COMMUNICATE THEN ADAPT: AN EFFECTIVE DECENTRALIZED ADAPTIVE METHOD FOR DEEP TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decentralized adaptive gradient methods, in which each node averages only with its neighbors, are critical to save communication and wall-clock training time in deep learning tasks. While different in concrete recursions, existing decentralized adaptive methods share the same algorithm structure: each node scales its gradient with information of the past squared gradients (which is referred to as the adaptive step) *before* or *while* it communicates with neighbors. In this paper, we identify the limitation of such *adapt-then/while-communicate* structure: it will make the developed algorithms highly sensitive to data heterogeneity, and hence deviate their limiting points from the stationary solution. To overcome this limitation, we propose an effective decentralized adaptive method with a *communicate-then-adapt* structure, in which each node conducts the adaptive step after finishing the neighborhood communications. The new method is theoretically guaranteed to converge to the stationary solution in the non-convex scenario. Experimental results on a variety of CV/NLP tasks show that the proposed algorithm has a clear superiority to other existing decentralized adaptive methods.

1 INTRODUCTION

Decentralized SGD (Lopes & Sayed, 2008; Nedic & Ozdaglar, 2009; Chen & Sayed, 2012; Lian et al., 2017; Assran et al., 2019) is an emerging training approach for deep learning known for its much less communication overhead. In contrast to parallel SGD in which a global averaging across all computing nodes is required per iteration, decentralized SGD does not involve any global operations. Building upon partial averaging, in which each node only needs to compute the locally averaged model within its neighborhood, decentralized SGD can save remarkable communications and training time in large-scale distributed deep learning tasks compared to parallel SGD.

Although simple to use, the vanilla decentralized SGD sometimes suffers from the slow convergence. Inspired by the well-documented success of adaptive methods such as AdaGrad (Duchi et al., 2011a), Adam (Kingma & Ba, 2014) and AMSGrad (Reddi et al., 2019), several decentralized adaptive methods (Nazari et al., 2019; Lin et al., 2021) have been proposed to accelerate decentralized SGD training. While these algorithms have achieved remarkable success in several practical applications, they have also been observed to not converge to the desired solution (i.e., global optimal solution in the convex scenario or stationary solution in the non-convex scenario) in some other settings. For example, it has been observed in the convex setting (see Sec. 3) that DAdam (Nazari et al., 2019) and QG-DAdam (Lin et al., 2021) do not converge to the global optimal solution.

This paper studies this situation in detail. We rigorously uncover the reason why DAdam and QG-DAdam fail to achieve the desired solution, and propose a novel decentralized adaptive method to resolve the convergence issue. In particular, we make the following key contributions:

- We find the algorithms DAdam and QG-DAdam, while different in concrete recursions, share a similar structure: each node scales its gradient with the past squared gradients (which is referred to as the adaptive step) *before* or *while* it communicates with neighbors. We identify the limitation of such *adapt-then/while-communicate* structure: it will make the developed algorithms highly sensitive to data heterogeneity, and hence deviate their limiting points from the desired solution.
- To overcome these limitations, we propose a novel *communicate-then-adapt* algorithm structure, in which each node will conduct the adaptive step after all neighborhood communications. It is *not* just trivially switching the order of communication step and adaptive step. The key component to guarantee the effectiveness of the communicate-then-adapt structure is the utilization of the

decentralized augmented gradient (DAG) rather than the standard stochastic gradient in algorithm development. The newly proposed algorithm, which is coined as DAG-Adam, can provably converge to the desired solution. While this paper mainly focuses on the vanilla Adam algorithm, the core idea behind DAG-Adam can be easilty extended to AMSGrad or other adaptive methods.

- Experimental results on a variety of computer vision and natural language processing tasks show that DAG-Adam outperforms various existing state-of-the-art decentralized adaptive baselines under different network and data configurations. Furthermore, the performance of our proposed algorithm is persistently competitive to the centralized counterpart.

Related work on decentralized optimization and deep training. Decentralized optimization was extensively studied in the control and signal processing community. The first decentralized algorithms on general optimization problems include decentralized gradient descent (Nedic & Ozdaglar, 2009), diffusion (Lopes & Sayed, 2008; Chen & Sayed, 2012; Sayed, 2014), and dual averaging (Duchi et al., 2011b). After that, various primal-dual algorithms come out to further speed up the convergence, and they are based on alternating direction method of multipliers (Adam) (Shi et al., 2014), explicit bias-correction (Shi et al., 2015; Yuan et al., 2019; Li et al., 2019), gradient tracking (Xu et al., 2015; Di Lorenzo & Scutari, 2016; Nedic et al., 2017; Qu & Li, 2018; Lu et al., 2019), and dual acceleration (Scaman et al., 2017; Uribe et al., 2020).

In deep learning tasks, decentralize SGD, which was established in (Lian et al., 2017) to achieve the same linear speedup as parallel SGD in convergence rate, has attracted a lot of attentions. Many efforts have been made to extend the algorithm to directed topologies (Assran et al., 2019), time-varying topologies (Koloskova et al., 2020), asynchronous settings (Lian et al., 2018), and data-heterogeneous scenarios (Tang et al., 2018; Xin et al., 2020; Lin et al., 2021; Yuan et al., 2021). With careful consensus control (Kong et al., 2021) or periodic global averaging (Chen et al., 2021b), decentralize SGD can achieve $1.3 \sim 2\times$ training time speedup without severe performance degradation. Techniques such as quantization/compression (Alistarh et al., 2017; Bernstein et al., 2018; Koloskova et al., 2019a;b; Tang et al., 2019; Liu et al., 2020), periodic updates (Stich, 2019; Koloskova et al., 2020; Yu et al., 2019), and lazy communication (Chen et al., 2018a; Liu et al., 2019b) were also integrated into decentralized SGD to further reduce communication overheads.

There are also a few studies on accelerated variants of decentralized SGD, and most of them are on (static) momentum acceleration. (Assran et al., 2019; Gao & Huang, 2020) propose to run a local momentum SGD step first before the partial averaging is conducted. Another work (Yu et al., 2019) imposes an additional partial averaging over momentum to increase stability. Recent works (Lin et al., 2021; Yuan et al., 2021) developed strategies that can remove the momentum-incurred bias in decentralized momentum SGD. All these methods are not with adaptive strategies to scale gradients.

Related work on adaptive gradient method. Adaptive gradient methods, with AdaGrad (Duchi et al., 2011a) and Adam (Kingma & Ba, 2014) as two representatives, have shown strong performance in training deep neural networks. With adaptive adjustment in the gradient direction and automatic tune in the learning rate, adaptive gradient methods can boost the performance of SGD training significantly. In spite of its remarkable empirical success, Adam suffers from a convergence issue: it may not converge to the desired solution with a fixed mini-batch size (Reddi et al., 2019; Zaheer et al., 2018). Many algorithms have been proposed to resolve this issue. AMSGrad (Reddi et al., 2019) preserves the long-term memory of past gradients to improve convergence, and (Zaheer et al., 2018) suggests the usage of increasing mini-batch sizes in convergence guarantees. (Chen et al., 2018b) has studied the convergence of a family of Adam-type algorithms. (Zhou et al., 2018) provides a better convergence rate for AMSGrad. From the empirical side, RAdam (Liu et al., 2019a) and AdamW (Loshchilov & Hutter, 2018) can significantly improve Adam performance.

The exploration in decentralized adaptive methods are rather limited. DAdam (Nazari et al., 2019) is the first consensus-based adaptive methods for distributed optimization to our knowledge. (Lin et al., 2021) proposes QG-DAdam, which utilizes quasi-global momentum to locally approximates the globally descent direction. A recent work (Chen et al., 2021a) proposes a unified framework that incorporates various adaptive methods into decentralized setting. While these methods have shown strong empirical performance in several practical applications, they either suffer from unstable convergence or heavy communications. We leave more discussion on their limitations in Sec. 2. Adaptive gradient methods have also been extended to the federated learning setting in which multiple clients cooperate to learn a model under the supervision of a central server (McMahan et al., 2017). Useful references in this direction can be found in (Xie et al., 2019; Reddi et al., 2020).

2 ADAPT-THEN/WHILE-COMMUNICATE STRUCTURE AND ITS LIMITATION

Problem. Suppose n computing nodes cooperate to solve the distributed optimization problem:

$$\min_{x\in\mathbb{R}^d} \; f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x) := \mathbb{E}_{\xi_i\sim D_i} F(x;\xi_i) \tag{1}$$

In the above problem, $f_i(x)$ is local to node i, and random variable ξ_i denotes the local data that follows distribution D_i. We do *not* assume each distribution D_i is the same across all nodes.

Network topology and weights. We assume all computing nodes are connected by a network topology. We define w_{ij}, the weight to scale information flowing from node j to node i, as follows:

$$w_{ij} \begin{cases} > 0 & \text{if node } j \text{ is connected to } i, \text{ or } i = j; \\ = 0 & \text{otherwise.} \end{cases} \tag{2}$$

$\mathcal{N}_i := \{j | w_{ij} > 0\}$ is defined as the set of neighbors of node i which also includes node i itself and the *weight matrix* $W := [w_{ij}]_{i,j=1}^n \in \mathbb{R}^{n\times n}$ are denoted as a matrix that stacks the weights of all nodes. This matrix W characterizes the sparsity and connectivity of the underlying topology.

Partial averaging. Decentralized methods are based on *partial averaging* within neighborhood defined by the network topology. With weights $\{w_{ij}\}$ and the set of neighbors $\mathcal{N}_i$, the partial averaging operation of node i can be expressed as

$$\text{Partial averaging:} \quad x_i^+ \leftarrow \sum_{j\in\mathcal{N}_i} w_{ij} x_j. \tag{3}$$

Partial averaging requires much less communication than global averaging on sparse topologies.

Assumptions. We will make the following standard assumptions throughout the paper:

A.1 [SMOOTHNESS] Each $f_i(x)$ is L-smooth, i.e., $\|\nabla f_i(x) – \nabla f_i(y)\| \le L\|x-y\|$ for any $x, y \in \mathbb{R}^d$.

A.2 [GRADIENT NOISE] The random sample $\xi_i^{(t)}$ is independent of each other for any k and i. We also assume $\mathbb{E}[\nabla F(x;\xi_i)] = \nabla f_i(x)$ and $\mathbb{E}\|\nabla F(x;\xi_i) - \nabla f_i(x)\|^2 \le \sigma^2$.

A.3 [WEIGHT MATRIX] The weight matrix W is symmetric and doubly-stochastic, i.e. $W\mathbb{1} = \mathbb{1}$ and $\mathbb{1}^T W = \mathbb{1}^T$, where $\mathbb{1}$ is an all-ones vector. Further, we assume $\|W - \frac{1}{n}\mathbb{1}\mathbb{1}^T\|_2 \le \rho \in (0,1)$.

A.4 [BOUNDED GRADIENT] The loss function F has bounded gradient, i.e., the maximum norm $\|\nabla F(x;\xi_i)\|_\infty \le G$ for all x and ξ_i, and $i \in [n]$.

Notation. Given a constant n, we let $[n] := \{1, \cdots, n\}$. Given a vector $x \in \mathbb{R}^d$, we let $\mathrm{diag}(x) = \mathrm{diag}\{x_1, \cdots, x_d\} \in \mathbb{R}^{d\times d}$ be a diagonal matrix. Given two vectors $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$, we let $x \odot y$ be the element-wise product between x and y.

2.1 THE ADAPT-WHILE-COMMUNICATE STRUCTURE AND ITS LIMITATION

Adapt-While-Communicate Structure. Algorithm 2 in Appendix B lists recursions for DAdam (Nazari et al., 2019). The critical step in DAdam, which is highlighted in Algorithm 2, is as follows

$$x_i^{(t+1)} = \overbrace{\sum_{j\in\mathcal{N}_i} w_{ij} x_j^{(t)}}^{\text{communicate}} - \overbrace{\frac{\gamma\, m_i^{(t)}}{\sqrt{\hat{v}_i^{(t)}} + \epsilon}}^{\text{adapt}}, \quad \forall\, i \in [n]. \tag{4}$$

where $m_i^{(t)}$ is the momentum, $\hat{v}_i^{(t)}$ is an estimate of $\nabla F(x_i^{(t)};\xi_i^{(t)}) \odot \nabla F(x_i^{(t)};\xi_i^{(t)})$, and ϵ is a small positive constant. Their updates can be referred to Algorithm 2. In recursion (4), it is observed that node i in DAdam conducts the adaptive step (i.e., scales the gradient by the square root of exponential moving averages of past squared gradients) *while* it communicates with neighbors. We call such structure utilized in DAdam as *adapt-while-communicate*.

Fixed point condition. Now we examine the fixed point achieved by DAdam and verify whether it satisfies the optimal condition to problem (1). To this end, we let $x_i^{(t)} \to x_i^\infty$ as $t \to \infty$ for $i \in [n]$. To remove the the influence of gradient noise, we assume each node i can access the full-batch gradient $\nabla f_i(x)$. If we define $H_i := \mathrm{diag}(\sqrt{\nabla f_i(x_i^\infty) \odot \nabla f_i(x_i^\infty)}\,)$ to be a diagonal matrix associated with node i, the fixed point condition for DAdam is (see derivation in Appendix B.2):

$$x_i^\infty = \sum_{j\in\mathcal{N}_i} w_{ij} x_j^\infty - \gamma(H_i + \epsilon I)^{-1}\nabla f_i(x_i^\infty), \quad \forall\, i \in [n]. \tag{5}$$

$$\Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n}(H_i + \epsilon I)^{-1}\nabla f_i(x^\infty) = 0 \quad (\text{if } x_1^\infty = \cdots = x_n^\infty = x^\infty \text{ in the limit}) \tag{6}$$

Fixed point in DAdam is deviated from the desired solution. In the data-heterogeneous scenario where data sample ξ_i follows different distribution D_i, it holds that $f_i(x) \neq f_j(x)$. If we further assume $\nabla f_i(x) \neq \nabla f_j(x)$, it follows that $H_i \neq H_j$, for any $i \neq j$. With this fact, we conclude that DAdam fixed point satisfying condition (6) does not necessarily satisfy the optimality condition for problem (1), i.e., $\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x^\infty) = 0$. This implies the fixed point of DAdam is *not* the stationary solution to problem (1). In particular, the following proposition derives the distance between the DAdam fixed point and global optimal solution to problem (1) when the cost function is strongly-convex (Proof is in Appendix B.3).

Proposition 1 *Under Assumptions A.1, A.3 and A.4, if each $f_i(x)$ is further assumed to be μ-strongly-convex, and the full-batch gradient $\nabla f_i(x)$ is accessed per iteration, then DAdam (Algorithm 2) cannot converge to $x^\star$, i.e., the global solution to problem (1). In particular, the distance between DAdam limiting point and $x^\star$ can be characterized as follows:*

$$\lim_{t\to\infty}\sum_{i=1}^{n}\|x_i^{(t)} - x^\star\|^2 = O\Big(\frac{\gamma^2 G^2}{(1-\rho)^2\epsilon^2} + B^2\Big) \tag{7}$$

where $B := \|\frac{1}{n}\sum_{i=1}^{n}(H_i + \epsilon I)^{-1}\nabla f_i(x_i^\infty) - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x_i^\infty)\| \neq 0$ is a constant bias term. Furthermore, the B^2 term in (7) is necessary and cannot be removed by an improved analysis; there exists strongly convex and smooth functions $f_i(x)$ such that $\lim_{t\to\infty}\sum_{i=1}^{n}\|x_i^{(t)} - x^\star\|^2 = \Omega(B^2)$.

It is observed in Proposition 1 that the distance between the DAdam fixed point and the optimal solution cannot diminish to zero as learning rate $\gamma \to 0$. Such non-vanishing limiting bias B^2 is caused by the adapt-while-communicate structure in DAdam. It is worth noting that a recent work (Chen et al., 2021a) has pointed out that DAdam cannot converge to the stationary solution by constructing a counter-example. However, (Chen et al., 2021a) does not clarify the reason why DAdam fails to converge to the stationary solution, and how far it can be from the desired solution.

2.2 The Adapt-Then-Communicate Structure and Its Limitation

Adapt-Then-Communicate Structure. Algorithm 3 in Appendix B lists the recursions for QG-DAdam (Lin et al., 2021). The critical step, which is highlighted in Algorithm 3, is as follows

$$x_i^{(t+1)} = \sum_{j\in\mathcal{N}_i} w_{ij}\Big(x_j^{(t)} - \gamma\frac{m_j^{(t)}}{\sqrt{v_j^{(t)}} + \epsilon}\Big) \tag{8}$$

It is observed that node i in the above recursion first conducts the adaptive step, and *then* it communicates with neighbors. We call the structure utilized in QG-DAdam as *adapt-then-communicate*.

Fixed point condition and limitation. Following the settings and arguments in Sec. 2.1, we can derive the fixed point condition for QG-DAdam as follows (see the derivation in Appendix B.5):

$$x_i^\infty = \sum_{j\in\mathcal{N}_i} w_{ij}\Big(x_j^\infty - \gamma\beta_1(\sqrt{\beta_2}H_j + \epsilon I)^{-1}\nabla f_j(x_j^\infty)\Big), \quad \forall\, i \in [n]. \tag{9}$$

$$\Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n}(\sqrt{\beta}_2 H_i + \epsilon I)^{-1}\nabla f_i(x^\infty) = 0 \quad (\text{if } x_1^\infty = \cdots = x_n^\infty = x^\infty \text{ in the limit}) \tag{10}$$

where $\beta_1, \beta_2 \in (0, 1)$ are momentum parameters listed in Algorithm 3. Similar to DAdam, the above condition implies that the fixed point of QG-DAdam is also deviated from the stationary solution. In particular, the distance between QG-DAdam fixed point and the global solution $x^\star$ can be characterized for the strongly-convex scenario as follows (proof is in Appendix B.6):

Proposition 2 *Under the same conditions as in Proposition 1, the distance between QG-DAdam fixed point and the global solution $x^\star$ can be characterized as follows:*

$$\lim_{t\to\infty}\sum_{i=1}^{n}\|x_i^{(t)}-x^\star\|^2 = O\Big(\frac{\gamma^2\beta_1^2G^2}{(1-\rho)^2\epsilon^2}+B^2\Big) \tag{11}$$

where $B := \|\frac{1}{n}\sum_{i=1}^n(\sqrt{\beta_2}H_i+\epsilon I)^{-1}\nabla f_i(x_i^\infty)-\frac{1}{n}\sum_{i=1}^n\nabla f_i(x_i^\infty)\|\neq 0$ is a constant bias term. Furthermore, the B^2 term in (7) is necessary and cannot be removed by an improved analysis; there exists strongly convex and smooth functions $f_i(x)$ such that $\lim_{t\to\infty}\sum_{i=1}^n\|x_i^{(t)}-x^\star\|^2=\Omega(B^2)$.

Similar to DAdam, it is observed that the distance between the QG-DAdam fixed point and the optimal solution cannot diminish to zero as learning rate $\gamma\to 0$.

2.3 GT-DAdam: A decentralized adaptive method based on gradient tracking

It is observed from the fixed-point condition (6) that DAdam cannot converge to the stationary solution due to the discrepancy between H_i's. A recent work (Chen et al., 2021a) proposes a gradient tracking based DAdam that enables each local different H_i to converge to $\bar{H}=\frac{1}{n}\sum_{i=1}^n H_i$ in theory, and therefore, its limiting fixed point satisfies $\frac{1}{n}\sum_{i=1}^n\nabla f_i(x^\infty)=0$, the optimality condition to problem (1). GT-DAdam also utilizes the adapt-while-communicate structure. While it can converge to the stationary solution to problem (1) as learning rate $\gamma\to 0$, it incurs *twice* amount of the communication overhead as DAdam. The distance between GT-DAdam fixed point and the global solution $x^\star$ can be characterized in Proposition 4 in Appendix B.7 for the strongly-convex scenario. In the convex scenario, the distance between GT-DAdam fixed point and the global solution can be worse than that of the proposed DAG-Adam, see Table 5 in Appendix C.3 and Fig.1 in Sec. 3.

3 Decentralized Augmented Gradient Adam

Now that we understand that the adapt-while/then-communicate structure in decentralized adaptive methods either deviates the limiting point from the desired solution (e.g., DAdam and QG-DAdam), or incurs more expensive communication overhead (GT-DAdam), this section will propose a new algorithm that can overcome the limitation of the adapt-while/then-communicate structure.

Decentralized SGD interprets as standard SGD. Without loss of generality, we assume each node i can access the accurate gradient $\nabla f_i(x)$ in decentralized SGD recursions. In this scenario, decentralized SGD, according to (Yuan et al., 2016), can be written as

$$x_i^{(t+1)} = \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{(t)} - \gamma\nabla f_i(x_i^{(t)}) \tag{12}$$

$$= x_i^{(t)} - \gamma\underbrace{\Big(\nabla f_i(x_i^{(t)})+\frac{1}{\gamma}(x_i^{(t)}-\sum_{j\in\mathcal{N}_i}w_{ij}x_j^{(t)})\Big)}_{\text{augmented gradient}},\quad \forall\, i\in[n]. \tag{13}$$

In other words, decentralized SGD (12) can be regarded as a standard SGD recursion (13) to solve

$$\min_{\{x_i\}_{i=1}^n}\quad \mathcal{L}(\{x_i\}_{i=1}^n) := \sum_{i=1}^n f_i(x_i)+\frac{1}{2\gamma}\Big(\sum_{i=1}^n\|x_i\|^2-\sum_{i=1}^n\sum_{j=1}^n w_{ij}x_ix_j\Big). \tag{14}$$

We call above as an augmented loss function, which is an approximate penalty problem to the target problem (1). The term $\sum_{i=1}^n\|x_i\|^2-\sum_{i,j}w_{ij}x_ix_j$ is convex when the weight matrix $W=[w_{ij}]$ is symmetric and doubly-stochastic, and it equals 0 if $x_1=\cdots=x_n$. Apparently, it is a regularization term that promotes consensus among all nodes.

Decentralized Augmented Gradient Adam (DAG-Adam). Inspired by the fact that decentralized SGD can be interpreted as a standard SGD with an augmented gradient (13), we can easily integrate the augmented gradient to the standard Adam method (Kingma & Ba, 2014). Decentralized Augmented Gradient Adam, or DAG-Adam for short is listed in Algorithm 1. It is observed that all communications occur when the stochastic augmented gradient is computed (see the highlighted part in Algorithm 1). After that, each node i will conduct an adaptive step, which is untangled with communication. We call the structure utilized in DAG-Adam as *communicate-then-adapt*. This adjustment of the algorithm structure is crucial. The stochastic gradient that used to compute $m_i^{(t)}$ and

Algorithm 1 DAG-Adam

Initialize $x_i^{(0)}$ arbitrarily; let $\beta_1, \beta_2, \in [0,1)$; let $m_i^{(0)} = v_i^{(0)} = 0$; set γ, ν properly;
For $t = 0, 1, 2, ..., T-1$, every node i **do in parallel**

Sample $\hat{g}_i^{(t)} = \nabla F(x_i^{(t)}; \xi_i^{(t)}) + \frac{\nu}{\gamma}\Big(x_i^{(t)} - \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(t)}\Big)$ ▷ stochastic augmented gradient;

$m_i^{(t)} = (1-\beta_1) m_i^{(t-1)} + \beta_1 \hat{g}_i^{(t)}$;

$v_i^{(t)} = (1-\beta_2) v_i^{(t-1)} + \beta_2 \hat{g}_i^{(t)} \odot \hat{g}_i^{(t)}$;

$x_i^{(t+1)} = x_i^{(t)} - \gamma \frac{m_i^{(t)}}{\sqrt{v_i^{(t)}} + \epsilon}$ ▷ adaptive step;

$v_i^{(t)}$ have incorporated the neighbor's information, which will correct the deviation in fixed point that introduced by different v_i (and H_i).

Note that a parameter $\nu > 0$ is added to the adjust the priority to gradient descent, or to promote consensus. Empirically, we can set $\nu = 1$ for the convex scenario. But for the non-convex deep learning problems, the choice of ν will heavily influence both convergence rate and performance. We will discuss the impact of ν in the experiment section. Moreover, while this paper we only focus on the extension of the vanilla Adam method, the core ideas such as the augmented gradient and the communicate-then-adapt structure can be extended to AMSGrad or other adaptive gradient methods.

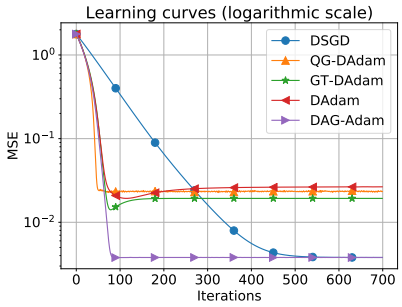

Figure 1: Convergence comparison between various algorithms for a full-batch logistic regression problem.

Fixed point condition. The stochastic augmented gradient and communicate-then-adapt structure utilized in DAG-Adam can remove the influence of the discrepancy in v_i^∞ and hence H_i. Following the settings and arguments in Sec. 2.1, we can derive the fixed point condition for DAG-Adam as follows (see the derivation in Appendix C):

$$\nabla f_i(x_i^\infty) + \frac{\nu}{\gamma}(x_i^\infty - \sum_{j \in \mathcal{N}_i} w_{ij} x_j^\infty) = 0, \quad \forall\, i \in [n]. \tag{15}$$

$$\Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^\infty) = 0 \qquad (\text{if } x_1^\infty = \cdots = x_n^\infty = x^\infty \text{ in the limit}) \tag{16}$$

Note that the fixed-point condition (15) for DAG-Adam, when $\nu = 1$, is exactly the same as that for vanilla decentralized SGD with full-bath gradient (Yuan et al., 2016). Relation (16) implies that DAG-Adam will converge to the stationary solution to problem (1). The following proposition characterizes the distance between the DAG-Adam fixed point and the global optimal solution $x^\star$ for the strongly-convex scenario.

Proposition 3 *Under the same conditions as in Proposition 1, the distance between DAG-Adam fixed point and $x^\star$ can be characterized as follows:*

$$\lim_{k \to \infty} \sum_{i=1}^{n} \|x_i^{(t)} - x^\star\|^2 = O\Big(\frac{\gamma^2 G^2}{(1-\rho)^2 \nu^2}\Big) \tag{17}$$

It is observed in (17) that the distance is on the order of $O(\gamma^2)$. Apparently, DAG-Adam will converge to the global optimal solution when $\gamma \to 0$. Moreover, DAG-Adam has the same communication overhead as in DAdam, and it is more communication-efficient than GT-DAdam. Proposition 3 also implies that DAG-Adam has the same limiting bias as decentralized SGD when $\nu = 1$.

Limiting point justification. We validate the limiting point analysis for DAdam, QG-DAdam, GT-DAdam, and DAG-Adam on the full-batch logistic regression problem. Table 5 in Appendix C.3 lists the results we establish in Propositions 1 – 4. When $\nu = 1$ in the DAG-DAdam method, which is a valid choice in convex settings in numerical experiments, the fixed point of DAG-DAdam is closet to the global solution $x^\star$ when the cost function is strongly-convex. The learning curves of

DAdam, GT-DAdam, QG-DAdam, and DAG-Adam are shown in Fig. 1. The MSE metric in the y-axis denotes $\frac{1}{n}\sum_{i=1}^{n}\|x_i^{(t)} - x^\star\|^2$. From the figure, it is observed that the DAG-Adam algorithm can converge as fast as other decentralized adaptive algorithms but to a more accurate limiting point. This limiting point is exactly the same as vanilla decentralized SGD, which is consistent with our claim in the limiting point analysis. More experimental setting and results refer to Appendix C.4.

Convergence Analysis. We provide the convergence analysis for our proposed DAG-Adam for non-convex functions. The full proof can be found in Appendix D.

Theorem 1 *Suppose the parameters γ, ν, and β_2 satisfy the conditions that $\gamma \leq \frac{\epsilon - 2\nu}{2L}$, $\nu < \min\{\epsilon/2, \kappa\gamma\}$, ($\kappa$ is any positive constant) and $1-\beta_2 \leq \frac{\epsilon^2}{16(G+\kappa C)^2}$. Under Assumptions A.1 - A.4 and the augmented gradient shown is bounded, the decentralized augmented gradient generated by DAG-Adam algorithm satisfies that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\Big\|\nabla f_i(x_i^{(t)}) + \frac{\nu}{\gamma}(x_i^{(t)} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{(t)})\Big\|^2 = O\Big(\frac{\mathcal{L}(\{x_i^{(0)}\}_{i=0}^n) - \mathcal{L}(\mathbf{x}^*)}{\gamma T}\Big) + O(\sigma^2) \tag{18}$$

where the augmented loss function $\mathcal{L}(\mathbf{x})$ is the one defined in (14), $\mathcal{L}^\star$ is the minimum of the augmented loss. Further, it holds for the globally averaged iterate $\bar{x}^{(t)}$ that

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(\bar{x}^{(t)})\|^2 \leq O\Big(\frac{\mathcal{L}(\{x_i^{(0)}\}_{i=0}^n) - \mathcal{L}^\star}{\gamma T}\Big) + O(\sigma^2) + O\Big(\frac{\gamma^2}{(1-\rho)^2\nu^2}\Big) \tag{19}$$

where $\bar{x}^{(t)} = 1/n\sum_{i=1}^{n} x_i^{(t)}$ and ρ is the second largest eigenvalue of W in magnitude.

Remark 1 *The theorem stated two conclusions. The first one is that the augmented gradient in DAG-Adam will converge to a neighborhood around 0 with constant learning rate γ, and the neighborhood, when $T \to \infty$, is on the order of $O(\sigma^2)$. The second one is that DAG-Adam will converge to a neighborhood around the stationary solution in terms of the globally averaged iterate $\bar{x}^{(t)}$. The neighborhood, when learning rate $\gamma = O(1/\sqrt{T})$ and $T \to \infty$, will be on the order of $O(\sigma^2)$. It is thus observed that DAG-Adam will converge to the exact stationary solution with decaying learning rate and increasing mini-batch sizes, which is consistent with the results of Adam using the standard stochastic gradient in (Zaheer et al., 2018).*

Remark 2 *The non-vanishing term $O(\sigma^2)$ in (18) and (19) is caused by the vanilla Adam algorithm structure, not the decentralized communication and update in DAG-Adam. Such term can be corrected by using AMSGrad rather than Adam in DAG-Adam. When there is no stochastic gradient noise, the augmented gradient will converge to zero. In consequence, the $v_i^{(t)}$ will converge to zero over all agents as well.*

4 Experiments

In this section, we systematically compare the proposed method, DAG-Adam, with all previous mentioned state-of-the-art decentralized adaptive methods on typical CV and NLP tasks.

Implementation Details. We implement all the aforementioned algorithms with PyTorch 1.8.0 (Paszke et al. (2019)) using NCCL 2.9.6 (CUDA 10.2) as the communication backend. For PAdam, we used PyTorch's native Distributed Data Parallel (DDP) module. For the implementation of decentralized methods, we utilize BlueFog. See Appendix F for more details.

4.1 Image Classification: Performance over various configurations

A series of experiments are carried out with CIFAR-10 (Krizhevsky et al. (2009)) and ImageNet (Deng et al. (2009)) datasets to evaluate the performance of the proposed DAG-Adam on both homogeneous and heterogeneous dataset over different sizes of network. We utilize ResNet-20 on CIFAR-10 and ResNet-50 on ImageNet (He et al., 2016).

Training configuration. All methods are trained using 90 epochs with mini-batch of size 32 in each computing node. PAdam uses a learning rate of 0.0125, while the ones for decentralized algorithms are scaled by the network size. In addition, QG-Adam (Lin et al. (2021)) uses a learning rate 10 times smaller than other decentralized methods to avoid divergence[1]. During the training procedure,

[1]We examine QG-DAdam based on our own implementation of the algorithm listed in the original paper.

Table 1: Classification accuracy comparison of different methods for CIFAR10 dataset.

METHODS	4 NODES	8 NODES			
	IID	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	IID
PADAM	90.28 ±0.07	51.65 ±0.88	84.33 ±0.18	90.82 ±0.23	91.19 ±0.09
DADAM	87.23 ±0.14	53.71 ±1.38	83.67 ±0.14	86.34 ±0.51	87.04 ±0.12
QG-DADAM	83.43 ±0.22	34.24 ±2.19	57.89 ±0.34	71.97 ±0.49	77.82 ±0.13
GT-DADAM	86.93 ±0.17	**56.48 ±1.26**	84.63 ±0.22	86.71 ±0.31	86.47 ±0.12
DAG-ADAM ($\nu = 10^{-3}$)	84.62 ±0.49	46.77 ±1.09	75.76 ±0.34	81.88 ±0.79	82.38 ±0.55
DAG-ADAM ($\nu = 10^{-2}$)	87.50 ±0.36	51.63 ±1.07	82.24 ±1.06	87.82 ±0.35	88.55 ±0.33
DAG-ADAM ($\nu = 10^{-1}$)	**91.39 ±0.09**	55.27 ±0.31	**86.64 ±0.03**	**91.00 ±0.18**	**91.23 ±0.09**

the learning rate is warmed up in the first 5 epochs and is then decayed by a factor of 10 at 30, 60 and 80 epochs. The weight decay is chosen as 5×10^{-5}. We also compared the results of the proposed DAG-Adam with various ν choices.

Heterogenerous data configuration. To simulate the heterogenerous data, the disjoint non-i.i.d. training data over agents are generated through Dirichlet distribution $\mathrm{Dir}(\alpha)$, where α stands for the degree of the data heterogeneity (Lin et al., 2021; Yurochkin et al., 2019). The training data for a particular class tends to concentrate in a single node as $\alpha \to 0$, i.e. becoming more heterogenous while the homogeneous data is formed as $\alpha \to \infty$. See Appendix E for more details.

Performance comparison. Table 1 compares the proposed DAG-Adam with other methods over networks with both homogeneous data distribution and heterogeneous ones. Two different network sizes (4 and 8) are tested using ring topology. With a proper choice of ν, the proposed method *consistently* outperforms all other methods under comparison, including the centralized optimization method PAdam, no matter what the data distribution is given. The left and middle figures in Fig. 2 shows how the test loss and test accuracy evolves in a heterogeneous data distribution with $\alpha = 10$. DAG-Adam (with $\nu = 1e{-}1$) converges faster than any other decentralized algorithms. As typical decentralized algorithm behaves, it was relatively slower compared with centralized PAdam algorithm in the beginning, but it eventually caught up when the algorithm converged. The right figure in Fig. 2 plots the consensus error $\sum_i \|x_i - \bar{x}\|$, which validates that the consensus error in Theorem 1 is proportional to the choice of ν.

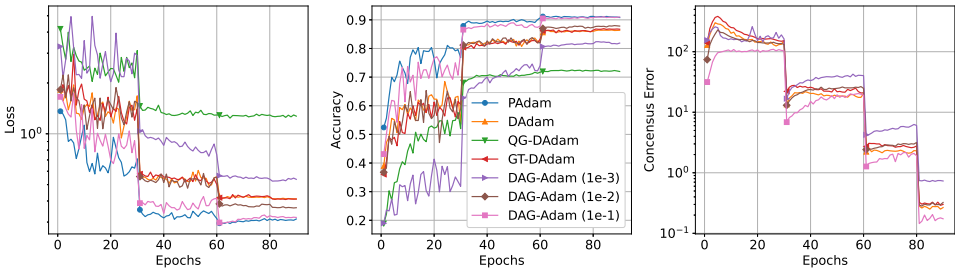

Figure 2: The evolution comparison of loss, accuracy and consensus error for CIFAR-10 dataset on a heterogeneous ($\alpha = 10$) network of size 8.

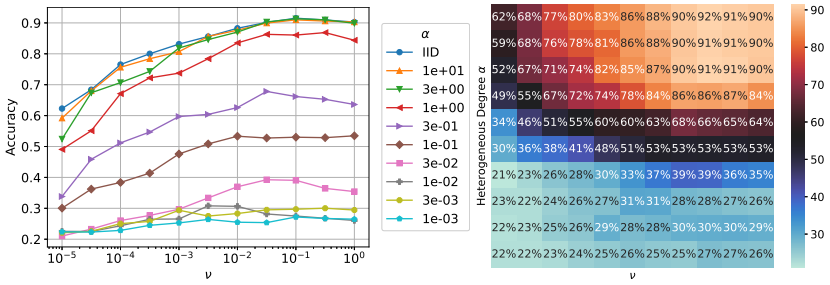

Figure 3: An ablation study of the hyperparameter ν in DAG-Adam for CIFAR-10 dataset on heterogeneous networks of size 8. The heatmap on the right shares the same selection of ν and α with the left figure. We reported the detailed accuracy value in it.

Sensitivity on ν constant. As shown in Table 1, the final classification performance depends on the choice of constant ν in DAG-Adam. To better understand how ν affects the DAG-Adam performance, we performed a corresponding ablation study with varying heterogeneity degree α as shown

in Fig. 3. For varying heterogeneity degree α, the constant ν that achieving best performance lies in the range of $[1e^{-2}, 5e^{-1}]$ for CIFAR-10 typically.

Performance over different network scales. We further examine the performance of the proposed algorithm over different network sizes to see how good it can scale up in large decentralized network case. As shown in Table 2, under the exponential graph (Assran et al., 2019) and fine-tuning of ν, DAG-Adam persistently achieves comparable performance with the centralized method until the node size reaches $n = 64$. Even under that case, the drop of performance is quite small.

Table 2: Comparison of different nodes and methods on CIFAR-10 classification over homogeneous data.

NODES	4	8	16	32	64
PADAM	91.28	91.19	90.67	90.15	**90.11**
DAG-ADAM ($\nu = 1e{-}3$)	84.62	82.38	84.35	83.73	83.90
DAG-ADAM ($\nu = 1e{-}2$)	87.50	86.55	86.89	87.83	87.62
DAG-ADAM ($\nu = 1e{-}1$)	**91.39**	**91.23**	**91.36**	**90.16**	89.98

ImageNet results. We also validate the performance of our proposed method on a relatively large-scale dataset ImageNet. We use PAdam to tune the learning rate (4×10^{-4}) and wight decay (1×10^{-4}) hyper-parameters to reach a strong baseline. Then, the same hyper-parameters are applied for decentralized counterparts. In the experiment, the algorithms run on 8 nodes (i.e. 8 × 8 = 64 GPUs) and train total 100 epochs with a cosine decay scheduler. Our proposed method reaches top-1 accuracy 75.96% which is slightly better than PAdam.

Table 3: Comparison of different methods and models on ImageNet classification.

METHOD	PADAM	DADAM	QG	GT	DAG/$1e{-}1$	DAG/$1e{-}2$	DAG/$1e{-}3$	DAG/$1e{-}4$
ACCURACY	75.62	70.53	73.89	74.736	66.506	73.750	75.445	**75.96**

4.2 LANGUAGE MODELING: PERFORMANCE OVER DIFFERENT TASKS

BERT (Devlin et al. (2018)) is a widely used pre-training language representation model for NLP tasks. We fine-tuned pretrained BERT models (Base/Large: 110/330M parameters) on the SQuAD (Rajpurkar et al. (2016)) dataset. All experiments are based on NVIDIA BERT[2] implementation with mixed precision support. We examine the task over both static exponential graph and dynamic exponential graph setting (Assran et al., 2019). As shown in Table 4, with proper tuned hyper-parameter ν, our proposed method reached best performance in both metrics.

Table 4: Comparison of different methods on BERT fine-tuning (SQuAD) on Exact Match and F1-score.

MODEL	BERT-BASE				BERT-LARGE			
TOPOLOGY	STATIC		DYNAMIC		STATIC		DYNAMIC	
EM	F1	EM	F1	EM	F1	EM	F1	EM
PADAM	81.50	88.85	-	-	83.54	90.69	-	-
DADAM	80.61	87.57	80.31	87.42	83.25	89.98	83.16	89.77
QG-DADAM	79.4	86.98	80.17	87.33	83.05	89.84	83.89	90.64
GT-DADAM	80.66	87.85	80.67	87.81	83.81	90.51	83.95	90.61
DAG-ADAM ($\nu = 1e{-}1$)	48.89	60.85	43.77	56.22	59.24	71.40	55.18	67.89
DAG-ADAM ($\nu = 1e{-}2$)	71.98	81.62	70.30	80.15	78.20	86.68	76.85	85.59
DAG-ADAM ($\nu = 1e{-}3$)	80.66	87.99	80.14	87.54	**83.93**	90.48	83.31	90.26
DAG-ADAM ($\nu = 1e{-}4$)	**81.79**	**88.93**	**81.89**	**88.96**	83.82	**90.71**	**84.28**	**91.08**

5 CONCLUSION AND FUTURE WORKS

In this paper, we identified the limitation of adapt-while/then-communicate structure utilized in existing adaptive gradient methods: the limiting discrepancy in the squared local gradient term (i.e., v_i^{∞}) can deviate the limiting point from the desired stationary point. To fix this issue, we proposed a novel DAG-Adam algorithm that is built upon the communicate-then-adapt structure instead. DAG-Adam can provably converge to the desired solution even with heterogeneous data. Experimental results on a variety of deep learning tasks show that DAG-Adam persistently outperforms existing decentralized adaptive baselines. However, it is also observed in these experiments that DAG-Adam is highly-sensitive to the hyper-parameter ν. How to tune ν automatically during the training stage is a crucial topic of the future works.

[2]https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT

REFERENCES

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*, pp. 1709–1720, 2017.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning (ICML)*, pp. 344–353, 2019.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. *arXiv preprint:1810.05291*, 2018.

Jianshu Chen and Ali H Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2018a.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2018b.

Xiangyi Chen, Belhal Karimi, Weijie Zhao, and Ping Li. On the convergence of decentralized adaptive gradient methods. *arXiv preprint arXiv:2109.03194*, 2021a.

Yiming Chen, Kun Yuan, Yingya Zhang, Pan Pan, Yinghui Xu, and Wotao Yin. Accelerating gossip sgd with periodic global averaging. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint:1810.04805*, 2018.

P. Di Lorenzo and G. Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011a.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3): 592–606, 2011b.

Hongchang Gao and Heng Huang. Periodic stochastic gradient descent with momentum for decentralized training. *arXiv preprint:2008.10435*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

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

Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2019a.

Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pp. 3478–3487, 2019b.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning (ICML)*, pp. 1–12, 2020.

Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian U Stich. Consensus control for decentralized deep learning. *arXiv preprint:2102.04828*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Z. Li, W. Shi, and M. Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, July 2019. early acces. Also available on arXiv:1704.07807.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pp. 3043–3052, 2018.

Tao Lin, Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2019a.

Xiaorui Liu, Yao Li, Rongrong Wang, Jiliang Tang, and Ming Yan. Linear convergent decentralized optimization with compression. *arXiv preprint arXiv:2007.00232*, 2020.

Yaohua Liu, Wei Xu, Gang Wu, Zhi Tian, and Qing Ling. Communication-censored admm for decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 67(10):2565–2579, 2019b.

Cassio G Lopes and Ali H Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 56(7):3122–3136, 2008.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Songtao Lu, Xinwei Zhang, Haoran Sun, and Mingyi Hong. Gnsd: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *2019 IEEE Data Science Workshop (DSW)*, pp. 315–321. IEEE, 2019.

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*, pp. 1273–1282. PMLR, 2017.

Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

A. Nedic, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint:1904.09237*, 2019.

Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

Ali H Sayed. Adaptation, learning, and optimization over networks. *Foundations and Trends in Machine Learning*, 7(ARTICLE):311–801, 2014.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pp. 3027–3036, 2017.

W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

Sebastian Urban Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. d^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pp. 4848–4856, 2018.

Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pp. 6155–6165. PMLR, 2019.

César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. *Optimization Methods and Software*, pp. 1–40, 2020.

Cong Xie, Oluwasanmi Koyejo, Indranil Gupta, and Haibin Lin. Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates. *arXiv preprint arXiv:1911.09030*, 2019.

Ran Xin, Usman A Khan, and Soummya Kar. An improved convergence analysis for decentralized online stochastic non-convex optimization. *arXiv preprint:2008.04195*, 2020.

J. Xu, S. Zhu, Y. C. Soh, and L. Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *IEEE Conference on Decision and Control (CDC)*, pp. 2055–2060, Osaka, Japan, 2015.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pp. 7184–7193. PMLR, 2019.

K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact dffusion for distributed optimization and learning – Part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708 – 723, 2019.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Kun Yuan, Yiming Chen, Xinmeng Huang, Yingya Zhang, Pan Pan, Yinghui Xu, and Wotao Yin. DecentLaM: Decentralized momentum SGD for large-batch deep training. *arXiv preprint arXiv:2104.11981*, 2021.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pp. 7252–7261. PMLR, 2019.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/90365351ccc7437a1309dc64e4db32a3-Paper.pdf.

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.