Cross-Gradient Aggregation for Decentralized Learning from Non-IID Data

Yasaman Esfandiari [1] **Sin Yong Tan** [1] **Zhanhong Jiang** [2] **Aditya Balu** [1] **Ethan Herron** [1] **Chinmay Hegde** [3] **Soumik Sarkar** [1]

Abstract

Decentralized learning enables a group of collaborative agents to learn models using a distributed dataset without the need for a central parameter server. Recently, decentralized learning algorithms have demonstrated state-of-the-art results on benchmark data sets, comparable with centralized algorithms. However, the key assumption to achieve competitive performance is that the data is independently and identically distributed (IID) among the agents which, in real-life applications, is often not applicable. Inspired by ideas from continual learning, we propose *Cross-Gradient Aggregation* (*CGA*), a novel decentralized learning algorithm where (i) each agent aggregates *cross*-gradient information, i.e., derivatives of its model with respect to its neighbors' datasets, and (ii) updates its model using a projected gradient based on quadratic programming (QP). We theoretically analyze the convergence characteristics of *CGA* and demonstrate its efficiency on non-IID data distributions sampled from the MNIST and CIFAR-10 datasets. Our empirical comparisons show superior learning performance of *CGA* over existing state-of-the-art decentralized learning algorithms, as well as maintaining the improved performance under information compression to reduce peer-to-peer communication overhead. The code is available here on GitHub.

1. Introduction

Distributed machine learning refers to a class of algorithms that are focused on learning from data distributed among multiple agents. Approaches to design distributed deep learning algorithms include: centralized learning (McMahan et al., 2017; Kairouz et al., 2019), decentralized learning (Lian et al., 2017; Nedić et al., 2018), gradient compression (Seide et al., 2014; Alistarh et al., 2018) and coordinate updates (Richtárik and Takáč, 2016; Nesterov, 2012). In centralized learning, a central parameter server collects, processes, and sends processed information back to the agents (Konečnỳ et al., 2016). As a popular approach for centralized learning, Federated Learning (FL) leverages a central parameter server and learns from dispersed datasets that are private to the agents. Another approach is Federated Averaging (McMahan et al., 2017) where agents avoid communicating with the server at each learning iteration and significantly decrease the communication cost.

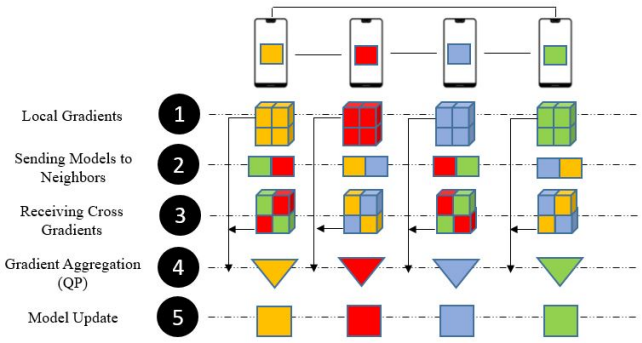


Figure 1. ***Algorithm overview****. In the proposed CGA algorithm (1) each agent computes gradients of model parameters on its own data set; (2) each agent sends its model parameters to its neighbors; (3) each agent computes the gradients of its neighbors' models on its own data set and sends the cross gradients back to the respective neighbors; (4) cross gradients and local gradients are projected into an aggregated gradient (using Quadratic Programming); which is then used to (5) update the model parameter.*

Decentralized learning: While having a central parameter server is acceptable for data center applications, in certain use cases (such as learning over a wide-area distributed sensor network), continuous communication with a central parameter server is often not feasible (Haghighat et al., 2020). To address this concern, several decentralized learning algorithms have been proposed, where agents only interact with their neighbors without a central parameter server.

Recent advances in decentralized learning involve gossip

[1]Department of Mechanical Engineering, Iowa State University, Ames, Iowa, USA [2]Johnson Controls, Milwaukee, Wisconsin, USA [3]Computer Science and Engineering Department, New York University, New York City, New York, USA. Correspondence to: Soumik Sarkar <soumiks@iastate.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Table 1. Comparison between different decentralized learning approaches. Rate: convergence rate for the optimization algorithm, Comm.: Communication overhead per mini-batch, Bo. Gr. Var.: Bounded gradient variances and variations as an assumption, Bo. Sec. Mom.: Bounded second moment of the gradient as an assumption, m_s: model size for the local agent, N_b: total number of non-zero elements in Π (total number of communications per mini-batch), γ: auxiliary costs due to forward and backward pass of the neural network, b: floating point precision of arithmetic computations (e.g. 64)

Method	Rate	Comm.	Bo. Gr. Var.	Bo. Sec. Mom.
DPSGD	$\mathcal{O}(\frac{1}{K}+\frac{1}{\sqrt{NK}})$	$\mathcal{O}(m_s N_b+\gamma)$	Yes	No
SGP	$\mathcal{O}(\frac{1}{K}+\frac{1}{K^{1.5}}+\frac{1}{\sqrt{NK}})$	$\mathcal{O}(m_s N_b+\gamma)$	Yes	No
SwarmSGD	$\mathcal{O}(\frac{1}{\sqrt{K}})$	$\mathcal{O}(m_s \frac{N_b}{2}+\gamma)$	No	Yes
CGA (ours)	$\mathcal{O}(\frac{1}{K}+\frac{1}{K^{1.5}}+\frac{1}{\sqrt{NK}}+\frac{1}{K^2})$	$\mathcal{O}(2m_s N_b+\gamma)$	Yes	No

* The communication overhead per mini-batch for *CompCGA* method is $\mathcal{O}(\frac{2m_s N_b}{b}+\gamma)$

averaging algorithms (Boyd et al., 2006; Xiao and Boyd, 2004; Kempe et al., 2003). Combining SGD with gossip averaging, Lian et al. (2017) shows analytically that decentralized parallel SGD (DPSGD) has far less communication overhead than its central counterpart (Dekel et al., 2012). Along the same line of work, Scaman et al. (2018) introduced a multi-step primal-dual algorithm while Yu et al. (2019) and Balu et al. (2021) introduced the momentum version of DPSGD. Tang et al. (2019) proposed DeepSqueeze, error-compensated compression is used in decentralized learning to achieve the same convergence rate as the one of centralized algorithms. Koloskova et al. (2019) utilized compression strategies to propose CHOCO-SGD algorithm which learns from agents connected in varying topologies. Similarly with the aid of compression, Lu and De Sa (2020) and Vogels et al. (2020) introduced compression-based algorithms that improves the memory usage and running time of existing decentralized learning approaches. Assran et al. (2019) proposed the SGP algorithm which converges at the same sub-linear rate as SGD and achieves high accuracy on benchmark datasets. Additionally, Koloskova et al. (2020) presented a unifying framework for decentralized SGD analysis and provided the best convergence guarantees. More recently, SwarmSGD was proposed by Nadiradze et al. (2019) which leverages random interactions between participating agents in a graph to achieve consensus. In a recent work, Arjevani et al. (2020) proposes using AGD to achieve optimal convergence rate both in theory and practice. Jiang et al. (2018) propose multiple consensus and optimality rounds and the tradeoff between the consensus and optimality in decentralized learning.

Handling non-IID data: It is well known that decentralized learning algorithms can achieve comparable performance with its centralized counterpart under the so-called IID (independently and identically distributed) assumption. This refers to the situation where the training data is distributed in a uniformly random manner across all the agents. However, in real life applications, such an assumption is difficult to satisfy. Considering centralized learning literature, Li et al. (2018) proposed a variant of FL by adding a penalty term in the local objective function in FedProx algorithm. They further showed that their algorithm achieves higher accuracy when learning from non-IID data compared to FedAvg. Motivated by life-long learning (Shoham et al., 2019), FedCurv was proposed by adding a penalty term to the local loss function, with respect to Fisher information matrix. In another research study, FedAvg-EMD (Zhao et al., 2018) utilized the earth mover's distance (EMD) as a metric to quantify the distance between the data distribution on each client and the population distribution, which was perceived as the root cause of problems arising in the non-IID scenario. Li et al. (2019b) showed the limitations with FedAvg on non-IID data analytically. Also, FedNova was proposed in which they use a normalized gradient in the update law of FedAvg after they show that the standard averaging of client models after heterogeneous local updates results in convergence to a stationary point (Wang et al., 2020). Similar to the case of decentralized learning, compression techniques (Sattler et al., 2019; Rothchild et al., 2020), momentum variant of algorithms (Wang et al., 2019; Li et al., 2019a), the use of adaptive gradients (Tong et al., 2020), and use of controllers in agent's and server's models (Karimireddy et al., 2019a) are also used in centralized learning for coping with non-IID data. Hsieh et al. (2019) proposes a solution for learning from non-IID data by Estimating the degree of deviation from IID by moving the model from one data partition to another. They then Evaluate the accuracy on the other data set and calculate the accuracy loss, and based on this measure, SkewScout controls the communication tightness by automatically tuning the hyper-parameters of the decentralized learning algorithm. In their experimental results, they consider until 80% non-IID data whereas in our approach our dataset is partitioned in a fully non-IID was based on the classes.

Although the above *centralized* approaches can handle departure from IID assumption, there still exists a gap in *decentralized* learning and several approaches fail under significant non-IID distribution of data among the agents (Hsieh

et al., 2019; Jiang et al., 2017).

Contributions: To overcome the issue of handling non-IID data distributions in a decentralized learning setting, we propose the *Cross-Gradient Aggregation* (*CGA*) algorithm in this paper. We show its effectiveness in learning (deep) models in a decentralized manner from both IID and non-IID data distributions. Inspired by continual learning literature (Lopez-Paz and Ranzato, 2017), we devise an algorithm which in each step of training, collects the gradient information of each agent's model on all its neighbors' datasets and projects them into a single gradient which is then used to update the model. We use quadratic programming (QP) to obtain such a projected gradient. We provide an illustration of this algorithm in Figure 1.

The communication cost for our proposed algorithm is higher than the other state-of-the-art algorithms due to additional cost for two-way communication of the model parameters to, and the gradient information from, the neighbors. A comparison of the communication costs is provided in Table 1. Therefore, we also propose a compressed variant (*CompCGA*) to reduce the communication cost. Finally we validate the performance of our algorithms on MNIST and CIFAR-10 with different graph typologies. Our code is publicly available on GitHub[1]. We then compare the effectiveness of our algorithm with *SwarmSGD* (Nadiradze et al., 2019), *SGP* (Assran et al., 2019), and *DPSGD* (Lian et al., 2017) and show that we can achieve higher accuracy in learning from non-IID data compared to the state-of-the-art decentralized learning approaches. Note that the goal here is to provide comparison between different decentralized learning algorithms; therefore, studies proposing novel compression schemes (Tang et al., 2019; Koloskova et al., 2019; Lu and De Sa, 2020; Vogels et al., 2020) are excluded from our comparison.

In summary, (i) we introduce the concept of *cross gradients* to develop a novel decentralized learning algorithm (*CGA*) that enables learning from both IID and non-IID data distributions, (ii) to reduce the higher communication costs of *CGA*, we propose a compressed variant, *CompCGA* that maintains a reasonably good performance in both IID and non-IID settings, (iii) we provide a detail convergence analysis of our proposed algorithm and show that we have similar convergence rates to the state-of-the-art decentralized learning approaches as summarized in Table 1, (iv) we demonstrate the efficacy of our proposed algorithms on benchmark datasets and compare performance with state-of-the-art decentralized learning approaches.

[1]https://github.com/yasesf93/CrossGradientAggregation

2. Cross-Gradient Aggregation

Let us first present a general problem formulation for decentralization deep learning, and then use it to motivate the Cross-Gradient Aggregation (*CGA*) algorithmic framework.

2.1. Problem Formulation

Very broadly, decentralized learning involves N agents collaboratively solving the empirical risk minimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathcal{F}(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}), \tag{1}$$

where $f_i(\mathbf{x}) := \mathbb{E}_{\zeta_i\sim\mathcal{D}_i}[F_i(\mathbf{x};\zeta_i)]$ denotes a loss function defined in terms of dataset $\mathcal{D}_i$ that is private to agent $i \in [N]$. The agents are assumed to be communication-constrained and can only exchange information with their neighbors (where neighborliness is defined according to a weighted undirected graph with edge set $\mathbb{C}$ and adjacency matrix Π). Note that the adjacency matrix Π is a doubly stochastic matrix constructed using the edge set of the graph, $\mathbb{C}$. For $(i,j) \notin \mathbb{C}$, we assign zero link weights (i.e., $\pi_{ij} = 0$), and if $(i,j) \in \mathbb{C}$, the link weights are assigned such that the Π is stochastic and symmetric, e.g., for a ring topology, $\pi_{ij} = \frac{1}{3}$ if $j \in \{i-1, i, i+1\}$. The goal is for the agents to come up with a consensus set of model parameters $\mathbf{x}$ (although during training each agent operates on its own copy of $\mathbf{x}$.)

Usual approaches in decentralized learning involve each agent alternating between updating the local copies of their parameters using gradient information from their private datasets, and exchanging parameters with its neighbors. We depart from this usual path by first introducing two key concepts.

Definition 1. *For agent j, consider the dataset $\mathcal{D}_j$, the differentiable objective function f_j, and the model parameter copy $\mathbf{x}^j$. The self-gradient is defined as:*

$$\mathbf{g}^{jj} := \nabla_{\mathbf{x}} f_j(\mathcal{D}_j; \mathbf{x}^j)\,. \tag{2}$$

Definition 2. *For a pair of agents j, l, consider the dataset $\mathcal{D}_l$, the differentiable objective function f_l, and the model parameter copy $\mathbf{x}^j$. The cross-gradient is defined as:*

$$\mathbf{g}^{jl} := \nabla_{\mathbf{x}} f_l(\mathcal{D}_l; \mathbf{x}^j)\,. \tag{3}$$

In words, the cross-gradient is calculated by evaluating the gradient of the loss function private to agent l at the parameters of agent j. Both the self-gradient $\mathbf{g}^{jj}$ and the cross-gradient $\mathbf{g}^{jl}$ immediately lend themselves to their stochastic counterparts (implemented by simply mini-batching the private datasets); in the rest of the paper, we will operate under this setting.

Algorithm 1 Cross-Gradient Aggregation (*CGA*)

Initialize: $\mathcal{D}_j, \mathbf{x}_0^j, \mathbf{v}_0^j, (j = 1, 2, \ldots, N), \alpha, \beta, K$, a $\mathtt{QP}$ solver
for $k = 1 : K$ **do**
  for $j = 1 : N$ **do**
    Randomly shuffle the data subset $\mathcal{D}_j$
    Compute $\mathbf{g}_k^{jj}$
    $\mathbf{G}^j = \{\}$
    for *each agent* l *s.t.* $(j, l) \in \mathbb{C}$ **do**
      Compute $\mathbf{g}_k^{jl}$
      $\mathbf{G}_k^j \leftarrow \mathbf{G}_k^j \cup \mathbf{g}_k^{jl}$
    end
    $\mathbf{w}_k^j = \sum_l \pi_{jl} \mathbf{x}_{k-1}^l$
    $\tilde{\mathbf{g}}_k^j \leftarrow \mathtt{QP}(\mathbf{g}_k^{jj}, \mathbf{G}_k^j)$
    $\mathbf{v}_k^j = \beta \mathbf{v}_{k-1}^j - \alpha \tilde{\mathbf{g}}_k^j$
    $\mathbf{x}_k^j = \mathbf{w}_k^j + \mathbf{v}_k^j$
  end
end

2.2. The CGA Algorithm

We now propose the *CGA* algorithm for decentralized deep learning. Figure 1 provides a visual overview of the method. Recall that $\mathbf{x}^j$ is the model parameter copy for each agent j which is initialized by training with $\mathcal{D}_j$. Pick the number of iterations K, step-size α, and the momentum coefficient β as user-defined inputs.

In the k^{th} iteration of *CGA*, each agent $j \in [N]$ calculates its self-gradient $\mathbf{g}_k^{jj}$. Then, agent j's model parameters are transmitted to all other agents (l) in its neighborhood, and the respective cross-gradients are calculated and transmitted back to agent j and stacked up in a matrix $\mathbf{G}_k^j$. Then $\mathbf{G}_k^j$ and $\mathbf{g}_k^{jj}$ are used to perform a quadratic programming (QP) projection step, which we discuss in detail below. To accelerate convergence, a momentum-like adjustment term is also incorporated to obtain the final update law.

The form of the algorithm is similar to momentum-accelerated consensus SGD (Jiang et al., 2017). The key difference in Algorithm 1 when compared to existing gradient-based learning methods is the QP projection step. We observe that the local gradient $\tilde{\mathbf{g}}^j$ is obtained via a nonlinear projection, instead of just the self-gradient $\mathbf{g}^{jj}$ (as is done in standard momentum-SGD), or a linear averaging of self-gradients in the neighborhood $\mathbb{C}$ (as is done in standard decentralized learning methods).

The motivation for this difference stems from the nature of the cross-gradients $\mathbf{g}_k^{jl}$. In the IID case, these should statistically resemble the self-gradient $\mathbf{g}_k^{jj}$, and hence standard momentum averaging would succeed. However, with non-IID data partitioning, the differences between the cross-gradients in different agents becomes so significant and consensus may be difficult to achieve, leading to overall poor convergence properties. Therefore, in the non-IID case we need an alternative approach.

We leverage the following intuition, borrowed from (Lopez-Paz and Ranzato, 2017). We seek a descent direction that is close to $\mathbf{g}_k^{ll}$ and *simultaneously* is positively correlated with all the cross-gradients. This can be modeled via a QP projection, posed in primal form as follows:

$$\begin{aligned} &\min_{\mathbf{z}} \ \frac{1}{2}\mathbf{z}^\top \mathbf{z} - \mathbf{g}^\top \mathbf{z} + \frac{1}{2}\mathbf{g}^\top \mathbf{g} \qquad (4a) \\ &\text{s.t.} \ \ \mathbf{G}\mathbf{z} \geq 0 \end{aligned}$$

where $\mathbf{g} := \mathbf{g}_k^{jj}$ and $\mathbf{G} := (\mathbf{g}^{jl}) \ \ \forall (j, l) \in \mathbb{C}$. The dual formulation of the above QP can be posed as:

$$\begin{aligned} &\min_{\mathbf{u}} \ \frac{1}{2}\mathbf{u}^\top \mathbf{G}\mathbf{G}^\top \mathbf{u} + \mathbf{g}^\top \mathbf{G}^\top \mathbf{u} \qquad (5a) \\ &\text{s.t.} \ \ \mathbf{u} \geq 0 \end{aligned}$$

which is more efficient from a computational standpoint. Once we solve for the optimal dual variable $\mathbf{u}^*$, we can recover the optimal projection direction $\mathbf{g}^*$ using the relation $\mathbf{g}^* = \mathbf{G}^\top \mathbf{u}^* + \mathbf{g}$.

2.3. The Compressed *CGA* Algorithm

The *CGA* algorithm requires multiple exchanges of model parameters and gradients between neighbor agents in each iteration, which can be a burden particularly in communication-constrained environments. To reduce the communication bandwidth, we propose adding a compression layer on top of the *CGA* framework. For that purpose, we use Error Feedback SGD (*EF-SGD*) (Karimireddy et al., 2019b) to compress gradients. The resulting algorithm is same as Algorithm 1; except that instead of regular self- and cross-gradients, a scaled *signed* gradient is calculated, the error between the compressed and non-compressed gradients will be computed (e_k^{ij} in the algorithm), and this error will be added as a penalty term to the gradients in the next step. The resulting algorithm is shown in Algorithm 2. In the pseudo code provided there, the quantity d corresponds to the dimension of the computed gradients for each agent.

3. Convergence Analysis for *CGA*

We now present a theoretical analysis of our proposed *CGA* approach. It should be noted that the communication among the agents is assumed to be synchronous in the following analysis. Let us begin with a definition of *smoothness*.

Definition 3. *A function $\mathcal{F}(\cdot)$ is L-smooth if $\forall \mathbf{x}, \mathbf{y}$:*

$$\mathcal{F}(\mathbf{x}) \leq \mathcal{F}(\mathbf{y}) + \nabla \mathcal{F}(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2. \quad (6)$$

Algorithm 2 Compressed Cross-Gradient Aggregation (*CompCGA*)

Initialize: $\mathcal{D}_j, \mathbf{e}_0^j, \mathbf{x}_0^j, \mathbf{v}_0^j, (j = 1, \ldots, N), \alpha, \beta, K$, a $\mathtt{QP}$ solver
for $k = 1 : K$ **do**
  for $j = 1 : N$ **do**
    Randomly shuffle the data subset $\mathcal{D}_j$
    Compute $\mathbf{g}_k^{jj}$
    $\mathbf{p}_k^{jj} = \mathbf{g}_k^{jj} + \mathbf{e}_k^{jj}$
    $\delta_k^{jj} = (\|\mathbf{p}_k^{jj}\|_1/d) sgn(\mathbf{p}_k^{jj})$
    $\mathbf{G}^j = \{\}$
    for *each agent l, s.t.* $(j, l) \in \mathbb{C}$ **do**
      Compute $\mathbf{g}_k^{jl}$
      $\mathbf{p}_k^{jl} = \mathbf{g}_k^{jl} + \mathbf{e}_k^{jl}$
      $\delta_k^{jl} = (\|\mathbf{p}_k^{jl}\|_1/d) sgn(\mathbf{p}_k^{jl})$
      $\mathbf{e}_k^{jl} = \mathbf{p}_k^{jl} - \delta_k^{jl}$
      $\mathbf{G}^j \leftarrow \mathbf{G}^j \cup \delta_k^{jl}$
    end
    $\mathbf{w}_k^j = \sum_l \pi_{jl} \mathbf{x}_{k-1}^l$
    $\tilde{\mathbf{g}}^j \leftarrow \mathtt{QP}(\delta_k^{jj}, \mathbf{G}^j)$
    $\mathbf{v}_k^j = \beta \mathbf{v}_{k-1}^j - \alpha \tilde{\mathbf{g}}^j$
    $\mathbf{x}_k^j = \mathbf{w}_k^j + \mathbf{v}_k^j$
    $\mathbf{e}_k^{jj} = \mathbf{p}_k^{jj} - \delta_k^{jj}$
  end
end

In order to analyze the convergence of decentralized learning algorithms, the following assumptions are standard.

Assumption 1. *Each function $f_i(\mathbf{x})$ is L-smooth.*

Assumption 2. *There exist $\sigma > 0$ and $\delta > 0$ such that*

$$\mathbb{E}_{\zeta \sim \mathcal{D}_i}[\|\nabla F_i(\mathbf{x}; \zeta) - \nabla f_i(\mathbf{x})\|] \leq \sigma^2, \tag{7}$$

and that

$$\frac{1}{N} \sum_{i=1}^{N} \|\nabla f_i(\mathbf{x}) - \nabla \mathcal{F}(\mathbf{x})\|^2 \leq \delta^2. \tag{8}$$

Assumption 3. *Define $\mathbf{g}^i = \nabla F_i(\mathbf{x}; \zeta)$. Then, there exists $\epsilon > 0$ such that*

$$\mathbb{E}_{\zeta \sim \mathcal{D}_i}[\|\tilde{\mathbf{g}}^i - \mathbf{g}^i\|^2] \leq \epsilon^2. \tag{9}$$

Assumption 1 implies that $\mathcal{F}(\mathbf{x})$ is L-smooth. Assumption 2 assumes bounded variances due to non-IID-ness. Equation 7 bounds the variance within the same agent ("intra-variance") while Equation 8 bounds the variance among different agents ("inter-variance").

Assumption 3 is necessitated by our adoption of the QP projection step. Intuitively, if the local optimization problem is meaningful, then this assumption holds. In this assumption, the value of ϵ is governed by the difference between the data distributions possessed by each agent. Note that thus far, Eq. 8 has been used to study the effect of non-IID data in most analyses of decentralized learning; previous methods operate upon $\mathbf{g}^i$. In our work, we combine both Eq. 8 and Eq. 9 to mathematically show convergence.

We next impose another assumption on the graph that serves to characterize consensus.

Assumption 4. *The mixing matrix $\mathbf{\Pi} \in \mathbb{R}^{N \times N}$ is a doubly stochastic matrix with $\lambda_1(\mathbf{\Pi}) = 1$ and*

$$max\{|\lambda_2(\mathbf{\Pi})|, |\lambda_N(\mathbf{\Pi})|\} \leq \sqrt{\rho} < 1, \tag{10}$$

where $\lambda_i(\mathbf{\Pi})$ is the ith-largest eigenvalue of $\mathbf{\Pi}$ and ρ is a constant.

3.1. Theoretical Results

We now present our theoretical characterization of *CGA*. We focus only on the case of non-convex objective functions. All detailed proofs are presented in the Appendix, and follow from basic algebra and sequence convergence theory. Below, i indicates the agent index; the average of all agent model copies is represented by $\bar{\mathbf{x}}$; throughout the analysis, we assume that the objective function value is bounded below by $\mathcal{F}^*$. We also denote $a_n = \mathcal{O}(b_n)$ if $a_n \leq c\, b_n$ for some constant $c > 0$.

We first present a lemma showing that *CGA* achieves consensus among the different agents, and then prove our main theorem indicating convergence of the algorithm.

Lemma 1. *Let Assumptions 1-4 hold. Define $\{\bar{\mathbf{x}}_k\}, \forall k \geq 0$ as the agent average sequence obtained by the iterations of CGA. If $\beta \in [0, 1)$ is the momentum coefficient, then for all $K \geq 1$, we have:*

$$\sum_{k=0}^{K-1} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}\left[\left\|\bar{\mathbf{x}}_k - \mathbf{x}_k^i\right\|^2\right] \leq \frac{2\alpha^2}{(1-\beta)^2}\left(\frac{\epsilon^2}{1-\rho} + \frac{3\sigma^2}{(1-\sqrt{\rho})^2} + \frac{3\delta^2}{(1-\sqrt{\rho})^2}\right)K + \frac{6\alpha^2}{(1-\beta)^2(1-\sqrt{\rho})} \sum_{k=0}^{K-1} \mathbb{E}[\|\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{x}_k^i)\|^2]. \tag{11}$$

A complete proof can be found in the *Supplementary Section* A.1. From Lemma 1, we can observe that the evolution of the deviation of the model copies from their average can be attributed to two terms. The first is the following constant:

$$\frac{2\alpha^2}{(1-\beta)^2}\Bigg(\underbrace{\frac{\epsilon^2}{1-\rho}}_{I} + \underbrace{\frac{3\sigma^2}{(1-\sqrt{\rho})^2}}_{II} + \underbrace{\frac{3\delta^2}{(1-\sqrt{\rho})^2}}_{III}\Bigg)K,$$

where (I) is controlled by Assumption 3 (which also implies how well the local QP is solved), (II) is related to sampling variance, and (III) indicates the gradient variations (determined by the data distributions). Additionally, the step size and momentum coefficient can be tuned to reduce the negative impact of these variance coefficients. The second is the following term:

$$\frac{6\alpha^2}{(1-\beta)^2(1-\sqrt{\rho})}\underbrace{\sum_{k=0}^{K-1}\mathbb{E}[\|\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{x}_k^i)\|^2]}_{IV},$$

where (IV) is the summation of the squared norms of average gradients, the effect of which can be controlled by leveraging the step size and momentum coefficient.

Lemma 1 also aligns with the well-known result phenomenon in decentralized learning that the consensus error is inversely proportional to the spectral gap of the graph mixing matrix. Using the above lemma, we obtain the following main result.

Theorem 1. *Let Assumptions 1-4 hold. Suppose that the step size α satisfies the following relationships:*

$$\begin{cases} 0 < \alpha \leq \frac{\beta L}{(1-\beta)^2} \\ 1 - \frac{6\alpha^2 L^2}{(1-\beta)(1-\sqrt{\rho})^2} - \frac{4L\alpha}{(1-\beta)^2} \geq 0. \end{cases} \tag{12}$$

For all $K \geq 1$, we have

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla\mathcal{F}\left(\bar{\mathbf{x}}_k\right)\right\|^2\right] \leq \frac{1}{C_1 K}\left(\mathcal{F}\left(\bar{\mathbf{x}}_0\right)-\mathcal{F}^*\right)+\left(2C_2+C_3\frac{\alpha^2\beta}{(1-\beta)^4}+C_4+C_5\frac{2\alpha^2}{(1-\beta)^2(1-\rho)}\right)\epsilon^2+\left(\frac{2}{N}(C_2+C_3\frac{\alpha^2\beta}{(1-\beta)^4})+C_5\frac{6\alpha^2}{(1-\beta)^2(1-\sqrt{p})^2}\right)\sigma^2+C_5\frac{6\alpha^2}{(1-\beta)^2(1-\sqrt{\rho})^2}\delta^2, \tag{13}$$

where $C_1 = \frac{\alpha}{2(1-\beta)} - \frac{(1-\beta)\alpha^2}{2\beta L}$, $C_2 = \left(\frac{\beta L\alpha^2}{2(1-\beta)^3}+\frac{\alpha^2 L}{(1-\beta)^2}\right)/C_1$, $C_3 = \frac{(1-\beta)L}{2\beta}/C_1$, $C_4 = \frac{\beta L}{2(1-\beta)^3}/C_1$, $C_5 = \frac{\alpha L^2}{2(1-\beta)}/C_1$.

A complete proof of Theorem 1 is discussed in the *Supplementary Section* A.2.

Theorem 1 shows that the average gradient magnitude achieved by the consensus estimates is upper-bounded by the difference between initial objective function value and the optimal value, as well as how well the local QP is solved, the sampling variance, and the non-IID-ness. The coefficients before these constants are determined by α, β, and L; judicious selection of α and β can be performed to reduce the error bound. Additionally, the step size is required to satisfy two conditions as listed in the above theorem statement. The second condition can be solved to get another upper bound, denoted by α^* (which will be shown in the Appendix section). Hence, if we choose $0 < \alpha \leq \min\{\frac{\beta L}{(1-\beta)^2}, \alpha^*\}$, the last inequality naturally holds. We next present a corollary to explicitly show the convergence rate of *CGA*.

Corollary 1. *Suppose that the step size satisfies $\alpha = \mathcal{O}(\frac{\sqrt{N}}{\sqrt{K}})$ and that $\epsilon = \mathcal{O}(\frac{1}{\sqrt{K}})$. For a sufficiently large $K \geq \max\{\frac{144NL^2}{r^2}, \frac{N}{\beta^2 L^2}\}, r = (1-\sqrt{\rho})\sqrt{16(1-\sqrt{\rho})^2+24(1-\beta)^3}-4(1-\sqrt{\rho})^2$, we have, for some constant C > 0,*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla\mathcal{F}\left(\bar{\mathbf{x}}_k\right)\right\|^2\right] \leq C\left(\frac{1}{\sqrt{NK}}+\frac{1}{K}+\frac{1}{K^{1.5}}+\frac{1}{K^2}\right). \tag{14}$$

An immediate observation is that when K is sufficiently large, the term $\mathcal{O}(\frac{1}{\sqrt{NK}})$ will dominate the convergence rate such that the *linear speed up* can be achieved, if increasing the number of agents N. This convergence rate matches the well-known best result in decentralized SGD algorithms in literature.

Analysis of *CompCGA*. We now provide some qualitative arguments to facilitate the understanding of *CompCGA*. Though we have not directly established its convergence rates, we can presumably extend the analysis of *CGA* to this setting. Observe that the core update laws for $\mathbf{x}$ are the same for *CompCGA* as in *CGA*, but equipped with gradient compression. Moreover, for our theoretical analysis presented for *CGA*, the specific way in which $\tilde{\mathbf{g}}$ is calculated does not play a role, and additional compression can perhaps be modeled by changing the variation constants. Therefore, we hypothesize that *CompCGA* also exhibits a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{NK}})$. This is also evidently seen from our empirical studies, which we present next.

4. Experimental Results

In this section, we analyze the performance of *CGA* algorithm empirically. We compare the effectiveness of our algorithms with other baseline decentralized algorithms such as *SwarmSGD* (Nadiradze et al., 2019), *SGP* (Assran et al., 2019), and the momentum variant of *DPSGD* (Lian et al., 2017) (*DPMSGD*).

Setup. We present the empirical studies on CIFAR-10 and MNIST datasets (MNIST results can be found in the *Supplementary Section A.6*). To explore the algorithm performance under different situations, the experiments are performed

with 5, 10 , and 40 agents. Here, we consider an extreme form of non-IID-ness by assigning different classes of data to each agent. For example, when there are 5 agents, each agent has the data for 2 distinct classes, and similarly when there are 10 agents, each agent has the data for 1 distinct class. When the number of agents are more than the number of classes, each class is divided into a sufficient number of subsets of samples and agents are randomly assigned distinct subsets. We use a deep convolutional neural network (CNN) model (with 2 convolutional layers with 32 filters each followed by a max pooling layer, then 2 more convolutional layers with 64 filters each followed by another max pooling layer and a dense layer with 512 units, ReLU activation is used in convolutional layers) for our validation experiments. Additionally, We use a VGG11 (Simonyan and Zisserman, 2014) model for CIFAR-10 (Detailed CIFAR-10 results can be found in the *Supplementary Section* A.5). A mini-batch size of 128 is used, the initial step-size is set to 0.01 for CIFAR-10, and step size is decayed with constant 0.981. The stopping criterion is a fixed number of epochs and the momentum parameter (β) is set to be 0.98. The consensus model is then used to be evaluated on the local test sets and the average accuracy is reported.

The experiments are performed on a large high-performance computing cluster with a total of 192 GPUs distributed over 24 nodes. Each node in the cluster is made of 2 Intel Xeon Gold 6248 CPUs with each 20 cores and 8 Tesla V100 32GB SXM2 GPUs. An experiment with 40 agents on a VGG11 model for CIFAR10 dataset takes about 55 seconds per epoch for execution. The code for performing the experiments is publicly available[2].

4.1. *CGA* convergence characteristics

We start by analyzing the performance of *CGA* algorithm on CIFAR-10. Figure 2 shows the convergence characteristics of our proposed algorithm via training loss versus epochs. Figure 2(a) shows the convergence characteristics of *CGA* for IID data distributions for different communication graph topologies. While the fully connected graph represents a dense topology, the ring and bipartite graphs represent relatively much sparser topologies.

We observe that the convergence behavior induced by the training loss remain similar across the different graph topologies, though at the final stage of training, the ring and bipartite networks moderately outperform the fully connected one. This can be attributed to more communication occurring for the fully connected case. The phenomenon of faster convergence with sparser graph topology is an observation that have been made by earlier research works in Federated Learning (McMahan et al., 2017) by reducing the client fraction which makes the mixing matrix sparser and decentralized learning (Jiang et al., 2017). However, as Figure 6(a) in the *Supplementary Section* A.5 shows, we observe that by training for more number of epochs, training losses associated with all graph topologies converge to similar values.

[2]https://github.com/yasesf93/CrossGradientAggregation

Table 2. Testing accuracy comparison for CIFAR10 with IID data distribution using CNN model architecture

Model	Fully-connected	Ring	Bipartite
DPMSGD	68.8% (5)	67.7% (5)	67.7% (5)
	68.1% (10)	67.7% (10)	67.3% (10)
	67.6% (40)	66.8% (40)	57.1% (40)
SGP	66.6% (5)	66.3% (5)	66.3% (5)
	59.3% (10)	59.2% (10)	58.4% (10)
	46.3% (40)	46.2% (40)	46.3% (40)
SwarmSGD	**70.6%** (5)	**70.7%** (5)	**70.7%** (5)
	68.3% (10)	65.4% (10)	60.3% (10)
	31.5% (40)	31.4% (40)	33.4% (40)
CGA (ours)	68.5 % (5)	67.9 % (5)	68.2 % (5)
	68.5% (10)	**67.8%** (10)	**68.2%** (10)
	64.6% (40)	**63.7%** (40)	**58.4%** (40)
CompCGA (ours)	68.4% (5)	68.3% (5)	68.4% (5)
	62.2% (10)	62.9% (10)	64.6% (10)
	63.3% (40)	53.4% (40)	56.6% (40)

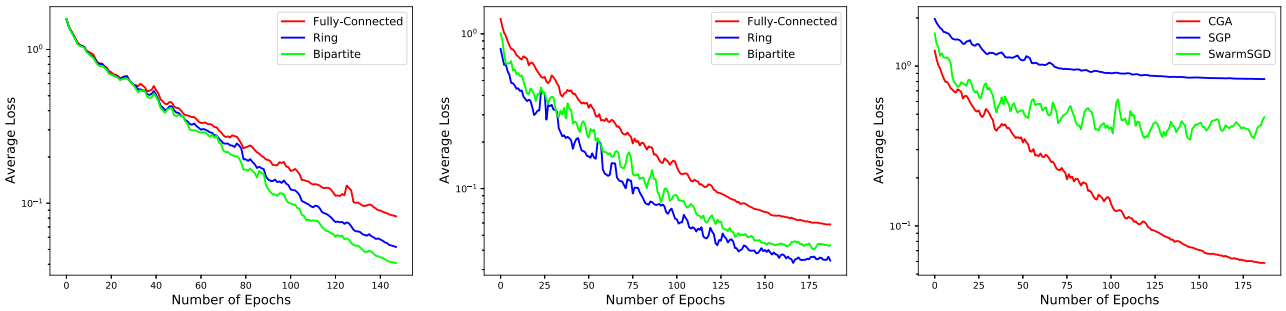
Figure 2(b) shows similar curves but for the non-IID case. In this case, we do observe a slight difference in convergence with faster rates for sparser topologies compared to their dense (fully connected) counterpart. Another phenomenon observed here is that for sparser topologies, the training process has more gradient variances and variations, which has been caused by the non-IID data distributions. This well matches the theoretical analysis we have obtained.

Finally, Figure 2(c) shows the comparison of convergence characteristics with other state-of-the-art decentralized algorithms with non-IID data distributions. *CGA* training is seen to be smoother compared to SwarmSGD, and to converge significantly faster compared to both SwarmSGD and SGP. From the theoretical analysis, we have shown that *CGA* enables to converge faster at the beginning although after a sufficiently large number of epochs, all methods listed here achieve the same rate $\mathcal{O}(\frac{1}{\sqrt{NK}})$.

Additionally, the SwarmSGD requires a geometrically distributed random variable to determine the number of local stochastic gradient steps performed by each agent upon interaction. That causes the largest variance shown in the loss curve in Figure 2(c). Note that since DPMSGD diverges for most of the non-IID experiments (see Table 3), we do not provide its loss plots here.

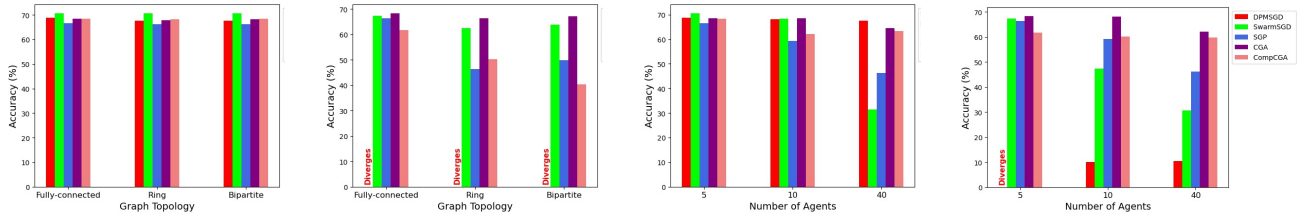
4.2. Comparative evaluation

We compare our proposed algorithm, *CGA* and its compressed version, *compCGA* with other state-of-the-art decentralized methods - DPMSGD, SGP and SwarmSGD. Note that in order to provide a fair comparison between the al-



(a) (b) (c)

Figure 2. Average training loss (log scale) for (a) CGA method on IID (b) CGA method on non-IID data distributions (c) different methods on non-IID data distributions for training 5 agents using CNN model architecture



(a) (b) (c) (d)

Figure 3. Average testing accuracy for different methods learning from (a) IID data distributions w.r.t graph topology (b) non-IID data distributions w.r.t graph topology (c) IID data distributions w.r.t the number of learning agents (d) non-IID data distributions w.r.t the number of learning agents

Table 3. Testing accuracy comparison for CIFAR10 with non-IID data distribution using CNN model architecture

Model	Fully-connected	Ring	Bipartite
DPMSGD	Diverges (5) 10.1% (10) 10.5% (40)	Diverges (5) Diverges (10) 10.0% (40)	Diverges (5) 10.0% (10) 10.7% (40)
SGP	66.4% (5) 59.3% (10) 46.2% (40)	46.3% (5) 25.8% (10) 31.4% (40)	49.8% (5) 24.9% (10) 11.8% (40)
SwarmSGD	67.3% (5) 47.3% (10) 30.6% (40)	62.5% (5) 38.5% (10) 25.8% (40)	63.9% (5) 33.8% (10) 23.5% (40)
CGA (ours)	**68.4%** (5) **68.2%** (10) **62.1%** (40)	**66.5%** (5) **48.8%** (10) **40.9%** (40)	**67.2%** (5) **38.9%** (10) **25.7%** (40)
CompCGA (ours)	61.7% (5) 60.2% (10) 59.8% (40)	50.3% (5) 39.5% (10) 32.7% (40)	40.4% (5) 36.7% (10) 23.6% (40)

gorithms, there are some minor adjustments that we made during the experiments. For SGP optimizer, we considered the graph to be undirected where the connected agents could both send and receive information to and from each other. On top of that, the adjacency matrix $\mathbf{\Pi}$ in our experiments (a.k.a mixing matrix $P^{(k)}$ in Assran et al. (2019)) is fixed throughout the training process. In the implementation of SwarmSGD, we defined the number of local SGD steps, $H = 1$, where the selected pair of agents perform only a single local SGD update before averaging their model parameters. In term of graph topologies, SwarmSGD was run not only on r-regular graphs (fully connected and ring) as described in Nadiradze et al. (2019), we also performed experiments using bipartite graph topology which is not r-regular.

As a baseline, we first provide a comparative evaluation for IID data distributions in Table 2. Results show that *CGA* performance is comparable with or slightly better than other methods in most cases with smaller number of agents, i.e., 5 and 10. However, we do observe a noticeable reduction in testing accuracy for SGP and SwarmSGD with 40 agents communicating over Ring or Bipartite graphs (which is an expected trend as reported in Assran et al. (2019) and Sattler et al. (2019)). While the testing accuracy of *CGA* also decreases in these scenarios, the performance reduction is not as drastic in comparison. The performance of *compCGA* deteriorates slightly compared to *CGA*, while still maintaining better accuracy than other methods in most scenarios.

The advantage of *CGA* is much more pronounced under non-IID data distributions as seen in Table 3. With extreme non-IID data distributions, *CGA* achieves the highest accuracy for all scenarios with different number of learning agents and communication graph topologies. In contrast, the baseline method DPMSGD struggles significantly in

all scenarios with non-IID data. Other methods (SGP and SwarmSGD) while having similar performance as *CGA* for 5 agents and fully connected topology, their performances drop significantly more than that of *CGA* with higher number of agents and sparser communication graphs. *CompCGA* performs slightly worse than *CGA*, while still maintaining better accuracy than other methods in most scenarios.

Finally, we graphically summarize the overall trends that we observed in Figure 3. From Figure 3 (a) and (b), it is clear that while there is no appreciable impact of graph topology on testing accuracy under IID data distributions, the impact is quite significant under non-IID data distributions. The results shown here are with 5 agents (see *Supplementary Section* A.5 for 10 and 40 agents). In this case, testing accuracy decreases for sparse graph topologies which conforms with observations made in Sattler et al. (2019). Figure 3 (c) and (d) show accuracy trends with respect to the number of agents. All results shown here are with fully connected topology (see *Supplementary Section* A.5 for other topologies). In this regard, Assran et al. (2019) shows a slight reduction in accuracy when the number of nodes/agents increase for both SGP and DPSGD methods. We see a similar trend here for both IID and non-IID data distributions. Clearly, the impact is more pronounced for non-IID data. However, the performance decrease with increase in number agents remain small for both *CGA* and *CompCGA* under non-IID data distributions. Also, as discussed in Table 1, we notice that the communication rounds for *CGA* is twice as that of SGP and 4 times the communication round of SwarmSGD. Therefore, we looked at their convergence properties *w.r.t* communication rounds. As Figure 4 shows, CGA converges to a lower loss value after 200 communication rounds.

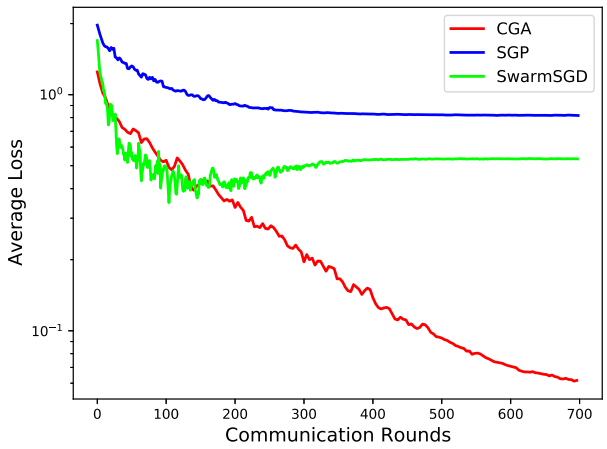


Figure 4. Average training loss (log scale) for different algorithms w.r.t communication rounds on non-IID data distributions (for 5 agents using CNN model architecture)

5. Conclusions

In this paper, we propose the Cross-Gradient Aggregation (*CGA*) algorithm to effectively learn from non-IID data distributions in a decentralized manner. We present convergence analysis for our proposed algorithm and show that we match the best known convergence rate for decentralized algorithms using *CGA*. To reduce the communication overhead associated with *CGA*, we propose a compressed variant of our algorithm (*CompCGA*) and show its efficacy. Finally, we compare the performance of both *CGA* and *CompCGA* with state-of-the-art decentralized learning algorithms and show superior performance of our algorithms especially for the non-IID data distributions. Future research will focus on addressing performance reduction in scenarios with a large number of agents communicating over sparse graph topologies.

6. Acknowledgements

This work was partly supported by the National Science Foundation under grants CAREER-1845969 and CAREER CCF-2005804. We would also like to thank NVIDIA® for providing GPUs used for testing the algorithms developed during this research. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant ACI-1548562 and the Bridges system supported by NSF grant ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

References

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5973–5983, 2018.

Yossi Arjevani, Joan Bruna, Bugra Can, Mert Gürbüzbalaban, Stefanie Jegelka, and Hongzhou Lin. Ideal: Inexact decentralized accelerated augmented lagrangian method. *arXiv preprint arXiv:2006.06733*, 2020.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

Aditya Balu, Zhanhong Jiang, Sin Yong Tan, Chinmay Hedge, Young M Lee, and Soumik Sarkar. Decentralized deep learning using momentum-accelerated consensus. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3675–3679. IEEE, 2021.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and De-

vavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13: 165–202, 2012.

Arya Ketabchi Haghighat, Varsha Ravichandra-Mouli, Pranamesh Chakraborty, Yasaman Esfandiari, Saeed Arabi, and Anuj Sharma. Applications of deep learning in intelligent transportation systems. *Journal of Big Data Analytics in Transportation*, 2(2):115–145, 2020.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B Gibbons. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. Collaborative deep learning in fixed topology networks. *Advances in Neural Information Processing Systems*, 2017:5905–5915, 2017.

Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. On consensus-optimality trade-offs in collaborative deep learning. *arXiv preprint arXiv:1805.12120*, 2018.

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.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019a.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847*, 2019b.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003.

Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U Stich. A unified theory of decentralized sgd with changing topology and local updates. *arXiv preprint arXiv:2003.10422*, 2020.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Chengjie Li, Ruixuan Li, Haozhao Wang, Yuhua Li, Pan Zhou, Song Guo, and Keqin Li. Gradient scheduling with global momentum for non-iid data distributed asynchronous training. *arXiv preprint arXiv:1902.07848*, 2019a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019b.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

Yucheng Lu and Christopher De Sa. Moniqua: Modulo quantized communication in decentralized sgd. *arXiv preprint arXiv:2002.11787*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Giorgi Nadiradze, Amirmojtaba Sabour, Dan Alistarh, Aditya Sharma, Ilia Markov, and Vitaly Aksenov. SwarmSGD: Scalable decentralized SGD with local updates. *arXiv preprint arXiv:1910.12308*, 2019.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. *arXiv preprint arXiv:2007.07682*, 2020.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *CoRR*, abs/1907.07346, 2019. URL http://arxiv.org/abs/1907.07346.

Qianqian Tong, Guannan Liang, and Jinbo Bi. Effective federated adaptive gradient methods with non-iid decentralized data. *arXiv preprint arXiv:2009.06557*, 2020.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Practical low-rank communication compression in decentralized deep learning. *Advances in Neural Information Processing Systems*, 33, 2020.

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

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1): 65–78, 2004.

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.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.