Layerwise and Dimensionwise Locally Adaptive Optimization Method

Anonymous Author(s)
Affiliation
Address
email

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 on the training of deep, as in multilayered, neural networks, under the FL settings. 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 a thorough finite-time convergence analysis for FED-LAMB characterizing how fast its gradient decreases. Finally, we provide experimental results under iid and non-iid settings to corroborate not only our theory, but also exhibit the faster convergence of our method, compared to the state-of-the-art.

1 Introduction

A growing and important task while learning models on observed data, is the ability to train the latter over a large number of clients which could either be devices or distinct entities. In the paradigm of Federated Learning (FL) [13; 21], the focus of our paper, a central server orchestrates the optimization over those clients under the constraint that the data can neither be centralized nor shared among the clients. 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 θ the global model parameter taking value in Θ, a subset of $\mathbb{R}^p$. While this formulation recalls that of distributed optimization, the core principle of FL is different than standard distributed paradigm.

FL currently suffers from two bottlenecks: communication efficiency and privacy. We focus on the former in this paper. While local updates, updates during which each client learn their local models, can reduce drastically the number of communication rounds between the central server and devices, new techniques are still necessary to tackle the challenge of communication due to, e.g., wireless bandwidth. Some quantization [1; 30] or compression [20] methods allow to decrease the number of bits communicated at each round and are efficient methods in a distributed setting. The other approach one can take is to accelerate the local training on each device and thus sending a better local model to the server at each round, thus reducing the number of communication rounds needed to get a well-trained global model.

Under the important setting of heterogenous data, i.e. the data in each device can be distributed according to different distributions, current local optimization algorithms are perfectible. One of the

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

most popular framework for FL is using multiple local Stochastic Gradient Descent (SGD) steps in each device, sending those local models to the server that computes the average over those received local model parameters and broadcasts it back to the devices. This method is called Fed-Avg [21].

In [2], the authors motivate the use of adaptive gradient optimization methods as a better alternative to the standard SGD inner loop in Fed-Avg. They propose an adaptive gradient method, namely Local AMSGrad, with communication cost sublinear in R that is guaranteed to converge to stationary points in $\mathcal{O}(\sqrt{p/Rn})$, where R is the number of communication rounds, p is the overall dimension of the problem and n corresponds to the number of clients available.

Based on recent progress in adaptive methods for accelerating the training procedure, see [31], we propose a variant of Local AMSGrad integrating dimensionwise and layerwise adaptive learning rate in each device's local update. Our contributions are as follows:

- We develop a novel optimization algorithm for federated learning, namely FED-LAMB, following a principled layerwise adaptation strategy to accelerate training of deep neural networks. Our method is provably and empirically communication-efficient for compositional structural models.
- We provide a rigorous theoretical understanding of the non asymptotic convergence rate of FED-LAMB. Based on the recent progress on nonconvex stochastic optimization, we derive for a any number of rounds performed by our method, a characterization of the rate at which the classical suboptimality condition, *i.e.,* , the second order moment of the gradient of the objective function, decreases. Our bound in $\mathcal{O}\left(\sqrt{\frac{p}{n}}\frac{1}{\sqrt{\mathsf{h}R}}\right)$, where h is the total number of layers and p denotes the dimension, matches state of the art methods in Federated Learning reaching a sublinear convergence in the total number of rounds.
- We exhibit the advantages of our method to reach similar, or better, test accuracy than baseline methods with less number of communication rounds, on several benchmarks supervised learning methods on both homogeneous and heterogeneous settings.

We provide below relevant works on federated and adaptive learning:

Adaptive gradient methods. In recent study on stochastic nonconvex optimization, adaptive methods have proven to be the spearhead in many applications. Those 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. Most used examples include AMSGrad [27], Adam [12], RMSprop [29], Adadelta [33], and Nadam [3].

Their popularity and efficiency are due to their great performance at training deep neural networks. They generally combine the idea of adaptivity from AdaGrad [4; 22], as explained above, and the idea of momentum from Nesterov's Method [23] or Heavy ball method [24] using past gradients. AdaGrad displays a great edge when the gradient is sparse compared to other classical methods. Its update has a notable feature: it leverages an anisotropic learning rate depending on the magnitude of the gradient for each dimension which helps in exploiting the geometry of the data.

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 [4]. Yet, when applying AdaGrad to train deep neural networks, it is observed that the learning rate might decay too fast, see [12] for more details. Consequently, Kingma and Ba [12] 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 [27] ought to fix Adam failures using a max operator.

A natural extension of AMSGrad has been developed in [31] specifically for multi layered neural network. A principled layerwise adaptation strategy to accelerate training of deep neural networks using large mini-batches is proposed using either a standard stochastic gradient update or a generalized adaptive method under the setting of a classical single server empirical risk minimization problem. 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 [13]. 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 [25; 16; 34] do not respect those constraints, it has been proposed in [21], 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.

2 Layerwise and Dimensionwise Adaptive Method

Beforehand, it is important to provide useful and important notations used throughout our paper.

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

2.1 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 [21] 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 [13; 36]. The heuristic efficiency of local updates using SGD and periodic averaging has been studied in [28; 32] 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 [2], extended both empirically and theoretically, the benefits of performing local updates coupled with periodic averaging.

2.2 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 depicted in Figure 1 and detailed in Algorithm 1, which is a natural adaptation of the vanilla AMSGrad method, for *multilayer* neural networks under the *federated* setting. In particular, while Line 1 and Line 1 corresponds to the standard approximation of the first and second moments, via the smooth updates allowed by the tuning parameters β_1 and β_2 respectively and that both Line 1 and Line 1 are correct the biases of

those estimates, the final local update in Line 11 is novel and corresponds to the specialization per layer of our federated method. Note that a scaling factor is applied to the learning rate α_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 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).

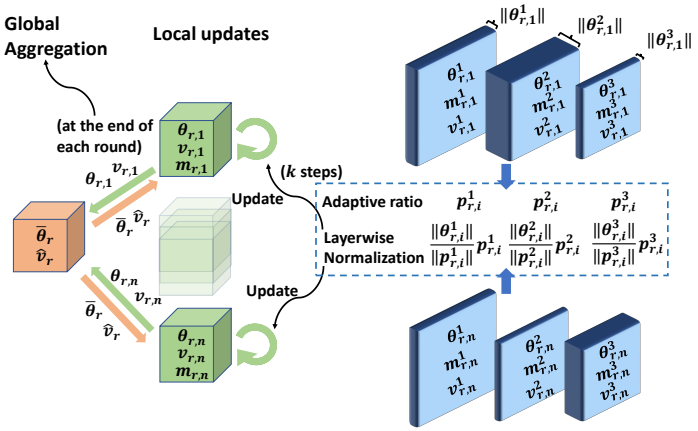

Figure 1: Illustration of Fed-LAMB (Algorithm 1), with a three-layer network and $\phi(x) = x$ as an example. The depth of each network layer represents the norm of its weights. For device i and each local iteration in round r, the adaptive ratio of j-th layer $p_{r,i}^j$ is normalized according to $\|\theta_{r,i}^j\|$, 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 θ and $\hat{v}$ to local devices to complete a round of training.

Algorithm 1 FED-LAMB for Federated Learning

1: **Input**: parameter $0 < \beta_1, \beta_2 < 1$, and learning rate α_t, weight decaying parameter $\lambda \in]0, 1[$.
2: Init: $\theta_0 \in \Theta \subseteq \mathbb{R}^d$, as the global model and $\hat{v}_0 = v_0 = \epsilon 1 \in \mathbb{R}^d$ 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: Compute stochastic gradient $g_{r,i}$ at $\theta_{r,i}^0$.
7: **for** $t = 1$ to T **do**
8: $m_{r,i}^t = \beta_1 m_{r-1,i}^{t-1} + (1-\beta_1) g_{r,i}$ and $m_{r,i}^t = m_{r,i}^t / (1-\beta_1^t)$.
9: $v_{r,i}^t = \beta_2 v_{r-1,i}^t + (1-\beta_2) g_{r,i}^2$ and $v_{r,i}^t = v_{r,i}^t / (1-\beta_2^t)$.
10: Compute the ratio $p_{r,i}^t = m_{r,i}^t / (\sqrt{\hat{v}_r} + \epsilon)$.
11: Update local model for each layer $\ell \in [\![\mathsf{h}]\!]$:

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

12: **end for**
13: Devices send $\theta_{r,i}^T = [\theta_{r,i}^{\ell,T}]_{\ell=1}^{\mathsf{h}}$ and $v_{r,i}^T$ to server.
14: **end for**
15: 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.
16: **end for**

3 On The Convergence of FED-LAMB

We develop in this section, the theoretical analysis of Algorithm 1.

Based on classical result for stochastic nonconvex optimization, we provide 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, 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 a 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.

3.1 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 in the unbiased stochastic optimization realm, 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 value* $a \in \mathbb{R}_+^*$*, there exists strictly positive constants such that* $\phi_m \leq \phi(a) \leq \phi_M$.

Important Intermediary Lemmas:

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:*

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

where ϕ_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.

Then, 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 $\left\|\frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}}\right\|$. Note that the end goal is to characterize how fast the gradient of the averaged/global parameter $\overline{\theta_r}$ goes to zero, and not the averaged gradient.

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

$$\left\|\frac{\overline{\nabla} f(\theta_r)}{\sqrt{v_r^t}}\right\|^2 \geq \frac{1}{2}\left\|\frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}}\right\|^2 - \overline{L}\alpha^2 M^2 \phi_M^2 \frac{(1-\beta_2)p}{v_0} \tag{3}$$

where M *is defined in H2,* $p = \sum_{\ell=1}^{\mathsf{h}} p_\ell$ *and* ϕ_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 α. Let the number of local epochs be $T \geq 1$ and $\lambda = 0$. Then, for any round $R > 0$, we have:*

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

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*

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{\theta_t})}{\hat{v}_t^{1/4}}\right\|^2\right] \leq \mathcal{O}\left(\sqrt{\frac{p}{n}}\frac{1}{\sqrt{\mathsf{h}R}} + \frac{\sigma^2}{Rn\sqrt{p}} + \frac{(T-1)^2 p}{\mathsf{h}^3 R^{3/2}}\right). \tag{5}$$

3.2 Comparisons and discussions

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. [31]: We first start our discussion with the comparison of convergence rate of FED-LAMB with that of LAMB, Theorem 3 in [31]. 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 (4) 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 [5]. (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 [31], performed 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 [2], and our *layerwise federated* method, namely FED-LAMB. We recall their main result under our notations:

Theorem 2 (Theorem 5.1 in [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:*

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}}\right\|^2\right] \leq \mathcal{O}(\sqrt{\frac{p}{Rn}} + \sqrt{\frac{p}{Rn}}\sigma^2 + \frac{n(T-1)^2}{R}). \tag{6}$$

where ϵ corresponds to their initialization of the vector $\hat{v}_0$ and L_s is the sum of the local smoothness constants.

The first two terms of their results and ours, standard in convergence analysis, 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. [35]). 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 can be handled for simplicity and clarity of the results when H3 is assumed globally.

In (5), 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 ϵ-stationary point compared with [2], hence improving the communication efficiency.

We now discuss our bound regarding several aspects in order to gain understanding on FED-LAMB :

Communication Complexity: The (sublinear) dependence on the number of communication rounds of our bound matches that of most recent methods in Federated Learning, see [11] developing SCAFFOLD, a solution to the problem posed by heterogeneity of the data in each client, and of [26], adapting state of the art method in optimization, here ADAM, under the federated setting. 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. Novelty in our bound occurs considering the sublinear dependence on the number of layer h and the dependence on the total number of dimension of our problem p. For the latter, the dependency can be simplified with stronger assumption on the control of the variance of the stochastic gradient. For the former,

Homogeneous and Heterogeneous Data: A common assumption regarding the stochastic gradient, and its variance, considers an upper-bound of the global variance, in the sense that it applies to both the aggregated objective function (1) and each of its local component. An alternative theoretical approach is to set apart a local variance for each local loss, and global variance for their sum, see [2] for instance. Heterogeneity is of utmost importance in FL since client may store radically different data points in local devices. Existing methods can lead to poor convergence as detailed in [18; 19]. Improvements are proposed through the use of gradient tracking techniques performed locally as seen in [6; 9; 11].

Dependence on the dimension p: The $\sqrt{p}$ term appearing in our bound is due to the assumption on coordinate-wise bounded variance. One may instead assume a bounded total variance to remove this term. In practice, the dependence on the overall size of the vectors being transmitted back and forth from the central to the devices can be improved in various ways. Indeed, recent efficient techniques aim at reducing the number of bits communicated at each round through sketches or compression techniques, see for instance [7; 10; 17] to name a few. This can be a great addition to FED-LAMB, and is naturally compatible, but is not the main focus of our contribution.

4 Numerical Experiments

In this section, we conduct numerical experiments on various datasets and network architectures to testify the effectiveness of our proposed method in practice. Our main objective is to validate the benefit of dimensionwise adaptive learning when integrated with adaptive Fed-AMS. We observe in the sequel that our method empirically confirms its edge in terms of convergence speed. Basically, our proposed method reduces the number of rounds and thus the communication cost required to achieve similar stationary points than baseline methods.

Settings. In our experiment, we will evaluate three federated learning algorithms: 1) Fed-SGD, 2) Fed-AMS 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 default and recommended [27]. Regarding federated learning environment, we use 50 local workers with 0.5 participation rate. That means, 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 a relatively small 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

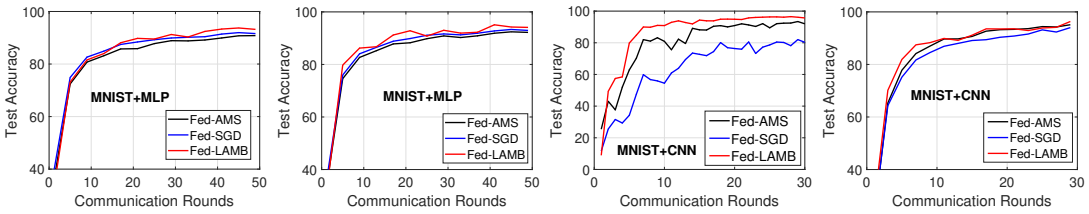

Figure 2: **Top Row**: Test accuracy on MNIST+MLP, with non-iid data distribution. **Bottom Row**: Test accuracy on MNIST+CNN, with non-iid data distribution. **Left panel:** 1 local epoch. **Right panel:** 5 local epochs. 50 clients for each run.

each dataset and number of local epochs, we tune the constant learning rate α for each algorithm over a fine grid in logarithm scale. For Fed-LAMB, the parameter λ in Algorithm 1 controlling the overall scale of the layerwise gradients is tuned from $\{0, 0.01, 0.1\}$. For each experiment, we use the identity function for $\phi(\cdot)$. For each run, we take the model performance with the best α and λ. The reported results are averaged over three independent runs, each with same initialization for every method.

Models. We test the performance of different federated learning algorithms on MNIST [15] and CIFAR10 [14] image classification datasets. 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) a ResNet-9 model proposed by [8]. For each, we use both iid and non-iid data.

4.1 MNIST with Multilayer Perceptron and Convolutional Neural Network

In Figure 2, we start by presenting the test accuracy on MNIST dataset trained by MLP. We compare each method under the heterogeneous (non-iid) data distribution settings, where each device only receives samples of one digit (out of ten). This is known to be the scenario where federated learning is harder to generalize well, see [21]. First of all, we observe that our proposed Fed-LAMB outperforms Fed-AMS and Fed-SGD with both 1 and 5 local epochs, illustrating its improvement over baseline federated methods. In addition, though it is not the main comparison of this paper, Fed-SGD slightly generalizes better than Fed-AMS, which means that SGD might be sufficient for this rather simple model. Yet, Fed-LAMB overachieves both methods on this task in terms of test accuracies. Similar comparison can be drawn with respect to the training and testing losses.

We also evaluate various algorithms on larger models. Since Fed-LAMB is specifically designed for multi-layer deep learning networks, we expect it to show more substantial advantage over Fed-AMS on larger network architectures, since we recall that in Corollary 1, the convergence bound decreases with larger number of layers h. In Figure 2, we present the results on MNIST with CNN, under non-iid data distribution. Firstly, we see that Fed-LAMB outperforms Fed-AMS in both cases. The advantage is in particular significant with 1 local epoch, where Fed-SGD generalizes poorly. Importantly, we would like to address the acceleration effect of Fed-LAMB, in the early stage of training. We observe that Fed-LAMB converges 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 adaptive gradient methods usually perform better than simple SGD in training large deep learning models. We refer the readers to a collection of prior studies such as [2; 26].

4.2 CIFAR-10 with Convolutional Neural Network and Residual Neural Network

In Figure 3, we report the test accuracies of a Convolutional Neural Network trained on CIFAR-10 dataset, where the data is either iid allocated among clients or non-iid. When we run 1 local epoch per device, we observe a clear advantage of FED-LAMBover Fed-AMS on both test accuracy and convergence speed. Note that Fed-SGD again fails to achieve close performance as those two adaptive gradient methods. Increasing the number of local iterations in each device per communication round leads to similar observations: our newly introduced method, namely FED-LAMB, converges the fastest, with similar generalization error as Fed-AMS.

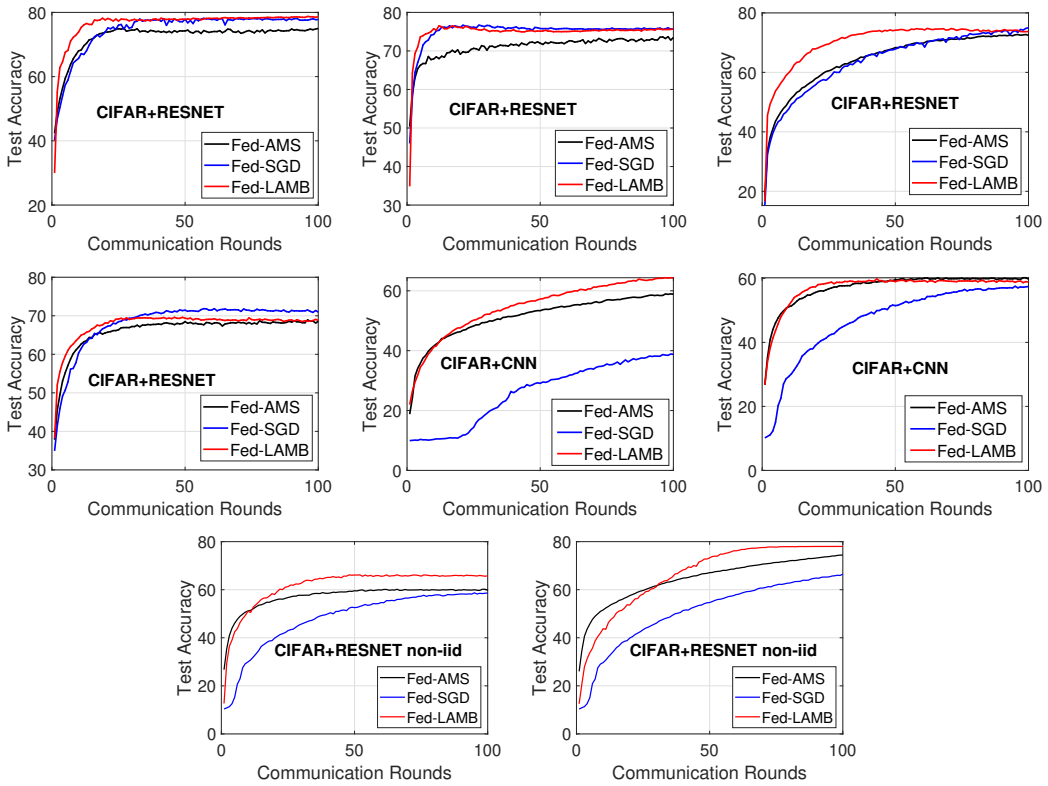

Figure 3: **Top Row**: Test accuracy on CIFAR+ResNet, with iid data distribution for 10 clients 1 and 3 local epochs and 50 clients 1 local epoch respectively. **Middle Row:** Test accuracy on CIFAR+ResNet with iid data distribution for 50 clients 3 local epochs and on CIFAR+CNN with iid data distribution for 50 clients 1 and 3 local epochs respectively. **Bottom Row:** Test accuracy on CIFAR+CNN with non iid data distribution for 50 clients and 3 local epochs.

Lastly, we test the algorithms on CIFAR-10 using a Residual Neural Network, the ResNet-9 model. We again observe in Figure 3 that Fed-LAMB improves the performance of vanilla local AMS method under various settings regarding the number clients and number local iterations, under both iid and non-iid data distribution settings, corroborating the solid improvement brought by our laywerwise adaptive strategy. We see remarkable acceleration in the accuracy curve of Fed-LAMB, especially with 50 clients and 1 local epoch (left bottom). In addition, Fed-AMS also compares favorably with Fed-SGD, though their performances are close on this specific task.

5 Conclusion

We study in this contribution 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 nonconvex 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 neural network, a core component of our method, namely 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 exhibits a sublinear dependence on the total number of communications rounds, the number of clients and the number of layers of the model. The multiple benefits of periodic averaging, adaptive optimization methods and layerwise updates are displayed in our bounds. We empirically confirm the advantage of our algorithm over baselines methods on a panel of numerical experiments.

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*, pages 1709–1720, 2017.

[2] Xiangyi Chen, Xiaoyun Li, and Ping Li. Toward communication efficient adaptive gradient method. In *ACM-IMS Foundations of Data Science Conference (FODS)*, Seattle, WA, 2020.

[3] Timothy Dozat. Incorporating nesterov momentum into adam. *ICLR (Workshop Track)*, 2016.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011.

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

[6] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. *arXiv preprint arXiv:2007.01154*, 2020.

[7] Farzin Haddadpour, Belhal Karimi, Ping Li, and Xiaoyun Li. Fedsketch: Communication-efficient and private federated learning via sketching. *arXiv preprint arXiv:2008.04975*, 2020.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[9] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[10] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13144–13154, Vancouver, Canada, 2019.

[11] 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.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[13] 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.

[14] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

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

[16] 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 *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 583–598, 2014.

[17] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972*, 2019.

[18] 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.

[19] 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.

[20] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[21] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[22] H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. *COLT*, 2010.

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

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

[25] 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.

[26] 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.

[27] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

[28] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

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

[30] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1299–1309, 2018.

[31] 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. *arXiv preprint arXiv:1904.00962*, 2019.

[32] 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.

[33] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv:1212.5701*, 2012.

[34] 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. *arXiv preprint arXiv:2003.05622*, 2020.

[35] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *CoRR*, abs/1808.05671, 2018.

[36] Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. *arXiv preprint arXiv:1708.01012*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section3
   (b) Did you include complete proofs of all theoretical results? [Yes] See Supplementary Material
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)? [Yes] Instructions and hyperparameters are given in Section4
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Settings for MNIST and CIFAR are standard
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Results curves are averaged over 5 runs
   (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)? [N/A]
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? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Data is public
   (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]