Exploring the Error-Runtime Trade-off in Decentralized Optimization

Jianyu Wang†, Anit Kumar Sahu*, Gauri Joshi†, Soummya Kar†
†Carnegie Mellon University, *Amazon Alexa AI
†{jianyuw1, gaurij, soummyak}@andrew.cmu.edu, *anit.sahu@gmail.com

***Abstract*—Decentralized stochastic gradient descent (SGD) has recently become one of the most promising methods to use data parallelism in order to train a machine learning model on a network of arbitrarily connected nodes/edge devices. Although the error convergence of decentralized SGD has been well studied in the last decade, most of the previous works do not explicitly consider how the network topology influences the overall convergence time. Communicating over all available links in the network may give faster error convergence, however, it will also incur higher communication overhead. The MATCHA algorithm proposed in [1] achieves a win-win in this error-runtime trade-off by judiciously sampling the communication graph. In this paper, we propose several variants of the MATCHA algorithm and show that MATCHA can work with many other activation schemes and decentralized computation tasks. It is a flexible framework to reduce the communication delay for free in decentralized environments.**

I. Introduction

The advent of federated learning [2], [3] has results in both data collection and model training to shift to edge devices. In cross-silo federated learning, the devices are connected through an arbitrary network topology, without a central coordinator. Decentralized stochastic gradient descent (SGD) is workhorse for training machine learning models [4]–[6] in such settings. Previous works have studied the error convergence of decentralized SGD in terms of iterations or communication rounds in both synchronous [6], [7] and asynchronous [8] settings. However, they do not consider how the topology affects the wall-clock runtime of the algorithm.

The performance of any decentralized optimization scheme centers around the communication-computation trade-off. With communication being the bigger bottleneck, have been lots of work from both the perspectives of algorithms [9]–[12] and systems [13]–[15] to improve the communication efficiency of synchronous distributed SGD in a fully-connected network. However, it is not clear whether aforementioned schemes can be extended to arbitrary decentralized networks. Other approaches towards communication efficiency in decentralized SGD include quantization of transmitted messages to reduce bandwidth [16], [17], increasingly sparse communication [18], [19] to reduce the number of messages over time. However, such approaches do not take into account communication delays and network latency.

This work was supported in part by NSF CCF grant # 1850029, a Qualcomm Innovation Fellowship (Jianyu) and an IBM Faculty Award.

To address the error vs runtime trade-off while keeping communication efficiency in mind for arbitrary decentralized graphs, MATCHA was recently proposed in [1]. In particular, MATCHA tries to obtain the best error vs runtime trade-off by carefully tuning inter-node communication. MATCHA decomposes the graph into sets of disjoint links called matchings which allows for non-blocking parallel communication. Furthermore, the probability of activations of these matchings are optimized so as to maximize the algebraic connectivity of the graph. This helps MATCHA in avoiding communication over redundant links, while ensuring frequent communication over connectivity-critical links thereby making way for faster information exchange in the graph. In spite of the tuned inter-node communication, MATCHA preserves the same error vs iterations rate as that of classical decentralized optimization on general non-convex objectives. However, in terms of empirical performance MATCHA outperforms other decentralized schemes in terms of error vs wall-clock time while preserving the error vs iterations rate.

In this paper, we extend the MATCHA algorithm in multiple ways, demonstrating that MATCHA is a flexible algorithmic framework to reduce the communication delay in decentralized computation tasks. The main contributions are listed as follows:

- In MATCHA, matchings are activated independently with different probabilities at each iteration. Instead, we show that one can also use other activation schemes: (1) ordering all matchings with respect to their importance to the algebraic connectivity and only activating the most important ones; (2) sampling matchings without replacement according to their importance. Both these two activation schemes can achieve similar performance as the original MATCHA algorithm.
- We show that MATCHA not only works for vanilla decentralized SGD (DecenSGD) but also can be applied to other decentralized optimization algorithms, such as DecenSGD with gradient tracking. Experimental results show that using MATCHA on the top of these algorithms can save communication delay as well as preserve the fast error convergence speed.
- We also examine the performance of MATCHA in the cases where nodes have non-IID local datasets. The results show that the effectiveness of MATCHA is not influenced by the distribution of data.

II. Preliminaries of MATCHA

In this section, we will introduce the problem formulation and review the MATCHA algorithm proposed in [1]. In decentralized optimization, suppose there are total m nodes and the model parameters are represented by a vector $\boldsymbol{x}$. Then, the objective function is formulated as follows:

$$F(\boldsymbol{x}) \triangleq \frac{1}{m}\sum_{i=1}^{m} F_i(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{s}\in\mathcal{D}_i}\ell(\boldsymbol{x};\boldsymbol{s}) \tag{1}$$

where $F_i(\boldsymbol{x})$ is defined as the local objective function, $\boldsymbol{s}$ denotes a single data sample, and $\ell(\boldsymbol{x};\boldsymbol{s})$ is the loss function for sample $\boldsymbol{s}$, defined by the learning model. For simplicity, we assume that all nodes have the same size of local datasets, that is, $|\mathcal{D}_i| = |\mathcal{D}_j|, \forall i \neq j$. Besides, we consider a setting where all nodes are organized in a network without a central coordinator. The network topology is denoted by $\mathcal{G}(\mathcal{V},\mathcal{E})$, where the vertex set $\mathcal{V} = \{1, 2, \dots, m\}$ corresponds to the nodes index and the edge set $\mathcal{E}$ represents the communication links among worker nodes. Node i and node j can communicate with each other if and only if $(i,j)\in\mathcal{E}$.

Vanilla Decentralized SGD (DecenSGD). The basic algorithm to minimize (1) using a network of nodes is vanilla decentralized SGD. Worker nodes first perform one step of local SGD and then average their local models with their neighbors. This procedure repeats until convergence. The update rule of DecenSGD on each node can be written as follows:

$$\boldsymbol{x}_i^{(k+1)} = \underbrace{\sum_{j=1}^{m} W_{ij}}_{\text{consensus step}} \underbrace{\left[\boldsymbol{x}_j^{(k)} - \eta g_j(\boldsymbol{x}_j^{(k)})\right]}_{\text{parallel local computation}} \tag{2}$$

where $\boldsymbol{x}_i^{(k)}$ denotes the model parameters of node i at the k-th iteration, $g_i(\boldsymbol{x}_i) = \frac{1}{|\xi|}\sum_{\boldsymbol{s}\in\xi}\nabla\ell(\boldsymbol{x};\boldsymbol{s})$ denotes the stochastic gradient evaluated on node i's local dataset, and η is the learning rate or step size. Additionally, the weight $W_{ij}\neq 0$ if and only if $(i,j)\in\mathcal{E}$. The matrix $\boldsymbol{W} = [W_{ij}]$ is defined as *mixing matrix*.

Matching Decomposition Sampling (MATCHA). In order to reduce the communication delay in the consensus step as shown in (2), [1] proposes to let worker nodes communicate more frequently over connectivity-critical links and less on others. Based on this insight, they develop the Matching Decomposition Sampling (MATCHA) algorithm. MATCHA contains the following four key components.

a) Matching Decomposition: The algorithm first decompose the base topology into disjoint subgraphs $\{\mathcal{G}_j(\mathcal{V},\mathcal{E}_j)\}_{j=1}^{M}$ such that $\mathcal{E} = \bigcup_{j=1}^{M}\mathcal{E}_j$ and $\mathcal{E}_i \bigcap \mathcal{E}_j = \varnothing, \forall i\neq j$. In particular, each subgraph is a matching of $\mathcal{G}$ in which each vertex is incident with at most one edge. The benefit of using matchings as the decomposition basis is that all links in one matching can be communicated over in parallel. At each training iteration, there is only a random subset of matchings will be activated for communication. As a result, the communication delay per iteration in MATCHA is proportional to the number of active matchings. One can reduce the communication delay by simply activate (i.e., let worker nodes communicate over) less matchings in each iteration.

b) Determine the Importance of Subgraphs: In MATCHA, matchings are activated independently for communication. In order to achieve this, we assign a Bernouli random variable B_j with mean p_j for each matching. The activation probabilities $\{p_j\}_{j=1}^{M}$ are determined by maximizing the algebraic connectivity subject to some communication delay constraints. Specifically, the algorithm solves the following optimization problem:

$$\begin{aligned} \max_{p_1,\dots,p_M} \quad & \lambda_2\left(\textstyle\sum_{j=1}^{M} p_j \boldsymbol{L}_j\right) \\ \text{subject to} \quad & \mathbb{E}[t(\textstyle\sum_{j=1}^{M}\mathsf{B}_j)] \le C_b\cdot t(M), \\ & 0\le p_j\le 1,\ \forall j\in\{1,2,\dots,M\}, \end{aligned} \tag{3}$$

where $\boldsymbol{L}_j$ denotes the Laplacian matrix of the j-th matching and $\sum_{j=1}^{M} p_j\boldsymbol{L}_j$ can be considered as the Laplacian of the expected graph. Note that λ_2 represents the algebraic connectivity and is a concave function [20], [21]. Moreover, $t(\cdot)$ denotes a monotonically increasing function and describes how the communication delay scales with the maximal degree in the graph. For simplicity, we assume $t(\cdot)$ is a linear function. In this case, (3) reduces to a concave optimization problem and can be solved efficiently.

c) Generate a Random Graph Sequence: Given the activation probabilities obtained by solving (3), in each iteration k, MATCHA generates an independent Bernoulli random variable $\mathsf{B}_j^{(k)}$ for each matching $j = 1,\dots M$. Thus, in the k-th iteration, the activated topology $\mathcal{G}^{(k)}(\mathcal{V},\mathcal{E}^{(k)})$, in which $\mathcal{E}^{(k)} = \bigcup_{j=1}^{M}\mathsf{B}_j^{(k)}\mathcal{E}_j$, can be sparse or even disconnected.

d) Construction of the Mixing Matrix: Based on the matching decomposition and random graph sequence, MATCHA constructs the mixing matrix as follows: $\boldsymbol{W}^{(k)} = \boldsymbol{I} - \alpha\sum_{j=1}^{M}\mathsf{B}_j^{(k)}\boldsymbol{L}_j$, where α is parameter that controls the weights of neighbors. This kind of construction is common in the literature. The best value of α that gives the fatest convergence is obtained by solving a semi-definite programming problem:

$$\begin{aligned} \min_{\rho,\alpha,\beta} \quad & \rho, \\ \text{subject to} \quad & \alpha^2-\beta\le 0, \\ & \boldsymbol{I} - 2\alpha\overline{\boldsymbol{L}} + \beta[\overline{\boldsymbol{L}}^2 + 2\widetilde{\boldsymbol{L}}] - \frac{1}{m}\boldsymbol{1}\boldsymbol{1}^\top \preceq \rho\boldsymbol{I} \end{aligned} \tag{4}$$

where β is an auxiliary variable, $\overline{\boldsymbol{L}} = \sum_{j=1}^{M} p_j\boldsymbol{L}_j$ and $\widetilde{\boldsymbol{L}} = \sum_{j=1}^{M} p_j(1-p_j)\boldsymbol{L}_j$.

Convergence Guarantee of MATCHA. In [1], the authors also provide a convergence analysis for the MATCHA algorithm. It is worth highlighting that, for a suitable communication budget, MATCHA can achieve the same or even faster convergence than vanilla DecenSGD. That is, unlike other communication-efficient methods which reduce the communication cost by sacrificing the convergence speed, MATCHA can reduce communication for free. Moreover, note that the convergence of MATCHA depends on the value of $\rho = \left\|\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right\|_2$. A smaller value of ρ yields a better convergence guarantee.

III. Experimental Settings

In this paper, we focus on training a deep neural network (ResNet-50 [22]) on the CIFAR-10 dataset [23], which has total 50000 training images and 10000 test images. We manually partition the dataset across all worker nodes with equal size. The default network topology we use is a randomly generated geometric graph with 16 nodes, as shown in Figure 1. In our experiments, each node is equipped with one NVIDIA TitanX Maxwell GPU and has a 40 Gbps (5000 MB/s) Ethernet interface. For vanilla DecenSGD, we tuned the learning rate from $\{0.01, 0.02, 0.04, 0.08, 0.1, 0.2\}$ and found that $\eta = 0.08$ works the best. For other algorithms, we use the same learning rate as vanilla DecenSGD in order to guarantee a fair comparison.

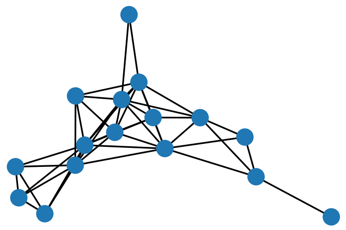


Fig. 1: Default network topology (a geometric graph) used in the experiments. There are total 16 nodes and the maximal degree is 10.

IV. Strategies to Generate Sparse Communication Topologies

In MATCHA, there are many different ways to construct the sparse graph sequence. One does not have to stick with the original method in [1] (i.e., activating matchings independently but with different probabilities). In this section, we explore two other matching activation/sampling methods and validate their effectiveness.

A. Removing the Least Important Matchings from the Base Topology

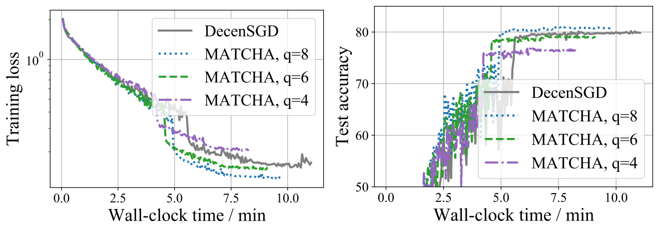
One simple way to sparsify the base topology is to remove the unimportant links. After obtaining the matching decomposition in MATCHA, one can also determine the importance of different matchings using the following optimization problem:

$$\begin{aligned} \max_{p_1,\dots,p_M} \quad & \lambda_2\left(\textstyle\sum_{j=1}^{M} p_j \boldsymbol{L}_j\right) \\ \text{subject to} \quad & \sum_{j=1}^{M} p_j = 1, \\ & 0 \le p_j \le 1,\ \forall j \in \{1,2,\dots,M\}, \end{aligned} \tag{5}$$

Then we can order all matchings based on their weights p_j and only use the top-q matchings for communication. Unlike the original MATCHA, this activation scheme uses a fixed (i.e., does not change with iteration k) sparse subgraph $\mathcal{E}^{(k)} = \bigcup_{j=1}^{q} \mathcal{E}_{(j)}$, where $\mathcal{E}_{(j)}$ denotes the edge set for the matching with the j-th largest weight.

Construction of the Mixing Matrix. When removing the least important matchings from the base topology, one can set the mixing matrix as $\boldsymbol{W} = \boldsymbol{I} - \alpha\sum_{j=1}^{q} \boldsymbol{L}_{(j)}$ where $\boldsymbol{L}_{(j)}$ denotes the Laplacian matrix for the matching with the j-th largest weight. The parameter α can be set by minimizing $\left\|\boldsymbol{W}^2 - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right\|_2$.

Experimental Results. The experimental results are presented in Figure 2. Recall that the maximal node degree in the base topology Figure 1 is 10, and hence, DecenSGD uses $q = 10$ matchings to perform communication at each iteration. We can observe that when the number of active matchings q reduces, the algorithm costs much less time to finish training. In terms of the error convergence, $q = 8$ can even achieve a lower training loss and a higher test accuracy than vanilla DecenSGD. If we further decrease the value of q, the final test accuracy of MATCHA has a slight drop.



(a) Training loss (b) Test accuracy

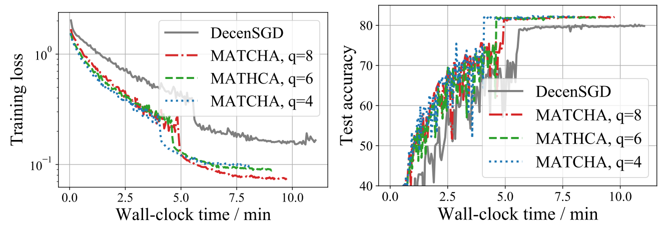
Fig. 2: The performance of a variant of MATCHA: removing the least important matchings from the base topology and using the remaining topology for communication.

B. Sampling q Matchings without Replacement

In MATCHA, different matchings are activated independently at each iteration. There also exists an alternative way to randomly activate matchings. One can sample q matchings without replacement according to their importance/weights as derived in (5). As long as $q < M$, the communication delay per iteration is reduced compared to vanilla DecenSGD.

Construction of the Mixing Matrix. Similar to the original MATCHA algorithm, the mixing matrix is defined as $\boldsymbol{W}^{(k)} = \boldsymbol{I} - \alpha\sum_{j=1}^{M} \mathsf{B}_j^{(k)} \boldsymbol{L}_j$. However, the activation random variables B_j's are no longer independent of each other. Besides, we cannot use (4) to determine the value of α automatically. Instead, we manually tune α for each value of q.

Experimental Results. The training curves are presented in Figure 3. When reducing the number of activated matchings and manually tuning the value of α, we observe that MATCHA can not only significantly reduce the total training time but also achieve even higher test accuracy than vanilla DecenSGD. For example, when q is set to 4, the final accuracy of MATCHA is 82.33% while vanilla DecenSGD is 80.20%.



(a) Training loss (b) Test accuracy

Fig. 3: The performance of a variant of MATCHA: sampling matchings without replacement at each iteration.

V. Applying MATCHA to Other Decentralized Optimization Algorithms

It is worth noting that the MATCHA framework can be applied to any decentralized algorithm to save communication delay. In this section, we provide an example on how to extend MATCHA to gradient tracking methods.

Decentralized SGD with Gradient Tracking (DecenSGD-GT). Gradient tracking (GT) [24] is a technique that is used to accelerate the convergence of DecenSGD. Instead of using the stochastic gradient to update local models, DecenSGD-GT uses an auxiliary variable $\boldsymbol{y}_i$, to approximate the global gradient $\frac{1}{m}\sum_{i=1}^{m} g_i(\boldsymbol{x}_i)$. The detailed update rule of DecenSGD-GT is provided as follows:

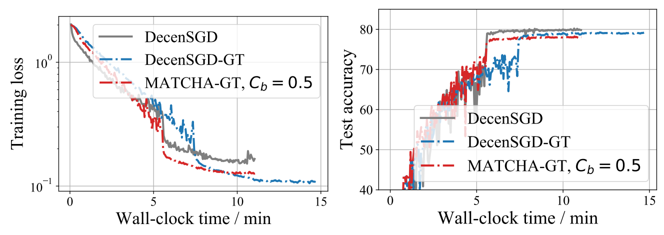
$$\boldsymbol{y}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}\left[\boldsymbol{y}_i^{(k)} + g_j(\boldsymbol{x}_j^{(k)}) - g_j(\boldsymbol{x}_j^{(k-1)})\right], \tag{6}$$

$$\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}\left[\boldsymbol{x}_j^{(k)} - \eta \boldsymbol{y}_j^{(k)}\right]. \tag{7}$$

Note that $\boldsymbol{y}_i^{(0)}$ and $g_i(\boldsymbol{x}_i^{(-1)})$ are initialized to zero. More properties and convergence analysis of DecenSGD-GT can be found in [25].

Combination with MATCHA. As shown in (6) and (7), there are two consensus steps in DecenSGD-GT. We choose to apply MATCHA on both of them. The activated sparse graph sequence is generated in the same way as original MATCHA and at each iteration, (6) and (7) use the same activated topology. The mixing matrix at the k-th iteration is defined as $\boldsymbol{W} = \boldsymbol{I} - \alpha \sum_{j=1}^{M} \mathsf{B}_j^{(k)} \boldsymbol{L}_j$. Unlike the original MATCHA algorithm where α is selected via optimizing (4), we found that α should be manually tuned when combining MATCHA and gradient tracking.

Experimental Results. We present the training curves of the combination of MATCHA and gradient tracking in Figure 4. Observe that although DecenSGD-GT can achieve lower loss value than vanilla DecenSGD at the end of training, it costs more wall-clock time since it requires doubled communication at each iteration. When applying MATCHA on the top of DecenSGD-GT, the training time can be significantly reduced and the training loss is still better than vanilla DecenSGD.



(a) Training loss (b) Test accuracy

Fig. 4: Combination of MATCHA and gradient tracking.

VI. Conclusion

In this paper, we have explored the error-runtime trade-off in decentralized optimization by extending the MATCHA framework to incorporate importance based matchings sampling and to decentralized optimization algorithms involving gradient tracking. We have empirically demonstrated the performance of our proposed extensions on deep network training for image recognition tasks. Future work includes extending MATCHA to network topologies having directed links.

References

[1] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," *arXiv preprint arXiv:1905.09435*, 2019.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

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

[5] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[6] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5336–5346.

[7] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[8] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," *arXiv preprint arXiv:1710.06952*, 2017.

[9] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9850–9861.

[10] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," *arXiv preprint arXiv:1710.09854*, 2017.

[11] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," *arXiv preprint arXiv:1705.07878*, 2017.

[12] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed SGD with slow momentum," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkxJ8REYPH

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2592–2600.

[14] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server." in *OSDI*, vol. 14, 2014, pp. 583–598.

[15] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and E. P. Xing, "Poseidon: An efficient communication architecture for distributed deep learning on {GPU} clusters," in *2017 {USENIX} Annual Technical Conference ({ USENIX } { ATC } 17)*, 2017, pp. 181–193.

[16] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *International Conference on Machine Learning*, 2019.

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

[18] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Communication-efficient distributed strongly convex stochastic optimization: Non-asymptotic rates," *arXiv preprint arXiv:1809.02920*, 2018.

[19] K. Tsianos, S. Lawlor, and M. G. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," in *Advances in neural information processing systems*, 2012, pp. 1943–1951.

[20] S. Kar and J. M. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, 2008.

[21] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[24] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *Mathematical Programming*, pp. 1–49, 2020.

[25] R. Xin, U. A. Khan, and S. Kar, "An improved convergence analysis for decentralized online stochastic non-convex optimization," *arXiv preprint arXiv:2008.04195*, 2020.