DecentLaM: Decentralized Momentum SGD for Large-batch Deep Training

Kun Yuan[1]*, Yiming Chen[1*], Xinmeng Huang[2*], Yingya Zhang[1], Pan Pan[1], Yinghui Xu[1], Wotao Yin[1]
[1]DAMO Academy, Alibaba Group [2]University of Pennsylvania

Abstract

The scale of deep learning nowadays calls for efficient distributed training algorithms. Decentralized momentum SGD (DmSGD), in which each node averages only with its neighbors, is more communication efficient than vanilla Parallel momentum SGD that incurs global average across all computing nodes. On the other hand, the large-batch training has been demonstrated critical to achieve runtime speedup. This motivates us to investigate how DmSGD performs in the large-batch scenario.

In this work, we find the momentum term can amplify the inconsistency bias in DmSGD. Such bias becomes more evident as batch-size grows large and hence results in severe performance degradation. We next propose DecentLaM, a novel decentralized large-batch momentum SGD to remove the momentum-incurred bias. The convergence rate for both strongly convex and non-convex scenarios is established. Our theoretical results justify the superiority of DecentLaM to DmSGD especially in the large-batch scenario. Experimental results on a a variety of computer vision tasks and models show that DecentLaM promises both efficient and high-quality training.

1. Introduction

Efficient distributed training across multiple computing nodes is critical for large-scale deep learning tasks nowadays. As a principal training algorithm, Parallel SGD computes a globally averaged gradient either using the *Parameter Server (PS)* [24] or the *All-Reduce* communication primitive [38]. Such global synchronization across all nodes either incurs significant bandwidth cost or high latency, which hampers the training scalability.

Decentralized SGD [36, 7, 25, 26, 4, 10] based on *partial averaging* has become one of the major approaches in the past decade to reduce communication overhead in distributed optimization. Partial averaging, as opposed to *global averaging* used in Parallel SGD, requires every node compute the average of the nodes in its neighborhood, see

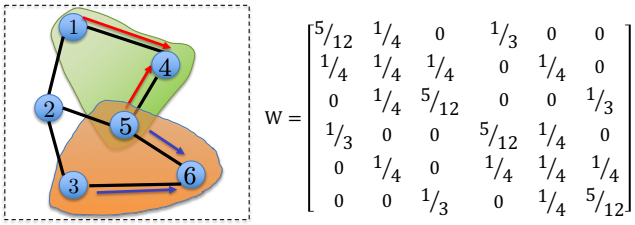

Figure 1. Illustration of decentralized methods. Nodes receive information from neighbors; they do not relay information. For example, nodes 4 and 6 collect information from their neighbors $\{1, 5\}$ and $\{3, 5\}$, respectively. Other nodes do the same but not depicted. Topology connectivity can be represented in a matrix, as shown in the right figure, see more details in Sec. 3.

Fig. 1. If a sparse topology such as one-peer exponential graph [4] is utilized to connect all nodes, each node only communicates with *one* neighbor *each* iteration and hence saves remarkable communications. Decentralized SGD can typically achieve $1.3 \sim 2\times$ training time speedup without performance degradation [25, 4, 23].

Existing Decentralized SGD methods [25, 26, 47, 27, 4] and their momentum accelerated variants [56, 44, 15, 5] primarily utilize small-batch in their algorithm design. However, recent hardware advances make it feasible to store large batches in memory and compute their gradients timely. Furthermore, the total batch size naturally grows when more computing nodes participate into training. These two primary reasons lead to the recent exploration of large-batch deep training algorithms.

In fact, large-batch training has been extensively studied in Parallel SGD. Pioneering works [17, 52, 54] find large-batch can significantly speed up Parallel SGD. First, the computation of a large-batch gradient can fully utilize the computational resources (e.g., the CUDA cores and GPU memories). Second, large-batch gradient will result in a reduced variance and hence enables a much larger learning rate. With newly-proposed layer-wise adaptive rate scaling (LARS) [52] and its variant [54], large-batch Parallel momentum SGD (PmSGD) can cut down the training time of BERT and Resnet-50 from days to hours [55].

This naturally motivates us to study *how Decentralized momentum SGD (DmSGD) performs with large batch-size*.

*Equal contribution. Correspondence can be addressed to Kun Yuan <kun.yuan@alibaba-inc.com>

Dataset	Cifar-10		ImageNet	
Batch-size	2K	8K	2K	32K
PmSGD	91.6%	89.2%	76.5%	75.3%
DmSGD	91.5%	88.3%	76.5%	74.9%

Table 1. Top-1 validation accuracy comparison between PmSGD and DmSGD under the small-batch and large-batch settings. No layer-wise adaptive rate scaling is used in any of these algorithms. All hypter-parameters are exactly the same. More experimental details can be referred to Appendix G.1

To this end, we compared PmSGD and DmSGD over Cifar-10 (Resnet-20) and ImageNet (Resnet-50) with both small and large batches. Their performances are listed in Table 1. While DmSGD achieves the same accuracy as PmSGD with small batch-size, it has far more performance degradation in the large-batch scenario. This surprising observation, which reveals that the extension of DmSGD to large-batch is non-trivial, raises two fundamental questions:

- Why does DmSGD suffer from severe performance degradation in the large-batch scenario?
- How to enhance the accuracy performance of large-batch DmSGD so that it can match with or even beat PmSGD?

This paper focuses on these questions and provides affirmative answers. In particular, our main contributions are:

- We find large-batch DmSGD has severe performance degradation compared to large-batch PmSGD and clarify the reason behind this phenomenon. It is discovered that the momentum term can significantly amplify the inconsistency bias in DmSGD. When batch-size is large and the gradient noise is hence dramatically reduced, such inconsistency bias gets dominant and thus degrades DmSGD's performance notably.
- We propose DecentLaM, a novel decentralized large-batch momentum SGD to remove the momentum-incurred bias in DmSGD. We establish its convergence rate for strongly convex and non-convex scenarios. Our theoretical results show DecentLaM has superior performance to existing decentralized momentum methods, and such superiority gets more evident as batch size grows.
- Experimental results on a variety of computer vision tasks and models show that DecentLaM outperforms DmSGD, DA/QG/D^2-DmSGD, PmSGD, and PmSGD with LARS in terms of both training speed and accuracy.

The rest of this paper is organized as follows: We briefly summarize related works in Sec. 2 and review DmSGD in Sec. 3. We identify the issue that causes performance degradation in DmSGD (Sec. 4) and propose DecentLaM to resolve it (Sec. 5). The convergence analysis and experiments are established in Sec. 6 and Sec. 7.

2. Related Works

Decentralized deep training. Decentralized optimization algorithms can be tracked back to [49]. Decentralized gradient descent [36, 59], diffusion [7, 42] and dual averaging [13] are among the first decentralized algorithms that target on general optimization problems arise from signal processing and control community. In the context of deep learning, Decentralized SGD (DSGD) have gained a lot of attentions recently. DSGD will incur model inconsistency among computing nodes. However, it is established in [25] that DSGD can reach the same linear speedup as vanilla Parallel SGD in terms of convergence rate. After that, [4] comes out to extend DSGD to directed topology. A recent work [21] proposes a unified framework to analyze DSGD with changing topologies and local updates. [26, 35] extended DSGD to the asynchronous setting, and [47, 51, 57] proposed decentralized stochastic primal-dual algorithms to remedy the influence of data heterogeneity. Various communication-efficient techniques can be further integrated into DSGD such as periodic updates [45, 21, 56], communication compression [3, 6, 22, 20, 48], and lazy communication [9, 32].

Momentum SGD training. Momentum SGD methods have been extensively studied due to their empirical success in deep learning. The works [33, 60, 16, 31, 56] establish that momentum SGD converges at least as fast as SGD. Momentum SGD for overparameterized models are shown to converge faster than SGD asymptotically [43]. The exploration in decentralized momentum SGD (DmSGD) is relatively limited. [4] proposes a widely-used DmSGD approach in which a local momentum SGD step is updated first before the partial averaging is conducted (see Algorithm 1). This approach is later extended by [44, 15] to involve communication quantization and periodic local updates. Another work (Doubly-averaging DmSGD, or DA-DmSGD) [56] imposes an additional partial averaging over momentum to increase stability. [5] proposes a new variant in which the partial-averaging step is mixed up with the local momentum SGD update. All these decentralize momentum methods were studied with small batch sizes.

Large-batch training. The main challenge to use large-batch lies in the generalization performance degradation. Recent works in large-batch training centered on adaptive learning rate strategies to enhance accuracy performance. For example, Adam and its variants [19, 39] adjust the learning rate based on the gradient variance, and [17] utilizes learning rate warm-up and linear scaling to boost the performance in large-batch scenario. The layer-wise adaptive rate scaling [52, 53] can reduce the training time of Resnet-50 and BERT from days to hours. However, the study of large-batch training in decentralized algorithms is quite limited. This paper does not propose any adaptive rate strategy for decentralized algorithms. Instead, we will de-

sign an optimization method that can reduce the intrinsic convergence bias in decentralized algorithms and hence improve their performance in the large-batch scenario.

Decentralized methods on heterogeneous data. Decentralized large-batch training within data-centers shares the same essence with decentralized training with heterogeneous data for EdgeAI applications. In large-batch training, the stochastic bias caused by gradient noise gets significantly reduced and the inconsistency bias caused by data heterogeneity will become dominant. [47, 34, 57, 51] proposed decentralized stochastic primal-dual algorithms to remedy the influence of data heterogeneity. However, none of these algorithms show strong effective empirical performances in commenly-used deep learning models such as ResNet-50 or EfficientNet. A concurrent work [27] proposes Quasi-Global momentum, which locally approximates the global optimization direction, to mitigate the affects of heterogeneous data. It is worth noting that while DecentLaM is proposed for the large-batch setting within data-centers, it is also suitable for EdgeAI applications.

3. Decentralized Momentum SGD

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

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

where $f_i(x)$ is local to node i, and random variable ξ_i denotes the local data that follows distribution D_i. Each node i can locally evaluate stochastic gradient $\nabla F(x;\xi_i)$; it must communicate to access information from other nodes.

Notation. We let $[n] := \{1,\cdots,n\}$, and $\mathbb{1} \in \mathbb{R}^d$ be a vector with each element being 1.

Network topology and weights. Decentralized methods are based on partial averaging within neighborhood that is defined by the network topology. We assume all computing nodes are connected by a undirected network topology. Such connected topology can be of any shape, but its degree and connectivity will affect the communication efficiency and convergence rate of the decentralized algorithm. For a given topology, we define w_{ij}, the weight to scale information flowing from node j to node i, as follows:

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

We further define $\mathcal{N}_i := \{j|w_{ij} > 0\}$ as the set of neighbors of node i which also includes node i itself. We define *weight matrix* $W := [w_{ij}]_{i,j=1}^n \in \mathbb{R}^{n\times n}$ to stack all weights into a matrix. Such matrix W will characterize the sparsity and connectivity of the underlying network topology. An example of the topology and its associated weight matrix W is illustrated in Fig. 1.

Algorithm 1: DmSGD

Require: Initialize γ, $x_i^{(0)}$; let $m_i^{(0)}=0, \beta\in(0,1)$

for $k=0,1,2,...,T-1$, *every node* i **do**

Sample $\xi_i^{(k)}$ and update $g_i^{(k)}=\nabla F(x_i^{(k)};\xi_i^{(k)})$

$m_i^{(k+1)} = \beta m_i^{(k)} + g_i^{(k)}$ ▷ momentum update

$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma m_i^{(k+1)}$ ▷ local model update

$x_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})}$ ▷ partial average

Partial averaging. With weights $\{w_{ij}\}$ and the set of neighbors $\mathcal{N}_i$, the neighborhood partial averaging operation of node i can be expressed as

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

Partial averaging has much lower communication overheads. When the network topology is sparse, partial averaging typically incurs $O(1)$ latency plus $O(1)$ bandwidth cost, which are independent of the number of computing nodes n. Consequently, decentralized methods are more communication efficient than those based on global averaging.

Decentralized SGD (DSGD). Given a connected network topology and weights $\{w_{ij}\}$, each node i in DSGD will iterate in parallel as follows:

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma\nabla F(x_i^{(k)};\xi_i^{(k)}) \quad \text{(local update)} \quad (4)$$

$$x_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} w_{ij}\, x_j^{(k+\frac{1}{2})} \quad \text{(partial averaging)} \quad (5)$$

where $x_i^{(k)}$ is the local model of node i at iteration k, $\xi_i^{(k)}$ is the realization of ξ_i at iteration k, and γ is the learning rate. When the network topology is fully connected and $w_{ij} = 1/n$, DSGD will reduce to the Parallel SGD algorithm.

Decentralized momentum SGD (DmSGD). Being the momentum accelerated extension of DSGD, DmSGD has been widely-used in existing literatures [26, 4, 44, 15, 56, 5]. The primary version of DmSGD is listed in Algorithm 1. When small-batch is used, DmSGD will achieve $1.3\sim 2\times$ speedup in training time compared to PmSGD without visibly loss of generalization performance.

Assumptions. We introduce several standard assumptions to facilitate future analysis:

A.1 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 The random sample $\xi_i^{(k)}$ is independent of each other for any k and i. We also assume each stochastic gradient is unbiased and has bounded variance, i.e., $\mathbb{E}[\nabla F(x;\xi_i)] = \nabla f_i(x)$ and $\mathbb{E}\|\nabla F(x;\xi_i) - \nabla f_i(x)\|^2 \le \sigma^2$.

A.3 The network topology is strongly connected, and the weight matrix is symmetric and satisfies $W\mathbb{1} = \mathbb{1}$.

Assumption A.3 indicates $\sum_{j\in\mathcal{N}_i} w_{ij} = 1$ for $i \in [n]$, which is critical to guarantee the partial averaging (3) to converge to the global averaging asymptotically. The weight matrix satisfying Assumption A.3 can be easily constructed, see [42, Table 14.1].

4. DmSGD Incurs Severe Inconsistency Bias

Table 1 shows that large-batch DmSGD has a severe performance degradation compared to PmSGD. This section targets to explore the reason behind this phenomenon. To highlight the insight, we assume each $f_i(x)$ to be strongly convex, and $x^\star$ to be the global solution to problem (1).

Limiting bias of decentralized methods. The convergence of decentralized methods such as DSGD and DmSGD will suffer from two sources of bias: **Stochastic bias** is caused by the utilization of stochastic gradient in algorithms; **Inconsistency bias** is caused by the data inconsistency between nodes.
In addition, these two bias are orthogonal to each other, i.e.,

$$\lim_{k\to\infty}\sum_{i=1}^{n}\mathbb{E}\|x_i^{(k)}-x^\star\|^2 = \text{sto. bias} + \text{inconsist. bias}$$

An example illustrating the stochastic and inconsistency bias of DSGD is established in Appendix C.1.

Inconsistency bias dominates large-batch scenario. In the large-batch scenario, the gradient nosie will be notably reduced. In an extreme case where a full-batch gradient is utilized, the stochastic bias becomes zero. This leads to

Proposition 1. *The inconsistency bias dominates the convergence of* large-batch *decentralized algorithms.*

DmSGD's inconsistency bias: intuition. The inconsistency bias can be achieved by letting gradient noise be zero. To this end, we let $g_i^{(k)} = \mathbb{E}[\nabla F(x_i^{(k)};\xi_i^{(k)})] = \nabla f_i(x_i^{(k)})$ be the full-batch gradient. Note that both model x_i and gradient g_i are deterministic due to the removal of gradient noise. By substituting the momentum update and local model update into the partial averaging step in Algorithm 1, we can rewrite DmSGD (see Appendix B.1) as

$$\begin{aligned} x_i^{(k+1)} = &\underbrace{\sum_{j\in\mathcal{N}_i} w_{ij}\Big(x_j^{(k)} - \gamma\nabla f_j(x_j^{(k)})\Big)}_{\text{DSGD}} \quad \text{(DmSGD)} \\ &\underbrace{+\beta\Big(x_i^{(k)} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{(k-1)}\Big)}_{\text{momentum}}, \ \forall i\in[n]. \end{aligned} \tag{6}$$

Since the momentum term in DmSGD (6) cannot vanish as k increases, it will impose an extra inconsistency bias to DSGD. A simple numerical simulation on a full-batch linear regression problem confirms such conclusion. It is observed in Fig. 2 that DmSGD converges faster but suffers from a larger bias than DSGD.

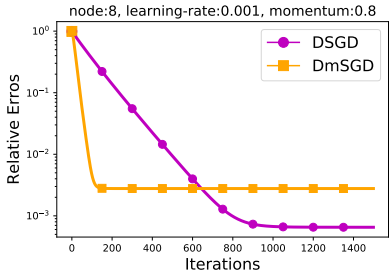

Figure 2. Convergence comparison between DSGD and DmSGD for a full-batch linear regression problem. Detailed experimental setting is in Appendix G.2.

DmSGD's inconsistency bias: magnitude. The following proposition quantitatively evaluates the magnitude of DmSGD's inconsistency bias:

Proposition 2. *Under Assumptions A.1 and A.3, if each $f_i(x)$ is further assumed to be strongly-convex, and the full-batch gradient $\nabla f_i(x)$ is accessed per iteration, then DmSGD (6) has the following inconsistency bias:*

$$\lim_{k\to\infty}\sum_{i=1}^{n}\|x_i^{(k)}-x^\star\|^2 = O\Big(\frac{\gamma^2 b^2}{(1-\beta)^2}\Big), \tag{7}$$

where $b^2 = (1/n)\sum_{i=1}^n\|\nabla f_i(x^\star)\|^2$ denotes the data inconsistency between nodes, and β is the momentum coefficient. (Proof is in Appendix C.2)

Note that we do not take expectation over $\sum_{i=1}^n\|x_i^{(k)}-x^\star\|^2$ in (7) because no gradient noise exists in recursion (6). Recall from Appendix C.1 that DSGD has an inconsistency bias on the order of $O(\gamma^2 b^2)$. Comparing it with (7), it is observed that the momentum term in DmSGD is essentially amplifying the inconsistency bias by a margin of $1/(1-\beta)^2$. This can explain why DmSGD converges less accurate than DSGD as illustrated in Fig. 2. Noting that β is typically set close to 1 in practice, DmSGD can suffer from a significantly large inconsistency bias. Since inconsistency bias dominates the large-batch scenario (see Proposition 1), DmSGD is thus observed to have a severely deteriorated performance as illustrated in Table 1.

5. Improving Inconsistency Bias: DecentLaM

To improve large-batch DmSGD's accuracy, we have to reduce the influence of momentum on inconsistency bias.

Intuition. The amplified inconsistency bias suffered by DmSGD is caused by the non-vanishing momentum term in (6). An intuitive remedy is to replace the partial averag-

ing $\sum_{j\in\mathcal{N}_i} w_{ij}x_j$ with x_i inside the momentum, i.e.,

$$x_i^{(k+1)} = \underbrace{\sum_{j\in\mathcal{N}_i} w_{ij}\Big(x_j^{(k)} - \gamma\nabla f_j(x_j^{(k)})\Big)}_{\text{DSGD}} \quad \text{(DecentLaM)}$$
$$+ \underbrace{\beta\Big(x_i^{(k)} - x_i^{(k-1)}\Big)}_{\text{momentum}}, \ \forall i \in [n]. \tag{8}$$

In the above recursion, the momentum term will diminish to zero as k goes to infinity[1]. As a result, it is expected that recursion (8) will converge to the same solution as DSGD. We simulate the convergence behaviour of recursion (8) under the same setting as Fig. 2. It is observed in Fig. 3 that the proposed algorithm converges as fast as DmSGD but to a more accurate solution. Since recursion (8) is with an improved inconsistency bias, it better fits into the large-batch scenario than DmSGD. We thus name it as decentralized large-batch momentum SGD, or DecentLaM for short.

DecentLaM's inconsistency bias. The following proposition quantitatively evaluates the magnitude of DecentLaM's inconsistency bias (Proof is in Appendix C.3):

Proposition 3. *Under the same assumptions as Proposition 2, DecentLaM (8) has an inconsistency bias as follows:*

$$\lim_{k\to\infty}\sum_{i=1}^{n}\|x_i^{(k)} - x^\star\|^2 = O\Big(\gamma^2 b^2\Big), \tag{9}$$

Remark 1. *Comparing with DmSGD's inconsistency bias (7), it is observed that DecentLaM completely removes the negative effects of momentum; it improves DmSGD's inconsistency bias by a margin of $1/(1-\beta)^2$. When b^2 is large or β is close to 1, such improvement is remarkable.*

Remark 2. *Comparing with DSGD's inconsistency bias $O(\gamma^2 b^2)$, it is observed that DecentLaM has exactly the same inconsistency bias as DSGD. However, the momentum term in DecentLaM will significantly speed up its convergence rate. The simulation results in Fig. 3 are consistent with the conclusions discussed in Remarks 1-2.*

Efficient implementation. To apply DecentLaM (8) to train deep neural networks, we need two additional modifications: (i) the full-batch gradient $\nabla f_i(x)$ needs to be replaced with the stochastic gradient $\nabla F(x;\xi_i)$; (ii) recursion (8) needs to be transformed to fit into the momentum SGD optimizer provided by PyTorch [37] or TensorFlow [2]. To this end, we introduce an important auxiliary variable:

$$g_i^{(k)} = \frac{1}{\gamma}x_i^{(k)} - \frac{1}{\gamma}\sum_{j\in\mathcal{N}_i} w_{ij}\big(x_j^{(k)} - \gamma\nabla F(x_j^{(k)};\xi_j^{(k)})\big) \tag{10}$$

[1]It is because $x_i^{(k)} - x_i^{(k-1)} \to 0$ when $x_i^{(k)}$ converges.

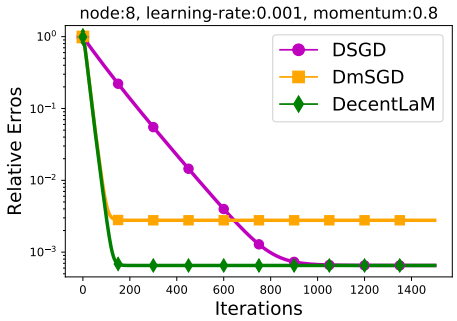

Figure 3. Convergence comparison between DSGD, DmSGD and decentLaM for a full-batch linear regression problem.

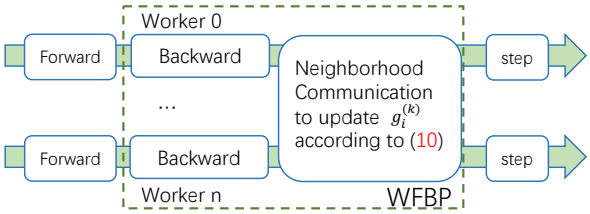

Figure 4. The workflow of DecentLaM in a training iteration.

Note that the construction of $g_i^{(k)}$ involves one round of back-propagation (to evaluate ∇F) and one round of neighborhood communications. With $g_i^{(k)}$ in (10), we can rewrite DecentLaM (8) as in Algorithm 2, see Appendix B.2 for detail derivation. This implementation is almost the same as the vanilla momentum SGD within a single computing node, with the exception to construct $g_i^{(k)}$ as in (10). The momentum update and local model update in Algorithm 2 can be conducted via the mSGD optimizer provided by PyTorch or TensorFlow. As shown in Fig. 4, the dashed part, which is different from the DmSGD workflow, enables an efficient wait-free backpropagation (WFBP) implementation that can overlap communication and computation.

6. Convergence analysis of DecentLaM

Sec. 5 proposes a novel algorithm DecentLaM and explains why it better fits into the large-batch scenario. This section establishes the formal convergence analysis in the strongly-convex and non-convex scenarios, respectively. We let λ_n denotes the smallest eigenvalue of matrix W.

Algorithm 2: DecentLaM

Require: Initialize γ, $x_i^{(0)}$; let $m_i^{(0)} = 0, \beta \in (0,1)$

for $k = 0, 1, 2, ..., T-1$, *every node* i **do**

Sample $\xi_i^{(k)}$ and update $g_i^{(k)}$ according to (10)

$m_i^{(k+1)} = \beta m_i^{(k)} + g_i^{(k)}$ ▷ momentum update

$x_i^{(k+1)} = x_i^{(k)} - \gamma m_i^{(k+1)}$ ▷ local model update

6.1. Strongly-convex scenario

Assumption A.4 Each $f_i(x)$ is μ-strongly convex, i.e., $\langle \nabla f_i(x) - \nabla f_i(y), x-y\rangle \geq \mu\|x-y\|^2$ for any $x, y \in \mathbb{R}^d$.

Theorem 1. *Under Assumption A.1–A.4, if W is further assumed to be positive definite, and the constant learning rate γ is on the order of $O(\frac{1}{L'}\min\{(1-\beta)^2, \frac{\mu}{L'}\})$, the DecentLaM algorithm in Algorithm 2 will converge as (Proof is in Appendix E.1):*

$$\begin{aligned}&\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|x_i^{(k)} - x^\star\|^2 \\ =&O\Big(\underbrace{(1-\frac{\gamma\mu}{1-\beta})^k}_{\text{convg. rate}} + \underbrace{\frac{\gamma\sigma^2}{1-\beta}}_{\text{sto. bias}} + \underbrace{\gamma^2 b^2}_{\text{inconsist.bias}}\Big)\end{aligned} \tag{11}$$

The constant L' is closely related to the Lipschitz constant L, see the definition in Appendix E.1. Expression (11) indicates that DecentLaM will converge exponentially fast to a limiting bias. The limiting bias (which can be achieved by letting $k \to \infty$) can be separated into the stochastic bias and the inconsistency bias, which is consistent with our discussion in Sec. 4. In addition, the inconsistency bias in (11) is consistent with the result in (9). Theorem 1 has restrictions on the positive-definiteness of W so that the analysis can be simplified. However, they are not needed in the real practice and not used in any of our experiments in Sec. 7.

Theorem 1 establishes the convergence of DecentLaM with a constant learning rate. When learning rate is decaying, DecentLaM converges exactly to the global solution:

Corollary 1. *Under the same assumptions as Theorem 1, if learning rate $\gamma = \tilde{O}(\frac{1-\beta}{k})$, DecentLaM will converge as follows (Proof is in Appendix E.2):*

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|x_i^{(k)} - x^\star\|^2 = \tilde{O}\Big(\frac{1}{k}\Big) \tag{12}$$

where $\tilde{O}(\cdot)$ hides all logarithm factors.

Corollary 1 implies that DecentLaM has the same convergence rate $O(1/k)$ as PmSGD in the strongly-convex scenario [31]. However, it saves more communication per iteration due to the partial averaging (see Sec. 3).

6.2. Non-convex scenario

We introduce a standard data inconsistency assumption for the non-convex scenario:

Assumption A.5 There exits a constant $\hat{b} > 0$ such that $\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x) - \nabla f(x)\|^2 \leq \hat{b}^2$ for any x, where $f(x)$ is the global cost function defined in (1).

Assumption A.5 is much more relaxed than the commonly used assumption that each $\nabla f_i(x)$ is upper bounded (i.e., $\|\nabla f_i(x)\| \leq M$) [15, 44, 5]. When data distribution D_i in each computing node i is identical to each other, it holds that $f_i(x) = f_j(x)$ for any i and j, which implies $\hat{b} = 0$.

	Strongly-convex	**Non-convex**
DmSGD [15]	N.A.	$O(\frac{\gamma^2 M^2}{(1-\beta)^2})$
DmSGD [44]	$O(\frac{\gamma^{5/2} M^2}{(1-\beta)^6})$	$O(\frac{\gamma^2 M^2}{(1-\beta)^4})$
DmSGD (eq.(7))	$O(\frac{\gamma^2 b^2}{(1-\beta)^2})$	N.A
DA-DmSGD [56]	N.A.	$O(\frac{\gamma^2 \hat{b}^2}{(1-\beta)^2})$
DecentLaM	$\boldsymbol{O(\gamma^2 b^2)}$	$\boldsymbol{O(\gamma^2 \hat{b}^2)}$

Table 2. Inconsistency bias comparison between various algorithms. Quantity M is the gradient's upper bound, which is typically much larger than the data inconsistency b or $\hat{b}$.

Theorem 2. *Under the same assumptions as Theorem 1 (replacing Assumption A.4 with A.5), the DecentLaM algorithm in Algorithm 2 will converge as follows (Proof is in Appendix F.1, and $\bar{x}^{(k)} = \frac{1}{n}\sum_{i=1}^{n} x_i^{(k)}$):*

$$\begin{aligned}&\frac{1}{T}\sum_{k=0}^{T-1}\Big(\mathbb{E}\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x_i^{(k)})\|^2 + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|x_i^{(k)} - \bar{x}^{(k)}\|^2\Big) \\ =&O\Big(\underbrace{\frac{1-\beta}{\gamma T}}_{\text{convg. rate}} + \underbrace{\frac{\gamma\sigma^2}{1-\beta}}_{\text{sto. bias}} + \underbrace{\gamma^2\hat{b}^2}_{\text{inconsist.bias}}\Big)\end{aligned} \tag{13}$$

Corollary 2. *Under the same assumptions as Theorem 2, if learning rate $\gamma = O(\frac{1-\beta}{\sqrt{T}})$, DecentLaM will converge at rate $O(1/\sqrt{T})$ (Proof is in Appendix F.2), which is consistent with PmSGD [31].*

Remark 3. *In the long report [58], we provide an enhanced version of Theorems 1 and 2, in which DecentLaM is proved to have a linear speedup in the iteration complexity with the increase of the number of computing nodes n.*

6.3. Comparison with existing methods

As we discussed in Sec. 4, the inconsistency bias will dominate the convergence performance as batch-size gets large. For this reason, we list the inconsistency bias comparison between DecentLaM and existing DmSGD variants in Table 2. By removing the influence of momentum, DecentLaM has the smallest inconsistency bias. This implies DecentLaM can perform better in the large-batch scenario.

After completing this work, we become aware of the concurrent work QG-DmSGD [27] that can also achieve $O(\gamma^2\hat{b}^2)$ for non-convex scenario. However, QG-DmSGD is based on a strategy to mimic the global momentum, which is different from the core idea to develop DecentLaM. Furthermore, our analysis is also different from QG-DmSGD. With a delicately designed Lyapunov function, we can cover the convergence analysis for both non-convex

and strongly-convex scenarios. In contrast, the analysis in QG-DmSGD is for the non-convex scenario. Finally, we theoretically uncovered the reason why traditional DmSGD [56, 15, 44] has degraded performance in Sec. 4 while QG-DmSGD only justifies it empirically.

7. Experiments

In this section, we systematically compare the proposed method, DecentLaM, with well-known state-of-the-art methods on typical large-scale computer vision tasks: image classification and object detection. In particular, we compare DecentLaM with the following algorithms. **PmSGD** is the standard Parallel momentum SGD algorithm in which a global synchronization across all nodes are required per iteration. **PmSGD + LARS** exploits a layer-wise adaptive rate scaling strategy [52] to boost performance with large batch-size. **DmSGD** [4, 15, 44], **DA-DmSGD** [56], **QG-DmSGD** [27] and **D^2-DmSGD** [47] are among the state-of-the-art decentralized momentum SGD methods. The details of these decentralized baselines can be refereed to our long report [58]. All baseline algorithms are tested with the recommended hyper-parameters in their papers, and comparison with more baselines (such as **AWC-DmSGD** [5] and **SlowMo** [50]) is listed in [58]. Note that DecentLaM requires *symmetric* (but not necessarily positive definite) weight matrix to guarantee empirical convergence.

We implement all the aforementioned algorithms with PyTorch [37] 1.6.0 using NCCL 2.8.3 (CUDA 10.2) as the communication backend. For PmSGD, we used PyTorch's native Distributed Data Parallel (DDP) module. For the implementation of decentralized methods, we utilize BlueFog [1], which is a high-performance decentralized deep training framework, to facilitate the topology organization, weight matrix generation, and efficient partial averaging. We also follow DDP's design to enable computation and communication overlap. Each server contains 8 V100 GPUs in our cluster and is treated as one node. The inter-node network fabrics are 25 Gbps TCP as default, which is a common distributed training platform setting. To eliminate the effect of topology with different size in decentralized algorithms, we used 8 nodes (i.e. $8 \times 8 = 64$ GPUs) in all decentralized training and changed the batch size of every single GPU respectively.

7.1. Image Classification

Implementation. We conduct a series of image classification experiments with the ImageNet-1K [12] dataset, which consists of 1,281,167 training images and 50,000 validation images in 1000 classes. We train classification models with different batch sizes to verify our theoretical findings. As suggested in [17], we choose batch size equal or less than 8k as small-batch setting and the training protocol in [17] is used. In details, we train total 90 epochs. The learning rate is warmed up in the first 5 epochs and is decayed by a factor of 10 at 30, 60 and 80 epochs. For large-batch setting (batch size larger than 8K), we train total 120 epochs. The learning rate is warmed up in the first 20 epochs and is decayed in a cosine annealing scheduler as suggested in [53]. The choice of hyper-parameter setting is not cherry-picked and we try to keep our PmSGD's baseline around 76% while [17]'s setting has severe performance drop even for large-batch PmSGD. The momentum SGD optimizer is with linear scaling by default. Experiments are trained in the mixed precision using Pytorch native amp module.

METHOD	BATCH SIZE 2K	8K	16K	32K
PMSGD	76.32	76.08	76.27	75.27
PMSGD+LARS[52]	76.16	75.95	76.65	75.63
DMSGD[4, 15, 44]	76.27	76.01	76.23	74.97
DA-DMSGD[56]	76.35	**76.19**	76.62	75.51
QG-DMSGD[27]	76.23	75.96	76.60	75.86
D^2-DMSGD[47]	75.44	75.30	76.16	75.44
DECENTLAM	**76.43**	**76.19**	**76.73**	**76.22**

Table 3. Top-1 validation accuracy of aforementioned methods when training ResNet-50 model with different batch sizes.

Performance with different batch-sizes. We first compare the top-1 accuracy of the aforementioned methods when training ResNet-50 [18] model (∼25.5M parameters) with different batch-sizes on ImageNet. The network topology is set as the symmetric exponential topology (see G.3). The results are listed in Table 3. It is observed that:

- DecentLaM always has better accuracy than other decentralized algorithms, and its superiority gets evident as batch size grows. DecentLaM outperforms DmSGD and DA-DMSGD by a large margin in the 32K batch size scenario due to its improved inconsistency bias derived in Proposition 3, Theorems 1 and 2.

- DecentLaM even outperforms PmSGD and PmSGD + LARS in each scenario. One conjecture is that its model inconsistency between nodes caused by the partial averaging helps the algorithm escape from shallow local minimums. As a result, DecentLaM can be better as well as faster than DmSGD (with LARS).

- DmSGD and DA-DmSGD have a severe degraded performance in the 32K batch size scenario. It is because the momentum have amplified their inconsistency bias, see Proposition 2 and the results listed in Table 2. QG-DmSGD has relatively small performance degradation, but it still performs worse than DecentLaM.

We also depict the training loss and top-1 validation accuracy curves on ResNet-50 with 2K and 16K batch sizes

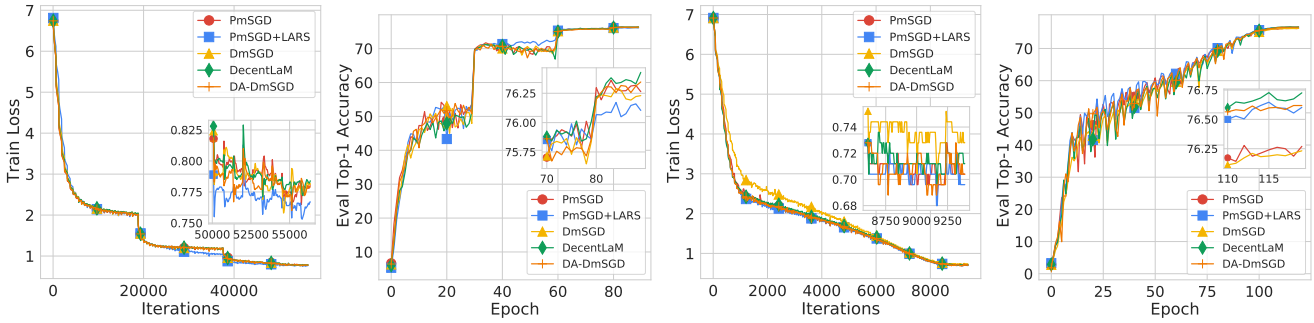

Figure 5. Convergence results on the ImageNet with respect to training loss and validation top-1 accuracy. The batch size is 2k for left-side two figures and 16k for right-side two figures.

MODEL	RESNET-18			RESNET-34			RESNET-50			MOBILENET-V2			EFFICIENTNET		
BATCH SIZE	2K	8K	16K	2K	8K	16K	2K	8K	16K	2K	8K	16K	2K	8K	16K
PMSGD	70.0	69.5	68.3	**73.8**	72.8	72.9	76.3	76.1	76.3	70.6	71.1	69.5	77.6	77.0	78.1
PMSGD+LARS	70.2	69.9	**70.6**	73.6	**73.2**	73.3	76.2	76.0	**76.7**	**70.7**	71.3	**72.3**	77.7	77.2	78.2
DMSGD	69.9	69.3	68.7	73.1	72.7	72.4	76.3	76.0	76.2	70.5	71.0	72.1	77.5	76.9	77.5
DA-DMSGD	70.2	**70.0**	70.2	73.3	73.1	73.1	**76.4**	**76.2**	76.6	70.4	**71.4**	72.1	77.6	**77.4**	77.8
QG-DMSGD	70.2	**70.5**	70.3	**73.8**	73.1	73.3	76.2	76.0	76.6	**70.8**	70.6	72.0	76.8	76.5	76.7
D^2-DMSGD	69.1	68.9	70.1	72.7	72.3	73.1	75.4	75.3	76.2	70.2	70.6	71.8	76.6	76.4	76.5
DECENTLAM	**70.3**	69.9	70.5	73.4	73.1	**73.4**	**76.4**	**76.2**	**76.7**	70.3	**71.4**	72.2	**77.8**	77.2	**78.3**

Table 4. Top-1 validation accuracy comparison with different models and batch sizes on ImageNet dataset.

in Fig. 5. When batch size is 2K, it is observed that DecentLaM achieves roughly the same training loss as DmSGD. However, as batch size grows to 16K, DecentLaM achieves visibly smaller training loss than DmSGD. This observation illustrates that DecentLaM improves inconsistency bias which can boost its convergence when batch-size is large. For small-batch settings, DA-DmSGD and QG-DmSGD can also achieve the best accuracy for some models.

Performance with different models. We now validate whether DecentLaM is effective to different neural network architectures. Table 4 compares the top-1 accuracy with backbones widely-used in image classification tasks including ResNet [18], MobileNetv2 [41] and EfficientNet [46]. The accuracy of PmSGD is slightly different from those reported in [41, 46] because we do not utilize tricks such as AutoAugment [11] that are orthogonal to our methods. It is observed in Table 4 that either PmSGD + LARS or DecentLaM achieves best performance with large batch-size (16K) across different models. However, DecentLaM has runtime speedup due to its efficient neighborhood-communication.

7.2. Object Detection

We compared the aforementioned methods with well-known detection models, e.g. Faster-RCNN [40] and RetinaNet [29] on popular PASCAL VOC [14] and COCO [30] datasets. We adopt the MMDetection [8] framework as the building blocks and utilize ResNet-50 with FPN [28] as the backbone network. We choose mean Average Precision (mAP) as the evaluation metric for both datesets. We used 8 GPUs (which are connected by the symmetric exponential topology) and set the total batch size as 256 in all detection experiments. Table 5 shows the performance of each algorithm. Compared with classification tasks, the total batch size in object detection cannot be set too large (typical object detection task sets batch size as 2 within each GPU). For this reason, our proposed method just reaches a slightly higher mAP than other methods.

DATASET	PASCAL VOC		COCO	
MODEL	R-NET	F-RCNN	R-NET	F-RCNN
PMSGD	79.0	80.3	36.2	36.5
PMSGD+LARS	78.5	79.8	35.7	36.2
DMSGD	79.1	80.5	36.1	36.4
DA-DMSGD	79.0	80.5	36.4	37.0
DECENTLAM	**79.3**	**80.7**	**36.6**	**37.1**

Table 5. Comparision of aforementioned methods with different models on PASCAL VOC and COCO datasets. R-Net and F-RCNN refer to RetinaNet and Faster-RCNN respectively.

8. Conclusion

We investigated the performance degradation in large-batch DmSGD and proposed DecentLaM to remove the momentum-incurred bias. Theoretically, we demonstrate the convergence improvement in smooth convex and non-convex scenarios. Empirically, experimental results of various models and tasks validate our theoretical findings.

References

[1] BlueFog: A High-performance Decentralized Training Framework for Deep Learning. https://github.com/Bluefog-Lib/bluefog. 7, 33

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. 5

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017. 2

[4] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning (ICML)*, pages 344–353, 2019. 1, 2, 3, 7, 33

[5] Aditya Balu, Zhanhong Jiang, Sin Yong Tan, Chinmay Hedge, Young M Lee, and Soumik Sarkar. Decentralized deep learning using momentum-accelerated consensus. *arXiv preprint:2010.11166*, 2020. 1, 2, 3, 6, 7

[6] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. *arXiv preprint:1810.05291*, 2018. 2

[7] 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. 1, 2

[8] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint:1906.07155*, 2019. 8

[9] 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*, pages 5050–5060, 2018. 2

[10] 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)*, 2021. 1

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint:1805.09501*, 2018. 8

[12] 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)*, pages 248–255. Ieee, 2009. 7

[13] 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, 2011. 2

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 8

[15] Hongchang Gao and Heng Huang. Periodic stochastic gradient descent with momentum for decentralized training. *arXiv preprint:2008.10435*, 2020. 1, 2, 3, 6, 7

[16] Igor Gitman, Hunter Lang, Pengchuan Zhang, and Lin Xiao. Understanding the role of momentum in stochastic gradient methods. *arXiv preprint:1910.13962*, 2019. 2

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint:1706.02677*, 2017. 1, 2, 7

[18] 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)*, pages 770–778, 2016. 7, 8

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014. 2

[20] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2019. 2

[21] 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)*, pages 1–12, 2020. 2

[22] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487, 2019. 2

[23] Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian U Stich. Consensus control for decentralized deep learning. *arXiv preprint:2102.04828*, 2021. 1

[24] 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. 1

[25] 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*, pages 5330–5340, 2017. 1, 2

[26] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052, 2018. 1, 2, 3

[27] 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. 1, 3, 6, 7

[28] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the*

IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017. 8

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 8

[31] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. *arXiv preprint:2007.07989*, 2020. 2, 6, 18

[32] 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, 2019. 2

[33] Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *Computational Optimization and Applications*, 77(3):653–710, 2020. 2

[34] 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)*, pages 315–321. IEEE, 2019. 3

[35] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 401–416, 2020. 2

[36] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009. 1, 2

[37] 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)*, pages 8024–8035, 2019. 5, 7

[38] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009. 1

[39] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint:1904.09237*, 2019. 2

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 8

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 8

[42] Ali H Sayed. Adaptation, learning, and optimization over networks. *Foundations and Trends in Machine Learning*, 7(ARTICLE):311–801, 2014. 2, 4, 33

[43] Othmane Sebbouh, Robert M Gower, and Aaron Defazio. On the convergence of the stochastic heavy ball method. *arXiv preprint:2006.07867*, 2020. 2

[44] Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi. Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization. *arXiv preprint:2005.07041*, 2020. 1, 2, 3, 6, 7

[45] Sebastian Urban Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[46] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 8

[47] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. d^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4848–4856, 2018. 1, 2, 3, 7

[48] 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*, pages 6155–6165. PMLR, 2019. 2

[49] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986. 2

[50] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. In *International Conference on Learning Representations*, 2019. 7

[51] Ran Xin, Usman A Khan, and Soummya Kar. An improved convergence analysis for decentralized online stochastic nonconvex optimization. *arXiv preprint:2008.04195*, 2020. 2, 3

[52] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint:1708.03888*, 2017. 1, 2, 7

[53] Yang You, Jonathan Hseu, Chris Ying, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large-batch training for lstm and beyond. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2019. 2, 7

[54] 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:1904.00962*, 2019. 1

[55] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–10, 2018. 1

[56] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Confer-*

ence on Machine Learning, pages 7184–7193. PMLR, 2019. 1, 2, 3, 6, 7, 18

[57] Kun Yuan, Sulaiman A Alghunaim, Bicheng Ying, and Ali H Sayed. On the influence of bias-correction on distributed stochastic optimization. *IEEE Transactions on Signal Processing*, 2020. 2, 3, 13

[58] Kun Yuan, Yiming Chen, Xinmeng Huang, Yingya Zhang, Pan Pan, Yinghui Xu, and Wotao Yin. Decentlam: Decentralized momentum sgd for large-batch deep training (long report). *arXiv preprint:2104.11981*, 2021. 6, 7

[59] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016. 2

[60] Kun Yuan, Bicheng Ying, and Ali H Sayed. On the influence of momentum acceleration on online learning. *The Journal of Machine Learning Research*, 17(1):6602–6667, 2016. 2