An Empirical Study on Compressed Decentralized Stochastic Gradient Algorithms with Overparameterized Models

Arjun Ashok Rao* and Hoi-To Wai*
* Dept. of SEEM; The Chinese University of Hong Kong; Shatin, Hong Kong
E-mail: arjunrao@link.cuhk.edu.hk, htwai@cuhk.edu.hk

***Abstract*—This paper considers decentralized optimization with application to machine learning on graphs. The growing size of neural network (NN) models has motivated prior works on decentralized stochastic gradient algorithms to incorporate communication compression. On the other hand, recent works have demonstrated the favorable convergence and generalization properties of overparameterized NNs. In this work, we present an empirical analysis on the performance of compressed decentralized stochastic gradient (DSG) algorithms with overparameterized NNs. Through simulations on an MPI network environment, we observe that the convergence rates of popular compressed DSG algorithms are robust to the size of NNs. Our findings suggest a gap between theories and practice of the compressed DSG algorithms in the existing literature.**

***Index Terms*—decentralized optimization, communication efficiency, overparameterized models**

I. Introduction

A popular approach for enabling scalable machine learning is applying decentralized algorithms to tackle the large-scale optimization problem through collaboration between a group of workers connected on a network/graph. Let $N \in \mathbb{N}$ be the number of workers and $d \in \mathbb{N}$ be the problem's dimension, we consider the following unconstrained optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \; J(\boldsymbol{\theta}) \quad \text{where} \quad J(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N} J_i(\boldsymbol{\theta}), \tag{1}$$

where $J_i : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function representing the private data held by the ith worker. Furthermore, the N workers are connected on an undirected graph denoted by $G = (V, E)$, where $V = [N] = \{1, ..., N\}$ is the set of workers and $E \subseteq V \times V$ is the set of edges of G with self loops such that $(i, i) \in E$ for all $i \in V$. Let G be a connected graph, we note that (1) is equivalent to the consensus optimization problem:

$$\min_{\boldsymbol{\theta}_i\in\mathbb{R}^d, i\in V} \; \sum_{i=1}^{N} J_i(\boldsymbol{\theta}_i) \;\; \text{s.t.} \;\; \boldsymbol{\theta}_i = \boldsymbol{\theta}_j, \; \forall \; (i,j) \in E, \tag{2}$$

where $\boldsymbol{\theta}_i \in \mathbb{R}^d$ is a private/local variable held by the ith worker. The applications of (1) include decentralized regression, sensor fusion for wireless sensor network [1], etc.

In this paper, we are concerned with the application of (1) to machine learning (ML) tasks via training a neural network (NN) model. Following the design of (1), our goal is to train a common model $\boldsymbol{\theta}$ at all workers. For example, if we consider a supervised learning problem for classification, the ith private function takes the form of the empirical risk:

$$J_i(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_i|}\sum_{j=1}^{|\mathcal{D}_i|} \mathsf{loss}(f(\boldsymbol{x}_j;\boldsymbol{\theta}); y_j), \tag{3}$$

where $\boldsymbol{x}_j \in \mathbb{R}^f$ and $y_j \in \mathbb{R}$ are the jth feature and label known by worker i, respectively, and $|\mathcal{D}_i|$ is the number of samples held by worker i. The loss function $\mathsf{loss}(\cdot)$ can be taken as the cross-entropy, or the quadratic loss. The nonlinear function $f(\boldsymbol{x};\boldsymbol{\theta})$ is the output of a neural network, e.g., a two-layer neural network with ReLU activation is given by

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{\sqrt{m}}\sum_{j=1}^{m} b_j \max\{0, \langle \boldsymbol{x} \, , \boldsymbol{\theta}^{(j)}\rangle\}, \tag{4}$$

where b_j is the jth output weight and we have defined the parameters as $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(m)}) \in \mathbb{R}^{mf}$ such that $d = mf$. Notice that despite its simplicity, the NN architecture (4) exhibits good representation power provided that $m \to \infty$ [2].

To tackle (1) when only local communications are allowed, decentralized first-order optimization algorithms have been developed for over a decade [1], [3], [4]. The main idea behind these algorithms is a simultaneous "consensus + optimize" strategy where workers communicate with each other to reach a common model $\boldsymbol{\theta}$ (i.e., achieving consensus) while optimizing their local models $\boldsymbol{\theta}_i$ via gradient steps on the private functions. For a non-convex optimization model, the convergence of decentralized optimization algorithms has been analyzed in [5]–[7] under the deterministic setting. Recent works have also analyzed the extension to stochastic gradient (SG) based methods [8]–[12]; also see [13] for a recent overview.

A major obstacle in applying the above algorithms to ML tasks (3) lies in the *communication overhead*. Taking the decentralized gradient (DGD) method from [3] as an example, each training iteration requires workers to transmit the entire d-dimensional local model. Notice that the state-of-the-art NN models are typically large with $d \gg 1$, e.g., the VGG16 NN comprises of $d \approx 1.38 \times 10^8$ parameters [14]. When applying the DGD method to train such a model, workers would be required to send 526 MB of data over the network

per-iteration; This may cause a significant slowdown to the practical convergence of the algorithms.

In light of the above, previous works have been motivated to develop *communication efficient* variants of decentralized SG algorithms. To list a few examples, [15], [16] proposed the CHOCO-SGD algorithm which compresses incremental vectors at the workers prior to transmission; also see its generalization in [17]; [18] proposed the ECD-PSGD algorithm based on an extrapolation technique. It is worth noting that the above algorithms share similarities with the compression techniques proposed for distributed SG such as [19].

On the other hand, recent works have studied the *global convergence* of centralized SG algorithms for training *overparameterized* NN models [20] via the non-convex optimization (1), (3). Particularly, these works consider simple architectures such as (4) and show that the optimal weights $\boldsymbol{\theta}^\star$ to (1) can be found using SG algorithms when $m \to \infty$, e.g., [21], [22]. Interestingly, it is demonstrated that the convergence rate for the L^2 function distance will be independent of m, implying that there is no loss in iteration complexity (for centralized learning) when deploying *overparameterized* NNs.

The above observation illustrates a dilemma when deciding the NN architecture to be used for decentralized training: on one hand, it is desired to adopt a compact NN model to reduce communication cost, on the other hand, using an *overparameterized* NN model provides enticing theoretical convergence properties and higher representation power. The aim of this paper is to empirically study the effects of overparameterization on the performance of (communication efficient) decentralized SG algorithms. Particularly, we verify the claim that *decentralized SG algorithms are practical (in terms of overall communication efficiency), even for training overparameterized NNs*. Our contributions are as follows:

- In Section II, we highlight the pitfalls of existing theories for training NNs with communication-efficient decentralized SG algorithms by showing that the curse of dimensionality persists. We argue that the latter can be an artifact of the analysis as it does not exploit the structure of the NN training problems.
- To support our claim, in Section III, we concentrate on the CHOCO-SGD algorithm [16] and experiment with this algorithm in various setting, providing the *first set of experiments* that highlights on the effects of an increasing NN width m in the model (4). We show that, when the amount of information transmitted per iteration is fixed, the algorithm achieves a lower training loss / testing error at the same speed (or even faster) as the NN's width is increased.

In addition, we discuss the intuitive reason behind the current gap in the theories and suggest potential solutions to fix it.

II. Decentralized Learning: Algorithms and Communication Efficiency

In this section, we discuss the algorithms for decentralized learning/training and their communication efficient variants. Furthermore, we will review the convergence guarantees of these algorithms and discuss how they may fail to yield meaningful insights for training overparameterized NNs.

To fix ideas, we define the doubly stochastic mixing matrix $\boldsymbol{W} \in \mathbb{R}_+^{N\times N}$ satisfying the row/column sum condition $\boldsymbol{W}\mathbf{1} = \boldsymbol{W}^\top\mathbf{1} = \mathbf{1}$; it respects the graph topology such that $W_{ij} = W_{ji} = 0$ whenever $(i,j) \notin E$; moreover, it satisfies the fast mixing condition of a Markov chain such that

$$\|\boldsymbol{W} - \mathbf{1}\mathbf{1}^\top/N\| \leq 1-\rho, \tag{5}$$

where $\rho \in (0,1]$ is the spectral gap. Notice that such matrix exists for any connected graph G.

With the above mixing matrix, the DGD (or its stochastic variant, DSGD) algorithm [3], [8] is given by: starting with any $\boldsymbol{\theta}_i^{(0)} \in \mathbb{R}^d$. $i \in V$, we have

$$\boldsymbol{\theta}_i^{(t+1)} = \sum_{j=1}^N W_{ij}\boldsymbol{\theta}_j^{(t)} - \eta_t \boldsymbol{g}_i^{(t)},\ \forall\ i \in V,\ \forall\ t \geq 0, \tag{6}$$

where $\eta_t > 0$ is the step size and $\boldsymbol{g}_i^{(t)}$ is the local stochastic gradient which is an unbiased estimate of $\nabla J_i(\boldsymbol{\theta}_i^{(t)})$. The algorithm (6) operates through a "consensus-then-optimize" strategy. At each iteration, the worker i first calculates an average of the local models held by the neighboring workers ($\sum_{j=1}^N W_{ij}\boldsymbol{\theta}_j^{(t)}$), then a stochastic gradient step w.r.t. the private function ($-\eta_t \boldsymbol{g}_i^{(t)}$) is performed.

We now consider implementing algorithm (6) for decentralized training of an NN model such as (4). As the latter is parameterized by the weights $\boldsymbol{\theta} \in \mathbb{R}^{mf}$, the workers are required to transmit mf real numbers to the neighbors on the network/graph at each iteration. For $m \gg 1$, this poses a significant challenge due to the limited communication bandwidth. A natural remedy is to compress information before transmitting.

However, directly compressing the local parameters $\boldsymbol{\theta}_i^{(t)}$ in (6) can result in a non-convergent algorithm since the compression operation can lead to unrecoverable information loss. A better idea is to exploit smoothness and focus on compressing the differences in the iterates. In particular, [15] proposed the CHOCO-SGD algorithm for communication efficient decentalized training, as summarized in Algorithm 1.

We observe that step 8 in the algorithm is the only step of CHOCO-SGD involving peer-to-peer communication between the workers. To understand the algorithm better, we observe that when the compression operator is an identity operator, i.e., $\mathcal{Q}(\boldsymbol{\theta}) = \boldsymbol{\theta}$ for any $\boldsymbol{\theta} \in \mathbb{R}^d$, together with the consensus parameter $\gamma = 1$, the CHOCO-SGD algorithm is reduced into:

$$\boldsymbol{\theta}_i^{(t+1)} = \sum_{j=1}^N W_{ij}\{\boldsymbol{\theta}_j^{(t)} - \eta_t \boldsymbol{g}_j^{(t)}\}, \tag{7}$$

which resembles the classical DSGD algorithm with a swapped order for the 'optimize' and 'consensus' steps.

In general, the communication step depends on a compression operator $\mathcal{Q} : \mathbb{R}^d \to \mathbb{R}^d$ which reduces the amount of information transmitted. Furthermore, we are compressing the *difference* between the successive iterates with the local

Algorithm 1 CHOCO-SGD Algorithm [15]

INPUT: initial weights $\{\boldsymbol{\theta}_i^{(0)}\}_{i=1}^N$, max. no. of iterations T, consensus parameter $\gamma \in (0,1)$, step sizes $\{\eta_t\}_{t\geq 0}$.
Set the auxilliary variables $\hat{\boldsymbol{\theta}}_{i,j}^{(0)} = \mathbf{0}$, $j \in \mathcal{N}_i$, $i \in [N]$.
Draw the stopping iteration number $\mathsf{T} \sim \mathcal{U}\{0, ..., T\}$.
for $t = 0, 1, ..., \mathsf{T}$ **do**
  for $i = 1, ..., N$ **do** `//Local SGD step//`
    Compute the local SGD:

$$\boldsymbol{\theta}_i^{(t+\frac{1}{2})} = \boldsymbol{\theta}_i^{(t)} - \eta_t \boldsymbol{g}_i^{(t)},$$

    where $\boldsymbol{g}_i^{(t)}$ is the stochastic estimate of $\nabla J_i(\boldsymbol{\theta}_i^{(t)})$.
  end for
  For each worker $i = 1, ..., N$, broadcast the compressed *difference vector* $\mathcal{Q}(\boldsymbol{\theta}_i^{(t+\frac{1}{2})} - \hat{\boldsymbol{\theta}}_{i,i}^{(t)})$ to the neighbors, where $\mathcal{Q}(\cdot)$ is a compression operator satisfying (8).
  for $i = 1, ..., N$ **do** `//Combination step//`
    Update the auxiliary variable:

$$\hat{\boldsymbol{\theta}}_{i,j}^{(t+1)} = \hat{\boldsymbol{\theta}}_{i,j}^{(t)} + \mathcal{Q}(\boldsymbol{\theta}_j^{(t+\frac{1}{2})} - \hat{\boldsymbol{\theta}}_{j,j}^{(t)}),\ \forall\ j \in \mathcal{N}_i.$$

    Update the local NN weights:

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t+\frac{1}{2})} + \gamma \textstyle\sum_{j\in\mathcal{N}_i} W_{ij}\{\hat{\boldsymbol{\theta}}_{i,j}^{(t+1)} - \hat{\boldsymbol{\theta}}_{i,i}^{(t+1)}\}.$$

  end for
end for
OUTPUT: trained weights $\{\boldsymbol{\theta}_i^{(\mathsf{T})}\}_{i=1}^N$.

stochastic gradient. Common communication compression techniques include gradient quantization and gradient sparsification. Broadly, these communication compression methods can be classified into *biased* and *unbiased* operators. For the CHOCO-SGD algorithm, it is assumed that the compression operator is a random operator satisfying

$$\mathbb{E}_\Omega\left[||\mathcal{Q}(\boldsymbol{\theta};\Omega) - \boldsymbol{\theta}||^2\right] \leq (1-\delta)||\boldsymbol{\theta}||^2,\quad \forall\ \boldsymbol{\theta} \in \mathbb{R}^d, \tag{8}$$

where Ω is the implicit random state of the compression operator, and $\delta \in (0,1]$ is a parameter characterizing the expected error resulted from the compression. Intuitively, with the condition (8), the CHOCO-SGD algorithm behaves similarly as the DSGD algorithm as only the differences between successive iterates are compressed.

Examples of *unbiased* $\mathcal{Q}(\cdot)$ satisfying (8) include [19] which *quantizes* a rescaled vector; [23] which *sparsifies* the vectors by setting a random subset of $d-k$ co-ordinates to zero. Concretely, let $\Omega \subseteq \{1, ..., d\}$, $|\Omega| = k$ be the selected random subset, we set the operator as $\mathsf{rand}_k : \mathbb{R}^d \to \mathbb{R}^d$ with:

$$\left[\mathsf{rand}_k(\boldsymbol{\theta};\Omega)\right]_i = \begin{cases} [\boldsymbol{\theta}]_i & i \in \Omega, \\ 0 & i \notin \Omega, \end{cases} \tag{9}$$

where $[\boldsymbol{\theta}]_i$ denotes the ith element of the vector $\boldsymbol{\theta}$. We observe that $\mathsf{rand}_k(\boldsymbol{\theta};\Omega)$ is a vector with at most k non-zero elements. Notice that for the rand_k compression operator, condition (8) is satisfied with $\delta = \frac{k}{d}$ [15].

Meanwhile, *biased* compression operators $\mathcal{Q}(\cdot)$ are also widely adopted. For example, the sign compression [24] and top_k sparsification which retains the top-k coordinates in the d-dimensional vector with the highest magnitude [23], [25], [26]. Notably, the $_k$ sparsification satisfies (8) with $\delta = \frac{k}{d}$ since the latter compressor always yield an error lower than that of the rand_k sparsifier.

A. Convergence Guarantee and Its Pitfalls

We discuss the convergence guarantees of the CHOCO-SGD algorithm and highlights on its pitfalls in decentralized training of overparameterized NNs.

We describe the assumptions used. First, we assume that each private function is smooth, i.e., the gradient map is Lipschitz continuous:

Assumption 1 *For any $i \in [N]$, there exists $L \geq 0$ such that*

$$\|\nabla J_i(\boldsymbol{\theta}) - \nabla J_i(\boldsymbol{\theta}')\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|,\ \forall\ \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d. \tag{10}$$

Next, we specify conditions on the stochastic gradient estimates. Denote $\mathcal{F}_t$ as the filtration generated by the random variables $\{\boldsymbol{\theta}_i^{(\tau)} : i \in V, 0 \leq \tau \leq t\}$. We assume that:

Assumption 2 *There exists $\sigma, G \geq 0$ such that for any $i \in [N]$, $t \geq 0$, the stochastic gradient $\boldsymbol{g}_i^{(t)}$ satisfies*

$$\begin{aligned} &\mathbb{E}[\boldsymbol{g}_i^{(t)}|\mathcal{F}_t] = \nabla J_i(\boldsymbol{\theta}_i^{(t)}),\quad \mathbb{E}[\|\boldsymbol{g}_i^{(t)}\|^2|\mathcal{F}_t] \leq G^2, \\ &\mathbb{E}[\|\boldsymbol{g}_i^{(t)} - \nabla J_i(\boldsymbol{\theta}_i^{(t)})\|^2|\mathcal{F}_t] \leq \sigma^2. \end{aligned} \tag{11}$$

Notice that the first condition is satisfied by a uniform sampler for the (mini-batch) stochastic gradient, e.g., when $\boldsymbol{g}_i^{(t)} = \nabla_{\boldsymbol{\theta}}\mathsf{loss}(f(\boldsymbol{x}_{j_t};\boldsymbol{\theta}_i^{(t)}); y_{j_t})$ such that j_t is selected uniformly at random from $\{1, ..., |\mathcal{D}_i|\}$. We have also assumed that the stochastic gradients have bounded second order moments.

We observe the following result that is borrowed from [16, Theorem 4.1] on the CHOCO-SGD algorithm:

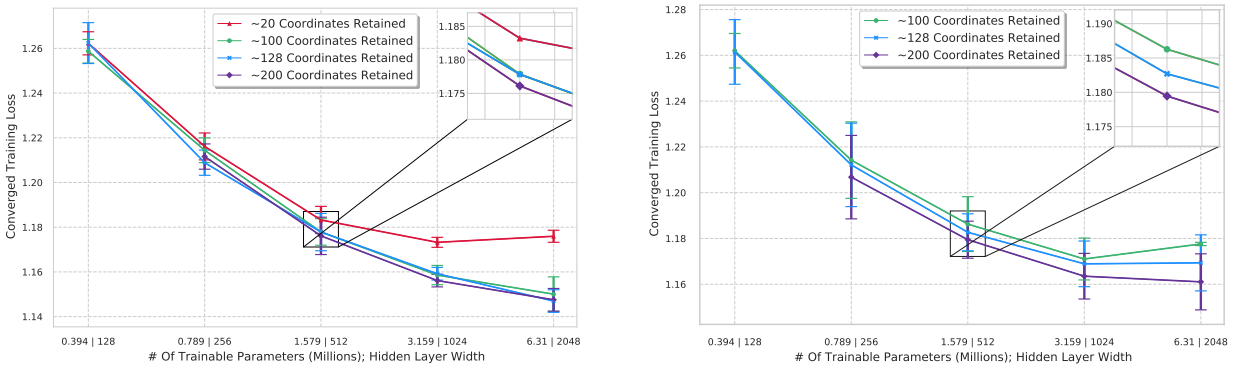
Theorem 1 *Under Assumptions 1, 2 and suppose that the compressor satisfies* (8)*. There exists $\eta, \gamma > 0$ such that if we consider a constant step size with $\eta_t \equiv \eta$, then for any $T \geq 1$, the output generated by Algorithm 1 satisfy:*

$$\mathbb{E}[\|\nabla J(\overline{\boldsymbol{\theta}}^{(\mathsf{T})})\|^2] = \mathcal{O}\left(\sqrt{\frac{L\sigma^2 J_0}{NT}} + \left(\frac{LGJ_0}{\rho^2\delta T}\right)^{\frac{2}{3}}\right),$$

where the expectation is taken over T and the stochastic quantities in the algorithm, δ was defined in (8)*, $\rho \in (0,1]$ is the spectral gap of $\boldsymbol{W}$ defined in* (5)*, $\overline{\boldsymbol{\theta}}^{(\mathsf{t})} = \sum_{i=1}^N \boldsymbol{\theta}_i^{(\mathsf{t})}/N$ is the network average iterate, and $J_0 = J(\overline{\boldsymbol{\theta}}^{(0)}) - \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.*

The theorem suggests that the CHOCO-SGD algorithm finds an $\mathcal{O}(1/\sqrt{T})$-stationary solution to (1) in at most T iterations as we note that $\mathsf{T} \leq T$.

Convergence for Overparameterized Models. We concentrate on the performance of CHOCO-SGD when $d \gg 1$, for example, when training an overparameterized NN such as (4)



(a) top_k sparsification. (b) rand_k sparsification

Fig. 1. Converged training losses for large-width NNs compressed with biased top_k 1(a) and unbiased rand_k 1(b) sparsification on CIFAR-10. Reported accuracies averaged over 5 independent trials with error bars indicating 99% confidence interval over the 5 trials. For $k = 20$ with rand_k sparsification, divergence is observed in the case of 2048-layer and 1024-layer NNs on all 5 trials.

with $m \gg 1$ neurons. Furthermore, to control the bandwidth usage, we choose the rand_k sparsifier or top_k sparsifier as the compressor. Now, we fix the number of iterations as T, and the number of coordinates sent per iteration at k, i.e., we fix the amount of data transmitted in the CHOCO-SGD algorithm. In this setting, we have

$$\mathbb{E}[\|\nabla J(\overline{\boldsymbol{\theta}}^{(\mathsf{T})})\|^2] = \mathcal{O}\Big(\sqrt{\frac{L\sigma^2 J_0}{NT}} + d^{\frac{2}{3}}\Big(\frac{LGJ_0}{\rho^2 kT}\Big)^{\frac{2}{3}}\Big). \quad (12)$$

Furthermore, applying Theorem 1 shows that to reach an ϵ-stationary solution (i.e., $\mathbb{E}[\|\nabla J(\overline{\boldsymbol{\theta}}^{(\mathsf{T})})\|^2] \leq \epsilon$), the number of CHOCO-SGD iterations required grows in the order:

$$T = \Omega\left(LJ_0 \cdot \max\left\{\frac{\sigma^2}{N\epsilon^2}, \frac{d}{k}\frac{G}{\rho^2\epsilon^{1.5}}\right\}\right). \quad (13)$$

As the amount of data transmitted per iteration is constant (i.e., k real numbers), the above calculation indicates that *the CHOCO-SGD algorithm may require a higher communication complexity as the NN model becomes inncreasingly overparameterized (i.e., when $d \gg 1$)*, in order to maintain the same performance level, despite the compression being applied at each iteration. In fact, for any $1 \leq k \leq n$, it is predicted from (13) that the number of real numbers transmitted is $\Omega(\max\{k\sigma^2/(N\epsilon^2), dG/(\rho^2\epsilon^{3/2})\})$ when the top_k or rand_k sparsifier is used as the compressor in CHOCO-SGD. We notice that similar dependence on the problem dimension d is also observed in other compressed DSGD methods, e.g., [17]–[19].

The observations in (12), (13) indicate a pitfall in the existing theories when applying compressed DSGD methods such as CHOCO-SGD to overparameterized NNs. Particularly, it suggests that although the compression scheme can reduce the communication cost *per iteration*, the number of iterations required would be increased. In the next section, we conduct an extensive set of numerical experiments to test the above observation. Interestingly, we show that in most cases, *increasing the degree of overparameterization can lead to a better performance for the trained NN without increasing the communication cost during training*, which is in contrast to the said observation. We conjecture that such phenomena is a consequence of an artifact in the existing analysis and discuss the possible fix for the observation.

III. Empirical Studies

In this section, we perform numerical experiments to examine the performance of compressed DSGD method when applied to training overparameterized NNs. We concentrate on the effects of the number of parameters on the communication complexity using the CHOCO-SGD method [16] [cf. Algorithm 1]. For simplicity, we consider a two-layer NN with ReLU activation described in (4) and adjust the width, m, of the NN.

We consider the task of training a classifier with the CIFAR-10 dataset [27] that contains 50K (resp. 10K) training (resp. test) samples. Each sample consists of a 32×32 RGB image which can be represented as a 3072-dimensional vector, and is associated with a label selected from 10 image classes. To simulate the decentralized training environment, samples from the 10 image classes are uniformly split among N workers and shuffled at every epoch – as in [16], [28]. To establish a challenging generalization task, we test the trained models on CIFAR-10.1 [29].

For the CHOCO-SGD method, we use a minibatch size of $\xi = 128$ for every iteration. Our chosen mode of communication compression is top_k and random_k with a fixed number of co-ordinates k allowed to be communicated between workers. We choose a constant consensus parameter $\gamma = 0.0375$ in Algorithm 1 and an SGD stepsize $\eta = 0.1$ which is decreased by a factor of 10 on epochs $100, 150, 200$. Our top_k and random_k simulations are run on $N = 8$ nodes of a ring topology. The decentralized training environment is simulated on an MPI-based [30] network where we assign an independent CPU process to each worker.

Our first empirical example aims at comparing the quality of solution found by CHOCO-SGD after performing 300 epochs of iterations against the number of parameters of the trained NN (d). Notice that as k is fixed in the compressor design, the

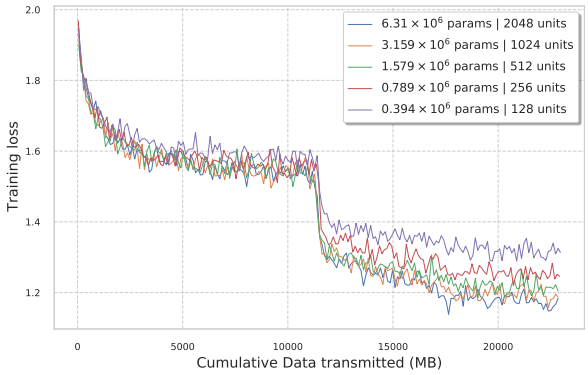


Fig. 2. Training loss with cumulative communication cost in (MB) for large-width NNs of varying widths constrained with a constant sparsification coordinate $k = 100$. While *convergence rate* of all models is invariant to layer width, overparameterized models ($d \geq 2 \times 10^6$ parameters) converge to a solution of lower training loss with identical cumulative data usage.

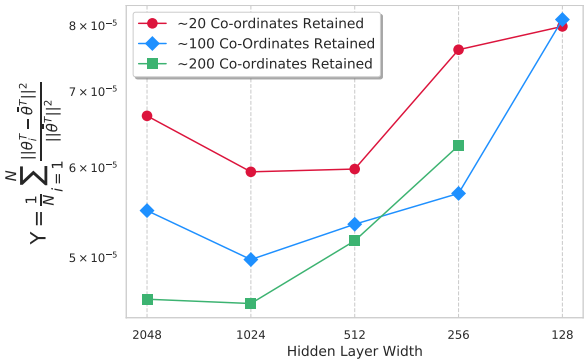


Fig. 3. Converged, normalized consensus distance between $N = 8$ workers for different NN layer widths. Overparameterized models enjoy significantly greater consensus among workers with only marginal dependence on sparsification co-ordinate bandwidth for larger models.

Layer Width	Normalized Consensus Distance [cf. (14)]		
	Epoch = 200	Epoch = 100	Epoch = 50
2048	5.499×10^{-5}	9.8206×10^{-3}	1.3977×10^{-2}
1024	4.980×10^{-5}	1.0346×10^{-2}	1.5307×10^{-2}
512	5.349×10^{-5}	1.0026×10^{-3}	1.3478×10^{-2}
256	5.694×10^{-5}	8.7639×10^{-3}	1.2423×10^{-2}
128	8.098×10^{-5}	7.3181×10^{-3}	9.2698×10^{-3}

Table II. **Converged consensus distances at intermediate training epoch numbers.** Overparameterized models converge to better-consensus solutions at a slower rate compared to low-width NNs

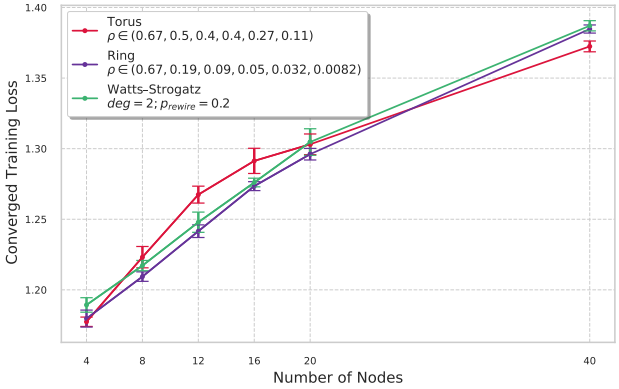


Fig. 4. Converged training loss versus number of workers with different graph topologies. Error bars indicate 95% confidence interval over 5 independent trials.

communication complexity (i.e., number of bits transmitted over the network) is fixed. Our results are presented in Fig. 1(a) and 1(b). From the figures, we observe that the training loss *decreases* with d, especially when the top_k sparsifier is used. Notice that the above is actually *in contrast* to the observations made in (12) based on Theorem 1, which predicts that the solution quality decreases as the problem dimension increases. In Table I we compare the *converged* test accuracy evaluated on CIFAR10.1 with different widths for the NN, where similar observations are obtained as in Fig. 1. Lastly, we plot the trajectory of the training loss against the cumulative communication cost (in MB) in Fig. 2. Again, we observe that increasing model dimension (overparameterization) leads to faster convergence with respect to the communication cost.

Our second empirical example examines the consensus error in the converged solution after 300 epochs of CHOCO-SGD with $N = 8$ workers. We aim at studying the consensus error of the converged solutions and the effects of problem dimension. Here, the normalized consensus error is defined as:

$$\Upsilon = \frac{1}{N}\sum_{i=1}^{N}\frac{||\boldsymbol{\theta}_i^T - \overline{\boldsymbol{\theta}}^T||^2}{||\overline{\boldsymbol{\theta}}^T||^2}, \tag{14}$$

where $\boldsymbol{\theta}_i^T$ denotes the CHOCO-SGD solution after 300 epochs. The above metric is compared against the width of the hidden layer, m, in the NN in Fig. 3. Naturally, we observe that the consensus error increases when the number of coordinates kept in the sparsifier k decreases. Moreover, an intriguing observation is that the consensus error *increases* when the width of the NN *increases* from $m = 1024$ to $m = 2048$. We recall from Fig. 1, Table I that the training loss/testing accuracy actually *decreases* with this change in the number of parameters. This suggests that a consensual solution may not be necessary for achieving a lower training loss in the overparameterized setting.

Our third empirical example examines the effects of graph topology on the converged training loss after 300 epochs of CHOCO-SGD iterations. In this experiment, we fix the number of hidden neurons at $m = 256$ and use a top_k sparsifier with $k = 51$. We experiment with common graph topologies such as ring ($deg_{ring} = 2$), torus ($deg_{torus} = 4$), and Watts–Strogatz (small-world) graphs [31], and for different numbers of connected neighbors. Notice that the spectral gap parameter ρ decreases with the number of workers N, especially for the ring topology. The result is presented against the number of workers in Fig. 4. As observed in the figure, for all topology settings, the training loss increases with the number of workers N. The deterioration in performance is the most severe with the ring topology. Unlike the previous example, we notice that the observed behavior is consistent with the theory in Theorem 1,

# Neurons \ # Coordinates k	top_k sparsifier				rand_k sparsifier		
	20	100	128	200	100	128	200
$m = 2048$	46.481 ± 0.767	46.837 ± 0.354	46.663 ± 1.544	46.711 ± 0.755	46.919 ± 1.401	43.288 ± 0.769	46.365 ± 1.1
$m = 1024$	46.776 ± 0.423	46.184 ± 0.937	46.688 ± 1.05	47.156 ± 0.999	48.245 ± 0.855	47.135 ± 1.103	47.309 ± 0.984
$m = 512$	45.651 ± 2.219	46.266 ± 0.902	45.787 ± 1.111	46.292 ± 0.584	46.497 ± 0.653	46.47 ± 0.401	45.505 ± 0.81
$n = 256$	45.286 ± 0.413	45.148 ± 1.103	45.051 ± 0.802	45.123 ± 0.855	45.707 ± 0.751	44.928 ± 0.846	44.733 ± 0.616
$m = 128$	44.59 ± 0.984	43.967 ± 0.851	42.778 ± 0.45	N/A	43.288 ± 1.435	43.568 ± 1.282	N/A
comm. cost per iteration (MB)	0.482	2.410	3.012	4.820	2.410	3.012	4.820

Table I. **Generalization performance of CHOCO-SGD to CIFAR-10.1 with overparameterized models**. Contrary to convergence performance, large-width NNs trained with top_k and rand_k sparsification exhibit marginal dependance to model dimension at test-time (average accuracy disparity of 2.39% between 2048 and 128 unit models). Test results averaged over 5 independent trials with altered seeds on a ring-topology communication graph with 8 nodes. CHOCO-SGD diverges when parameters are compressed with rand_k sparsification of $k = 20$ co-ordinates.

which predicts a slower convergence when the graph topology has a smaller ρ.

Lastly, we study a case of decentralized training with heterogeneous data. Particularly, we allocate samples from a single class to each of the $N = 8$ workers. We evaluate the training loss with *global data* using the local model $\boldsymbol{\theta}_i^T$ obtained by CHOCO-SGD with top_k sparsifier after 300 epochs; compared to the averaged model $(1/N)\sum_{i=1}^N \boldsymbol{\theta}_i^T$. The simulation result is shown in Fig. 5. We observe that a similar trend as Fig. 1 holds as the training loss decreases with the hidden layer width m given that the algorithms are run with a similar communication budget. Furthermore, we note that the training loss is higher with the local model.

Discussions. The above numerical examples demonstrate a consistent discrepancy between existing theories on compressed DSGD methods and their practical performances when training overparameterized NN models. In particular, we show that with the same communication cost allowed, the performance of the CHOCO-SGD trained NN improves with the number of neurons m employed in the NN, contrary to (12).

The careful readers may notice that the poor dependence on dimension d in the convergence bounds (12) can be seen as a direct consequence of the worst-case analysis of the compression error in (8). This is because one is forced to take $\delta = d/k$ to account for the condition that holds for all $\boldsymbol{x} \in \mathbb{R}^d$ in case of the top_k/rand_k sparsifier. Furthermore, we remark that the existing theories are developed for general distributed optimization problems, assuming only high level properties such as smoothness of the objective function.

On the other hand, recent works on overparameterized models [20]–[22] have shown that the training of such models with SGD shall be compared to the learning of a kernel model in the RKHS. Among others, a key innovation therein is that the optimality gap is measured in terms of the distance in the (infinite-dimensional) function space, where the rate of convergence is shown to be independent of m even as $m \to \infty$. We believe that extending these results to a decentralized setting such as the CHOCO-SGD method can help break the curse of dimensionality in the analysis. We remark that an interesting observation was made in [32] on the stability of overparameterized NN models with dropout, an operation that is akin to applying the sparsifier.

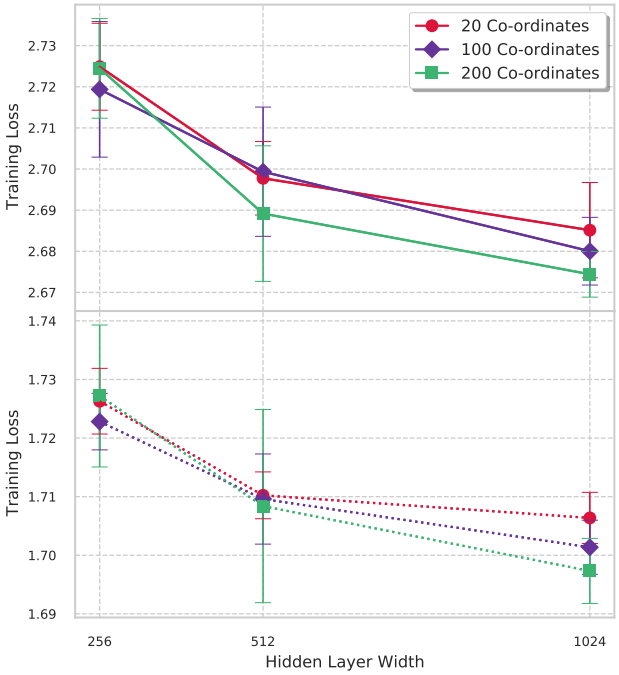


Fig. 5. Converged training loss versus model dimensionality for *heterogeneously* distributed data among $N = 8$ workers on a ring topology where each worker's private distribution contains labels from a single class. Top row (solid line) indicates average performance of the *local model* on the complete training dataset. Bottom row (dotted line) indicates performance of *averaged model* ($\bar{\theta}^*$) on the complete training dataset. Error bars indicate 95% confidence interval over 5 independent trials.

IV. Conclusions

This paper provides the first empirical study on the performance of compressed DSGD methods on *overparameterized NN models*. On the positive side, our result shows that utilizing overparameterized NN models in a decentralized learning is both practical and beneficial, contradicting existing theories that suggest otherwise. Furthermore, we identify a gap in the existing theories that attempt to analyze compressed DSGD methods. We believe that our article will serve as a motivating study to develop improved communication efficient algorithms

for decentralized training of NNs.

ACKNOWLEDGEMENTS

This work is supported by CUHK Direct Grant #4055113.

REFERENCES

[1] A. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[2] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 629–681, 2017.

[3] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[4] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[5] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE transactions on automatic control*, vol. 58, no. 2, pp. 391–405, 2012.

[6] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[7] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1529–1538.

[8] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *arXiv preprint arXiv:1705.09056*, 2017.

[9] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on signal processing*, vol. 66, no. 11, pp. 2834–2848, 2018.

[10] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: decentralized training over decentralized data," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4848–4856.

[11] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9217–9228.

[12] R. Xin, U. A. Khan, and S. Kar, "An improved convergence analysis for decentralized online stochastic non-convex optimization," *IEEE Transactions on Signal Processing*, vol. 69, pp. 1842–1858, 2021.

[13] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 26–38, 2020.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3478–3487.

[16] A. Koloskova*, T. Lin*, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *International Conference on Learning Representations*, 2020.

[17] D. Kovalev, A. Koloskova, M. Jaggi, P. Richtarik, and S. Stich, "A linearly convergent algorithm for decentralized optimization: Sending less bits for free!" in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 4087–4095.

[18] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7652–7662, 2018.

[19] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[20] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *arXiv preprint arXiv:1806.07572*, 2018.

[21] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 322–332.

[22] A. Bakshi, R. Jayaram, and D. P. Woodruff, "Learning two layer rectified neural networks in polynomial time," in *Conference on Learning Theory*. PMLR, 2019, pp. 195–268.

[23] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[24] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 560–569.

[25] S. Shi, X. Chu, K. C. Cheung, and S. See, "Understanding top-k sparsification in distributed deep learning," *arXiv preprint arXiv:1911.08772*, 2019.

[26] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[27] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[28] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[29] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do cifar-10 classifiers generalize to cifar-10?" *arXiv preprint arXiv:1806.00451*, 2018.

[30] R. L. Graham, T. S. Woodall, and J. M. Squyres, "Open mpi: A flexible high performance mpi," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2005, pp. 228–239.

[31] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[32] A. Shevchenko and M. Mondelli, "Landscape connectivity and dropout stability of sgd solutions for over-parameterized neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8773–8784.