Understanding Generalization in Distributed Machine Learning

April 13, 2022

Arjun Ashok Rao ARJUNRAO@LINK.CUHK.EDU.HK

Advisor: Hoi-To Wai HTWAI@SE.CUHK.EDU.HK

Final Report, FTEC4998 – Final Year Thesis

Abstract

This work studies the decentralized consensus optimization problem from the perspective of understanding the relation between consensus constraints between individual workers, and the generalization ability of the proposed algorithm. We first conduct an independent study on the efficiency of compressed Decentralized Stochastic Gradient Algorithms (DSGD) with overparameterized models, and note the weak dependence on worker consensus with deep neural networks (DNNs). Inspired by recent work proposing stochastic optimization algorithms for multi-agent systems over Reproducing Kernel Hilbert Spaces (RKHS), we propose a novel functional-consensus-based decentralized stochastic gradient descent algorithm. Advantages and drawbacks of the functional consensus algorithm is highlighted, and a secondary inexact-consensus based on computed lottery-tickets is proposed. Finally, this project turns to the problem of inexact consensus learning on time-varying graphs.

1. Introduction

The rapid increase in the size of real-world datasets make it infeasible to store large training data on a centralized server. Learning algorithms must thus learn to leverage data distributed over multiple workers. Learning from distributed data also lends valuable benefits to data privacy, ownership, and efficiency. There are several subfields that propose learning algorithms over distributed data

- Distributed learning: In distributed learning, workers communicate with each other, and a centralized parameter server to learn a shared model θ^*. This parameter-server architecture aids to simplicity for analysis. Figure of a typical distributed learning architecture in shown in 1

- Federated learning [Wang et al., 2021] – Federated learning is a subset of the distributed learning methodology, that has found practical use due to extensive work done in combating heterogeneous data, heterogeneous clients [Diao et al., 2020], and privacy-preserving machine learning.

- Decentralized Learning: In decentralized learning, workers cannot communicate with all other workers; Instead, they are only allowed to communicate with their immediate neighbors while arriving to a consensus about each worker's underlying parameter distribution θ_i^t [Blondel et al., 2005, Nedic and Ozdaglar, 2009].

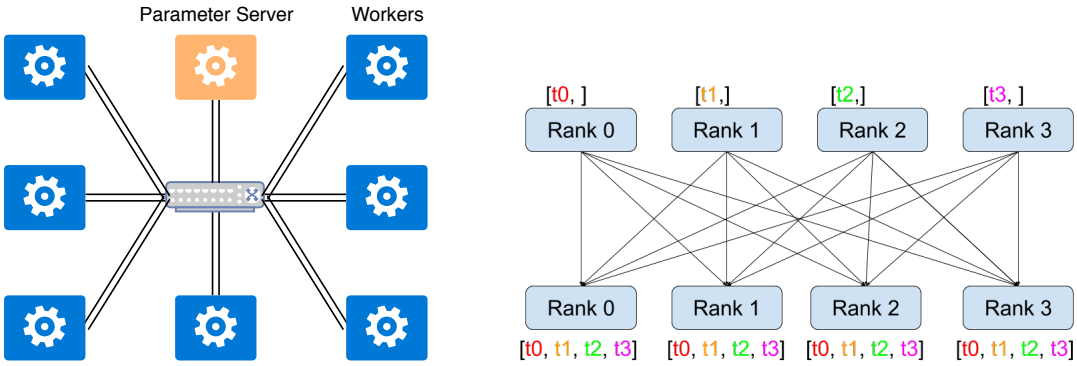

Figure 1: (Left) Standard Parameter-Server architecture for distributed learning. In distributed learning, the server is tasked with aggregating optimization variables from each worker. Usually, at each iteration, after performing local updates, worker i sends its optimization variable w_i^n to the server, who aggregated $\sum_i w_i^t$ over all workers, and broadcasts the updated w_i^{t+1} back to each worker. (Right) A Ring-*AllReduce* algorithm to combat the expensive communication caused by sending variables to the server. In Ring-*AllReduce*, each process first reduces, and then sends the reduced variable to the next worker.

2. Background

2.1 Decentralized Optimization

This work concerns itself with the final method – *decentralized learning*. 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.

In decentralized learning, we are interested in the following finite-sum optimization problem of a d-dimensional variable $\boldsymbol{\theta}$:

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

and $J_i : \mathbb{R}^d \to \mathbb{R}$ is a continuous, differential private objective function of worker i. 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:

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

where $\boldsymbol{\theta}_i \in \mathbb{R}^d$ is a private/local variable held by the ith worker. 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 $m \to \infty$.

2.2 Consensus in Decentralized Learning

An important consideration is to note the heavy dependence of 2 on the consensus constraint $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j \ \forall \ (i,j) \in E$. In practice, the consensus constraint is reformulated as an average consensus among N workers' local vectors $\boldsymbol{\theta}_i$:

$$\bar{\boldsymbol{\theta}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\theta}_i \tag{5}$$

A standard method to compute this aggregate in a distributed environment is through *gossip* algorithms that communicate with their immediate neighbors. For a parameter distribution $\boldsymbol{\theta}_i^t$ on node i for iteration t, we can compute $\boldsymbol{\theta}_i^{t+1}$ by:

$$\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^t - \eta \sum_{j=1}^{n} w_{ij} \delta_{ij} \quad \forall j \in V, \ \forall t \geq 0$$

Where we compute δ_{ij} as a difference $\boldsymbol{\theta}_i^t - \boldsymbol{\theta}_j^t$ for gossip with exact communication (vanilla gossip). In matrix form, we may write gossip averaging for a parameter vector across all

agents $\boldsymbol{\theta}_N^t = \begin{bmatrix} \boldsymbol{\theta}_1^t & \boldsymbol{\theta}_2^t & \dots & \boldsymbol{\theta}_N^t \end{bmatrix}$ as:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \boldsymbol{\theta}^t (\boldsymbol{W} - I)$$

Where $W \in \mathbb{R}_+^{N \times N}$ is a 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

$$\|\boldsymbol{W} - \mathbf{1}\mathbf{1}^\top / N\| \leq 1 - \rho, \tag{6}$$

where $\rho \in (0, 1]$ is the spectral gap, and $\beta = max\{|\lambda_2(\boldsymbol{W})|, |\lambda_n(\boldsymbol{W})|\}$ be the second-largest eigenvalue of $\boldsymbol{W}$. Notice that such matrix exists for any connected graph G. For a constant stepsize $\eta \in (0, 1]$, vanilla gossip converges linearly to the averaged iterate $\bar{\boldsymbol{\theta}}$ in linear time. While previous analysis depended on G being a connected graph, [Boyd et al., 2006] show that averaging time of the gossip algorithm over a graph of arbitrary (and possibly sparse) topology depends on β of the mixing matrix $\boldsymbol{W}$.

2.3 Decentralized Gradient Descent (DGD)

In a distributed multi-agent setting, where workers aim to minimize a local objective functions, algorithms follow a "consensus + optimize" strategy, where local updates are performed on each worker, and updated parameter vectors are communicated to immediate neighbors using a gossip update mechanism. Under the assumption that each private function $J_i(\boldsymbol{\theta})$ is smooth, i.e., the gradient map is Lipschitz continuous, DGD algorithm, which follows the "consensus + optimize strategy" converges in linear time:

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

$$\|\nabla J_i(\boldsymbol{\theta}) - \nabla J_i(\boldsymbol{\theta}')\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \ \forall \ \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d. \tag{7}$$

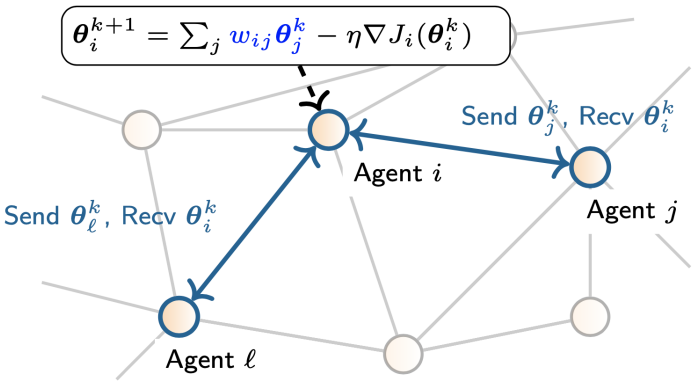

Figure 2: Decentralized Gradient Descent over a network on N workers. At each node, worker i computes a local gradient on its local variable $\boldsymbol{\theta}_i^t$, and broadcasts it to all neighbors $j \in V, w_{ij} \geq 0$. Gossip averaging with neighbors' parameters is done with a gossip step.

Thus, at each iteration k, and agent i, the following consensus and optimize strategy can take place asynchronously:

$$\boldsymbol{\theta}_i^{t+1} = \sum_{j=1}^{N} w_{ij}\boldsymbol{\theta}_j^t - \eta\nabla J_i(\boldsymbol{\theta}_i^t) \tag{8}$$

Note that 8 can be written as a special case of the standard vanilla gossip. [Lian et al., 2017] propose a stochastic alternative to 8 with data-point or minibatch ξ_i on worker i. We are then interested in determining:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad J(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\xi_i \sim \mathcal{D}} J_i(\boldsymbol{\theta}; \xi) \tag{9}$$

Where $\mathcal{D} = \cup_{i=1}^{M} \mathcal{D}_i$ is the private data distribution available to worker i. In the stochastic setting, we require a bounded svariance on the stochastic gradient, and unbiased estimate of the stochastic gradient:

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

$$\begin{aligned} &\mathbb{E}[\boldsymbol{g}_i^{(t)}|\mathcal{F}_t] = \nabla J_i(\boldsymbol{\theta}_i^{(t)}), \quad \mathbb{E}[\|\boldsymbol{g}_i^{(t)}\|^2|\mathcal{F}_t] \leq G^2, \\ &\mathbb{E}[\|\boldsymbol{g}_i^{(t)} - \nabla J_i(\boldsymbol{\theta}_i^{(t)})\|^2|\mathcal{F}_t] \leq \sigma^2. \\ &\mathbb{E}[\|\nabla J_i(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\|^2] \leq \tau^2 \end{aligned} \tag{10}$$

In DPSGD, each worker communicates $\mathcal{O}(deg(G))$ models per iteration. [Lian et al., 2018] propose an asynchronous version of 1 where update and communication are done in parallel; leading to a linear speedup with number of workers N. With assumptions 2, and 1, for a constant total number of iterations $T \geq 0$, minibatch size $M \geq 0$, and total number of workers N, we have:

$$\frac{\sum_{i=0}^{N} \mathbb{E}\|\nabla J\left(\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\theta}_i^t\right)\|^2}{T} \leq \frac{2(\mathbb{E}[J(\boldsymbol{\theta}_i^0) - [J(\boldsymbol{\theta}_i^T) \cdot N])}{\gamma T M} + \frac{2\gamma L(\sigma^2 + 6M\tau^2)}{N} \tag{11}$$

Furthermore, [Lian et al., 2018] remark that when setting stepsize $\gamma = \frac{N}{10ML + \sqrt{\sigma^2 + 6M\tau^2\sqrt{TM}}}$, and as $T \to \infty$ we recover a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$.

Algorithm 1 Decentralized Parallel Stochastic Gradient Descent (DPSGD)

Require: SGD step-size $\eta > 0$, mixing matrix $\boldsymbol{W}$, weights $\{\boldsymbol{\theta}_i^0\}_{i=0}^N$, consensus stepsize $\gamma \in (0, 1)$
  for $t = 1, 2, \ldots, T$ **do**
    for worker $i = 1, \ldots, N$ **do**
      Sample minibatch from node i: $\xi_{k,i} \sim \mathcal{D}_i$ and calculate local stochastic gradient $\boldsymbol{g}_i^t = \nabla J(\boldsymbol{\theta}_i^k, \xi_i^k)$
      Receive parameter variables from neighbors $\boldsymbol{\theta}_j^k \ \forall j \in V, \ w_{ij} > 0$
      Compute consensus step: $\boldsymbol{\theta}_i^{k+\frac{1}{2}} = \sum_j w_{ij} \boldsymbol{\theta}_j^k$
      Update the local variable $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^{k+\frac{1}{2}} - \eta \boldsymbol{g}_i^t$
    end for
  end for
  Return Averaged model $\bar{\boldsymbol{\theta}^*} = \sum_N \boldsymbol{\theta}_i^T$

3. Contributions and Related Work

While there has been a large body work in investigating newer decentralized learning algorithms, this study focusses on two constraining requirements:

1. The decentralized learning algorithm must be efficient for overparameterized deep neural network models ($d \gg 1$). Overparameterization is achieved when the number of parameters is significantly higher than the number of training samples (ResNet (s), VGG, AlexNet are all overparameterized models). A majority of these schemes propose a form of communication compression to counteract the increased communication cost with large NN models.

2. This study is focussed on *inexact consensus* – where workers are no longer required to send a parameter vector $\boldsymbol{\theta}_i^t$ at every iteration to each neighbor $j \in V \ \forall w_{ij} > 0$.

The above constraints significantly sparsify the available literature relevant to the problem of interest. Therefore, this section will present related work that satisfy one of our above constraints at a time.

Communication Efficient Deep Learning: The past few years have witnessed a significant increase in interest surrounding multi-agent stochastic nonconvex optimization methods, particularly with neural networks. The rise of deep neural networks have inevitably led to an increased communication bandwidth between workers, expecially for overparameterized models that find themselves in use for practical applications [Brown et al., 2020]. Previous works have developed communication-efficient alternatives to standard DPSGD algorithms. [Koloskova et al., 2019, Koloskova* et al., 2020] proposed CHOCO-SGD which compresses incremental vectors at each worker prior to communication. [Tang et al., 2019] proposed an error-compensation strategy to DPSGD with significant communication cost reduction without affecting convergence. [Seide et al., 2014] quantize gradients to one-bit per value, and propose an error compensation algorithm that corrects quantization error from the previous iteration. Finally, [Alistarh et al., 2018, Shi et al., 2019] attempt to provide an

analysis of DPSGD using the top_k sparsifier.

Inexact Consensus: [Kong et al., 2021] attempt to explain the conensus difference between individual workers as a key factor influencing generalization gap between centralized and decentralized training. This work proposes a critical consensus distance - below which, decentralized training can converge as quickly as its centralized counterpart with a larger stepsize. [Xiang et al., 2020] propose a privacy preserving *private consensus scheme*. [Koppel et al., 2019, Koppel et al., 2018] use a penalized variant of functional SGD to enforce a consensus constraint – a concept used in the proposed algorithm of this study.

Primary Contributions:

1. We highlight a pitfall in existing theory regarding the performance of convergence and consensus of DSGD algorithms with overparameterized models. We empirically show the consensus properties of overparameterized models – overparameterized models enjoy higher consensus. However, NN models with wider hidden layers are slower to reach consensus (more iterations required) compared to smaller models.

2. We study work on decentralized learning with regressors from RKHS primarily used for strongly convex problems, and make a preliminary extension in inexact consensus learning. We propose Inex-SGD, a decentralized learning algorithms for DNN models that leverage model output as a consensus metric as opposed to the parameter matrix.

3. We simulate a series of numerical experiments with inex-SGD, making comparisons to CHOCO-SGD and standard decentralized parallel stochastic gradient descent algorithm (DPSGD). We find that inex-SGD, while not converging as efficiently as CHOCO-SGD with a top-k sparsifier, still converges to a valid solution with no consensus involved, and at a fraction of the communication cost.

4. Motivated by recent theoretical advances to the Lottery Ticket Hypothesis [Frankle and Carbin, 2018]: an algorithms to iteratively find sparse subnetworks from their overparameterized counterparts, we propose an extension to inex-SGD that bases consensus on pruned neural network models. Simulation results indicate that performing DPSGD on pruned networks can replicate performance and communication cost of CHOCO-SGD.

5. Finally, the project attempts to pivot into a critical application of inexact consensus algorithms: decentralized learning on time-varying graphs. We provide a brief literature review of traditional algorithms built to learn on time-varying graphs, and offer a forecast on inexact consensus algorithms for time-varying graphs.

4. Decentralized Learning in the Overparameterized Regime

We first conduct an empirical study on CHOCO-SGD (Algorithm 2), highlighting a pitfall in existing theories surrounding overparameterization and communication cost for decentralized learning. CHOCO-SGD relies on compressing the difference in iterated before communication. More formally, 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 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

$$\mathbb{E}_\Omega \left[\|\mathcal{Q}(\boldsymbol{\theta}; \Omega) - \boldsymbol{\theta}\|^2 \right] \leq (1 - \delta)\|\boldsymbol{\theta}\|^2, \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^d, \tag{12}$$

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

Algorithm 2 CHOCO-SGD Algorithm [Koloskova et al., 2019]

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 (12).
    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 $\{\boldsymbol{\theta}_i^{(\mathsf{T})}\}_{i=1}^N$.

the condition (12), the CHOCO-SGD algorithm behaves similarly as the DSGD algorithm as only the differences between successive iterates are compressed.

Under Assumptions 1, 2, and 12, We observe the following result that is borrowed from [Koloskova* et al., 2020, Theorem 4.1] on the CHOCO-SGD algorithm:

Theorem 1 *Under Assumptions 1, 2 and suppose that the compressor satisfies* (12)*. 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 2 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* (12)*,* $\rho \in (0, 1]$ *is the spectral gap of* $\boldsymbol{W}$ *defined in* (6)*,* $\overline{\boldsymbol{\theta}}^{(\mathsf{t})} = \sum_{i=1}^{N} \boldsymbol{\theta}_i^{(\mathsf{t})}/N$ *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 $\mathsf{T} \leq T$.

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

$$\mathbb{E}[\|\nabla J(\overline{\boldsymbol{\theta}}^{(\mathsf{T})})\|^2] = \mathcal{O}\Big(\sqrt{\frac{L\sigma^2 J_0}{NT}} + \boldsymbol{d}^{\frac{2}{3}}\Big(\frac{LGJ_0}{\rho^2 kT}\Big)^{\frac{2}{3}}\Big). \tag{13}$$

Furthermore, 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:

$$T = \Omega\left(LJ_0 \cdot \max\left\{\frac{\sigma^2}{N\epsilon^2}\,,\, \frac{d}{k}\frac{G}{\rho^2\epsilon^{1.5}}\right\}\right). \tag{14}$$

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 (14) 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., [Kovalev et al., 2021, Tang et al., 2018, Alistarh et al., 2017].

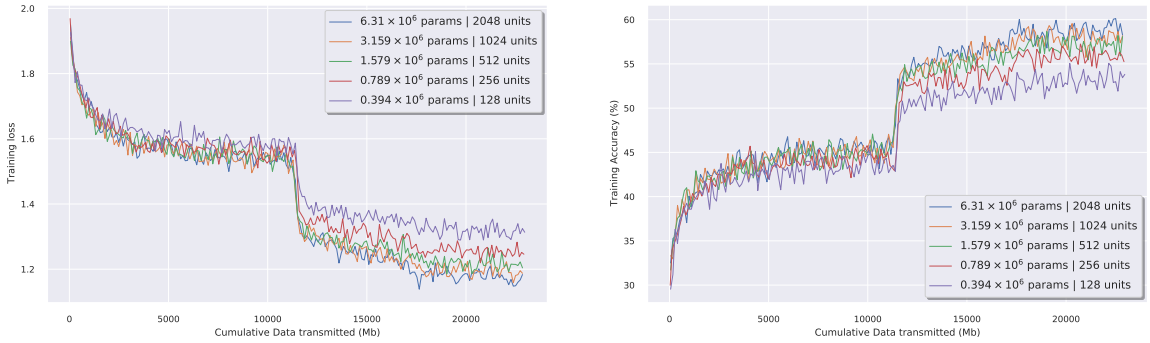

Figure 3: Training loss with cumulative communication cost in (MB) for large-width NNs of varying widths constrained with a constant sparsification coordinate $k = 100$. While *convergence rate* of all models is invariant to layer width, overparameterized models ($d \geq 2 \times 10^6$ parameters) converge to a solution of lower training loss with identical cumulative data usage.

4.1 Empirical Study

Theorem 13, motivated an empirical study on the performance and consensus of compressed DSGD algorithms with overparameterized models. For simplicity, we consider a two-layer NN with ReLU activation described in (4) and adjust the width, m, of the NN. A full experimental setup can be found in A, and results concerned with impact of dimensionality on performance of convergence can be found in [Rao and Wai, 2021]. This section focusses on rather interesting results that show weak dependence of *consensus error* on the size of the NN model used. More formally, we define the normalized consensus error as:

$$\Upsilon = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\boldsymbol{\theta}_i^T - \overline{\boldsymbol{\theta}}^T\|^2}{\|\overline{\boldsymbol{\theta}}^T\|^2}, \tag{15}$$

[Kong et al., 2021] propose a bound on the consensus distance – If the following bound is satisfied,

$$\Upsilon_t^2 \leq \left(\frac{1}{Ln}\gamma\sigma^2 + \frac{1}{8L^2}\|\nabla J(\bar{\boldsymbol{\theta}}^T)\|^2\right)$$

we can recover centralized SGD's convergence rate with a larger stepsize $\gamma \leq \gamma_{max}$. Interestingly, figure 9 shows that overparameterized models enjoy marginally higher consensus

compared to their underparameterized counterparts, and figure 5 shows that both overparameterized and underparameterized NNs converge with the same cumulative data usage.

However, note that Table 1 shows that overparameterized models reach consensus with significantly larger number of iterations. Thus, it is expensive for overparameterized models to reach consensus – even if they are in greater consensus compared to underparameterized models. This motivates our study of inexact consensus methods.

These empirical results conveying the invariance of consensus distance Υ to the convergence rate, especially for overparameterized deep NNs motivate the algorithm proposal with inexact functional consensus.

5. Inexact Consensus in [Koppel et al., 2018]

To overcome the inefficiency of establishing consensus with entire parameter matrices, [Koppel et al., 2018] propose a function consensus constraint on individual regressor functions. Consider a class of functions $f \in \mathcal{H}$ belonging to a hypothesized function class $\mathcal{H}$, and a strictly convex loss function used to penalize the deviation of regressor f from the output label y given by $l \ : \ \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ where $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are the feature vectors and scalar label outputs respectively.

$$J^T = \text{argmin}_{f_i \subset \mathcal{H}} \left(\sum_{i \in V} \left(\mathbb{E}_{x_i, y_i} \left[l_i f_i(x), y_i \right] \right] + \frac{\lambda}{2} \| f_i \|_{\mathcal{H}}^2 \right) \tag{16}$$

$$\text{such that } f_i = f_j \ \forall \ (i, j) \in E \tag{17}$$

To solve 16, [Koppel et al., 2018] equip the hypothesized function class $\mathcal{H}$ with a kernel function over the feature vector space $\kappa \ : \ \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that satisfies:

$$\langle f, \kappa(x_i, \cdot) \rangle_{\mathcal{H}} = f(x_i) \qquad \mathcal{H} = \overline{span(\kappa(x_i), \cdot)}$$

A series of experiments were carried to reproduce results in [Koppel et al., 2018] on decentralized stochastic optimization with regressors from a RKHS. The nonparametric optimization problem is formulated with an inexact consensus constraint: $f_i = f_j \ \forall (i,j) \in \mathcal{E}$. The problem may be formulated as:

$$\min \sum_{i \in \mathcal{V}} (\mathbb{E}_{(x_i,y_i)} \left[l_i(f_i(x_i, y_i)] + \frac{\lambda}{2} \|f_i\|^2_{\mathcal{H}} + \frac{c}{2} \sum_{j \in n_i} \mathbb{E}_{x_i}([f_i(x_i) - f_j(x_i)]^2)\right.$$

[Koppel et al., 2018] minimize a stochastic approximation of the above problem: i.i.d samples $(x_{i,t}, y_{i,t})$ are revealed to each worker f_i to create a new penalty function:

$$\min \sum_{i \in \mathcal{V}} (l_i(f_i(x_{i,t}), y_{i,t}) + \frac{\lambda}{2} \|f_i\|^2_{\mathcal{H}} + \frac{c}{2} \sum_{j \in n_j} (f_i(x_{i,t}) - f_j(x_{i,t}))^2)$$

where the representer theorem implies that at time t, the regressor f can be expanded as:

$$f_{i,t}(x) = \sum_{n=1}^{t-1} w_{i,n} \kappa(x_{i,n}, x) = w_{i,t}^T \kappa_{x_{i,t}}(x)$$

Note that κ is assumed to be Positive definite to qualify as an RKHS. A series of experiments were conducted to understand the effect of kernel choice $\kappa(\cdot)$ on consensus term $\frac{c}{2} \sum_{j \in n_i} \mathbb{E}_{x_i}([f_i(xi) - f_j(xi)]^2)$. Along with the provided Linear and Chi-Squared kernels, a radial basis kernel and a gaussian kernel were implemented in MATLAB. Secondly, we run numerical experiments to understand the dependence of consensus on the consensus penalty term c, and increase the penalty term every 200 iterations. Results of the simulations are given in figure 1 and figure 2. All experiments were run on a Gaussian Mixture Model dataset with 5000 training features and label pairs. Similar to [Koppel et al., 2018], the number of classes were fixed at 5, and the graph topology is a random network of 20 nodes, with equal rewiring probability of 0.2.

Algorithm 3 Inexact Consensus Learning on Node i (Inex-SGD)

INPUT: initial weights $\{\boldsymbol{\theta}_i^{(0)}\}_{i=1}^N$, max. no. of iterations T, SGD step sizes $\{\eta_t\}_{t\geq 0}$, batch size M
Draw the stopping iteration number $\mathsf{T} \sim \mathcal{U}\{0, ..., T\}$.
for $t = 0, 1, ..., \mathsf{T}$ **do**
  for $i = 1, ..., N$ **do** ▷ `//Sample batch from Local Distribution` D_i`//`
    $\xi_{i,k} = [\xi_i^{k,1}, \xi_i^{k,2}, \ldots, \xi_i^{k,M}]$
    Evaluate model on batch $\sum_{j=1}^M J(\boldsymbol{\theta}_i^k, \xi_i^{k,j})$
    For each worker $i = 1, ..., N$, send:

$$\left(\sum_{j=1}^{M} J(\boldsymbol{\theta}_i^k, \xi_i^{k,j}),\ \xi_{i,k} \right) \text{ to } j \in \mathcal{N}_i$$

    Receive:

$$\left(\sum_{p=1}^{M} J(\boldsymbol{\theta}_j^k, \xi_j^{k,p}),\ \xi_{j,k} \right) \ \forall j \text{ in } \mathcal{N}_i$$

    Calculate Stochastic gradient on worker i:

$$\begin{aligned} g_i^k &= \nabla_{\boldsymbol{\theta}_i} l_i(J(\xi_{i,k})) + \lambda \boldsymbol{\theta}_i^k \\ &\quad + c \sum_{j \in \mathcal{N}_i} \left(J(\xi_i^k; \boldsymbol{\theta}_i^k) - J(\xi_i^k; \boldsymbol{\theta}_j^k) \right) \nabla J(\xi_{i,k}, \boldsymbol{\theta}_i^k) \\ &\quad + c \sum_{j \in \mathcal{N}_i} \left(J(\xi_j^k; \boldsymbol{\theta}_i^k) - J(\xi_j^k; \boldsymbol{\theta}_j^k) \right) \nabla J(\xi_{j,k}, \boldsymbol{\theta}_i^k) \end{aligned} \tag{18}$$

    Perform SGD Update: $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k - \eta^k g_i^k$
  end for
end for
OUTPUT: trained weights $\{\boldsymbol{\theta}_i^{(\mathsf{T})}\}_{i=1}^N$.

5.1 Inex-SGD

Motivated by the formulation in 16, we extend the functional consensus constraint for a NN model. The complete logical flow of the proposed algorithm with mini-batch SGD is given in 3. Consider a dense feedforward NN model on the ith worker $f_i(x_i;\ \boldsymbol{\theta}_i)$. We are interested

in the following optimization problem:

$$\min_{f_i} \sum_{i=1}^{N} \left(\left[\mathbb{E}[l_i(f_i(x_i, y_i))] + \frac{\lambda}{2} \|f_i\|^2 \right] + \frac{c}{2} \sum_{j \in \mathcal{N}_i} \mathbb{E}_{x_i}[|f_i(x_i) - f_j(x_i)|^2] \right) \tag{19}$$

For the NN model parameterized by $\boldsymbol{\theta}$, we have: (20)

$$min_{\boldsymbol{\theta}_i \forall i=1,\ldots,N} \sum_{i=1}^{N} \left(\mathbb{E}_{x_i} \left[l_i(f(x_i;\ \boldsymbol{\theta}_i), y_i) \right] + \frac{\lambda}{2} \|\boldsymbol{\theta}_i\|^2 \right) + \sum_{j \in \mathcal{N}_i} \mathbb{E}_{x_i} \left[\frac{c}{2} |f(x_i; \boldsymbol{\theta}_i) - f(x_i; \boldsymbol{\theta}_j)|^2 \right] \tag{21}$$

We note several benefits to our proposed Inex-SGD algorithm, as proposed in algorithm 3

- Communication involves only sending data-point or minibatch information to immidiate neighbors, instead of large parameter matrices, that grow to occupy a large proportion of the communication cost in the case of overparameterized models. However, sending local data to immediate neighbors has two main drawbacks:
  - Communicating datapoints partially defeats the purpose of privacy-preserving machine learning, since neighboring nodes are allowed to read incoming data. Future work will propose to remove personal information/ obfuscate private information before transmission to neighbors.
  - While communication cost is low for transmitting a single data-point, communication cost grows with increase in mini-batch size, and is thus not scaleable to large batch sizes
- We eliminate the standard consensus constraint and substitute it with an inexpensive functional consensus constraint. For standard datasets, $f(x_i, \boldsymbol{\theta}_i$ returns either a scalar, or sparse vector – thus being less expensive.

5.2 Numerical Simulations

[Results Obtained in Term 2] The inexact consensus algorithm proposed in 3 was implemented in Python3.9, and simulations were carried out on a $N = 80$ CPU environment, with an openMPI network environment. For specificity, we detail the technical requirements and specifications of the network environment in Appendix **??**.

All simulations were performed on an $n = 8$ worker MPI environment, where workers were connected over a ring topology. The network graph, in absence of a formal adjacency matrix due to the inexact consensus nature, was simulated using the Python environment `networkX`. We train a series of overparameterized NN models with $d = 512$ hidden units per layer. Code snippet 5.2 is an academic version of inex-SGD, where all workers are simulated in one code run, instead of the simulation version where each worker is run independently $n = 8$ times with MPI.

Listing 1: Rough Python3.9 Implementation of Inex-SGD

```
def inexsgd(task, world, learning_rate, num_steps, penalty_inex=1e-2):
    state: torch.Tensor = task.init_state()  # shape [num_workers, ...]

    for step in range(num_epochs):
        batch = retrieve(dataloader)
        output = model(batch)          #Pass batch to local NN model

        for worker in world.workers:
            neighbors = world.neighbors(worker)
            for neighbor in neighbors:
                worker.broadcast_to(neighbor, (output, batch))
                 input, batch_in = worker.recv(neighbor)


        grad[step] = autograd(loss(output)))
                  + penalty_inex * ((model_local(batch)
                  - model_neighbor(batch_in))*autograd(output)
                  + penalty_inex * ((model_local(batch_in)
                  - model_neighbor(batch_in))*autograd(model(batch_in))


                perform_sgd_update(model_local, eta=1e=4, grad[step])
```

Train Dataset	% Visible (CIFAR-10)
Ring (d=2,n=12)	24.91
Ring (d=2,n=8)	37.5
Torus (d=4,n=12)	41.89
Torus (d=4,n=8)	62.11

Table 1: For a $n = 8$ ring topology graph, the average agent has seen 37.5% of all CIFAR-10 images by the 25000 iteration number when trained with inex-SGD.

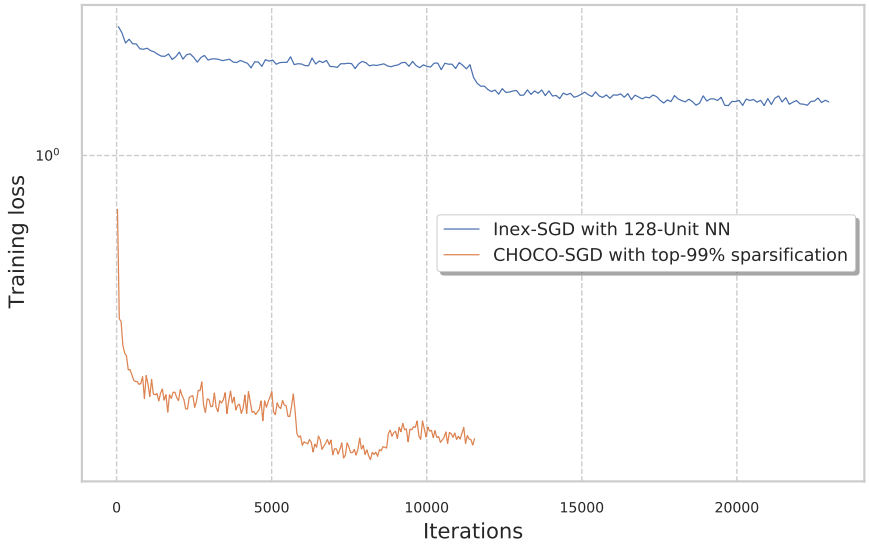

Figure 6: Quality of solution obtained by inex-SGD and CHOCO-SGD after 100 epochs of decentralized training. CHOCO-SGD is trained with top-1% sparsification, and inex-SGD is trained on a 512-width NN model. CHOCO-SGD finds higher quality solutions compared to inex-SGD. Consensus penalty c of inex-SGD fixed at 10^{-2} and both learning rates were scheduled to allow for better convergence.

5.3 Result

Figure 6 shows the quality of converged solutions of inex-SGD and CHOCO-SGD. Note that a top-99% sparsification is chosen for CHOCO-SGD.

Preliminary simulations also observed divergence of inex-SGD for a majority of simulation trials. Figure 2 shows the "data visibility" percentage of the average worker in an $n = 8$ ring topology. For the CIFAR-10 image dataset, the data-visibility is recorded as the percentage

of the CIFAR-10 dataset that is sent to the average worker. Decentralized learning requires the data visibility to be minimized. For the ring topology with $n = 8$ workers, the average worker is exposed to approximately 37.5% of the CIFAR-10 dataset. This represents another important drawback of the inexact consensus algorithm proposed. Note that while data-visibility is high for $n = 8$ setting, larger topologies recorded significantly lesser visibility. The data-visibility is also clearly dependent on network topology.

Implementation Boost: Passing minibatch index: An additional improvement was proposed after a review of previous simulation results. In inex-SGD v2, $\xi_{i,k}$ is sent to all $j \in \mathcal{N}_i$ only for the first epoch. Each worker records a scalar index of the minibatch recieved during this first iteration, that is consistent with sender and receiver. The index value is unique, and is stored permanently for the remainder of training. Following the first epoch, workers do not re-send their minibatches, and instead send the batch's index value, that is received by each neighbor. Thus, communication cost only increases with minibatch size for the first epoch, and sharply decreases for following iterations of inex-SGD. Preliminary simulations with inex-SGD v2 on identical network environment, NN models, and datasets as standard vanilla inex-SGD report an decreased running time, with a near-identical data visibility score for each agent.

Index Storage Implementation: To improve lookup time, given a minibatch index from a neighbor, a hash-map like structure was implemented where index (hash values) were correlated to a ($\mathsf{batchsize} \times 32 \times 32 \times 3$) tuple for CIFAR-10 dataset.

Galvanized by the drawbacks of inex-SGD, we now summarize algorithm and theoretical results of the Lottery Ticket Hypothesis, followed by a new propose inexact consensus algorithm based on identifying sparse subnetworks of overparameterized NN models.

6. Lottery Ticket Hypothesis

Inspired by all kinds of pruning techniques, [Frankle and Carbin, 2018] proposed the famous Lottery Ticket Hypothesis which states that a randomly-initialized dense neural network contains a subnetwork which is able to be trained to match the test accuracy of the original network under the same initialization of the original network. An analogy to lottery ticket is mentioned where different subnetwork of the initialized neural network is similar to a lottery ticket and a larger model have more combination of subnetwork, thus it has a larger chance of winning the lottery, i.e. converge to the parameters that obtains high test accuracy. More formally, the lottery ticket hypothesis was proposed in [] as:

Conjecture 1 (Standard Lottery Ticket Hypothesis) *Consider a dense feed-forward neural network $f(x;\theta)$ with initial parameters $\theta = \theta_0 \sim \mathcal{D}_\theta$. When optimizing with stochastic gradient descent (SGD) on a training set, f reaches minimum validation loss l at iteration j with test accuracy a. In addition, consider training $f(x; m \odot \theta)$ with a mask $m \in \{0,1\}^{|\theta|}$ on its parameters such that its initialization is $m \odot \theta_0$. When optimizing with SGD on the same training set (with m fixed), f reaches minimum validation loss l' at iteration j' with test accuracy a'. The lottery ticket hypothesis predicts that $\exists m$ for which $j' \leq j$ (commensurate training time), $a' \geq a$ (commensurate accuracy), and $\|m\|_0 \ll |\theta|$ (fewer parameters).*

In [Frankle and Carbin, 2018] the author also proposed an iterative pruning method such that it can identify the winning ticket (i.e. the best subnetwork) after n iterations. Note that an important artefact of the standard Lottery Ticket Hypothesis is the requirement of training the sparse subnetwork. The algorithm is detailed, as present in [Frankle and Carbin, 2018] below:

1. . Randomly initialize a neural network $f\left(x;\theta_0\right)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j.

3. Prune $p\%$ of the parameters in θ_j, creating a mask m.

4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

Winning Tickets obtained through the above algorithms matched the accuracy of the original network at smaller sizes than does one-shot pruning.

6.1 Lottery Tickets in Random Networks

While the lottery ticket hypothesis applies appropriately to smaller NN models, the standard LTH conjecture is weak when considering large, overparameterized models. According to [Ramanujan et al., 2020] such large networks contain randomly initialized subnetworks (untrained) that can approximate a target network's performance. Note that the target network is trained on a particular dataset, and the sparse subnetwork is untrained with weights assigned from a uniform distribution. The target network is also smaller when compared to the original overparameterized network. Figure 8 explains the hypothesis presented in [Ramanujan et al., 2020]. Formally, we write the strong LTH conjecture as follows:

Conjecture 2 (Strong LTH) *Within a sufficiently over-parameterized neural network with random weights (e.g. at initialization), there exists a subnetwork that achieves (without training) competitive accuracy compared to the full network after training.*

The strong LTH conjecture has led to several streams of work that aim to:

1. Quantify the degree of overparameterization required to find a sparse subnetwork that can reliably approximate a target network with at least a probability $1 - \delta$.

2. Design algorithms that can extract a sparse subnetwork $\tilde{G}$ from the original overparameterized network G

We first formalize a definition of the subgraph/ subnetwork.

Conjecture 3 (Definition of a subnetwork [Malach et al., 2020]) *Fix a network* $G(x) = W_d\sigma\left(W_{d-1}\sigma\left(\ldots W_1 x\right)\right)$*, where* $\sigma(x) = \max\{x, 0\}$ *(ReLU).*

A subnetwork of G *is any network of the form* $\tilde{G}(x) = \tilde{W}_d\sigma\left(\tilde{W}_{d-1}\sigma\left(\ldots \tilde{W}_1 x\right)\right)$*, where* $\tilde{W}_i = B_i \odot W_i$ *for some* $B_i \in \{0,1\}^{n_{in} \times n_{out}}$ *(here* $\odot$ *is Hadamard product)*

Given the definition of the subnetwork, [Malach et al., 2020] prove that the minimum over-parameterization required of the original network G is of the order $\mathcal{O}\left(\frac{d^4 l^2}{\epsilon^2}\right)$ for a target network F with width l and depth of d layers for an $\epsilon-$ approximate approximation. Formally, the theorem is stated below.

Conjecture 4 (Degree of Overparameterization Required for Strong-LTH [Malach et al., 2020]) *Fix some target fully-connected ReLU-network* F *of width* k*, depth* d *and input dimension* n*. Fix* $\delta > 0$*.*

Then, a randomly-initialized network G *of width poly* $(d, n, k, 1/\epsilon, \log(1/\delta))$ *and depth* $2d$*, has w.p.* $\geq 1 - \delta$ *a subnetwork* $\tilde{G}$ *that approximates* F *up to* ϵ*.*

This rate of overparameterization was improved in [Pensia et al., 2020]. Below stated is the informal theorem:

Conjecture 5 *[Pensia et al., 2020] A randomly initialized network with width* $O(d\log(dl/\min\{\epsilon, \delta\}))$ *and depth* $2l$*, with probability at least* $1-\delta$*, can be pruned to approximate any neural network with width* d *and depth* l*, up to error* ϵ*.*

6.2 Determining optimal subnetwork $\tilde{G}$

Notation We use lower-case letters to represent scalars. For all experiments and equations, we use fully-connected neural networks (FC NNs). Each FC NN has n_l nodes (or neurons) on each layer l. We denote the set $V_l = [(v_l^1, \ldots, v_{n_l}^l)]$ to denote all nodes on layer l. Let I_v be the input to node v and let Z_v denote the output of node v. We define w_{uv} to be a connected edge between node u and node v, where u and v are not in the same layer.

Preliminaries Note that for a FC NN model, for a single unit, we have $Z_v = \mathsf{ReLU}(I_v)$. Therefore, the forward propagation can be written as:

$$I_v = \sum_{u \in V_{l-1}} w_{uv} Z_u$$

Potential candidates for determining a sparse subnetwork are important, as an optimal subnetwork can significantly improve the quality of the solution found by the decentralized learning algorithm. [Zhou et al., 2019] propose the concept of a supermask – a binary mark applied to the overparameterized model. Empirical results in [Zhou et al., 2019] indicate a 40-50% top-1 accuracy on the CIFAR-10 dataset after applying a supermask on a randomly initialized overparameterized network. The reasoning behind the success of the supermask is the low-magnitude of several weights during the training stage, and the nature of gradient-based optimization algorithms that incentivize such weights to tend to zero.

Furthermore, [Zhou et al., 2019] provide an algorithm to determine $\tilde{G}$ by determining a probability p for every weight w_{uv} between node u and node v. This algorithm assumes a constant p across all iterations, and behaves in a dropout-like mechanism. [Ramanujan et al., 2020] improvement to the above method serves as the foundation for the simulations done in section 6.3. In [Ramanujan et al., 2020], the forward and backpropagation scores are modified as follows:

- Assume desired subnetwork $\tilde{G} = (V, \mathcal{E})$.

- Assign a scalar score s_{uv} to each edge w_{uv}.
- Forward propagate:

$$I_v = \sum_{u \in V_{l-1}} w_{uv} Z_u \mathbb{I}(s_{uv})$$

  Where $\mathbb{I}(s_{uv}) = \begin{cases} 1 & s_{uv} \in \mathsf{top}_k(l) \\ 0 & \mathsf{else} \end{cases}$
- Calculate $\nabla s_{uv}^t = \frac{d(\mathsf{loss})}{dI_v} w_{uv} Z_u$
- Update: $s_{uv}^{t+1} = s_{uv}^t - \alpha \nabla s_{uv}^t$

Where loss is a smooth loss function. Here, s_{uv}^{t+1} is the updated score of the edge. The intuition behind making the score s_{uv} learnable is to allow a particular edge to appear in the top-k weights if it consistently aligns with the negative gradient. The proof of the decrease in loss with the above algorithm is given in appendix B1 of [Ramanujan et al., 2020].

6.3 Proposed Algorithm and Result

We propose the pruning technique outlined in the previous section to yield the subnetwork $\tilde{G}$. While the above algorithm is an iterative process that requires access to either the entire, or a subset of the data, the above method is computationally less expensive to traditional lottery ticket hypothesis networks.

We propose training overparameterized networks with the traditional D-SGD algorithm after pruned to a subnetwork. Algorithm 1 contains details of the proposed algorithm. In figure 1, a comparison of traditional D-SGD, CHOCO-SGD, and the proposed strong-Lottery ticket-based D-SGD algorithm is performed. From the figure, the proposed algorithm nearly matches the performance of CHOCO-SGD with lesser communication cost due to the sparsity of the underlying network. The proposed method is arguably similar to a top-k compression used by CHOCO-SGD during each communication stage. However, the proposed

algorithm requires no error correction, and can simply be attached to the D-SGD algorithm after distributing $\tilde{G}$ to all agents.

Algorithm 4 Strong LTH + DPSGD

Require: SGD step-size $\eta > 0$, weights $\{\boldsymbol{\theta}_i^0\}_{i=0}^N$, node-score step-size $\alpha > 0$, pruning steps $\mathcal{T}$
  for $t = 1, 2, \ldots, \mathcal{T}$ **do**
    for Each Layer $l \in L$ **do**
      $I_v = \sum_{u \in V_{l-1}} w_{uv} Z_u \mathbb{I}(s_{uv})$
      $\nabla s_{uv}^t = \frac{d(\mathsf{loss})}{dI_v} w_{uv} Z_u$
      Update: $s_{uv}^{t+1} = s_{uv}^t - \alpha \nabla s_{uv}^t$
    end for
  end for
  Output $\tilde{G}$
  Send $\tilde{G}$ to all workers
  for $t = 1, 2, \ldots, T$ **do**
    for worker $i = 1, \ldots, N$ **do**
      Sample minibatch from node i: $\xi_{k,i} \sim \mathcal{D}_i$ and calculate local stochastic gradient $\boldsymbol{g}_i^t = \nabla J(\tilde{G}_i^k, \xi_i^k)$
      Receive parameter variables from neighbors $\tilde{G}_j^k \ \forall j \in V, \ w_{ij} > 0$
      Compute consensus step: $\tilde{G}_i^{k+\frac{1}{2}} = \sum_j w_{ij} \tilde{G}_j^k$
      Update the local variable $\tilde{G}_i^{k+1} = \tilde{G}_i^{k+\frac{1}{2}} - \eta \boldsymbol{g}_i^t$
    end for
  end for
  Return Averaged model $\bar{\tilde{G}}^* = \sum_N \tilde{G}_i^T$

7. Conclusion

In this final report, we showed the preferable convergence and consensus properties of over-parameterized models used in compressed DSGD algorithms. To improve on the rate of convergence, we propose Inex-SGD: a decentralized algorithms where workers reach in consensus from the model output as opposed to the parameter matrices of workers. We summarize the chief result of the Strong-Lottery Ticket Hypothesis, and propose a modification to existing algorithm proposed in [Ramanujan et al., 2020] for application to the decentralized setting. With a pruned, randomly initialized NN model, D-SGD empirically shows compa-

rable performance to CHOCO-SGD without any need for error-correction and training-time compression.

8. Future Work

- inex-SGD for time-varying graphs

References

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

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

[Blondel et al., 2005] Blondel, V. D., Hendrickx, J. M., Olshevsky, A., and Tsitsiklis, J. N. (2005). Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2996–3000. IEEE.

[Boyd et al., 2006] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Diao et al., 2020] Diao, E., Ding, J., and Tarokh, V. (2020). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

[Frankle and Carbin, 2018] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

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

[Graham et al., 2005] Graham, R. L., Woodall, T. S., and Squyres, J. M. (2005). Open mpi: A flexible high performance mpi. In *International Conference on Parallel Processing and Applied Mathematics*, pages 228–239. Springer.

[Koloskova* et al., 2020] Koloskova*, A., Lin*, T., Stich, S. U., and Jaggi, M. (2020). Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*.

[Koloskova et al., 2019] Koloskova, A., Stich, S., and Jaggi, M. (2019). Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487. PMLR.

[Kong et al., 2021] Kong, L., Lin, T., Koloskova, A., Jaggi, M., and Stich, S. U. (2021). Consensus control for decentralized deep learning. *arXiv preprint arXiv:2102.04828*.

[Koppel et al., 2018] Koppel, A., Paternain, S., Richard, C., and Ribeiro, A. (2018). Decentralized online learning with kernels. *IEEE Transactions on Signal Processing*, 66(12):3240–3255.

[Koppel et al., 2019] Koppel, A., Warnell, G., Stump, E., and Ribeiro, A. (2019). Parsimonious online learning with kernels via sparse projections in function space. *The Journal of Machine Learning Research*, 20(1):83–126.

[Kovalev et al., 2021] Kovalev, D., Koloskova, A., Jaggi, M., Richtarik, P., and Stich, S. (2021). A linearly convergent algorithm for decentralized optimization: Sending less bits for free! In *International Conference on Artificial Intelligence and Statistics*, pages 4087–4095. PMLR.

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

[Lian et al., 2018] Lian, X., Zhang, W., Zhang, C., and Liu, J. (2018). Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052. PMLR.

[Malach et al., 2020] Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. (2020). Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR.

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

[Pensia et al., 2020] Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. (2020). Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in Neural Information Processing Systems*, 33:2599–2610.

[Ramanujan et al., 2020] Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. (2020). What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902.

[Rao and Wai, 2021] Rao, A. A. and Wai, H.-T. (2021). An empirical study on compressed decentralized stochastic gradient algorithms with overparameterized models. *arXiv preprint arXiv:2110.04523*.

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

[Seide et al., 2014] Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. (2014). 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. In *Interspeech 2014*.

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

[Tang et al., 2018] Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J. (2018). Communication compression for decentralized training. *Advances in Neural Information Processing Systems*, 31:7652–7662.

[Tang et al., 2019] Tang, H., Lian, X., Qiu, S., Yuan, L., Zhang, C., Zhang, T., and Liu, J. (2019). Deepsqueeze: Decentralization meets error-compensated compression. *arXiv preprint arXiv:1907.07346*.

[Wang et al., 2021] Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., et al. (2021). A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*.

[Xiang et al., 2020] Xiang, L., Wang, L., Wang, S., and Li, B. (2020). Achieving consensus in privacy-preserving decentralized learning. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 899–909. IEEE.

[Zhou et al., 2019] Zhou, H., Lan, J., Liu, R., and Yosinski, J. (2019). Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32.

Appendix A. Experimental Setup

We consider the task of training a classifier with the CIFAR-10 dataset [**?**] 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 [Koloskova* et al., 2020, Goyal et al., 2017]. To establish a challenging generalization task, we test the trained models on CIFAR-10.1 [Recht et al., 2018].

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 coordinates k allowed to be communicated between workers. We choose a constant consensus parameter $\gamma = 0.0375$ in Algorithm 2 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 [Graham et al., 2005] network where we assign an independent CPU process to each worker.

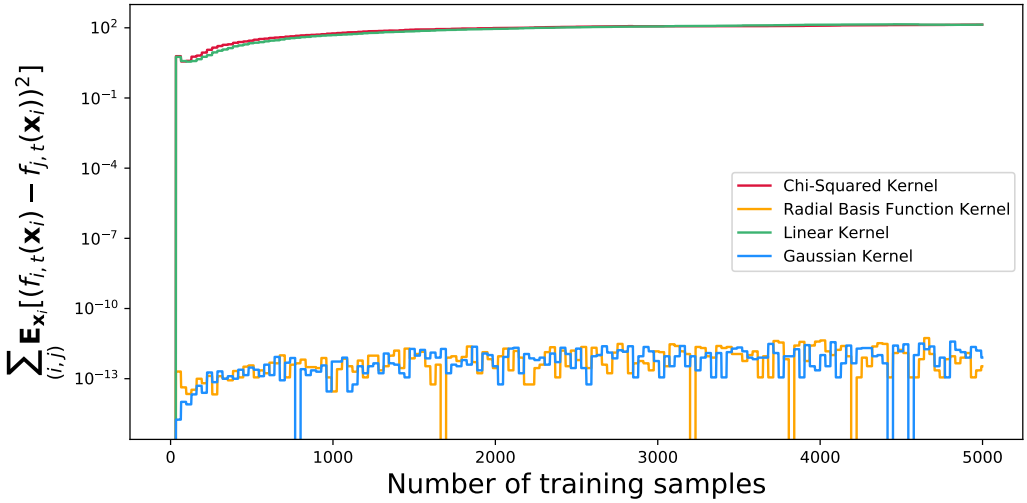

Figure 4: Inexact consensus error $\sum_{(i,j)} \mathbf{E}_{\mathbf{x}_i}[(f_{i,t}(\mathbf{x}_i) - f_{j,t}(\mathbf{x}_i))^2]$ calculated as en expectation over number of training samples. Radial basis kernel $\kappa(x, x') = exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ and Gaussian function kernels lead to consensus errors several magnitudes lower than the simpler linear and chi-squared kernels.

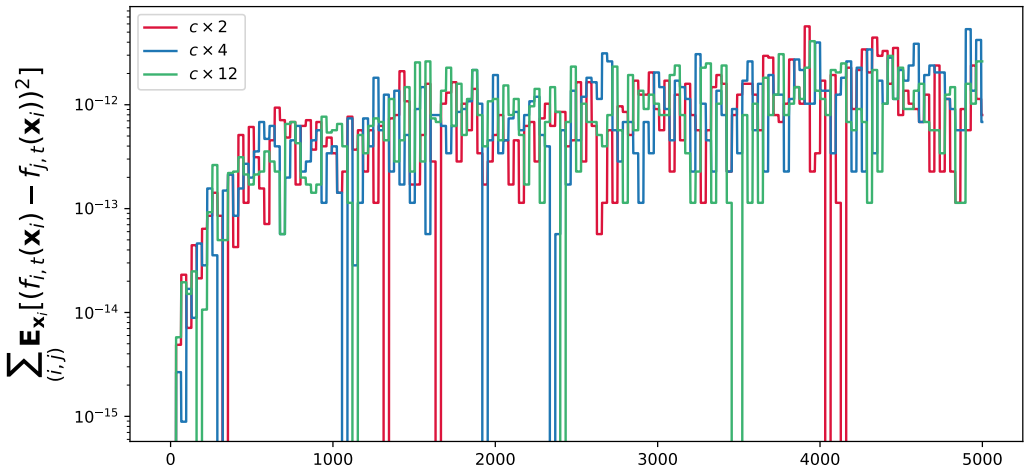

Figure 5: Inexact consensus error $\sum_{(i,j)} \mathbf{E}_{\mathbf{x}_i}[(f_{i,t}(\mathbf{x}_i) - f_{j,t}(\mathbf{x}_i))^2]$ calculated as en expectation over number of training samples with a radial basis kernel $\kappa(x, x') = exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ with different values of penalty coefficient increments. Consensus errors converge at the same rate for different penalty parameters for a given kernel and datapoints.

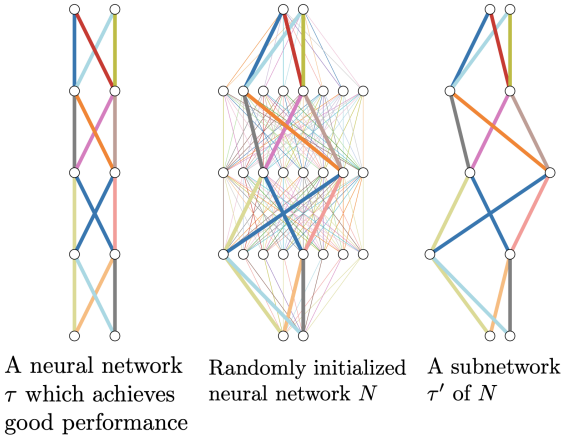

Figure 7: [Ramanujan et al., 2020] If a neural network with random weights (center) is sufficiently overparameterized, it will contain a subnetwork (right) that perform as well as a trained neural network (left) with the same number of parameters.

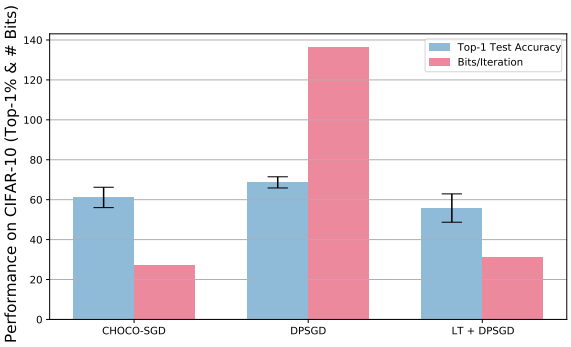

Figure 8: Simulation result with comparison of Algorithm 4 to CHOCO-SGD and D-SGD. The proposed strong-LTH D-SGD method with high sparsity reaches a similar quality of solution as CHOCO-SGD with only a marginally higher value of bits/iteration value (indicative of communication cost). The method performs better than standard D-SGD algorithm in bits/iteration.

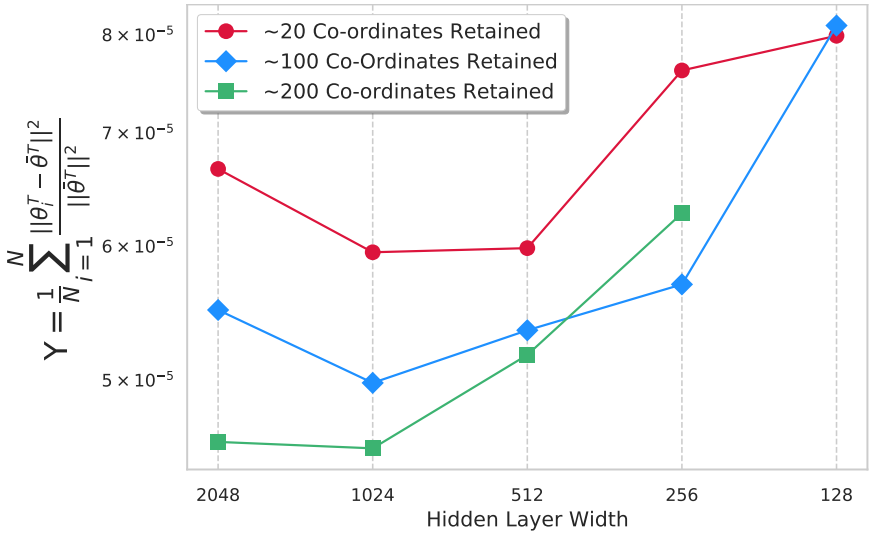

Figure 9: Converged, normalized consensus distance between $N = 8$ workers for different NN layer widths. Overparameterized models enjoy significantly greater consensus among workers with only marginal dependence on sparsification co-ordinate bandwidth for larger models.

Layer Width	Normalized Consensus Distance [cf. (15)]		
	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 2: **Converged consensus distances at intermediate training epoch numbers.** Overparameterized models converge to better-consensus solutions at a slower rate compared to low-width NNs