Live Gradient Compensation for Evading Stragglers in Distributed Learning

Jian Xu
Tsinghua-Berkeley Shenzhen Institute, Tsinghua University

***Abstract*—The training efficiency of distributed learning systems is vulnerable to stragglers, namely, those slow worker nodes. A naive strategy is performing the distributed learning by incorporating the fastest K workers and ignoring these stragglers, which may induce high deviation for non-IID data. To tackle this, we develop a Live Gradient Compensation (LGC) strategy to incorporate the one-step delayed gradients from stragglers, aiming to accelerate learning process and utilize the stragglers simultaneously. In LGC framework, mini-batch data are divided into smaller blocks and processed separately, which makes the gradient computed based on partial work accessible. Extensive simulation experiments of image classification on CIFAR-10 dataset are conducted, and the numerical results demonstrate the effectiveness of our proposed strategy.**

***Index Terms*—Straggler, Distributed Learning, Non-IID, Gradient Compensation**

I. Introduction

Distributed implementations of gradient-based methods [1], [2] have been essential for training large machine learning models on massive datasets, e.g., deep neural networks for image classification and speech recognition [3], [4]. Typical distributed learning architecture consists of a parameter server (PS) and distributed worker nodes – the workers compute and send local gradients to PS in parallel, while the PS aggregates the gradients and then broadcasts back to workers to update local parameters [5]. In synchronous settings, the time overhead of each iteration in such system architecture is subject to the stragglers, i.e., slow or unresponsive workers that are caused by performance variability as well as unexpected incidents like network congestion and hardware failures.

Much research attention has recently focused on mitigating stragglers either by leveraging coding-theoretic techniques [10]–[14] or by utilizing partial work completed by stragglers [15], [16]. In particular, it is possible to collect gradients from only the fast workers and discard the computations on stragglers, while still achieving convergence [8], [17], [18]. We refer to this naive strategy as K-SGD. However, this approach relies on the IID assumption[1] of training data and is shown to induce gradient/sampling bias in more general settings. Another line of work leverages gradient coding to obtain the exact gradient value despite of the stragglers [7], [9], [19]–[21]. However in such approaches, a certain amount of overhead (computation/data duplication) must always be present, in order to successfully address the worst-case stragglers. Further, gradient coding schemes are brittle in the sense that they work perfectly only up to a fixed number of stragglers.

[1]Training data among workers are independent and identically distributed (IID), so that local gradient is an unbiased estimation of global gradient.

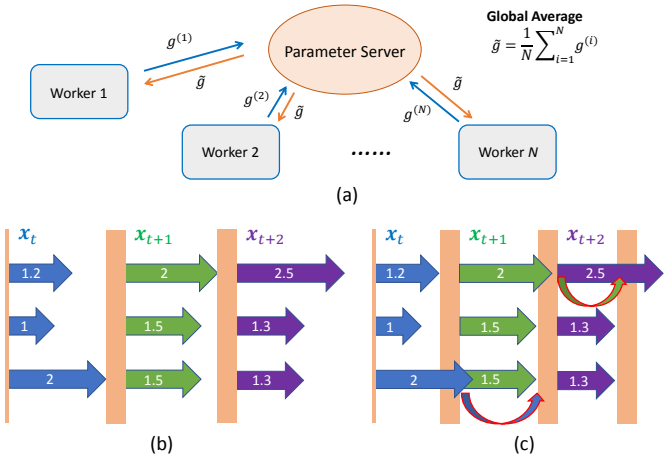


Fig. 1. Distributed learning with a parameter server. (a) System architecture. (b) Full-SGD method. (c) LGC-SGD method. LGC-SGD can significantly reduce training time overhead while maintaining convergence, by evading and compensating for stragglers.

In this project, we propose a novel Live Gradient Compensation (LGC) framework for mitigating stragglers in distributed learning, which is built on top of K-SGD with partial work tolerance and gradient bias compensation mechanisms. The central idea can be seen through a simple example shown in Fig. 1 with one PS and three workers. In full synchronous SGD (Full-SGD), the training time overhead of each iteration is determined by the worst performing worker, which results in a total training time of $2 + 2 + 2.5 = 6.5$ for all three iterations. On the other hand, we can bypass the slowest worker in each iteration (thus collecting the results only from the $K = 2$ fastest workers) and then compensate for the impact by performing a combined gradient update in the next iteration. This reduces the total training time to $1.2 + 1.5 + 1.3 = 4$, albeit minor gradient noise introduced due to the one-step delay of compensation. This motivates the design of LGC-SGD. We would like to emphasize that in contrast to the gradient coding approach, LGC-SGD does not require any extra computation or data storage overhead. While gradient compensation has been developed as a technique in gradient compression [22]–[26], we make novel use of that to mitigate stragglers in distributed learning.

The proposed LGC framework is evaluated on CIFAR-10 dataset with various cases by changing the level of non-IID and the straggling period length, which verify the effectiveness

in speeding up training while keeping a high model generalization ability. Our simulation results show that ∼35% saving in training time can be obtained with only slight accuracy loss. To summarize, the main contributions of this project are as follows:

- A new distributed training strategy based on one-step delayed gradient compensation, namely LGC-SGD, is proposed for evading stragglers and utilizing partial work.
- The effectiveness of proposed LGC-SGD is verified on CIFAR-10 dataset, where LGC-SGD can significantly reduce training time while converging to the same training error compared with Full-SGD.

II. The Proposed LGC Framework

A. System Model for Distributed Learning

We focus on distributed optimization of a non-convex problem on non-IID data. We assume that training data are distributed over multiple worker nodes in a network, and all workers jointly optimize a shared model based on local data. Mathematically, the underlying distributed optimization problem can be formalized as follows:

$$\min_{\mathbf{x}\in R^d} F(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\xi_i\sim D_i}\left[F(\mathbf{x};\xi_i)\right] \tag{1}$$

where N is the number of workers, D_i denotes the local dataset of i-th worker and could have different distribution from other workers (which means the IID assumption is relaxed), and $F(\mathbf{x};\xi_i)$ denotes the local loss function given shared model parameters $\mathbf{x}$ and training data ξ_i (one sample point or a mini-batch) sampled from D_i of the i-th worker.

We make all workers initialized to the same point $\mathbf{x_0}$, then Full-SGD can be employed to solve the problem. At each iteration, the i-th worker randomly draws a mini-batch samples ξ_i from D_i, and computes local stochastic gradient with respect to global shared parameter $\mathbf{x}_t$:

$$g_t^{(i)} = g(\mathbf{x}_t;\xi_i) = \frac{1}{|\xi_i|}\sum_{j=1}^{|\xi_i|}\nabla F(\mathbf{x}_t;\xi_i^{(j)}) \tag{2}$$

The parameter server aggregates all the local gradients to get a global gradient:

$$\tilde{g}_t = \frac{1}{N}\sum_{i=1}^{N} g_t^{(i)} \tag{3}$$

Then the result will be broadcast to all worker nodes to update their local models and start a new iteration. This process will repeat until the model converges.

B. Our Proposed Solution

The proposed LGC framework is described in Algorithm 1 and the training process is illustrated in Fig. 2. Specifically, for each worker, mini-batch of data are divided into s smaller blocks and computed incrementally. Slow worker may not be able to completely finish its task by next iteration, but perhaps it has processed r of s blocks and can send an approximate

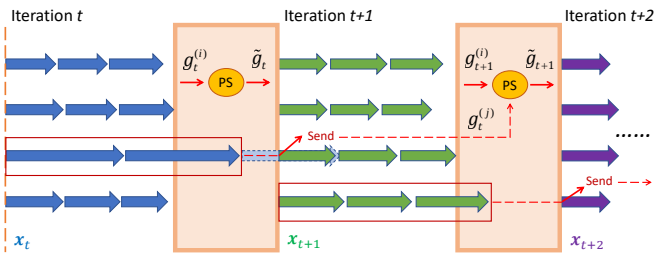


Fig. 2. Illustration of the workflow of proposed LGC-SGD. Mini-batch data are divided into multiple blocks and processed in order.

result afterwards. At each iteration, the server collects the fastest K fresh gradients that evaluated on entire mini-batch as well as any delayed gradients from the previous iteration, and obtains a parameter update by combining average fresh gradient and a proper compensation for gradient bias of the previous iteration. The gradient bias induced by ignoring stragglers is quantified by Eq. (7). Meanwhile, the remaining slow workers are allowed to continue computing until the entire mini-batch are evaluated or new global update is received, after which the delayed gradients are sent to the server for bias compensation. It is worth noting that the delayed gradients could be computed based on full mini-batch data or a portion, depending on the computation speed of stragglers. Suppose the i-th worker computed r blocks of samples within an iteration and the mini-batch size is m, the variance of local stochastic gradient evaluated on a sample point is bounded by σ^2. Then the expectation and variance of gradient satisfy the following:

$$\mathbb{E}\left[g_t^{(i)}\right] = \nabla F_i(\mathbf{x}_t) \tag{4}$$

$$\mathbb{E}\left[\left\|g_t^{(i)} - \nabla F_i(\mathbf{x}_t)\right\|^2\right] \le \frac{s}{r}\cdot\frac{\sigma^2}{m} \tag{5}$$

It means that the gradient based on partial work is still a reliable estimation, equivalent to that obtained by scaling down mini-batch size. Therefore, if gradients of stragglers could be measured and compensated, the training performance can be guaranteed. That is the main motivation of the proposed strategy. Considering the enlarged variance may influence the training process, we adopt a linear scaling rule on the gradient to address this issue as Eq. (6), which is similar to the linear scaling rule on the learning rate as [33]. The scaling operation may reduce the magnitude of gradient, but not affect the estimation of direction, having the effect of variance reduction.

$$g_t^{(i)} = \frac{r}{s}\cdot\frac{1}{r}\sum_{k=1}^{r} g(\mathbf{x}_t;\xi_i[k]) \tag{6}$$

Let $\tilde{g}_t$ and $\tilde{g}'_t$ denote the average gradient of N and K workers respectively, and S_t the set of fastest K workers. Then the gradient bias caused by ignoring stragglers can be obtained

Algorithm 1 Live Gradient Compensation SGD

Input: learning rate η, total iteration T, partition number s, mini-batch size m, total workers N, threshold K
Initial: $\mathbf{x}_0 \in R^d$; $e_{-1} = 0$
for $t = 0, 1, ..., T-1$ **do**
  On each worker *i* :
  divide mini-batch samples ξ_i into s partitions
  $g_t^{(i)} = 0, r_t^{(i)} = 1$
  while $r_t^{(i)} \leq s$ **and** update not received **do**
    $g_t^{(i)} = g_t^{(i)} + g(\mathbf{x}_t; \xi_i[r_t^{(i)}])$
    $r_t^{(i)} = r_t^{(i)} + 1$
  end while
  send $g_t^{(i)} = g_t^{(i)}/s$ to server
  wait for global update $\tilde{g}_t$ from server
  update local model: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\tilde{g}_t$
  On server:
  collect fastest K gradients from workers
  average: $\tilde{g_t}' = \frac{1}{K}\sum_{i\in S_t} g_t^{(i)}$
  if $t \geq 1$ **then**
    collect delayed $(N-K)$ gradients from stragglers
    calculate compensation:
    $e_{t-1} = \frac{1}{N}\sum_{i\notin S_{t-1}} g_{t-1}^{(i)} - \frac{N-K}{N}\tilde{g}'_{t-1}$
  end if
  obtain the global update: $\tilde{g}_t = \tilde{g_t}' + e_{t-1}$
  send $\tilde{g}_t$ to all workers
end for

as follows:

$$e_t = \tilde{g}_t - \tilde{g}'_t = \frac{N-K}{N}\left[\frac{1}{N-K}\sum_{k\notin S_t} g_t^{(k)} - \frac{1}{K}\sum_{k\in S_t} g_t^{(k)}\right] \tag{7}$$

The first term in parentheses in the last step yields the average gradient of stragglers, and the second term is the aforementioned average gradient of fastest K workers.

III. Simulation

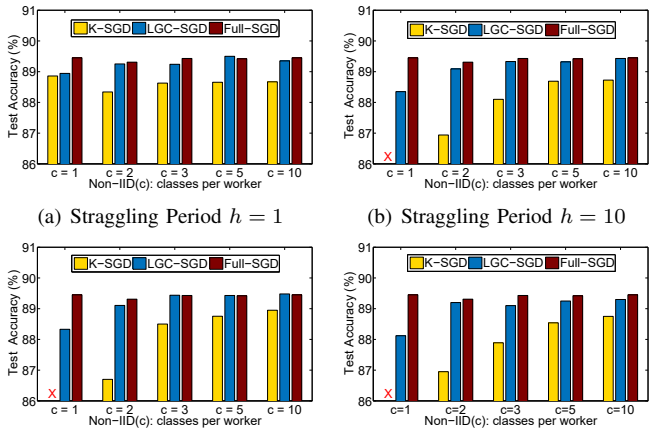
A. Experimental Setup

Dataset and Model. The well-known CIFAR-10 dataset contains 10 object classes with 50,000 training samples and 10,000 testing samples. Here we use the notation non-IID(c) to mean that each worker is allocated with c categories of samples. We constructed our model based on VGG-11 [35], where we adjusted the neural network to fit the input size and kept only one fully connected layer without dropout layer.

Simulation Setting. To simulate the straggling behaviors, we use shifted exponential distribution to generate the per-iteration computation time, on which the stragglers are identified. The mini-batch size is set to 32 and each training algorithm is run for total 60 epochs. The initial learning rate is set to 0.1 and divided by 10 after 30 epochs. The momentum is set to 0.9 and weight decay is set to 0.0005. For K-SGD and LGC-SGD, the first 2 epochs are run in Full-SGD fashion as warmup. All algorithms are implemented in PyTorch.

B. Numerical Results

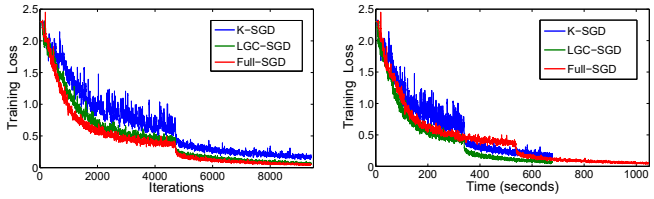
We conduct simulations for N = 10 workers and choose K = 7 as the straggler threshold value. As [21] we introduce dependency between stragglers across iterations by fixing per-iteration computation time for h iterations, after which the computation time for each worker will be generated randomly and independently again. To begin with, we simply assume that the stragglers can complete all computations before the beginning of next iteration. We repeated each experiment for three times and reported the average result. Fig. 3 shows the main results of model test accuracy under different level of non-IID for different straggling behaviors.



(a) Straggling Period $h = 1$ (b) Straggling Period $h = 10$

(c) Straggling Period $h = 50$ (d) Straggling Period $h = 150$

Fig. 3. Test accuracy comparison of training methods under various level of non-IID and different straggling period length. LGC-SGD outperforms K-SGD and catches up with Full-SGD.



(a) Training loss v.s. iterations (b) Training loss v.s. time

Fig. 4. The convergence of training loss over iterations and time. LGC-SGD has faster training speed and comparable convergence error than Full-SGD.

1) Robustness to Non-IID: We reduce the value of c to generate data distributions with increasing non-IID level and test the robustness of LGC-SGD. It can be found that Full-SGD performs well despite of data skewness among workers. However, the K-SGD has lower test accuracy, especially under large data skewness and persistent straggling behavior, such as non-IID(2) and non-IID(3) in Fig. 3(b)(c)(d), and even diverges under non-IID(1). In contrast, the proposed

LGC-SGD can effectively leverage gradient compensation to eliminate gradient bias during the training process, achieving comparable model generalization ability as Full-SGD after the same number of training iterations.

2) Robustness to Straggling Period: We also change the h to simulate different straggling behavior to investigate the impact on training. Fig. 3 provides the test results of 3 cases, where $h = 1$ means the stragglers are random and independent every iteration while $h = 10$ means that stragglers are randomly selected every 10 iterations. It's interesting to see that when $h = 1$, the test result of K-SGD is less affected by non-IID, verifying our result in Theorem 1. As we increase the value of h, the model trained by K-SGD results in lower test accuracy due to gradient bias induced by discarding the computation of stragglers. However, the model trained by LGC-SGD still achieves nearly equal test result to Full-SGD.

3) Efficiency Improvement: Take the case of $c = 3$ and $h = 10$, the convergence of training loss in terms of the number of iterations and generated wall-clock time are plotted in Fig. 4, where we use $X_i \sim 0.05 + Exp(0.02)$ to generate and simulate per-iteration time. The Full-SGD can achieve lowest convergence error at the cost of longer overall training time, while K-SGD can save per-iteration time but result in higher convergence error. However, the LGC-SGD can have the best of both worlds by significantly reduce training time as K-SGD while achieving almost the same training loss as Full-SGD. The simulation result demonstrates that LGC-SGD can reduce training time by up to $\sim$35% compared with the Full-SGD, while achieving nearly the same convergence error.

C. Discussions

Finally, we perform analysis on other factors that may affect the performance of LGC-SGD. Specifically, we evaluate the behaviors of LGC-SGD with different choices of K, different percentages of the mini-batch data that are processed by stragglers for each iteration as well as different system sizes. In this part, we fix $c = 3$ and $h = 10$ while the results are similar for other values of c and h.

1) Tradeoff through K: As mentioned previously, the selection of threshold K is non-trivial and highlights an important tradeoff between minimizing training error and training time. We gradually increased the value of K from 5 to 10 for fixed number of iterations to plot the optimal frontier of training time and training loss as Fig. 5, in which different colors represent different values of K and the red digital labels denote test accuracy. It can be found that as K decreases, the model test accuracy of K-SGD degrades substantially while LGC-SGD only has slight accuracy loss. It experimentally reveals that the selection of K is an explicit tradeoff bewteen training time and model accuracy. And the optimal frontier achieved by LGC-SGD significantly improves that of K-SGD.

2) Partial Work: We artificially make the slow workers only process different percentages of mini-batch data to study the influence of tolerating partial work of stragglers. We keep the batch-size as $m = 32$ and set the number of blocks as $s = 4$, then simulations are conducted under fixed and random

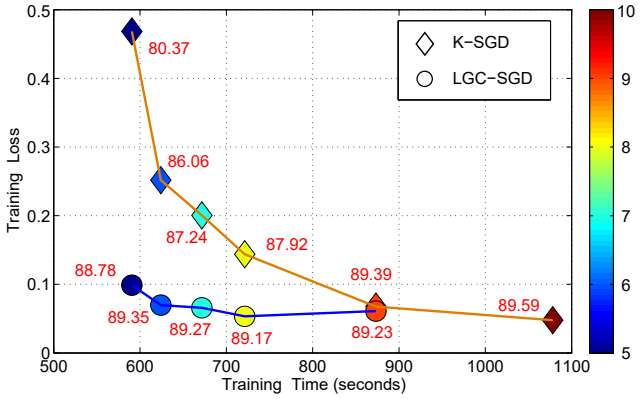


Fig. 5. Training loss and training time as well as test accuracy for various value of K under fixed training iterations.

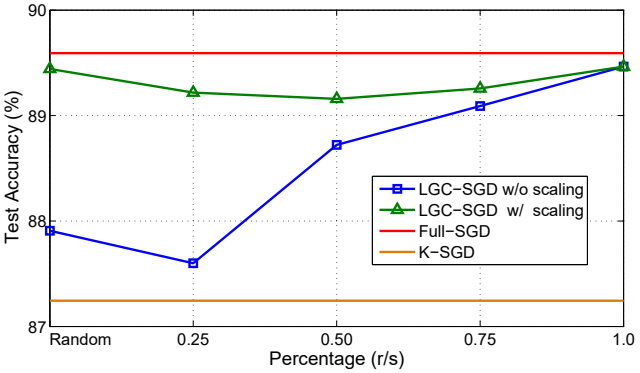


Fig. 6. Tolerating partial work by linear scaling on the gradient of stragglers.

amount of partial work ($r = 1 \sim 4$). Particularly, we compare the results of LGC-SGD with and without linear scaling on the delayed gradients of stragglers as shown in Fig. 6. It can be seen that LGC-SGD with linear scaling on delayed gradient can effectively utilize the partial work of stragglers.

IV. Conclusion

In this work, we proposed a live gradient compensation framework to evade stragglers in distributed learning system. It can overcome the drawbacks of naively ignoring stragglers in synchronous SGD and unlike gradient coding approaches does not require any extra computation/storage overhead. We particularly investigated the performance of LGC-SGD on non-IID training data, providing theoretical analysis on the convergence error and quantifying the tradeoff by selecting different straggler threshold value. Simulation results on CIFAR-10 dataset verified our theoretical findings and demonstrated the effectiveness of proposed LGC-SGD. Future work includes developing strategies to dynamically adjust different hyper-parameters in LGC-SGC in practical distributed systems.

REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1223–1231.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] M. Li, D. G. Andersen, A. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," *Advances in Neural Information Processing Systems (NIPS)*, vol. 1, pp. 19–27, 2014.

[6] J. Dean and L. A. Barroso, "The tail at scale," *Communications of The ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning (ICML)*, 2017.

[8] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd," in *AISTATS*, 2018.

[9] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *Journal of Machine Learning Research*, vol. 20, no. 72, pp. 1–47, 2019.

[10] C. Karakus, Y. Sun, S. N. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5434–5442.

[11] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *IEEE International Parallel and Distributed Processing Symposium Workshops(IPDPS)*, 2018, pp. 857–866.

[12] K. Lee, M. Lam, R. Pedarsani, D. S. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[13] H. Park, K. W. Lee, J.-Y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1630–1634.

[14] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr, "Polynomially coded regression: Optimal straggler mitigation via data encoding," *arXiv:1805.09934*, 2018.

[15] N. Ferdinand, H. Al-Lawati, S. Draper, and M. Nokleby, "Anytime minibatch: Exploiting stragglers in online distributed optimization," in *International Conference on Learning Representations (ICLR)*, 2019.

[16] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," in *IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2729–2733.

[17] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv:1604.00981*, 2016.

[18] S. Kas Hanna, R. Bitar, P. Parag, V. Dasari, and S. El Rouayheb, "Adaptive distributed stochastic gradient descent for minimizing delay in the presence of stragglers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[19] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *International Conference on Machine Learning (ICML)*, 2018, pp. 5606–5615.

[20] H. Wang, Z. B. Charles, and D. S. Papailiopoulos, "Erasurehead: Distributed gradient descent without delays using approximate gradient coding," *arXiv:1901.09671*, 2019.

[21] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 277–291, 2020.

[22] S. U. Stich, J. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[23] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2019.

[24] S. Zheng, Z. Huang, and J. T. Kwok, "Communication-efficient distributed blockwise momentum sgd with error-feedback," in *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[25] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "`DoubleSqueeze`: Parallel stochastic gradient descent with double-pass error-compensated compression," in *International Conference on Machine Learning (ICML)*, 2019, pp. 6155–6165.

[26] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: Sgd with delayed gradients," *Journal of Machine Learning Research*, vol. 21, no. 237, pp. 1–36, 2020.

[27] F. Niu, B. Recht, C. Re, and S. J. Wright, "Hogwild! a lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 693–701.

[28] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for dnn training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[29] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2737–2745.

[30] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "An asynchronous mini-batch algorithm for regularized stochastic optimization," *IEEE Trans. Automat. Contr.*, vol. 61, no. 12, pp. 3740–3754, 2016.

[31] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv:1806.00582*, 2018.

[32] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations (ICLR)*, 2020.

[33] P. Goyal, P. Dollár, R. B. Girshick, and P. Noordhuis, "Accurate, large minibatch SGD: training imagenet in 1 hour," *arXiv:1706.02677*, 2017.

[34] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.