach, CA, 2017.

[2] Xiangyi Chen, Xiaoyun Li, and Ping Li. Toward communication efficient adaptive gradient method. In *Proceedings of the ACM-IMS Foundations of Data Science Conference (FODS)*, pages 119–128, Virtual Event, USA, 2020.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[5] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

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

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, 2016.

[8] 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. *arXiv preprint arXiv:1910.06378*, 2019.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[10] 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.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[12] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[13] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[14] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 583–598, Broomfield, CO, 2014.

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

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

[17] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[18] 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 the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, Fort Lauderdale, FL, 2017.

[19] H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Conference on Learning Theory (COLT)*, pages 244–256, Haifa, Israel, 2010.

[20] Yurii Nesterov. Introductory lectures on convex optimization: A basic course. *Springer*, 2004.

[21] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *Mathematics and Mathematical Physics*, 1964.

[22] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24:693–701, 2011.

[23] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[24] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, Austria, 2021.

[25] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8253–8265. PMLR, 2020.

[26] Sebastian U. Stich. Local SGD converges fast and communicates little. In *Proceedings of the 7th*

*International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

[27] T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

[28] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1306–1316, Montréal, Canada, 2018.

[29] Yang You, Jing Li, Sashank J. 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 *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[30] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 7184–7193, Long Beach, CA, 2019.

[31] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[32] Weijie Zhao, Deping Xie, Ronglai Jia, Yulei Qian, Ruiquan Ding, Mingming Sun, and Ping Li. Distributed hierarchical GPU parameter server for massive scale deep learning ads systems. In *Proceedings of Machine Learning and Systems (MLSys)*, Austin, TX, 2020.

[33] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

[34] Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3219–3227, Stockholm, Sweden, 2018.

[35] Yingxue Zhou, Belhal Karimi, Jinxing Yu, Zhiqiang Xu, and Ping Li. Towards better generalization of adaptive gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, 2020.