OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Local Gradient Aggregation for Decentralized Learning from Non-IID data

Yasaman Esfandiari YASESF@IASTATE.EDU
Iowa State University

Sin Yong Tan TSYONG98@IASTATE.EDU
Iowa State University

Zhanhong Jiang STARKJIANG@GMAIL.COM
Johnson Controls International

Aditya Balu BADITYA@IASTATE.EDU
Iowa State University

Chinmay Hegde CHINMAY.H@NYU.EDU
New York University

Soumik Sarkar SOUMIKS@IASTATE.EDU
Iowa State University

Abstract

Distributed centralized learning refers to a class of machine (deep) learning algorithms that enable a group of collaborative learning agents to train models using a dataset distributed among them with the aid of a central parameter server. Recently, decentralized learning algorithms which are independent of the central parameter server have demonstrated state-of-the-art results. However, an essential requirement to achieve such performance has been balanced distribution (among classes) of data among the agents, referred to as iid data. In real-life applications, having a precise iid distribution of data among the agents is often not feasible. We propose a Local Gradient Aggregation (*LGA*) that is a decentralized learning algorithm, where each agent collects the gradient information from its neighboring agents and updates its model with a projected gradient. We demonstrate the efficacy of *LGA* on non-iid data distributions on benchmark datasets. By comparing against state-of-the-art decentralized algorithms, we show that our algorithm achieves the highest accuracy rate on non-iid data distribution while preserving the iid counterpart's performance.

1. Introduction

Distributed machine learning refers to algorithms focused on learning from data distributed among many agents using distributed optimization schemes. Researchers have designed distributed deep learning algorithms with several approaches including: centralized learning (also known popularly as Federated Learning, FL) [6, 17], decentralized learning [15, 19], gradient compression [1, 25] and Coordinate updates [20, 21]. In this paper, we study two aspects of distributed deep learning:

Decentralized learning: While having a central parameter server is acceptable in data center type applications, in certain use cases such as learning over a robotic network, continuous communication with a central parameter server is often not feasible. Several decentralized learning algorithms have been proposed to address this concern, where agents directly interact with each other (their neighbors) without a central parameter server. Recent contributions in decentralized learning have been made with the aid of gossip averaging algorithms [3, 8, 32]. Combining SGD with gossip averaging algorithm, [15, 24, 33] proposed effective algorithms. In another line of work, researchers proposed utilizing compression techniques to achieve a consensus model [9, 28]. Around the same time, Assran et al. [2] proposed SGP algorithm which converges at the same sub-linear rate as SGD and SwarmSGD was proposed by Nadiradze et al. [18], which leverages random interactions between participating agents in a graph to achieve consensus.

Learning to cope with non-iid data: Researchers have found empirically and shown analytically that decentralized learning algorithms achieve comparable performance to its centralized counterpart under the so-called iid (independently and identically distributed) assumption. However, in real life application, such an assumption is difficult to satisfy. Li et al. [14] analytically showed the limitations with FedAvg on non-iid data. Li et al. [13] proposed a variant of FL by adding a penalty term in the local objective function. Motivated by life-long learning, Shoham et al. [26] proposed FedCurv by adding a penalty term to the local loss function using the Fisher information matrix. In another research study, Zhao et al. [34] proposed FedAvg-EMD utilizing the earth mover's distance (EMD) as a metric to quantify the distance between the data distribution on each client and population distribution. Also, FedNova was proposed in which they use a normalized gradient in the update law of FedAvg [31]. Similar to the case of decentralized learning, compression techniques [22, 23], momentum variant of algorithms [12, 30], the use of adaptive gradients [29], and use of controllers in agent's and server's models [7] are also used in centralized learning for coping with non-iid data. Although several FL approaches can handle departure from the iid assumption, there is still a gap in decentralized learning [4]. To overcome this issue, we are proposing the *Local Gradient Aggregation* (LGA) algorithm and show its effectiveness in learning models in a decentralized manner from both iid and non-iid data distributions. Inspired by continual learning literature [16], we are devising an algorithm which in each step of training, collects the gradient information from each agent's model on all other agent's datasets and projects them into a single gradient which is then used to update the model. We interpret the algorithm and analytically show how to obtain such an optimal local gradient using quadratic programming (QP). Finally, we empirically validate the performance of our algorithm. We then compare the effectiveness of our algorithm with the state-of-the-art and show that we can achieve higher accuracy in learning from non-iid data.

2. Problem Formulation

In this section, we describe the decentralized distributed learning framework. Following that, we propose the Local Gradient Aggregation (*LGA*) method for decentralized distributed learning.

2.1. Distributed Empirical Risk Minimization

The standard (unconstrained) empirical risk minimization problem which is used in a generic deep learning problem can be represented as:

$$\min \frac{1}{n} \sum_{i=1}^{n} f^i(x), \tag{1}$$

where $x \in \mathbb{R}^d$ denotes the set of parameters of deep learning model and $f : \mathbb{R}^d \to \mathbb{R}$ is a prescribed loss function, f^i is the function value corresponding to a data point i, and the set $\mathcal{D}$ represents the training dataset with n as the total number of data samples ($n = |\mathcal{D}|$).

In decentralized learning setting, we have N agents ($N > 1$) and each agent j has a subset of training data, $\mathcal{D}_j,\ \ j \in \{1, \ldots, N\}$. There are n_j samples in each of the subsets $\mathcal{D}_j$. Also, we assume that $\sum_{j=1}^{N} n_j = n$. We also construct a set for $(j, l) \in \{1, \ldots, N\} \times \{1, \ldots, N\}$ to represent all the pairs of agent communications possible in the communication network. This is often formulated using a communication graph topology [10], where agents represent the vertices of the graph and the set (j, l) represents the edges of the graph. In this paper, we represent this set as $\mathbb{C} := \{(j, l) \in \{1, \ldots, N\} \times \{1, \ldots, N\}\}$. Throughout the rest of analysis, we assume that the graph is *undirected and static*. We now reformulate Eq. 1 as:

$$\min f(x) = \sum_{j=1}^{N} \sum_{i \in \mathcal{D}_j} f^i(x^j) := \frac{N}{n} \sum_{j=1}^{N} f_j(x^j); \text{s.t. } x^j = x^l \ \forall \ (j, l) \in \mathbb{C} \tag{2}$$

where, $f_j(x) = \frac{1}{N} f(x)$ represents the empirical loss specific to j^{th} agent.

2.2. Limitations in Existing Update Laws

The optimization problem in Eq. 2 has been solved using numerous proposed techniques, and some have been covered in the related works. Generically, they can be summarized into three categories: parameter consensus, gradient consensus, and both. Next, discuss why the performance of most of these existing algorithms degrades when the assumption that $\mathcal{D}_j$ is the independently and identically distributed (IID) is violated. We denote by $\phi(\cdot)$ the implicit core update law for each x^j. Thus, the following expressions show respectively, three categories:

- Parameter consensus:$x_k^j = \phi(\sum_l \pi_{jl} x_{k-1}^l, \nabla f_j(x_{k-1}^j, \zeta_k))$
- Gradient consensus: $x_k^j = \phi(x_{k-1}^j, \sum_l \pi_{jl} \nabla f_l(x_{k-1}^l, \zeta_k))$
- Both: $x_k^j = \phi(\sum_l \pi_{jl} x_{k-1}^l, \sum_l \pi_{jl} \nabla f_l(x_{k-1}^l, \zeta_k))$

where π_{jl} is the j-th row and l-th column element of a doubly stochastic matrix Π, ζ_k is the random seed at time step k realized by randomly selecting a mini-batch of samples. These three categories typically can perform well when $\mathcal{D}_j$ is IID, regardless of different convergence rates. However, in most real-world problems, the IID assumption has rarely possible. From a local agent's perspective, non-IID data makes a significant impact on local parameters and gradients. Thus, utilizing the parameter consensus scheme could get stuck with bad local minima. Though the gradient consensus may reduce the variance, the optimal solution obtained using this approach can be biased as x_{k-1}^j is quite different for $j \in \{1, ..., N\}$. Combining both parameter consensus and gradient consensus is beneficial for getting better local optima due to the reduction of variance and bias, but still, the local gradient can have a negative impact such that the direction of $\sum_l \pi_{jl} \nabla f_l(x_{k-1}^l, \zeta_k)$ may not be the optimal one for x_k^j. To mitigate these issues, we take a different viewpoint on finding the *local optimal gradient* for each agent, motivated by the recent advances in continual learning.

2.3. Inspiration From Gradient Episodic Memory

Lopez-Paz and Ranzato [16] perceived the problem of continual learning involving several tasks, which lead to a data distribution shift, from the perspective of gradient episodic memory by formulating a constrained optimization problem.

Specifically, they solved the following problem once observing a new sample at time k, s,

$$\min_x f(s; x), \text{ s.t. } f(\mathcal{M}_t; x) \leq f(\mathcal{M}_t; x_{k-1}) \ \forall t < k, \tag{3}$$

where $\mathcal{M}_t$ signifies the previous tasks, x_{k-1} indicates the model state at the end of learning of task $k-1$. To solve the above problem efficiently, the authors first assume that the function is locally linear due to the small optimization steps and then diagnose the increase in previous tasks' loss by computing angle between the gradient vectors. Hence, the following relationship can be obtained as

$$\langle g, g_t \rangle := \langle \frac{\partial f(s;x)}{\partial x}, \frac{\partial f(\mathcal{M}_t;x)}{\partial x} \rangle \geq 0, \ \forall t < k. \tag{4}$$

Two scenarios are taken into account for Eq. 4. If all inequalities are satisfied, g is the correct direction to update x. If one or more inequalities are violated, at least one previous task will experience an increase in loss after the parameter update. One intuitively simple correction is to find another gradient vector $\tilde{g}$ that is closest to the proposed initially one g satisfy the inequality constraint, which can be realized by solving a quadratic programming (QP) problem. Hence, the optimization problem Eq. 3 has been turned equivalently into a *bi-level* optimization problem that aims at finding the best model parameter, which also enables the continuous loss descents over tasks. Comparing the data distribution shifts in the continual learning with the non-IID data distributions in the decentralized learning, we can leverage the techniques into the decentralized learning framework by finding the local optimal gradient for each local model. Thus, we present the proposed algorithm in the next section, followed by a detailed analysis of the algorithm interpretation.

3. Local Gradient Aggregation Algorithm for Distributed Deep Learning

We introduce the following definitions to characterize the analysis.

Definition 1 *Given a dataset $\mathcal{D}_j$, an differentiable objective function f^j, and model parameter x^j, a self-gradient is defined as$g^{jj} := \nabla_{x^j} f^j(\mathcal{D}_j; x^j)$;; given a dataset $\mathcal{D}_l$, an differentiable objective function f^l, and model parameter x^j, a cross-gradient is defined as $g^{jl} := \nabla_{x^j} f^l(\mathcal{D}_l; x^j)$.*

Note that in the following g^{jj} and g^{jl} are calculated as stochastic gradients. Based on the formulation discussed in the last section, we devise the *LGA* algorithm suitable for distributed deep learning. In this algorithm, x^j is the model parameter for each agent j, which is initialized along with $\mathcal{D}_j$. K is the number of iterations, α is the step size, and N is the number of agents, which are the inputs to the algorithm. In each iteration k, the gradient of the model parameters for each agent j on its data subset is calculated by g_k^{jj}. Then agent j's model parameters are sent to the other agents, and the gradients are calculated on the other agents' data subsets $g_k^{jl}; \forall l \neq j$ and stored in a matrix G^j associated with agent j. Using G^j and g_k^{jj}. the QP solver obtains the local optimal gradient $\tilde{g}$, which is then used to update agent j's model parameter. We also add momentum update for faster convergence.

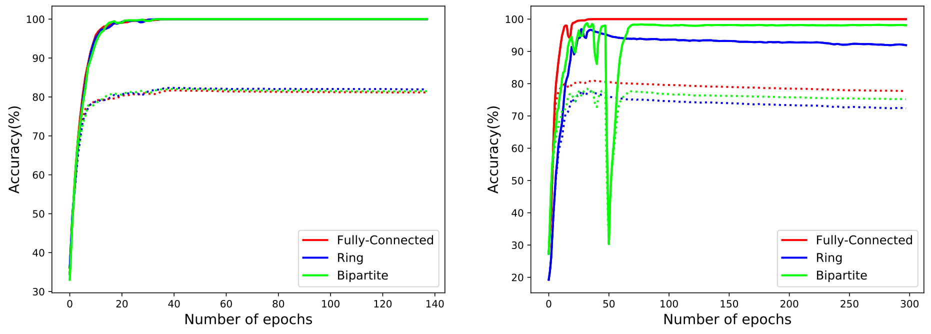
Different from the aforementioned update laws in the last section, in Algorithm 1, we can observe that the local optimal gradient $\tilde{g}^j$ is obtained for each agent by solving a simple QP problem. Another noticeable difference in this context is the cross-gradient g^{jl}. With non-IID data distributions, the difference between self-gradients in different agents is significant such that large variations in g^{jj} result in poor convergence. However, the cross-gradient alleviates this issue and provides a local optimal direction for the parameter's local loss descent. Intuitively, the local optimal gradient obtained from the neighbors enable the local loss function to continuously descend on a path to a local minimum that all agents in the neighborhood will reach.

4. Experimental Results

In this section, we analyze the performance of *LGA* algorithm empirically. We compare the effectiveness of our algorithms with baseline algorithms *SwarmSGD* [18], *SGP* [2], and *CDSGD* [5].
Experiment Setup: We present the empirical studies on CIFAR-10 and MNIST datasets with both iid and non-iid data distributions. The experiments are performed for 5 and 10 agents. To simulate

Algorithm 1 Local Gradient Aggregation

```
1:  Initialization: $\mathcal{D}_j, x_0^j, v_0^j, (j = 1, 2, \ldots, N), \alpha, K$, a QP solver
2:  for $k = 0 : K$ do
3:      for $j = 1 : N$ do
4:          Randomly shuffle the corresponding data subset $\mathcal{D}_j$
5:          Compute $g_k^{jj}$
6:          $G^j = \{\}$
7:          for each agent $l$, s.t. $(j, l) \in \mathbb{C}$ do
8:              Compute $g_k^{jl}$
9:              $G^j \leftarrow G^j \cup g_k^{jl}$
10:             $w_k^j = \sum_l \pi_{jl} x_{k-1}^l$
11:         end for
12:         $\tilde{g}^j \leftarrow \mathtt{QP}(g_k^{jj}, G^j)$
13:         $v_k^j = \mu v_{k-1}^j - \alpha \tilde{g}^j$
14:         $x_k^j = w_k^j + v_k^j$
15:     end for
16: end for
```



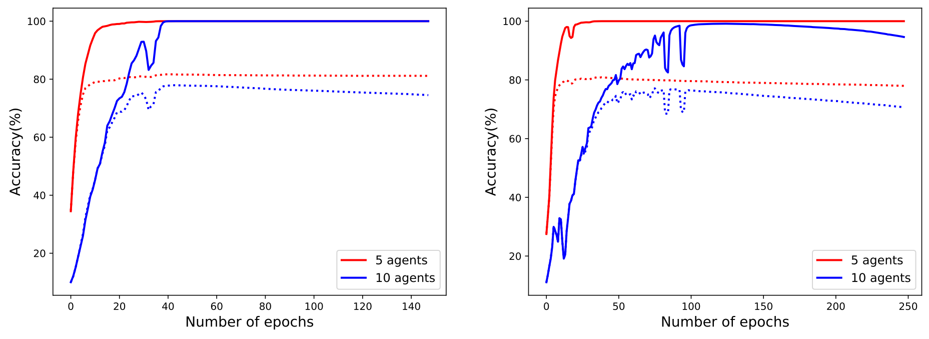
(a) (b)

Figure 1: *Average training and validation accuracy for LGA method on (a) iid (b) non-iid data*

non-iid data distribution, we assign each agent with unique classes (unseen by the other agents). In other words, 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. We use a simple LeNet model [11] for training MNIST and use a VGG11 [27] model for CIFAR10. Mini-batch sizes of 64 and 256 are used for 5 and 10 agents, respectively. The initial step-size is set to 0.01 for CIFAR-10 and 0.001 for MNIST experiments. The decaying constant for the step size is 0.98. We present detailed results on experiments with MNIST Data in the Appendix.

LGA method maintains the high accuracy and smooth performance for different graph topologies: Figure 1 shows that LGA can maintain the high accuracy when learning from both iid and non-iid data distributions. However, based on Figure 4 other methods suffer from more complex graph typologies and their performance gets adversely affected when the graph is not fully connected.

LGA can achieve high accuracy when the number of participating agents increases: Based on Figure 2, LGA can still achieve comparably high accuracy when the number of agents is doubled in a fully connected graph topology. However, when graph topology becomes sparser, LGA also succumbs to a decrease in the performance. As Figure 4 b shows, LGA accuracy diminishes when

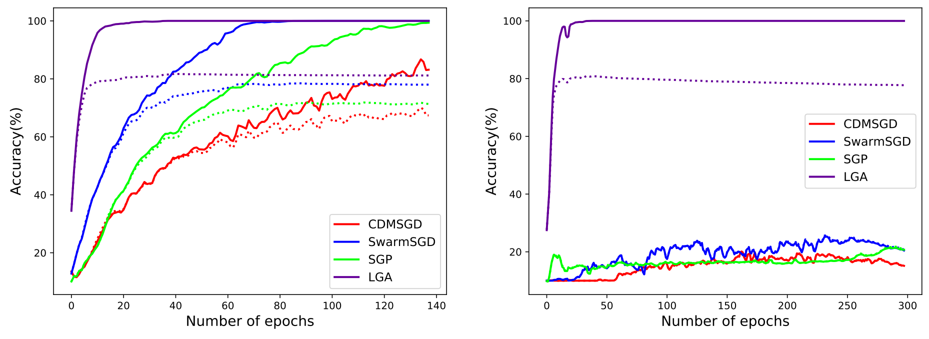


(a) (b)

Figure 2: *Average training and validation accuracy for LGA method on (a) iid (b) non-iid data*

the graph topology gets more complicated in a 10 agent scenario from the non-iid dataset. However, the high accuracy is still maintained in the iid scenario (see Appendix).

LGA achieves the highest accuracy in less number of epochs smoothly and maintains it in both iid and non-iid scenarios: Figure 3 shows the accuracy comparison for different methods under both iid and non-iid data distribution for fully connected graph topology (the results for other topologies are in the Appendix and the trend matches with Figure 3). LGA achieves the highest accuracy on both non-iid and iid data distribution in less number of epochs. This also matches what we observe in Figure 4 with a different number of agents. Another observation, as Figure 3 shows, is that an immediate result of the LGA algorithm is smoothness and stability of the training process compared to other algorithms. Since the optimal gradient is achieved after aggregating the gradients from all the participating agents in the neighborhood and applying QP projection, the ultimate gradient is much less noisy than the other algorithms.



(a) (b)

Figure 3: *Average training (solid line) and validation (dash line) accuracy for different methods on (a) iid (b) non-iid data distributions for fully connected graph topology with* 5 *agents*

5. Conclusions

In this paper, we propose a Local Gradient Aggregation (*LGA*) algorithm to effectively learn from non-iid data distributions in a decentralized manner that resolves the scalability and connectivity concerns associated with using a central parameter server. We present the convergence characteristics of the algorithm and investigate the effect of different topologies, with different combinations of agent numbers empirically. Also, we compare the performance of *LGA* algorithm with state-of-the-art decentralized learning algorithms as baseline methods. Future research directions include: (i) Computation analysis of *LGA* (ii) investigating projection methods other than QP (iii) empirical comparison between different extents of non-iidness in the data distribution.

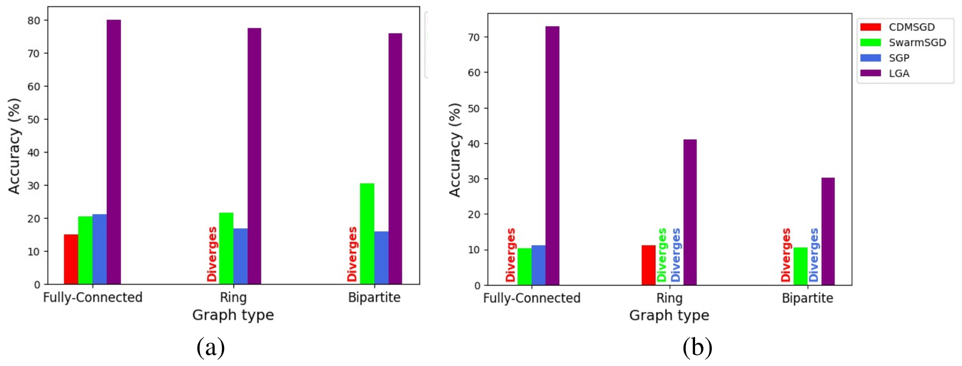


Figure 4: *Average validation accuracy for different methods with (a) 5 (b) 10 agents on non-iid data distributions on CIFAR-10 dataset*

6. Acknowledgements

This paper is based upon research partially supported by the National Science Foundation under grant No. CAREER-1845969. Any claims, methodology and conclusions discussed in this research work correspond to those of the authors and do not necessarily reflect the opinions of the funding agency.

References

[1] 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.

[2] 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.

[3] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

[4] 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.

[5] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. Collaborative deep learning in fixed topology networks, 2017.

[6] 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.

[7] 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*, 2019.

[8] 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.

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

[10] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

[11] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

[12] 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*, 2019.

[13] 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.

[14] 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*, 2019.

[15] 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.

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

[17] 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.

[18] Giorgi Nadiradze, Amirmojtaba Sabour, Dan Alistarh, Aditya Sharma, Ilia Markov, and Vitaly Aksenov. Swarmsgd: Scalable decentralized sgd with local updates, 2019.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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`.

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

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

[31] 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.

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

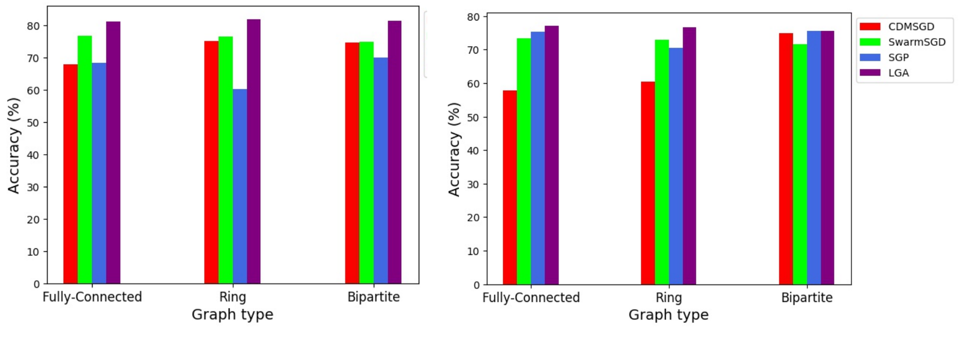
[33] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

[34] 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.

Appendix A. Appendix

A.1. Additional CIFAR-10 Results and Observations

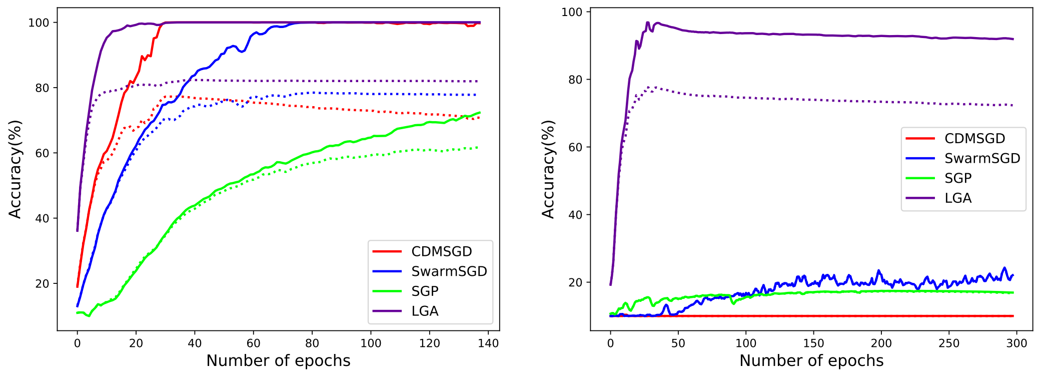
LGA maintains its performance for different graph topologies when the number of participating agents increases while learning from iid data distribution: Figure 5 shows that LGA achieves almost constant accuracies for different topologies when learning from iid distributions with increasing number of agents. We showed earlier in Figure 1 that LGA can be topology agnostic also in non-iid setting when the number of agents are smaller. However, when both the graph topology and the number of agents becomes complicated, LGA also succumbs to decrease in the performance. As Figure 4 b shows.



(a) (b)

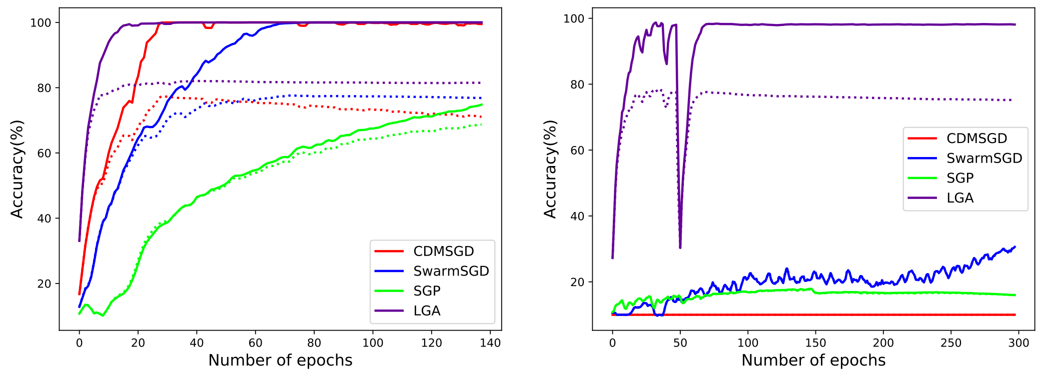
Figure 5: *Average validation accuracy for different methods with (a) 5 (b) 10 agents on iid data distributions on CIFAR-10 dataset*

LGA achieves the highest accuracy in less number of epochs smoothly and maintains it in both iid and non-iid scenarios with different topologies: Figure 6 and 7 shows the accuracy comparison for different methods under both iid and non-iid data distribution for Ring and Bipartite graph topologies. Confirming the observation that we made based on Figure 3, LGA achieves the highest accuracy on both non-iid and iid data distribution in less number of epochs for other graph topologies as well.



(a) (b)

Figure 6: *Average training (solid line) and validation (dash line) accuracy for different methods on (a) iid (b) non-iid data distributions for Ring graph topology with* 5 *agents*

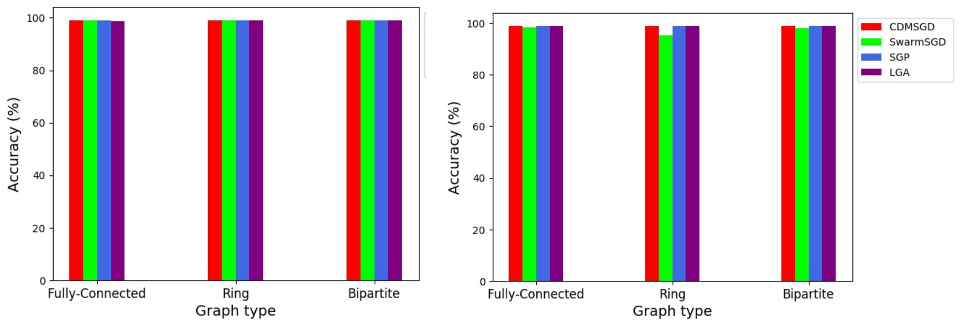


(a) (b)

Figure 7: *Average training (solid line) and validation (dash line) accuracy for different methods on (a) iid (b) non-iid data distributions for Bipartite graph topology with* 5 *agents*

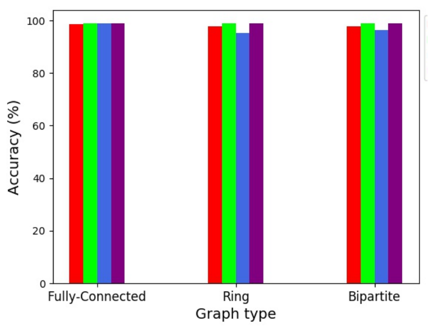
A.2. MNIST Results

Same as what we did for CIFAR-10, we are comparing different methods performance on MNIST dataset. The results are summarized in Figure 8 and 9. Although the accuracies are much higher in MNIST, and most of the methods work in most of the settings, we can see that CDMSGD [5] and SGP [2] diverge in some non-iid settings, whereas LGA maintains its performance in all of the experiments.

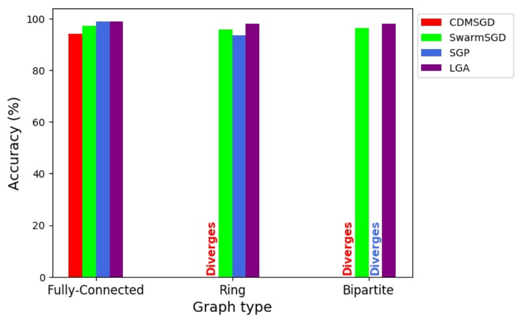


(a) (b)

Figure 8: *Average validation accuracy for different methods with (a) 5 (b) 10 agents on iid data distributions on MNIST dataset*



(a)



(b)

Figure 9: *Average validation accuracy for different methods with (a) 5 (b) 10 agents on non-iid data distributions on MNIST dataset*