DECENTRALIZED DEEP LEARNING USING MOMENTUM-ACCELERATED CONSENSUS

Aditya Balu[⋆] *Zhanhong Jiang*[†] *Sin Yong Tan*[⋆] *Chinmay Hegde* [§] *Young M Lee*[†] *Soumik Sarkar*[⋆]

[⋆] Iowa State University [†] Johnson Controls [§] New York University

ABSTRACT

We consider the problem of decentralized deep learning where multiple agents collaborate to learn from a distributed dataset. While several decentralized deep learning approaches exist, the majority consider a central parameter-server topology for aggregating the model parameters from the agents. However, such a topology may be inapplicable in networked systems such as ad-hoc mobile networks, field robotics, and power network systems where direct communication with the central parameter server may be inefficient. In this context, we propose and analyze a novel decentralized deep learning algorithm where the agents interact over a fixed communication topology (without a central server). Our algorithm is based on the heavy-ball acceleration method used in gradient-based optimization. We propose a novel consensus protocol where each agent shares with its neighbors its model parameters and gradient-momentum values during the optimization process. We consider nonconvex objective functions and theoretically analyze our algorithm's performance. We present several empirical comparisons with competing decentralized learning methods to demonstrate the efficacy of our approach under different communication topologies.

***Index Terms*—** Decentralized deep learning, nonconvex, momentum, convergence

1. INTRODUCTION

Spurred by the need to accelerate deep neural network training with massive distributed datasets, several recent research efforts [1, 2, 3, 4] have put forth a variety of distributed, parallel learning approaches. One line of work has focused on adapting traditional deep learning algorithms that use a single CPU-GPU environment to a distributed setting with a network of several GPUs [5, 2, 6, 7]. Some of these approaches also can be used in conjunction with gradient compression schemes between compute nodes in the network [8]. A different line of works falls under the umbrella of *federated learning* [9] which deals with inherently decentralized datasets, i.e., each compute node has its own corresponding set of data samples that are not shared. The majority of works in this area consider a central parameter-server topology that aggregates estimates of model parameters from the agents.

In this paper, our particular focus is on decentralized learning where there is *no* central server: each node in the network maintains its model parameters (which it can communicate with its neighbors defined according to a pre-specified, but otherwise arbitrary, communication topology), and the goal is to arrive at a consensus model for the whole network. See [10, 11, 12, 13, 14, 15, 16] for examples of such decentralized learning approaches.

While the above works are representative of key advances in the algorithmic front, several gaps remain in our understanding of centralized versus distributed learning approaches. Conspicuous among these gaps is the notion of *momentum*, which is a common technique to speed up convergence in gradient-based learning methods [22, 23]. However, few papers (barring exceptions such as [17, 11, 12, 20]) in the decentralized learning literature have touched upon momentum-based acceleration techniques, and to our knowledge, rigorous guarantees in the context of nonconvex and stochastic optimization have not been presented. Our objective in this paper is to fill this key gap from both a theoretical as well as empirical perspective.

Our contributions. We propose and analyze a stochastic optimization algorithm that we call decentralized momentum SGD (DMSGD), based on the classical notion of momentum (or the *heavy-ball* method [24]). See Table 1 for more comparisons.

For smooth and nonconvex objective functions, we show the convergence to a *first-order stationary point*, that is, the algorithm produces an estimate x with sufficiently small gradient ($\|\nabla f(x)\| \leq \varepsilon$) after $\mathcal{O}(1/\varepsilon + 1/(N\varepsilon^2))$ iterations, where N is the number of agents. Additional results on strongly-convex and its relaxation using the Polyak-Łojasiewicz criterion is provided in the extended version of this paper [25].

We empirically compare DMSGD with baseline decentralized methods such as D-PSGD/CDSGD [10, 11]. We show that when the momentum term is appropriately weighted, DMSGD is faster and more accurate than these baseline methods, suggesting the benefits of its use on practice.

2. PROBLEM SETUP AND PRELIMINARIES

Let the parameters of the deep neural network be denoted as $x \in \mathbb{R}^d$. We define a loss function $f : \mathbb{R}^d \to \mathbb{R}$ and denote its corresponding stochastic gradient by g.

Decentralized learning. Consider a static undirected graph $G = (V, E)$, where V is the node set and E is an edge set. Consequently, if we assume that there exist N nodes (agents) in the networked system, we can denote $V = \{1, 2, ..., N\}$ while $E \subseteq V \times V$. If $(j, l) \in E$, then agent j can communicate with agent l. A node $j \in V$ has its neighbors $Nb(j) \triangleq \{j \in V : (j, l) \in E \text{ or } l = j\}$. We assume that the network G is connected without loss of generality throughout this paper. In this paper, we consider a finite sum minimization problem defined as follows:

$$\min \frac{1}{n} \sum_{j=1}^{N} \sum_{i \in \mathcal{D}_j} f_j^i(\mathbf{x}), \tag{1}$$

where $\mathcal{D}_j$ denotes the subset of the training data (comprising n_j samples) only known by the j^{th} agent such that $\sum_{j=1}^{N} n_j = n$, n is the size of dataset, N is the number of agents, $f_j : \mathbb{R}^d \to \mathbb{R}$ are local loss functions of each node. Let $x^j \in \mathbb{R}^d$ be a local copy of x. Then, define $\mathbf{x} = [x^1; x^2; \ldots; x^N] \in \mathbb{R}^{Nd \times 1}$. All vector and matrix norms are Euclidean and Frobenius norms respectively.

In this paper, for simplicity of presentation, we assume that $d = 1$, while noting that exactly the same proof ideas hold when $d > 1$ albeit at the expense of extra notation.

Equation 1 can be rewritten as the constrained problem:

corr address: soumiks@iastate.edu

Table 1: Comparisons between different optimization approaches. Gra.Lip.: Gradient Lipschitz. Str.Con.: strongly convex. Cen.: centralized. Con.: convex. Dec: decentralized. ρ: a positive constant in $(0, 1)$. k is the number of iterations. N: the number of agents. PL: Polyak-Łojasiewicz condition. It should be noted that each ρ in different methods vary in real values.

Method	f	Rate	Setting	Gra.Lip.	Stochastic	Momentum
HBM	Str.Con.	$\mathcal{O}(\rho^k)$	Cen.	Yes	No	Yes
MSWG [17]	Str.Con.	$\mathcal{O}(\rho^k)$	Dec.	Yes	No	Yes
SHB [18]	Con.	$\mathcal{O}(\rho^k)$	Cen.	Yes	Yes	Yes
DMSGD (This paper)	**PL (Quasi-convex)**	$\mathcal{O}(\rho^k)$	**Dec.**	**Yes**	**Yes**	**Yes**
SUM [19]	Nonconvex	$\mathcal{O}(1/\sqrt{k})$	Cen.	Yes	Yes	Yes
CDSGD/D-PSGD	Nonconvex	$\mathcal{O}(1/k + 1/\sqrt{Nk})$	Dec.	Yes	Yes	No
MSGD [20]	Nonconvex	$\mathcal{O}(1/k + 1/\sqrt{Nk})$	Cen./Dec.	Yes	Yes	Yes
SlowMo[21]	Nonconvex	$\mathcal{O}(1/k + 1/\sqrt{Nk})$	Cen.	Yes	Yes	Yes
DMSGD (This paper)	**Nonconvex**	$\mathcal{O}(1/k + 1/\sqrt{Nk})$	**Dec.**	**Yes**	**Yes**	**Yes**

$$\min F(\mathbf{x}) \triangleq \frac{1}{n}\sum_{j=1}^{N}\sum_{i\in\mathcal{D}_j} f_j^i(x^j), \quad \text{s.t. } \Pi\mathbf{x} = \mathbf{x}, \tag{2}$$

where the matrix Π is the mixing matrix encoding the adjacency structure of G (which is assumed to be *doubly stochastic*). By turning the hard constraint $\Pi\mathbf{x} = \mathbf{x}$ into a soft constraint that penalizes the corresponding decision variables $\mathbf{x}$, the following equivalent objective function can be obtained:

$$\mathcal{F}(\mathbf{x}) := F(\mathbf{x}) + \frac{1}{2\xi}(\mathbf{x}^T(I-\Pi)\mathbf{x}) \tag{3}$$

where $\xi > 0$. In the next section, we will show that ξ can be related to the step size α.

In order to study the behavior of the proposed algorithm, we now present basic definitions and assumptions.

***Definition* 1.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, if for all $x, y \in \mathbb{R}^d$, we have

$$f(y) \le f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2.$$

***Definition* 2.** A function $c(\cdot)$ is said to be coercive if it satisfies $c(x) \to \infty \; when \; \|x\| \to \infty$.

***Assumption* 1.** The objective functions $f_j : \mathbb{R}^d \to \mathbb{R}$ are assumed to satisfy the following conditions: a) Each f_j is L_j-smooth; b) each f_j is proper (not everywhere infinite) and coercive.

An immediate consequence of Assumption 1 a) is that $\sum_{j=1}^{N} f_j(x^j)$ is L_m-smooth where $L_m := \max\{L_1, L_2, ..., L_N\}$.

***Assumption* 2.** The unified objective function $\mathcal{F}(\mathbf{x})$ has bounded gradient such that $\|\nabla\mathcal{F}(\mathbf{x})\| \le M$.

Denote $\mathcal{S}(\mathbf{x})$ by the stochastic gradient of $\mathcal{F}$ at point $\mathbf{x}$ such that it is the unbiased estimate of $\nabla\mathcal{F}(\mathbf{x})$. We next make another assumption on the variance of $\mathcal{S}(\mathbf{x})$ to ensure that it is bounded from above.

***Assumption* 3.** The stochastic gradients of $\mathcal{F}$ satisfy: $Var(\mathcal{S}(\mathbf{x})) = \mathbb{E}[||\mathcal{S}(\mathbf{x}) - \nabla\mathcal{F}(\mathbf{x})||^2] \le \sigma^2$

With Assumption 2 and Assumption 3(b), we have

$$\begin{aligned}\mathbb{E}[||\mathcal{S}(\mathbf{x})||] &= \sqrt{(\mathbb{E}[||\mathcal{S}(\mathbf{x})||])^2} \le \sqrt{\mathbb{E}[||\mathcal{S}(\mathbf{x})||^2]} \\ &= \sqrt{||\mathbb{E}[\mathcal{S}(\mathbf{x})]||^2 + Var(S(\mathbf{x}))} \\ &\le \sqrt{M^2+\sigma^2}.\end{aligned}$$

3. PROPOSED ALGORITHM

We first present our proposed approach in Algorithm 1.

In the above update law, $g^j(x_k^j)$ is a stochastic gradient which is calculated by randomly selecting at uniform a mini-batch for each agent. Let $\mathcal{D}'$ be a mini-batch of the dataset $\mathcal{D}_j$ of the j-th agent. Therefore,

$$g^j(x_k^j) = \frac{1}{b}\sum_{i\in\mathcal{D}'} \nabla f_j^i(x_k^j),$$

where b is the size of $\mathcal{D}'$.

Algorithm 1 DMSGD

Input : m, Π, x_0^j, x_1^j, α, N, ω, β
Output : x^*
for $k = 1 : m$ **do**
  for $j = 1 : N$ **do**
    Consensus step: Nodes run average consensus:
    $v_k^j = \sum_{l\in Nb(j)} \pi_{jl} x_k^l$;
    Momentum step:
    $\delta_k = \omega(x_k^j - x_{k-1}^j) + (1-\omega)(v_k^j - v_{k-1}^j)$;
    Local gradient step for node j:
    $x_{k+1}^j = v_k^j - \alpha g^j(x_k^j) + \beta\delta_k$;
  end
end

In [11, 12], decentralized variants of classic momentum have been proposed (without analysis). On the other hand, our proposed DMSGD method uses a special parameter, ω, to trade off between two different momentum terms. The first momentum term is implemented over the true *decision* variables (x_k^j) while the second momentum term is implemented over the *consensus* variables (v_k^j), which are graph-smoothed averages of the decision variables.

We present a fairly general analysis for DMSGD; as special cases, we obtain known convergence properties for other methods. For example, we recover the decentralized classic momentum SGD by setting the parameter $\omega = 1$. When $\omega = 0$, DMSGD produces a new decentralized MSGD algorithm in which the momentum relies on the consensus variables. When the parameter β is set to 0, the proposed DMSGD boils down to the decentralized SGD method without momentum [11, 10]. Another slightly different alternative of DMSGD is to replace v_k^j with x_k^j in the local gradient step such that the consensus only affects the momentum term. The intuition behind this variant is that for the local gradient step, agent j only relies on its current state information instead of the consensus, "refusing" to proceed the update on top of an "agreement". For convenience and simplicity, the initial values of x^j are set to 0 throughout the analysis.

We now rewrite the core update law with in a vector form as:

$$\begin{aligned}\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha(\mathbf{g}(\mathbf{x}_k) + \frac{1}{\alpha}(I-\Pi)\mathbf{x}_k) \\ + \beta(\omega I + (1-\omega)\Pi)(\mathbf{x}_k - \mathbf{x}_{k-1})\end{aligned} \tag{4}$$

Here, we define $\mathcal{S}(x_k) = \mathbf{g}(\mathbf{x}_k) + \frac{1}{\alpha}(I-\Pi)\mathbf{x}_k$ and $\tilde{\Pi} = \omega I +$

$(1-\omega)\Pi$. Consequently, Eq. 4 can be written in a compact form as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathcal{S}(\mathbf{x}_k) + \beta\tilde{\Pi}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{5}$$

The simplification in Eq. 5 enables us to construct a function that unifies the true objective function with a term that captures the constraint of consensus among agents (nodes of the communication graph).

$$\mathcal{F}(\mathbf{x}) := F(\mathbf{x}) + \frac{1}{2\alpha}(\mathbf{x}^T(I-\Pi)\mathbf{x}) \tag{6}$$

Comparing Eqs. 3 and 6, we can know that they have exactly the same form and in our specific case corresponding to DMSGD, the parameter ξ is the step size α. $\mathcal{F}$ is smooth with $L' = L_m + \frac{1}{\alpha}(1-\lambda_2)$ where λ_2 is the second-largest eigenvalue of Π.

4. CONVERGENCE ANALYSIS

Consensus. We first prove that the agents achieve consensus, i.e., each agent eventually obtains a parameter that is close to the ensemble average $\bar{x}_k = \frac{1}{N}\sum_{j=1}^{N} x_k^j$, using the metrics of $\mathbb{E}[\|x_k^j - \bar{x}_k\|]$. In the setting of $d = 1$, though x_k^j and $\bar{x}_k$ are both scalars, we use the norm notation here for generality. As defined above, $\mathbf{x}$ has dimension of N. Define $\bar{\mathbf{x}}_k = [\bar{x}_k; \bar{x}_k; ...; \bar{x}_k]_N$. Therefore, it holds that $\|x_k^i - \bar{x}_k\| \leq \|\mathbf{x}_k - \bar{\mathbf{x}}_k\|$ [26] and instead of directly bounding $\|x_k^i - \bar{x}_k\|$, we investigate the upper bound for $\|\mathbf{x}_k - \bar{\mathbf{x}}_k\|$. We first obtain:

***Proposition* 1.** (**Consensus**) Let all assumptions hold. The iterates generated by DMSGD satisfies the following inequality $\forall k \in \mathbb{N}$, $\exists\alpha > 0$:

$$\mathbb{E}[|x_k^j - \bar{x}_k|] \leq \frac{8\alpha\sqrt{N}\sqrt{M^2+\sigma^2}}{\sqrt{\eta(1-\beta\Lambda)}(1-\sqrt{\beta\Lambda})}, \tag{7}$$

where η is defined as an arbitrarily small constant such that $\tilde{\Pi} \succcurlyeq \eta I$, $0 < \eta < 1$, $\Lambda = \omega + (1-\omega)\lambda_2$.

Proof. The proof for this proposition is fairly technical and we provide the sketch here, referring interested readers to the extended version of this paper [25]. We first define $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \bar{\mathbf{x}}_k$ and construct the linear time-invariant system for $[\tilde{\mathbf{x}}_{k+1}; \tilde{\mathbf{x}}_k]$. Then by induction and setting initialization 0, we can express $[\tilde{\mathbf{x}}_{k+1}; \tilde{\mathbf{x}}_k]$ using only the coefficient matrices and stochastic gradient inputs. By leveraging the decomposition techniques in matrices, the upper bound of matrix norms is obtained correspondingly. Hence, the iterates converge to the consensus estimate. □

***Remark* 1.** Proposition 1 provides a uniform consensus error upper bound among agents, proportional to the step size α and the number of agents N and inversely proportional to the gap between the largest and second-largest (in magnitude) eigenvalues of $\beta\tilde{\Pi}$. When $\omega = 0$, DMSGD achieves the "best" consensus; the upper bound simplifies to $\frac{8\alpha\sqrt{N}\sqrt{M^2+\sigma^2}}{\sqrt{\eta(1-\beta\lambda_2)}(1-\sqrt{\beta\lambda_2})}$. When $\omega \to 1$, we get a worse-case upper bound on consensus error. Further, a more connected graph has a smaller value of λ_2, implying better consensus (which makes intuitive sense).

Nonconvex functions. We summarize the main result on the convergence of DMSGD for nonconvex function in Theorem 1. But first, we give an auxiliary lemma to simplify the proof process for Theorem 1. Throughout the rest of the analysis, $\mathcal{F}^* := \mathcal{F}(\mathbf{x}^*) > -\infty$ is denoted as the minimum of the value sequence $\{\mathcal{F}(\mathbf{x}_k)\}, \forall k \in \mathbb{N}$. Recall the update law (Equation 5). For convenience of analysis, we let $\hat{\mathbf{p}}_k = \beta\tilde{\Pi}(I-\beta\tilde{\Pi})^{-1}(\mathbf{x}_k - \mathbf{x}_{k-1})$ and rewrite the above equality in the following expression:

$$\mathbf{x}_{k+1} + \hat{\mathbf{p}}_{k+1} = \mathbf{x}_k + \hat{\mathbf{p}}_k - \alpha(I-\beta\tilde{\Pi})^{-1}\mathcal{S}(\mathbf{x}_k) \tag{8}$$

(Due to the space limit, we derive Eq. 8 in the extended version of this paper [25].) Let $\hat{\mathbf{z}}_k = \mathbf{x}_k + \hat{\mathbf{p}}_k$ such that the update rule finally becomes:

$$\hat{\mathbf{z}}_{k+1} = \hat{\mathbf{z}}_k - \alpha(I-\beta\tilde{\Pi})^{-1}\mathcal{S}(\mathbf{x}_k) \tag{9}$$

which resembles a regular form of SGD. Before showing the convergence analysis, we present a lemma for characterizing the main theorem.

***Lemma* 1.** Let all assumptions hold. The iterates $\{\hat{\mathbf{z}}_k\}$ generated by Eq. 9 satisfy:

$$\mathbb{E}[\mathcal{F}(\hat{\mathbf{z}}_{k+1}) - \mathcal{F}(\hat{\mathbf{z}}_k)] \leq -A_1\mathbb{E}[\|\nabla\mathcal{F}(\mathbf{x}_k)\|^2] + A_2 \tag{10}$$

where $A_1 = \frac{\alpha}{2(1-\beta\Lambda)} - \frac{L'\alpha^2}{2(1-\beta\Lambda)^2}$, $A_2 = \frac{\alpha^3 L'^2(\beta\Lambda)^2}{(1-\beta\Lambda)^5}(M^2+\sigma^2) + \frac{L'\alpha^2\sigma^2}{2(1-\beta\Lambda)^2}$.

With the above lemma in hand, we are ready to state the main theorem for the nonconvex analysis for DMSGD. Before that, we discuss the choice of step size for the convergence analysis. While constant step size enables algorithms to converge faster, diminishing step size can achieve better accuracy in stochastic optimization. In this context, the step size should satisfy a condition that can guarantee the value sequence $\{\mathcal{F}(\mathbf{x})\}$ to sufficiently descend. According to Lemma 1, α should satisfy $\alpha \leq \frac{1-\beta\Lambda}{L'}$.

Theorem 1. *Let all assumptions hold. Suppose the step size satisfies* $\alpha = \min\{\frac{1-\beta\Lambda}{2L'}, \sqrt{\frac{N}{K}}\}$. *The iterates* $\{\mathbf{x}_k\}$ *generated by Eq. 5 satisfy the following inequality:*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla\mathcal{F}(\mathbf{x}_k)\|^2] \leq \max\left\{\frac{8(\mathcal{F}(\mathbf{x}_1)-\mathcal{F}^*)L'}{K}, \frac{4(1-\beta\Lambda)(\mathcal{F}(\mathbf{x}_1)-\mathcal{F}^*)}{\sqrt{NK}}\right\} + \frac{4NL'^2(\beta\Lambda)^2(M^2+\sigma^2)}{K(1-\beta\Lambda)^4} + \frac{2NL'\sigma^2}{(1-\beta\Lambda)\sqrt{NK}}. \tag{11}$$

Proof. Using the conclusion from Lemma 1, by induction, we can get the desired results. Please refer to the extended version of this paper [25] for more details. □

Theorem 1 shows that with a properly selected constant stepsize, for nonconvex functions, DMSGD can converge to the optimal solution $\mathbf{x}^*$ (which essentially is a stationary point) with a rate of $\mathcal{O}(1/K + 1/\sqrt{NK})$. This matches the best results in [21, 20]. Additionally, the selection of α satisfies the condition that $\alpha \leq \frac{1-\beta\Lambda}{L'}$ such that when $K \geq \frac{NL'^2}{(1-\beta\Lambda)^2}$, Theorem 1 suggests $\mathcal{O}(1/\sqrt{NK})$, which implies the linear speed up for DMSGD. Additional analytical results regarding strongly convex and quasi-convex objective functions are presented in the extended version of this paper [25] due to the space limit.

5. EXPERIMENTAL RESULTS

We now support the utility of our proposed DMSGD algorithm by simulating a distributed environment over a GPU cluster with multiple GPUs, similar to the experiments of [27, 11, 28]. We define a graph topology where each agent in the graph can communicate with another agent with an interaction matrix initialized by the user, ensuring that it is doubly stochastic (in our experiments, we explore a fully connected topology, a ring topology, and a bipartite graph just as in [12]).

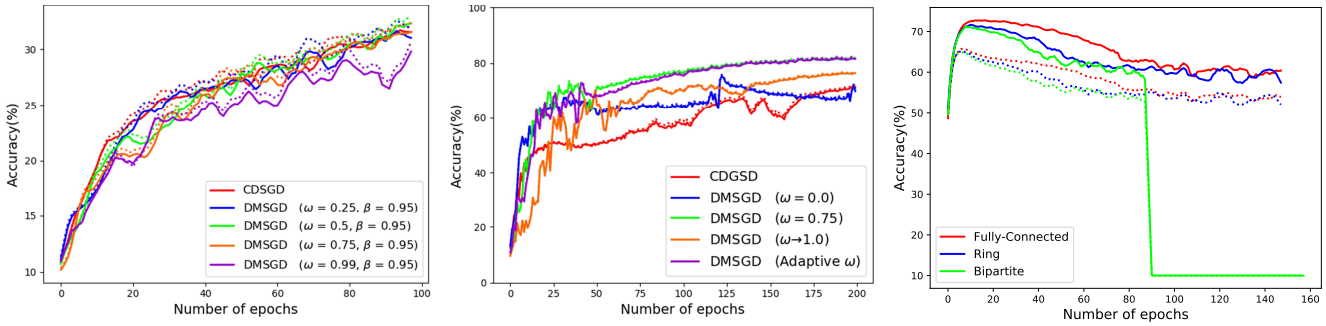

Fig. 1: (a) Performance of our proposed algorithm, DMSGD with different ω values, and its comparison with CDSGD. These performances are with iid data simulation strategy (b) Performance of our proposed algorithm in non-iid data simulation strategy (c) Performance on different topologies for mnist dataset.

We split the given (complete) training dataset among different agents equally, creating two data simulation strategies:

1. **iid:** the dataset is shuffled completely and distributed amongst the agents to simulate an environment where each of the agents has an independently identical draw from the data distribution.
2. **non-iid:** We first segregate the dataset based on the target labels, then we create chunks of data and distribute the chunks with unique target labels to all the agents. If the number of agents is larger than the number of target labels, each agent gets only a chunk of data corresponding to each target label, and if the number of agents is lesser than the number of target labels, each agent gets a set of multiple chunks with unique target labels unavailable with other agents. This strategy simulates an extreme imbalance across different agents and we expect to see significant loss in the performance of decentralized learning algorithms.

In this work, we implement proposed algorithms with both the data simulation strategies. The implementation is using Pytorch.

First, we demonstrate empirical evidence of good consensus using the lesser generalization gap as done by [11, 28]. In Figure 1, the dotted lines denote the performance of agents on test data, which closely follow the solid line (performance on training data) but lag slightly. We attribute to the averaging of several weights, which promotes generalization, as explained in [29, 30]. In [29], authors show that by averaging the weights of the network, they get wider and flat optima that generalize well. We note that the consensus step provides us with similar conditions. Another observation from our experiments is a validation of Remark 1; we see that as ω increases, the generalization gap increases with a weaker consensus bound occurring at $\omega \to 1$ as explained in Remark 1. Therefore, we see that at $\omega = 0.99$, our algorithm does not converge.

Now, we analyze the convergence and performance of the DMSGD algorithm. Due to space constraints, we only present a few anecdotal results here. In Figure 1(a and b), we show the performance of DMSGD with different ω values for CIFAR-10 dataset. All the results shown here are for a sufficiently large Convolutional Neural Network. While we could perform comparisons with the algorithm proposed by [12, 13], it would be unfair as the protocol for communication used by them is different (Push-Sum and Dynamic Model Averaging). Note that we could extend our momentum-accelerated consensus to these models, analysis of the same is beyond the scope of this work. Therefore, as a baseline, we use a non-momentum decentralized algorithm that would have a fair comparison. For this, we compare with CDSGD [11] in this simple experiment. We observe that DMSGD performs with similar performance as the CDSGD algorithm, i.e., without any acceleration. However, while working on a non-iid data simulation strategy, DMSGD performs better than the CDSGD algorithm. We believe that this is a trade-off between consensus and convergence, which [31] explores in detail.

We also note from the results shown and the analysis in the previous section that as $\omega \to 1$, the convergence bounds become weaker. This explains why the performance dies down as a function of ω, e.g. $\omega = 0.5$ performs better than $\omega = 0.75$. However, setting $\omega = 0$ performs very badly for non-iid data. The dynamics of ω with respect to the data distribution is not explored in this work and can be considered as future work.

Finally, we would like to add another result for the performance of our proposed DMSGD algorithm for different communication topologies in Figure 1(c). We consider three communication topologies: (1) Fully connected topology (2) Ring topology (3) Bipartite topology. As the communication topology has sparse communication, the consensus and convergence bounds also become weaker. In Figure 1(c), where we see that the Bipartite graph with very sparse connections performs worse than fully connected graph, which validates the analysis.

6. CONCLUSIONS AND FUTURE WORK

This paper addresses the problem of deep learning in a decentralized setting using momentum accelerated consensus. We establish a consensus-based decentralized learning algorithm using the stochastic heavy ball approach that can assist in finding the optimal solution faster than conventional SGD-style methods. We show that the proposed DMSGD with different choices of momentum terms can achieve linear convergence rate with appropriately chosen step size for strongly-convex, quasi-convex objective functions along with the assumption of smoothness, and convergence to a stationary point for nonconvex objective functions.

Relevant experimental results using benchmark datasets show that the proposed algorithms can achieve better accuracy with sufficient training epochs. While our current research focuses on extensive testing and validation of the proposed framework (especially for large networks), a few directions for future research include an extension to the analysis of Nesterov momentum with nonconvex objective functions, analysis of non-iid data setting and variance reduction strategies for further convergence speed-up techniques in the stochastic setting.

7. REFERENCES

[1] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al., "Large scale distributed deep networks," *Advances in neural information processing systems*, pp. 1223–1231, 2012.

[2] Sixin Zhang, Anna E Choromanska, and Yann LeCun, "Deep learning with elastic averaging sgd," in *Advances in neural information processing systems*, 2015, pp. 685–693.

[3] Peter H Jin, Qiaochu Yuan, Forrest Iandola, and Kurt Keutzer, "How to scale distributed deep learning?," *arXiv preprint arXiv:1611.04581*, 2016.

[4] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al., "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 1508–1518.

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

[7] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[8] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar, "signsgd: Compressed optimisation for non-convex problems," *arXiv preprint arXiv:1802.04434*, 2018.

[9] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[10] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5336–5346.

[11] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5906–5916.

[12] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat, "Stochastic gradient push for distributed deep learning," *arXiv preprint arXiv:1811.10792*, 2018.

[13] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 393–409.

[14] David Luengo, Luca Martino, Víctor Elvira, and Mónica Bugallo, "Efficient linear fusion of partial estimators," *Digital Signal Processing*, vol. 78, pp. 265–283, 2018.

[15] Luca Martino, Jorge Plata-Chaves, and Francisco Louzada, "A monte carlo scheme for node-specific inference over wireless sensor networks," in *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2016, pp. 1–5.

[16] Xiangyu Wang and David B Dunson, "Parallelizing mcmc via weierstrass sampler," *arXiv preprint arXiv:1312.4605*, 2013.

[17] Euhanna Ghadimi, Iman Shames, and Mikael Johansson, "Multi-step gradient methods for networked optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5417–5429, 2013.

[18] Nicolas Loizou and Peter Richtárik, "Linearly convergent stochastic heavy ball method for minimizing generalization error," *arXiv preprint arXiv:1710.10737*, 2017.

[19] Tianbao Yang, Qihang Lin, and Zhe Li, "Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization," *arXiv preprint arXiv:1604.03257*, 2016.

[20] Hao Yu, Rong Jin, and Sen Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," *arXiv preprint arXiv:1905.03817*, 2019.

[21] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat, "Slowmo: Improving communication-efficient distributed sgd with slow momentum," *arXiv preprint arXiv:1910.00643*, 2019.

[22] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.

[23] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[24] Boris T Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

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

[26] Albert S Berahas, Raghu Bollapragada, Nitish Shirish Keskar, and Ermin Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3141–3155, 2018.

[27] Qihang Lin, Zhaosong Lu, and Lin Xiao, "An accelerated proximal coordinate gradient method," in *Advances in Neural Information Processing Systems*, 2014, pp. 3059–3067.

[28] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu, "D2: Decentralized training over decentralized data," *arXiv preprint arXiv:1803.07068*, 2018.

[29] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[30] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[31] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang, "Communication efficient decentralized training with multiple local updates," *arXiv preprint arXiv:1910.09126*, 2019.