Re: Distributed Distillation for On-Device Learning

Shufan Wang
EPFL
shufan.wang@epfl.ch

Yian Wang
EPFL
yian.wang@epfl.ch

Ke Ma
EPFL
ke.ma@epfl.ch

Abstract—**In this project, we reproduce the work introduced in the paper Distributed Distillation for On-Device Learning [1] as part of the 2020 NeurIPS Reproducibility Challenge. In this paper, Ilai Bistritz et. al [1] introduce a distributed distillation algorithm where devices communicate and learn from soft-decision outputs that are inherently architecture-agnostic and scale only with the number of classes. This algorithm provides performance improvement while requiring only a little communication overhead compared to traditional distributed training. During the reproducing experiments, we mainly focus on the following topics: 1) if the baselines chosen in this paper are appropriate and state-of-the-art. 2) if the performance improvement and communication reduction are as good as described in this paper. 3) if this training algorithm is compatible with different network compression strategies. We also find plausible typos or problems in this paper, which would also be discussed. Overall, we think Distributed Distillation is a good training strategy with reliability and compatibility.**

I. Introduction

In recent years, there's an increasing need to transfer users' data to the central server for training, which entails massive communication overhead and privacy issues. On the other hand, one device alone does not have enough data to achieve state-of-the-art performance. Hence it's crucial to employ on-device learning to keep both the data and the training on the device by coordinating collaborative training across devices. However, the traditional on-device learning, like Federated Learning [2], [3], [4] and distributed SGD based DNN [5], [6] can only process devices that have the same model architecture, and the communication overhead is large. While Co-distillation methods [7] overcome these problems, they don't have the communication constraints or privacy restrictions of the on-device setting [8], [9].

To this end, the studied article introduced a new training algorithm called Distributed Distillation(D-Distillation), incorporating devices with different DNN architectures or even other types of classifiers. At the same time, this algorithm reduces the communication overhead while achieving comparable accuracy compared to Distributed based SGD training. Finally, Ilai Bistritz et. al also conduct an in-depth mathematical proof of the algorithm's convergence.

In section II, we introduced the mathematical formulation of the problems and how to tackle the difficulties. In section III, the replication of experiments and some ablations in compression are summarized. While some outputs are inconsistent with those in the original article, the obtained results provide the validity and reliability of the Distributed Distillation Algorithm. In section IV, we present a discussion about the problems we meet in the reproducing process, and we provide a rethinking about the details in the algorithm. Finally, in section V, we conclude the results and findings of replication. For simplicity, we will use Silo-SGD (training on the private dataset with no communication), D-SGD (distributed based SGD training), and D-Dist (Distributed Distillation) in the following.

II. Problem Formulation

The whole problem can be formulated as a classification task which is to learn a function that maps every input data point to the correct class out of K options. Each device n has access to its own private data $\mathcal{D}_n = \{\widetilde{x}_i^n\}_{i=1}^{M_n}$ of inputs with corresponding labels (hard decisions) $y(\widetilde{x}_i)$ which are one-hot vectors over the set of classes. Devices communicate through a reference dataset that all devices have access to, denoted $\mathcal{D}_r = \{x_j\}_{j=1}^{Q}$. Each device n has a classification model with p_n parameters, $\theta^n \in \mathbb{R}^{p_n}$, which produces a probability vector over the classes. This probability vector is also called soft-decision, denoted $s_n(\theta^n, x) : \mathbb{R}^{p_n} \times (\mathcal{D}_n \cup \mathcal{D}_r) \to \Delta^K$. Then we can define the local loss function of device n as

$$\mathcal{L}_n\left(\boldsymbol{\theta}^n\right) \triangleq \sum_{\tilde{x} \in \mathcal{D}_n} \mathcal{L}\left(s\left(\boldsymbol{\theta}^n, \tilde{x}\right), \boldsymbol{y}(\tilde{x})\right) \qquad (1)$$

Let $\mathcal{X}$ be the unknown distribution of a labeled data point $(\tilde{x}, \mathbf{y}(x))$, the goal is to minimize the total generalization error in the network:

$$\mathcal{L}^* \triangleq \underset{\theta^1,\ldots,\theta^N}{min} \mathbb{E}_{\tilde{x}\sim\mathcal{X}} \left\{ \sum_{n=1}^{N} \mathcal{L}(s(\theta^n, \tilde{x}), \mathbf{y}(\tilde{x})) \right\} \qquad (2)$$

To optimize (2) based on the private training samples without sharing them between devices, the authors proposed the following distributed training objective:

$$\begin{aligned} \min_{\boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^N} \sum_{n=1}^{N} & \left[\mathcal{L}_n\left(\boldsymbol{\theta}^n\right) + \right. \\ & \left. \rho \sum_{x \in \mathcal{D}_r} \left\| \frac{1}{N} \sum_{m=1}^{N} s\left(\boldsymbol{\theta}^m, x\right) - s\left(\boldsymbol{\theta}^n, x\right) \right\|^2 \right] \end{aligned} \qquad (3)$$

where $\rho > 0$ is a regularization parameter and the second term is a distillation regularization term. Solving this equation means that the algorithm converges to the following set of distillation stationary points:

$$\mathcal{S}_d^* = \left\{ \boldsymbol{\theta} \mid \nabla_{\boldsymbol{\theta}^n} \left[\mathcal{L}_n \left(\boldsymbol{\theta}^n \right) + \rho \sum_{x \in \mathcal{D}_r} \left\| \frac{1}{N} \sum_{m=1}^{N} s \left(\boldsymbol{\theta}^m, x \right) - s \left(\boldsymbol{\theta}^n, x \right) \right\|^2 \right] = 0, \forall n \right\} \quad (4)$$

where $\mathcal{D}_r$ is the set of size Q unlabeled reference data. To design a distributed algorithm that provably converges to the set in (4) for every strongly connected, directed communication graph, the authors then defined the communication graph and communication matrix, whose values are the weights devices assign to their peers to incorporate their received information.

The communication graph is represented by a randomly generated matrix $W_{m,n}$. Let $G = (\upsilon, \epsilon)$ be the directed communication graph, where $\upsilon = 1, ..., N$ and device n can send messages to device m if and only if $(n, m) \in$, assuming G is strongly connected. Let $W = \omega_{n,m} \geq 0$ be the communication matrix of G such that $\omega_{n,m} > 0$ if and only if $(n, m) \in$, assuming W is doubly stochastic, so $\Sigma_{m=1}^{N} \omega_{n,m} = 1$ and $\Sigma_{m=1}^{N} \omega_{m,n} = 1$ for each n, and that $\omega_{n,n} > 0$ for all n.

In order to converge to the distillation stationary point (4), the authors proposed to solve the following problem:

$$\min_{\substack{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N \\ \boldsymbol{z}^1, \ldots, \boldsymbol{z}^N}} \sum_{n=1}^{N} \left[\mathcal{L}_n \left(\boldsymbol{\theta}^n \right) + \rho \sum_{x \in \mathcal{D}_r} \left\| \boldsymbol{z}^n(x) - s \left(\boldsymbol{\theta}^n, x \right) \right\|^2 \right] \quad (5)$$

$$\text{s.t. } \boldsymbol{z}^n(x) = \boldsymbol{z}^m(x), \forall (m, n) \in \mathcal{E} \text{ and } \forall x \in \mathcal{D}_r$$

By solving (5), the devices learn a solution to (3).

Then, the studied article introduced Distributed Distillation Algorithm (D-Dist). As for the details, We refer to Algorithm 1 in [1].

III. Experiments and Analysis

In this section, we first describe our simulation settings of both network training and distributed communication networks. Then we provide the results we reproduce together with a detailed analysis of the findings and problems we come across during the experiments. Given limited training resources, we only reproduce the D-Dist and Silo-SGD results, and we directly take the D-SGD results from the original paper as the baseline. All the experiments are conducted on two NVIDIA-TITAN-X and one GTX-1060M. The main points we want to validate is: 1) if the baselines used in this paper are valid, reasonable and state-of-the-art. 2) if the improvement of using the algorithm can be reproduced (both in the test accuracy and communication reduction). 3) if the algorithm is naturally compatible with different compression strategies, as heterogeneous compression strategies may be chosen among different distributed participants in practice.

A. Settings

Networks Training We employed the same networks (LeNet5 [10], ResNet2, ResNet8, ResNet14 [11]) and datasets according to this paper. Also, we followed all the specifications mentioned in the appendix, such as pure SGD optimizer, no data augmentation, and other training hyper-parameters. Unlike the quantization used in this paper, we test the performance when irregular weight pruning [12], which prunes the weights with the least absolute value, is used with different pruning ratios across the networks.

Communication backend: In each trail, we generate a random connection matrix according to the paper to establish peers' connection. (We choose the minimum degree and maximum degree of each node according to the paper. But how the degree is sampled is not mentioned in the original paper. Here we manually set the probability of the degree proportional to the absolute value of the degree. For example, if $Degree_{min} = 1, Degree_{max} = 3$, then we set $Degree = 1 \quad with \quad probability \quad \frac{1}{6}, \quad Degree = 2 \quad with \quad \frac{2}{6}, \quad and \quad Degree = 3 \quad with \quad \frac{3}{6}$.) We use a naive communication mechanism. Each node broadcasts a UDP packet containing its softmax predictions on public reference data to its directly connected peers. The receiving function will finish only if all the message is received. As we conduct the distributed simulation on the local server, the network condition is stable, and no packet loss happened during our experiments.

Table I
Communication Overhead of 8 Networks trained on evenly distributed CIFAR10. Here we calculate the communication overhead by the size of tensors that peers broadcast in each batch. The slight difference between our calculation and that in the original paper may come from the different tensors representations in the packet.

Communication Overhead between Two Peers in 800 Epoch / MB			
Network	LeNet5	ResNet8	ResNet14
D-Dist	1.129	1.129	1.129
D-SGD	4989.71	9994.34	20920.90

B. Reproduce Results

We first conduct the Silo-SGD training over MNIST [10] and CIFAR10 [13]. To our surprise, we found a huge gap between our results with the results reported in the original

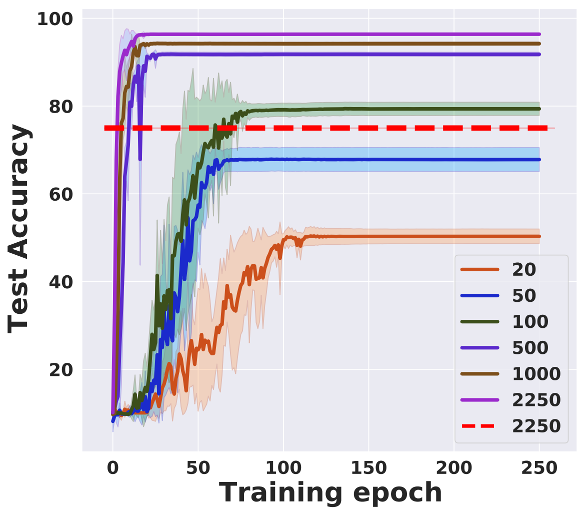

Figure 1. **Test Accuracy and Standard Deviation** of a LeNet5 trained with different amount of randomly selected images from MNIST. Each experiment is conducted for 5 trails. The **Red** dashed line represents the final accuracy in the original paper with 2250 training examples.

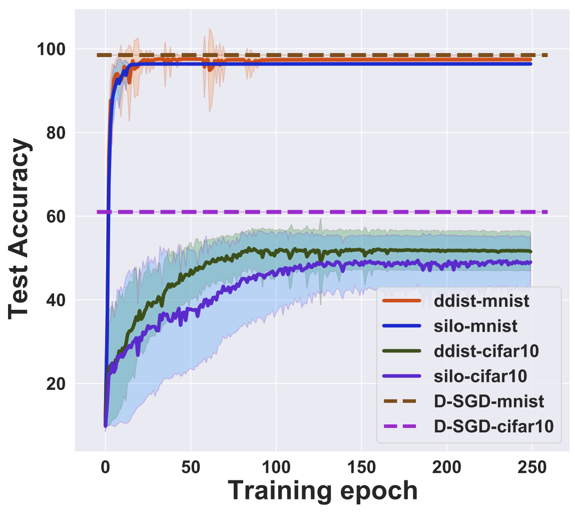

Figure 2. **Test Accuracy and Standard Deviation** of 16 LeNet5 trained on randomly selected images from MNIST, and 8 models (4 LeNet5 and 4 ResNet8) on CIFAR10. All the training hyper-parameters follow the original paper without further tuning. The brown dashed line and the purple dashed line represent the D-SGD performance according to the original paper, respectively.

paper on MNIST. The private data amount used in the original paper is 2250 images per participant. We conduct Silo-SGD with different amounts of private data, ranging from only 20 to 2250. As indicated in Fig. 1, a participant with only 100 images can reach a higher accuracy than the baseline in this paper (the red dashed line). On the other hand, a participant following the original setting (with 2250 images) can reach 95% test accuracy effortlessly (knowing that in the original paper, training with 2250 images only leads to around 75% test accuracy). Here we only display the first 250 epochs according to the same settings in this paper. We found the network is fully trained, and no noticeable performance improvement appears between 250 $\sim$ 800 epoch. Our results with 5 trails displayed only a little variance, so we wonder why the baseline is critically low in this paper. (We firstly suspect our results, but we also find resources that prove our results. we refer readers to a report[14] for further details about baselines on MNIST.) This issue does not appear on CIFAR10.

Then we conduct D-Dist according to the same settings in Figure.2 and Figure.3 of the original paper. As indicated in Figure.2 the D-Dist will improve the test accuracy compared to Silo-SGD. It will bring more improvement on CIFAR10 than on MNIST. However, it does not get a non-trivial improvement compared to D-SGD. As shown in Fig. 2, the brown and the purple dashed lines represent the D-SGD performance in the original paper respectively, which is always higher than D-Dist (this is actually shown in Figure.4 in the original paper as well, where D-SGD is nearly 10% higher than D-Dist on 8 ResNet14. However, it is not explained by the authors). We conjecture that more information is exchanged between models in D-SGD than in D-Dist, thus D-SGD still outperforms D-Dist on some occasions. Another observation not mentioned in the original paper is that we find the variance of participants in D-Dist is much lower than Silo-SGD. This means all the participants benefit from others' knowledge and get some improvement. For communication reduction, as shown in Table.I, we reach the same results that D-Dist is far more efficient, as the communication only contains a batch soft predictions, it will reduce the expense by a multiple of 4,419 on LeNet5 and 18,528 on ResNet14. If we cannot totally satisfy the D-SGD communication overhead, it will be a good choice to use D-Dist to improve the performance.

As network pruning is widely deployed in practice, we try to test if this algorithm is naturally compatible with other pruning methods, such as irregular weight pruning [12]. We choose this compression strategy mainly because it is efficient to deploy and can maintain both low accuracy degradation and high compression ratio compared to structured pruning [15]. Here we test on 8 ResNet14 with compression different ratios in $[20\%, 40\%, 60\%, 80\%]$, namely, each ratio with two models. Here the pruning ratio means we prune the same percentage of weights with the least L_1-norm across all layers consistently. As shown in Fig. 3, this algorithm works well among networks with different pruning ratios. As shown by the mean test accuracy and the standard deviation, all pruned networks shall be improved. This validates that, except for

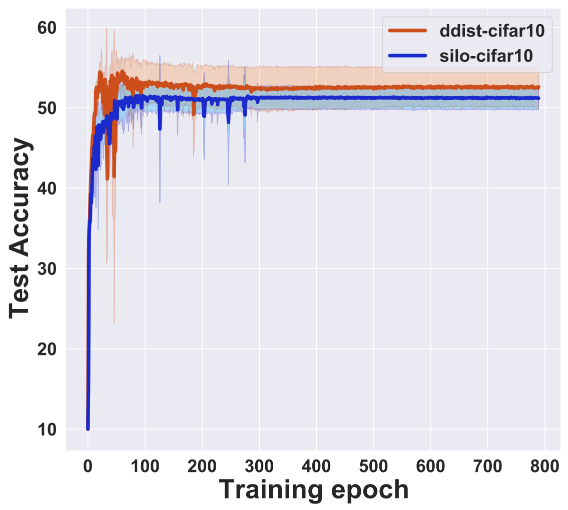

Figure 3. **Test Accuracy and Standard Deviation** of 8 ResNet14 trained with D-Dist or Silo-SGD on CIFAR10. All the training hyper-parameters follow the settings in the original paper in Figure.4 without further tuning. Obviously, D-dist lead to accuracy improvement on Silo-SGD.

quantization which is mentioned and tested in the original paper, network pruning is also compatible to this network compression algorithm.

In summary, The D-Dist algorithm in this paper will improve the test performance compared to Silo-SGD, and reduce the performance variance among the participants in the network. And this algorithm is compatible with different network pruning methods. However, there is a clear problem of the baseline on MNIST in this paper, and the accuracy degradation compared to D-SGD is not explained by the authors, which embodies some kind of disadvantages of this paper.

IV. Discussion

While this paper provide a detailed description of the training settings in the appendix, the algorithm itself described in this paper is somewhat ambiguous and even contradictory. As stated in Algorithm 1 in the paper, the peers **only communicate at the beginning of each epoch**, different from D-SGD where peers communicate at the beginning of each iteration, namely each batch. However, in the legend of Figure.1 in the original paper, it states the **communication is conducted in each iteration (batch)**. This is quite misleading because if there is no typo in Algorithm 1, then the total communication reduction compared to D-SGD shown in Figure.2 should be a multiple between $10^5 \sim 10^6$, much larger than what is reported in Figure.4 in the original paper. At the beginning of the experiments, we implement communication in an epoch manner, which lead to even worse test performance. Intuitively, one communication per epoch may provide outdated information for the following batches in this epoch, leading to accuracy degradation.

Moreover, we speculate the variable Z_n (so called network soft-decision in the original paper) in Algorithm 1, which represents the global consensus on soft prediction is not bounded (shown in the original paper Algorithm 1: equation 7). We think peers may reach a consensus where the prediction on a given image for a class may exceed 1, losing the meaning of probability and may lead the training process in the uncontrolled direction. In our implementation, we choose to clamp Z_n into $(0, 1)$, maintaining the original meaning of Z_n.

In summary, the training settings is clear enough in this paper. But it would be better if the communication Algorithm.1 is more detailed. The ambiguity in the algorithm degrades the reproducibility of the paper.

V. Conclusion

We reproduced the experiments in the paper [1] with some study of the baselines and other network compression methods. In general, the results are easy to reproduce given the hyper-parameters in the supplementary materials, even though there are some ambiguous descriptions about the algorithm itself. From the above experiments, we validate that 1) Although there is some problem in the MNIST baseline, Distributed Distillation can still improve the test performance compared to Silo-SGD. And all our experiments indicate that D-Dist cannot outperform D-SGD. 2) Distributed Distillation Algorithm does reduce the communication overhead significantly while achieving comparable accuracy and allows for devices with different classifier architectures to participate in the distributed training. 3) Distributed Distillation is compatible with different model compression algorithms, such as quantization and network pruning. In this project, we have also learned about the strict mathematical proofs and the implementation of a practical distributed machine learning algorithm.

There is also some weakness in our work. Given that D-SGD is highly time-consuming with limited resources, we cannot run the D-SGD baselines on ourselves. Thus it is hard to say that D-Dist is worse than D-SGD as shown in our experiment or that D-Dist is better than D-SGD. But we think it is enough for D-Dist to provide performance improvement compared to Silo-SGD, as the communication overhead compared to D-SGD is negligible.

Acknowledgements

The authors thank all professors who provide us with wonderful and detailed online resources and all TAs who are always willing to help us with patience and encouragement. The authors also thank Google Colab, which provides GPU resources for neural network training.

References

[1] I. Bistritz, A. Mann, and N. Bambos, "Distributed distillation for on-device learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[2] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[3] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[5] 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. 5330–5340.

[6] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[8] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," *arXiv preprint arXiv:1804.03235*, 2018.

[9] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.

[10] 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.

[11] 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.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[13] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[14] "Reduced mnist: how well can machines learn from small data?" 2017.

[15] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=rJqFGTslg