Abstract
Decentralized training of deep learning models is a key element for enabling data privacy and on-device learning over networks. In realistic learning scenarios, the presence of heterogeneity across different clients' local datasets poses an optimization challenge and may severely deteriorate the generalization performance.

In this paper, we investigate and identify the limitation of several decentralized optimization algorithms for different degrees of data heterogeneity. We propose a novel momentum-based method to mitigate this decentralized training difficulty. We show in extensive empirical experiments on various CV/NLP datasets (CIFAR-10, ImageNet, and AG News) and several network topologies (Ring and Social Network) that our method is much more robust to the heterogeneity of clients’ data than other existing methods, by a significant improvement in test performance (1%−20%). Our code is publicly available.

1. Introduction
Decentralized machine learning methods—allowing communications in a peer-to-peer fashion on an underlying communication network topology (without a central coordinator)—have emerged as an important paradigm in large-scale machine learning (Lian et al., 2017; 2018; Koloskova et al., 2019; 2020b). Decentralized Stochastic Gradient Descent (DSGD) methods offer (1) scalability to large datasets and systems in large data-centers (Lian et al., 2017; Assran et al., 2019; Koloskova et al., 2020a), as well as (2) privacy-preserving learning for the emerging EdgeAI applications (Kairouz et al., 2019; Koloskova et al., 2020a), where the training data remains distributed over a large number of clients (e.g. mobile phones, sensors, or hospitals) and is kept locally (never transmitted during training).

A key challenge—in particular in the second scenario—is the large heterogeneity (non-i.i.d.-ness) in the data present on the different clients (Zhao et al., 2018; Kairouz et al., 2019; Hsieh et al., 2020). Heterogeneous data (e.g. as illustrated in Figure 1) causes very diverse optimization objectives on each client, which results in slow and unstable global convergence, as well as poor generalization performance (shown in the inline table of Figure 1). Addressing these optimization difficulties is essential to realize reliable decentralized deep learning applications. Although such challenges have been theoretically pointed out in Shi et al. (2015); Lee et al. (2015); Tang et al. (2018b); Koloskova et al. (2020b), the empirical performance of different DSGD methods remains poorly understood. To the best of our knowledge, there currently exists no efficient, effective, and robust optimization algorithm yet for decentralized deep learning on heterogeneous data.

In the meantime, SGD with momentum acceleration (SGDm) remains the current workhorse for the state-of-the-art (SOTA) centralized deep learning training (He et al., 2016; Goyal et al., 2017; He et al., 2019). For decentralized deep learning, the currently used training recipes (i.e. DSGDm) maintain a local momentum buffer on each worker (Assran et al., 2019; Koloskova et al., 2020a; Nadiradze et al., 2020; Singh et al., 2020; Kong et al., 2021) while only communicating the model parameters to the neighbors. However, these attempts in prior work mainly consider homogeneous decentralized data—and there is no evidence that local momentum enhances generalization performance of decentralized deep learning on heterogeneous data.

As our first contribution, we investigate how DSGD and DSGDm are impacted by the degree of data heterogeneity and the choice of the network topology. We find that heterogeneous data hinders the local momentum acceleration in DSGDm. We further show that using a high-quality shared momentum buffer (e.g. synchronizing the momentum buffer globally) improves the optimization and generalization performance of DSGDm. However, such a global communication significantly increases the communication cost and violates the decentralized learning setup.
We instead propose Quasi-Global (QG) momentum, a simple, yet effective method that mitigates the difficulties for decentralized learning on heterogeneous data. Our approach is based on locally approximating the global optimization direction without introducing extra communication overhead. We demonstrate in extensive empirical results that QG momentum can stabilize the optimization trajectory, and that it can accelerate decentralized learning achieving much better generalization performance under high data heterogeneity than previous methods.

- We systematically examine the behavior of decentralized optimization algorithms on standard deep learning benchmarks for various degrees of data heterogeneity.
- We propose a novel momentum-based decentralized optimization method—QG-DSGDm and QG-DSGDm-N—to stabilize the local optimization. We validate the effectiveness of our method on a spectrum of non-i.i.d. degrees and network topologies—it is much more robust to the data heterogeneity than all other existing methods.
- We rigorously prove the convergence of our scheme.
- We additionally investigate different normalization methods alternative to Batch Normalization (BN) (Ioffe & Szegedy, 2015) in CNNs, due to its particular vulnerable to non-i.i.d. local data and the caused severe quality loss.

2. Related Work

Decentralized Deep Learning. The study of decentralized optimization algorithms dates back to Tsitsiklis (1984), relating to use gossip algorithms (Kempe et al., 2003; Xiao & Boyd, 2004; Boyd et al., 2006) to compute aggregates (find consensus) among clients. In the context of machine learning/deep learning, combining SGD with gossip averaging (Lian et al., 2017; 2018; Assran et al., 2019; Koloskova et al., 2020b) has gained a lot of attention recently for the benefits of computational scalability, communication efficiency, data locality, as well as the favorable leading term in the convergence rate $O\left(\frac{1}{\sqrt{n\varepsilon}}\right)$ (Lian et al., 2017; Scaman et al., 2017; Tang et al., 2018; Koloskova et al., 2019; 2020a,b) which is the same as in centralized mini-batch SGD (Dekel et al., 2012). A weak version of decentralized learning also covers the recent emerging federated learning (FL) setting (Konečný et al., 2016; McMahan et al., 2017; Kairouz et al., 2019; Karimireddy et al., 2020b; Lin et al., 2020b) by using (centralized) star-shaped network topology and local updates. Note that specializing our results to the FL setting is beyond the scope of our work. It is also non-trivial to adapt certain very recent techniques developed in FL for heterogeneous data (Karimireddy et al., 2020b; Lin et al., 2020b; Wang et al., 2020a; Das et al., 2020; Haddadpour et al., 2021) to the gossip-based decentralized deep learning.

A line of recent works on decentralized stochastic optimization, like $D^2$/Exact-diffusion (Tang et al., 2018b; Yuan et al., 2020a; Yuan & Alghunaim, 2021), and gradient tracking (Pu & Nedić, 2020; Pan et al., 2020; Lu et al., 2019), proposes different techniques to theoretically eliminate the influence of data heterogeneity between nodes. However, it remains unclear if these theoretically sound methods still endow with superior convergence and generalization properties in deep learning.
Other works focus on improving communication efficiency, from the aspect of communication compression (Tang et al., 2018a; Koloskova et al., 2019; 2020a; Lu & De Sa, 2020; Taheri et al., 2020; Singh et al., 2020; Vogels et al., 2020; Taheri et al., 2020; Nadiradze et al., 2020), less frequent communication through multiple local updates (Hendrikx et al., 2019; Koloskova et al., 2020b; Nadiradze et al., 2020), or better communication topology design (Nedić et al., 2018; Assran et al., 2019; Wang et al., 2019; 2020b; Neglia et al., 2020; Nadiradze et al., 2020; Kong et al., 2021).

Mini-batch SGD with Momentum Acceleration. Momentum is a critical component for training the SOTA deep neural networks (Sutskever et al., 2013; Lucas et al., 2019). Despite various empirical successes, the current theoretical understanding of momentum-based SGD methods remains limited (Bottou et al., 2016). A line of work on the serial (centralized) setting has aimed to develop a convergence analysis for different momentum methods as a special case (Yan et al., 2018; Gitman et al., 2019). However, SGD is known to be optimal in the worst case for stochastic non-convex optimization (Arjevani et al., 2019).

In distributed deep learning, most prior works focus on homogeneous data (especially for numerical evaluations) and incorporate momentum with a locally maintained buffer (which has no synchronization) (Lian et al., 2017; Assran et al., 2018a; Taheri et al., 2020; Singh et al., 2020; Vogels et al., 2020; Nadiradze et al., 2020). However, it often fails on distributed deep learning non-convex optimization problems $f : \mathbb{R}^d \to \mathbb{R}$ of the form

$$f^* := \min_{x \in \mathbb{R}^d} \left[ f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right],$$

where the components $f_i : \mathbb{R}^d \to \mathbb{R}$ are distributed among the $n$ nodes and are given in stochastic form: $f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [F_i(x, \xi)]$, where $\mathcal{D}_i$ denotes the local data distribution on node $i \in [n]$. In (centralized)SGD, each node $i$ maintains local parameters $\mathbf{x}_i^{(t)} \in \mathbb{R}^d$, and updates them as:

$$\mathbf{x}_i^{(t+1)} := \mathbf{x}_i^{(t)} - \eta \nabla F_i(\mathbf{x}_i^{(t)}, \xi_i^{(t)}) \quad \text{(SGD)},$$

that is, by a stochastic gradient step based on a sample $\xi_i^{(t)} \sim \mathcal{D}_i$, followed by gossip averaging with neighboring nodes in the network topology encoded by the mixing weights $w_{ij}$.

In this paper, we denote DSGD with local HeavyBall momentum by DSGDm, and DSGD with local Nesterov momentum by DSGDm-N; the naming rule also applies to our method. For the sake of simplicity, we use HeavyBall momentum variants in Section 3 and 4 for analysis purposes.

3.2. QG-DSGDm Algorithm

To motivate the algorithm design, we first illustrate the impact of using different momentum buffers (local vs. global) on distributed training on heterogeneous data.

Heterogeneous data hinders local momentum acceleration—an example 2D optimization illustration. In Figure 2, we show a toy 2D optimization example that simulates the biased local gradients caused by heterogeneous data. It depicts the optimization trajectories of two agents ($n = 2$) that start the optimization from the position $(0, 0)$ and receive in every iteration a gradient that points to the local minimum $(0, 5)$ and $(4, 0)$, respectively. The gradient is given by the direction from the current model (position) to the local minimum, and scaled to a constant update magnitude. Model synchronization (i.e. uniform averaging) is performed for every local model update step.

Heterogeneous data strongly influences the effectiveness of the local momentum acceleration. Though local momentum in Figure 2(b) assists the models to converge to the neighborhood of the global minimum (better convergence than when

\[\text{...}\]
excluding local momentum in Figure 2(a)), it also causes an unstable and oscillation optimization trajectory. The problem gets even worse in decentralized deep learning, where the learning relies on stochastic gradients from non-convex function and only has limited communication.

Synchronizing the local momentum buffers boosts decentralized learning. We here consider a hypothetical method, which synchronizes the local momentum buffer as in Yu et al. (2019), to use the global momentum buffer locally (avoid using ill-conditioned local momentum buffer caused by heterogeneous data, as shown by the poor performance in Figure 1). We can witness from Table 5 that synchronizing the buffer per update step by global averaging to some extent mitigates the issue caused by heterogeneity (1%–5% improvement comparing row 3 with row 7 in Table 5). Despite its effectiveness, the global synchronization fundamentally violates the realistic decentralized learning setup and introduces extra communication overhead.

Our proposal—QG-DSGDm. Motivated by the performance gain brought by employing a global momentum buffer—a communication-free approach to mimic the global optimization direction—to mitigate the difficulties for decentralized learning on heterogeneous data. Integrating quasi-global momentum with local stochastic gradients alleviates the drift in the local optimization direction, and thus results in a stabilized training and high robustness to heterogeneous data.

Algorithm 1 highlights the difference between DSGDm and QG-DSGDm. Instead of using local gradients from heterogeneous data to form the local momentum (line 4 for DSGDm), which may significantly deflect from the global optimization direction, for QG-DSGDm, we use the difference of two consecutive synchronized models (line 8)

$$d_i^{(t)} = \frac{\eta}{\beta} (x_i^{(t)} - x_i^{(t+1)})$$  \hspace{1cm} (2)

to update the momentum buffer (line 9) by \( \hat{m}_i^{(t)} = \mu m_i^{(t-1)} + (1 - \mu) d_i^{(t)} \). We set \( \mu = \beta \) for all our numerical experiments, without needing hyper-parameter tuning. The update scheme of QG-DSGDm can be re-formulated in matrix form (\( X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}, \) etc.) as follows

$$X^{(t+1)} = W \left( X^{(t)} - \eta \left( \beta M^{(t-1)} + G^{(t)} \right) \right),$$

$$M^{(t)} = \mu M^{(t-1)} + (1 - \mu) \frac{x_i^{(t)} - x_i^{(t+1)}}{\eta}. \hspace{1cm} (3)$$

Figure 2: The ineffectiveness of local momentum acceleration under heterogeneous data setup: the local momentum buffer accumulates “biased” gradients, causing unstable and oscillation behaviors. The size of marker will increase by the number of update steps; colors blue and green indicate the local models of two workers (after performing local update), while color black is the synchronized global model. Uniform weight averaging is performed after each update step, and the new gradients will be computed on the averaged model. We use the common \( \beta = 0.9 \) in this illustration and more results on different \( \beta \) values refer to Appendix D.2.
We provide a convergence analysis for our novel QG-DSGDm for non-convex functions. The proof details can be found in Appendix C.

**Assumption 1.** We assume that the following hold:

- The function \( f(x) \) we are minimizing is lower bounded from below by \( f^* \), and each node’s loss \( f_i \) is smooth satisfying \( \|\nabla f_i(y) - \nabla f_i(x)\| \leq L \|y - x\| \).

- The stochastic gradients within each node satisfies \( \mathbb{E}[g_i(x)] = \nabla f_i(x) \) and \( \mathbb{E}[|g_i(x) - \nabla f_i(x)|^2] \leq \sigma^2 \). The variance across the workers is also bounded as \( \frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2 \).

- The mixing matrix is doubly stochastic: for all ones \( \sum_i x_{ij} = 1 \) for all matrices \( X \) satisfying \( \sum_j x_{ij} = 1 \).

**Theorem 3.1** (Convergence of QG-DSGDm for non-convex functions). Given Assumption 1, the sequence of iterates generated by (3) for step size \( \eta = \mathcal{O}\left(\frac{1}{\sqrt{\sigma^2 T}}\right) \) and momentum parameter \( \beta \) satisfies

\[
\frac{1}{n} \sum_{t=1}^{T} \|\nabla f_i(\bar{x}^t)\|^2 \leq \epsilon \quad \text{in iterations}
\]

where \( \zeta^2 := \epsilon^2 + (1 + \frac{1 - \beta}{\beta \eta})\sigma^2 \).

**Remark 3.2.** The asymptotic number of iterations required, \( \mathcal{O}\left(\frac{n^2}{\epsilon^2}\right) \) shows perfect linear speedup in the number of workers \( n \), independent of the communication topology. This upper bound matches the convergence bounds of DSGD (Lian et al., 2017) and centralized mini-batch SGD (Dekel et al., 2012), and is optimal (Arjevani et al., 2019). This significantly improves over previous analyses of distributed momentum methods which need \( \frac{L_2 \epsilon^2}{n(1-\beta)^2} \) iterations, slowing down for larger values of \( \beta \) (Yu et al., 2019; Balu et al., 2020). The second drift term \( \frac{1}{\epsilon^2} \) arises due to the non-iid data distribution, and matches the tightest analysis of DSGD without momentum (Koloskova et al., 2020b). Finally, our theorem imposes some constraint on the momentum parameter \( \beta \) (but not on \( \mu \)). In practice however, QG-DSGDm performs well even when this constraint is violated.

### 3.4. Connection with Other Methods

We bridge quasi-global momentum with two recent works below. The corresponding algorithm details are included in Appendix B.1 for clarity.

**Connection with MimeLite.** MimeLite (Karimireddy et al., 2020a) was recently introduced in a preprint for FL on heterogeneous data. It shares a similar ingredient as ours: a “global” movement direction \( d \) is used locally to alleviate the issue caused by heterogeneity. The difference falls into the way of forming \( d \) (c.f. line 8 in Algorithm 1): in MimeLite, \( d \) is the full batch gradients computed on the previously synchronized model, while the \( d \) in our QG-DSGDm is the difference on two consecutive synchronized models.

MimeLite only addresses the FL setting, which results in a computation and communication overhead (to form \( d \)), and is non-trivial to extent to decentralized learning.

**Connection with SlowMo.** SlowMo and its “noaverage” variant (Wang et al., 2020c) aim to improve generalization performance in the homogeneous data-center training scenario, while QG-DSGDm is targeting learning with data heterogeneity. In terms of update scheme, SlowMo variants update the slow momentum buffer through the model difference \( d \) of \( \tau \gg 1 \) local update (and synchronization) steps, while QG-DSGDm only considers consecutive models (analogously \( \tau = 1 \)).3 Besides, in contrast to QG-DSGDm, the slow momentum buffer in SlowMo will never interact with the local update—setting \( \tau \) to 1 in SlowMo variants cannot recover QG-DSGDm.

SlowMo variants are orthogonal to QG-DSGDm; combining these two algorithms may lead to a better generalization performance, and we leave it for future work.

---

3 We also study the variant of QG-DSGDm with \( \tau > 1 \) in Appendix D.8—we stick to \( \tau = 1 \) in the main paper for its superior performance and hyper-parameter (\( \tau \)) tuning free.
4. Understanding QG-DSGDm

4.1. Faster Convergence in Average Consensus

We now consider the simpler averaging consensus problem (isolated from the learning part of QG-DSGDm): we simplify (3) by removing gradients and step-size:

\[ X^{(t+1)} = W \left( X^{(t)} - \beta M^{(t-1)} \right) \]
\[ M^{(t)} = \mu M^{(t-1)} + (1 - \mu) \left( X^{(t)} - X^{(t+1)} \right) \]

and compare it with gossip averaging \( X^{(t+1)} = WX^{(t)} \).

Figure 3 depicts the advantages of (4) over standard gossip averaging, where QG-DSGDm can quickly converge to a critical consensus distance (e.g. \(10^{-2}\)). It partially explains the performance gain of QG-DSGDm from the aspect of improved decentralized communication (which leads to better optimization)—decentralized training can converge as fast as its centralized counterpart once the consensus distance is lower than the critical one, as stated in Kong et al. (2021).

4.2. QG-DSGDm (Single Worker Case) Recovers QHM

Considering the single worker case, QG-DSGDm can be further simplified to (derivations in Appendix B.3.1):

\[ \dot{m}^{(t)} = \beta \dot{m}^{(t-1)} + g^{(t)} \]
\[ x^{(t+1)} = x^{(t)} - \eta \left( 1 - \frac{\mu}{\beta} \right) \dot{m}^{(t)} + \frac{\mu}{\beta} g^{(t)} \]

where \( \dot{\beta} := \mu + (1 - \mu) \beta \). Thus, the single worker case of QG-DSGDm (i.e. QG-SGDm) recovers Quasi-Hyperbolic Momentum (QHM) (Ma & Yarats, 2019; Gitman et al., 2019). We illustrate its acceleration benefits as well as the performance gain in Figure 12 and Figure 13 of Appendix D.3. We elaborate in Appendix B.3 that SGDm is only a special case of QG-SGDm/QHM (by setting \( \mu = 0 \)). Besides, it is non-trivial to adapt (centralized) QHM to (decentralized) QG-DSGDm due to discrepant motivation.

Stabilized optimization trajectory. We study the optimization trajectory of Rosenbrock function (Rosenbrock, 1960)\(^4\) \( f(x, y) = (y - x^2)^2 + 100(x - 1)^2 \) as in Lucas et al. (2019) to better understand the performance gain of QG-SGDm (with zero stochastic noise). Figure 15 illustrates the effects of stabilization in QG-SGDm (much less oscillation than SGDm).

Larger effective step-size. Recent works (Hoffer et al., 2018; Zhang et al., 2019a) point out the larger effective step-size (i.e. \( \eta/\|\dot{x}\|_2^2 \)) brought by weight decay provides the primary regularization effect for deep learning training. Figure 5 examines the effective step-size during the optimization procedure: QG-SGDm illustrates a larger effective

\[ \eta/\|x_t\|_2 \]

\[ 1e^{-5} \]

\[ \text{w/ BN, w/ weight decay, SGDm} \]

\[ \text{w/ GN, w/ weight decay, SGDm} \]

\[ \text{w/ BN, w/ weight decay, QG-SGDm} \]

\[ \text{w/ GN, w/ weight decay, QG-SGDm} \]

\[ \text{QG-SGDm and SGDm (i.e. single worker case) via a 2D toy function} \]

\[ f(x, y) = (y - x^2)^2 + 100(x - 1)^2 \]

\[ \text{This function has a global minimum at} (x, y) = (1, 1) \]

\[ \text{SGDm and QG-SGDm use} \beta = 0.9, \eta = 0.001, \text{with initial point} (0, 0) \]

\[ \text{Trajectories for different initial points and/or} \beta \text{values refer to Appendix D.4} \]

\[ \text{Figure 4: Understanding the optimization trajectory of QG-SGDm and SGDm (single worker case) on CIFAR-10. The weight norm curves refer to Figure 14 in Appendix D.3} \]

\[ \text{step-size than SGDm, explaining the performance gain e.g. in Figure 12 and Figure 13 of Appendix D.3} \]

5. Experiments

5.1. Setup

Datasets and models. We empirically study the decentralized training behavior on both CV and NLP benchmarks, on the architecture of ResNet (He et al., 2016), VGG (Simonyan & Zisserman, 2014) and DistilBERT (Sanh et al., 2019). • Image classification (CV) benchmark: we consider training CIFAR-10 (Krizhevsky & Hinton, 2009), ImageNet-32 (i.e. image resolution of 32) (Chrabaszcz et al., 2017), and ImageNet (Deng et al., 2009) from scratch, with standard data augmentation and preprocessing scheme (He et al., 2016). We use VGG-11 (with width factor 1/2 and without BN) and ResNet-20 for CIFAR-10, ResNet-20 with width factor 2 (noted as ResNet-20-x2) for ImageNet-32, and ResNet-18 for ImageNet. The width factor indicates the proportional scaling of the network width corresponding to the original neural network. Weight initialization schemes follow He et al. (2015); Goyal et al. (2017). • Text classification (NLP) benchmark: we perform fine-tuning on a 4-class classification dataset (AG News (Zhang et al., 2015)). Unless mentioned otherwise, all our experiments are repeated over three random seeds. We report the averaged performance of local models on the full test dataset.

Heterogeneous distribution of client data. We use the Dirichlet distribution to create disjoint non-i.i.d. client

\[ 2 \]

\[ \eta/\|x\|_2 \]

\[ 2 \]

\[ w/ BN, w/ weight decay, QG-SGDm \]

\[ w/ GN, w/ weight decay, QG-SGDm \]

\[ w/ BN, w/ weight decay, SGDm \]

\[ w/ GN, w/ weight decay, SGDm \]

\[ \text{Figure 5: The effective step-size} \eta/\|x_t\|_2 \text{of QG-SGDm and SGDm (single worker case) on CIFAR-10. The weight norm curves refer to Figure 14 in Appendix D.3} \]

\[ \text{step-size than SGDm, explaining the performance gain e.g. in Figure 12 and Figure 13 of Appendix D.3} \]
We also investigate the performance of DSGDm in Table 5 with learning rate scaling and warm-up. The learning rate is \(0\) for the first \(\alpha\) epochs. Besides, the learning rate will be divided by 10 when the model has accessed specified fractions of the total number of training samples—\(\{\frac{1}{2}, \frac{3}{4}\}\) for CIFAR and \(\{\frac{1}{4}, \frac{3}{5}, \frac{7}{8}\}\) for ImageNet.

Table 1: The test top-1 accuracy of different decentralized optimization algorithms evaluated on different degrees of non-i.i.d. local CIFAR-10 data, for various neural architectures and network topologies. The results are averaged over three random seeds, with learning rate tuning for each setting. We also include the results of centralized baseline for reference purposes, following the decentralized experiment configuration, except that the centralized baseline uses randomly partitioned local training data (i.e. independent of \(\alpha\)).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Neural Architectures</th>
<th>Methods</th>
<th>Ring (n = 16)</th>
<th>Social Network (n = 32)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\alpha = 10)</td>
<td>(\alpha = 1)</td>
<td>(\alpha = 0.1)</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td></td>
<td>SGDm-N (centralized)</td>
<td>92.95 (\pm 0.13)</td>
<td>92.88 (\pm 0.07)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGD</td>
<td>90.94 (\pm 0.15)</td>
<td>88.95 (\pm 0.59)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGDm-N</td>
<td>92.53 (\pm 0.27)</td>
<td>89.13 (\pm 0.81)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QG-DSGDm-N</td>
<td>92.65 (\pm 0.17)</td>
<td>91.21 (\pm 0.28)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGDm-N (centralized)</td>
<td>88.06 (\pm 1.12)</td>
<td>86.19 (\pm 1.08)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGD</td>
<td>86.86 (\pm 0.37)</td>
<td>85.93 (\pm 0.14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGDm-N</td>
<td>89.86 (\pm 0.15)</td>
<td>88.30 (\pm 0.49)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QG-DSGDm-N</td>
<td>90.18 (\pm 0.44)</td>
<td>89.68 (\pm 0.41)</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>ResNet-BN-20</td>
<td>SGDm-N (centralized)</td>
<td>92.18 (\pm 0.19)</td>
<td>91.92 (\pm 0.33)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGD</td>
<td>89.90 (\pm 0.26)</td>
<td>88.88 (\pm 0.26)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGDm-N</td>
<td>91.47 (\pm 0.23)</td>
<td>89.98 (\pm 0.10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QG-DSGDm-N</td>
<td>91.90 (\pm 0.17)</td>
<td>91.28 (\pm 0.38)</td>
</tr>
<tr>
<td></td>
<td>VGG-11 (w/o normalization layer)</td>
<td>SGDm-N (centralized)</td>
<td>88.87 (\pm 0.29)</td>
<td>87.38 (\pm 0.39)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGD</td>
<td>88.68 (\pm 0.30)</td>
<td>88.52 (\pm 0.24)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGDm-N</td>
<td>89.01 (\pm 0.04)</td>
<td>89.00 (\pm 0.22)</td>
</tr>
</tbody>
</table>

Training schemes. Following the SOTA deep learning training scheme, we use mini-batch SGD as the base optimizer for CV benchmark (He et al., 2016; Goyal et al., 2017), and similarly, Adam for NLP benchmark (Zhang et al., 2019b; Mosbach et al., 2021). In Section 5.2, we adapt these base optimizers to different distributed variants. For the CV benchmark, the models are trained for 300 and 90 epochs for CIFAR-10 and ImageNet-(32) respectively; the local mini-batch size are set to 32 and 64. All experiments use the SOTA learning rate scheme in distributed deep learning training (Goyal et al., 2017; He et al., 2019) with learning rate scaling and warm-up. The learning rate is always gradually warmed up from a relatively small value (i.e. 0.1) for the first 5 epochs. Besides, the learning rate will be divided by 10 when the model has accessed specified fractions of the total number of training samples—\(\{\frac{1}{2}, \frac{3}{4}\}\) for CIFAR and \(\{\frac{1}{4}, \frac{3}{5}, \frac{7}{8}\}\) for ImageNet.

For the NLP benchmark, we fine-tune the distilbert-base-uncased from HuggingFace (Wolf et al., 2019) with constant learning rate and mini-batch of size 32 for 10 epochs.

We fine-tune the learning rate for both CV and NLP tasks; we use constant weight decay (1e-4). The tuning procedure ensures that the best hyper-parameter lies in the middle of our search grids; otherwise we extend our search grid. Regarding momentum related hyper-parameters, we follow the common practice in the community (\(\beta = 0.9\) and without dampening for Nesterov/HeavyBall momentum variants, and \(\beta_1 = 0.9, \beta_2 = 0.99\) for Adam variants).

BN and its alternatives for distributed deep learning. The existence of BN layer is challenging for the SOTA distributed training, especially for heterogeneous data setting. To better understand the impact of different normalization schemes in distributed deep learning, we investigate:

- Distributed BN implementation. Our default implementation follows Goyal et al. (2017); Andreux et al. (2020) that computes the BN statistics independently for each client while only synchronizing the BN weights.
- Using other normalization layers: for instance on ResNet with BN layers (denoted by ResNet-BN-20), we can instead use ResNet-GN by replacing all BN with GN with group number of 2, as suggested in Hsieh et al. (2020). We also examine the recently proposed S0 variant of EvoNorm (Liu et al., 2020) (which does not use runtime mini-batches statistics), noted as ResNet-EvoNorm.

5 We by default use local momentum variants without buffer synchronization. We consider DSGDm-N as our primary competitor for CNNs, as Nesterov momentum is the SOTA training scheme. We also investigate the performance of DSGDm in Table 5.

6 We tune the initial learning rate and warm it up from 0.1 (if the tuned one is above 0.1).

7 We also try the BN variant (Li et al., 2021) proposed for FL, but we exclude it in our comparison due to its poor performance.
5.2. Results

Comments on BN and its alternatives. Table 1 and Table 3 examine the effects of BN and its alternatives on the training quality of decentralized deep learning on CIFAR-10 and ImageNet datasets. ResNet with EvoNorm replacement outperforms its GN counterpart on a spectrum of optimization algorithms, non-i.i.d. degrees, and network topologies, illustrating its efficacy to be a new alternative to BN in CNNs for distributed learning on heterogeneous data.

Superior performance of quasi-global momentum. We evaluate QG-DSGDm-N and compare it with several DSGD variants in Table 1, for training different neural networks on CIFAR-10 in terms of non-i.i.d. degrees on Ring \((n=16)\) and Social Network \((n=32)\). QG-DSGDm-N accelerates the training by stabilizing the oscillating optimization trajectory caused by heterogeneity and leads to a significant performance gain over all other competitors on all levels of data heterogeneity. The benefits of our method are further pronounced when considering a higher degree of non-i.i.d.-ness. These observations are consistent with the results on the challenging ImageNet(-32) dataset in Table 3 (and the learning curves in Figure 17 in Appendix D.5).

Table 3: Test top-1 accuracy of different decentralized optimization algorithms evaluated on different degrees of non-i.i.d. local ImageNet data. The results are over three random seeds. We perform sufficient learning rate tuning on ImageNet-32 for each setup while we use the same one for ImageNet due to the computational feasibility. "\(\alpha\)" indicates non-convergence.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>Methods</th>
<th>Ring (n = 16)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\alpha = 1)</td>
<td>(\alpha = 0.1)</td>
</tr>
<tr>
<td>ImageNet-32</td>
<td>ResNet-20-x2 (EvoNorm)</td>
<td>SGDm-N (centralized)</td>
<td>44.43 ± 0.20</td>
</tr>
<tr>
<td>(resolution 32)</td>
<td></td>
<td>DSGDm-N</td>
<td>30.35 ± 0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QG-DSGDm-N</td>
<td>31.24 ± 0.27</td>
</tr>
<tr>
<td>ImageNet</td>
<td>ResNet-20-x2 (EvoNorm)</td>
<td>SGDm-N (centralized)</td>
<td>37.89 ± 0.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSGDm-N</td>
<td>34.16 ± 1.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QG-DSGDm-N</td>
<td>38.57 ± 0.45</td>
</tr>
</tbody>
</table>

Decentralized Adam. We further extend the idea of quasi-global momentum to the Adam optimizer for decentralized learning, noted as QG-DAdam (the algorithm details are deferred to Algorithm 2 in Appendix B.1). We validate the effectiveness of QG-DAdam over D(decentralized)Adam in Table 6, on fine-tuning DistilBERT on AG News and training ResNet-EvoNorm-20 on CIFAR-10 from scratch: QG-DAdam is still preferable over DAdam. We leave a better adaptation and theoretical proof for future work.

Generalizing quasi-global momentum to time-varying topologies. The benefits of quasi-global momentum are not limited to the fixed and undirected communication topologies, e.g. Ring and Social network in Table 1—it also generalizes to other topologies, like the time-varying directed topology (Assran et al., 2019), as shown in Table 4 for training ResNet-EvoNorm-18 on ImageNet. These results are aligned with the insights of the critical consensus distance on the generalization performance of decentralized deep learning (Kong et al., 2021), supporting the fact that quasi-global momentum can be served as a simple plug-in to further improve the performance of decentralized deep learning.

Comparison with \(D^2\) and Gradient Tracking (GT). As shown in Table 2, \(D^2\) (Tang et al., 2018b) and GT methods (Pu & Nedić, 2020; Pan et al., 2020; Lu et al., 2019) cannot achieve comparable test performance on the standard deep learning benchmark, while QG-DSGDm-N outperforms them significantly. Additional detailed comparisons are deferred to Appendix D.9.

It is non-trivial to integrate \(D^2\) with momentum. Besides, \(D^2\) requires constant learning rate, which does not fit the SOTA learning rate schedules (e.g. stage-wise) in deep learning. We include an improved \(D^2\) variant\(^8\) (denoted as \(D^2_+\)) to

\(^8\)\(D^2\) can be rewritten as \(w(x(t) - \mu(x(t-1) - x(t))) / \eta + \nabla f(x(t)) - \nabla f(x(t-1)))\), and the update would break if the magnitude of \(x(t-1) - x(t)\) is a factor of \(10\eta\) (i.e. performing learning rate decay at step \(t\)).

\(^9\)The update scheme of \(D^2_+\) follows \(w(x(t) - q(t)) (x(t-1) - x(t)) / q(t) + \nabla f(x(t)) - \nabla f(x(t-1)))\).
Table 5: *An extensive investigation for a wide spectrum of DSGD variants*, for training ResNet-EvoNorm-20 on CIFAR-10. The results are averaged over three seeds, each with learning rate tuning. We use “communication topology” to synchronize the model parameters, while some methods involve “extra communication”, with specified objective to be communicated on the given network topology.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Communication Topology</th>
<th>Extra Communication</th>
<th>Momentum Type</th>
<th>Test Top-1 Accuracy (α = 1)</th>
<th>Test Top-1 Accuracy (α = 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGDm-N</td>
<td>complete</td>
<td>-</td>
<td>global</td>
<td>92.18 ± 0.19</td>
<td></td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>complete</td>
<td>-</td>
<td>local</td>
<td>91.47 ± 0.10</td>
<td>71.24 ± 3.08</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>momentum buffer (complete)</td>
<td>local</td>
<td>90.96 ± 0.33</td>
<td>81.22 ± 1.78</td>
</tr>
<tr>
<td>SlowMo</td>
<td>ring</td>
<td>model parameters (complete)</td>
<td>local &amp; global</td>
<td>91.06 ± 0.26</td>
<td>79.20 ± 1.16</td>
</tr>
<tr>
<td>DSGDm</td>
<td>ring</td>
<td>-</td>
<td>-</td>
<td>88.88 ± 0.26</td>
<td>74.55 ± 2.07</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>-</td>
<td>local</td>
<td>89.67 ± 0.33</td>
<td>77.66 ± 0.95</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>momentum buffer (ring)</td>
<td>local</td>
<td>89.98 ± 0.10</td>
<td>77.48 ± 2.67</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>momentum buffer (ring)</td>
<td>local</td>
<td>90.42 ± 0.32</td>
<td>78.69 ± 2.39</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>local gradients (ring)</td>
<td>local</td>
<td>90.48 ± 0.67</td>
<td>79.83 ± 2.29</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>-</td>
<td>local</td>
<td>90.10 ± 0.61</td>
<td>78.58 ± 4.12</td>
</tr>
<tr>
<td>DSGDm-N</td>
<td>ring</td>
<td>-</td>
<td>local</td>
<td>90.06 ± 0.04</td>
<td>79.89 ± 0.97</td>
</tr>
<tr>
<td>QG-DSGDm</td>
<td>ring</td>
<td>-</td>
<td>local</td>
<td>91.22 ± 0.41</td>
<td>82.24 ± 1.05</td>
</tr>
<tr>
<td>QG-DSGDm-N</td>
<td>ring</td>
<td>-</td>
<td>local</td>
<td><strong>91.28 ± 0.38</strong></td>
<td>82.20 ± 1.27</td>
</tr>
</tbody>
</table>

Besides, Figure 6 showcases the generality of quasi-global momentum for achieving remarkable performance gain on different topology scales and non-i.i.d. degrees.

Figure 6: *Test top-1 accuracy of different decentralized algorithms evaluated on different topology scales and non-i.i.d. degrees*, for training ResNet-EvoNorm-20 on CIFAR-10. The results are over three random seeds, each with sufficient learning rate tuning. Colors blue and red indicate DSGDm-N and QG-DSGDm-N respectively. Numerical results refer to Table 7 in Appendix D.7.

**Conclusion**

We demonstrated that heterogeneity has an out sized impact on the performance of deep learning models, leading to unstable convergence and poor performance. We proposed a novel momentum-based algorithm to stabilize the training and established its efficacy through thorough empirical evaluations. Our method, especially for mildly heterogeneous settings, leads to a 10–20% increase in accuracy. However, a gap still remains between the centralized training. Closing this gap, we believe, is critical for wider adoption of decentralized learning.

**Acknowledgements**

We acknowledge funding from a Google Focused Research Award, Facebook, and European Horizon 2020 FET Proactive Project DIGIPREDICT.
References


Quasi-Global Momentum


