Exploring Decentralized Collaboration in Heterogeneous Edge Training

Xiang Chen, Zhuwei Qin
Department of Electrical and Computer Engineering, George Mason University
{xchen26, zqin}@gmu.edu

2020 IEEE/ACM Symposium on Edge Computing (SEC) | 978-1-7281-5943-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/SEC50012.2020.00069

***Abstract*—Recent progress in deep learning techniques enabled collaborative edge training, which usually deploys identical neural network models globally on multiple devices for aggregating parameter updates over distributed data collection. However, as more and more heterogeneous edge devices are involved in practical training, the identical model deployment over collaborative edge devices cannot be guaranteed: On one hand, the weak edge devices with less computation resources may not catch up stronger ones' training progress, and appropriate local model training customization is necessary to balance the collaboration. On the other hand, a particular local edge device may have specific learning task preference, while the global identical model would exceed the practical local demand and cause unnecessary computation cost. Therefore, we explored the collaborative learning with heterogeneous convolutional neural networks (CNNs) in this work, expecting to address aforementioned real problems. Specifically, we proposed a novel decentralized collaborative training method by decoupling a training target CNN model into independently trainable sub-models correspond to a sub-set of learning tasks for each edge device. After sub-models are well-trained on edge nodes, the model parameters for individual learning tasks can be harvested from local models on every edge device and ensemble the global training model back to a single piece. Experiments demonstrate that, for the AlexNet and VGG on the CIFAR10, CIFAR100 and KWS dataset, our decentralized training method can save up to 11.8× less computation load while achieve central sever test accuracy.**

I. Introduction

Promoted by the evolution of artificial intelligence and deep learning, more and more intelligent applications have emerged on edge devices. These applications keep collecting new and sensitive data from different users while being expected to have the ability to continually train the embedded neural network model on these newly collected data. Traditionally, neural network model are trained by powerful server machines, which requires sending collected data from the edge device to the server. As security threats to edge computing systems increase, server dependence needs to be reduced [1]. Thus, distributed training has been applied on edge devices to collaboratively update a global neural network model by averaging the parameter of local models. This collaboration strategy distributes the data processing workload over multiple edge devices, which will train identical neural network models with the same learning task and model structure. (*e.g.*, Federated Learning [2].)

However, this distributed training approach cannot well adapt to vast heterogeneous edge devices with limited computation resources as well as specific learning task preference: The computation capacity of edge devices still cannot satisfy with the heavy computation workload of neural network model training. The weak edge devices with less computation resources may not catch up stronger ones' training progress, and appropriate training model local customization is necessary to balance the collaboration. Also, a particular local edge device may have unique data domains or different cognitive tasks [3], [4]. The global identical model would exceed the practical local demand and cause unnecessary computation cost [5]–[7].

To effectively adapt distributed neural network models to the aforementioned heterogeneity in practical utilization, we explored decentralized neural network training by harvesting lightweight independently trainable sub-models' learning tasks/model structures across decentralized edge nodes. Our decentralized collaboration can divide a large target convolutional neural network (CNN) model into independent functionality structures corresponding to individual learning tasks (*e.g.*, classification targets). By selectively combining several functionality structures, a trainable CNN sub-model can be composed and trained to an edge node. Compared to the original model, these sub-models have significantly smaller sizes, and can be parallelly trained on edge nodes with minimum computation and communication requirement. To build decentralized training collaboration across edge nodes, partially overlapped training tasks are deployed

978-1-7281-5943-0/20/$31.00 ©2020 IEEE
DOI 10.1109/SEC50012.2020.00069

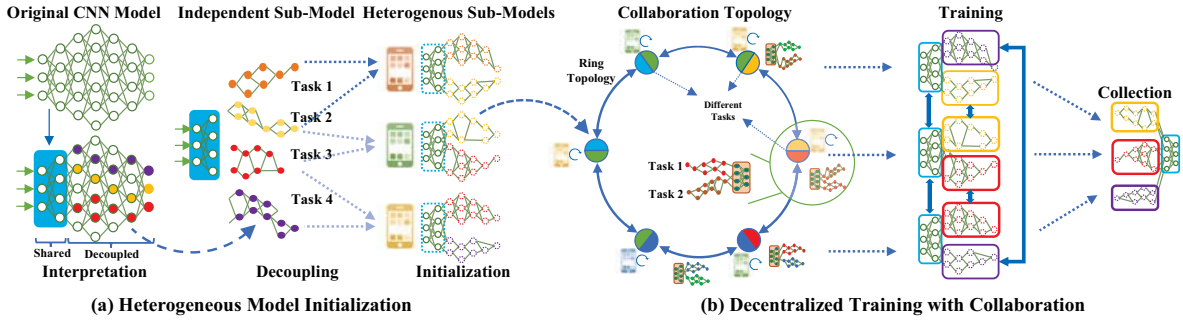

Fig. 1. Decentralized Heterogeneous Collaborative Edge Training Method. (a) Heterogeneous Model Initialization: Nodes of different colors are associated with different learning tasks. (b) Decentralized Training Collaboration: Connected neighbor edge nodes with partially overlapped training tasks will synchronize with each other to exchange the parameters of the task-identical model structures.

between neighbor edge nodes. By aggregating corresponding functionality structures, parameter consensus can be achieved in a chain manner across edge nodes. Such a collaboration consequently enhances the global convergence when training these sub-models in the decentralized setting. Eventually, with optimal computation and communication efficiency, the target CNN model parameter for all training tasks can be harvested from sub-models trained on edge nodes.

We implemented our decentralized collaboration method with AlexNet and VGG neural network on the CIFAR10, CIFAR100 and KWS speech dataset. Experiments shows that, we can save up to 11.8× less computation load while achieve central sever test accuracy.

II. Decentralized Collaborative Training

In previous distributed edge training schemes, different nodes are assumed to have integrated training datasets and learn identical CNN model structures. Collaboration between nodes is implemented by a straightforward parameter average of each identical local models. However, this identical model setting causes high burden for edge computing and vulnerability to local task and data heterogeneity. To address these problems, we propose a decentralized collaborative training method with heterogeneous sub-models. Figure. 1 illustrates the main components in our method, including **Heterogeneous Model Initialization**, and **Decentralized Training with collaboration**. An overview of our method is illustrated Fig. 1.

*a) **Heterogeneous Model Initialization**:* We first propose a task-oriented CNN model decoupling scheme to exploit the task learning exclusiveness of different neurons in a pre-trained model. As shown in the Figure. 1 (a), our method can decouple a pre-trained CNN model into two parts: the "shared layers" and the decoupled functionality structures. Recent works on CNN interpretation have shown that the CNN neurons in the shallow layers ("shared layers") act as basic feature extractor to extract universal features for all classification tasks. While the neurons in the deep layers can be activated by specific classification task [8], [9]. Therefore, by removing unnecessary connections between neurons with different task learning exclusiveness, we can decouple the CNN model into multiple independent functionality structures, each of which is associated with a learning task and works independently for it (*e.g.*, a particular image class in *ImageNet* dataset) (Figure. 1 (a) decoupling). Based on the investigation of the task-oriented model decoupling from pre-trained CNN models, we further propose a heterogeneous model initializing scheme to compose functionality structures into task-specific and trainable CNN sub-models: Each edge node can select individual or composed multiple sub-model structures based on its computation capacity and local heterogeneous tasks (Figure. 1 (a) initialization). By adding a shared layer structure to the selected functionality structures, each edge node can initialize a heterogeneous sub-model for local training. Such a model decomposition can significantly reduce the learning tasks for each node and therefore the computation and communication workload. By assigning all learning tasks with corresponding sub-models into nodes, a thorough model parallelism is achieved for collaborative learning.

*b) **Decentralized Training with Collaboration**:* Then, we construct a decentralized collaborative training method to minimize global training loss while preserving global consensus. To enable decentralized collaboration for parameter consensus, partially overlapped learning tasks are deployed between neighbor edge nodes. The

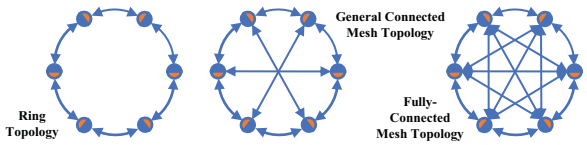

Fig. 2. Decentralized Communication Topology.

sub-model on each node is constructed by deploying decoupled functionality structures of the general CNN model in accordance with allocated learning tasks. Local nodes with overlapped tasks are connected to exchange parameters for task-identical structures. An underlying mechanism is specifically applied for global synchronization of "shared layers". As shown in in Figure. 1 (b), the adjacent 3 edge nodes collaboratively learns 3 different tasks, each trains for 2 local tasks.

When a specific amount of local training is finished, connected neighbor edge nodes with partially overlapped learning tasks will synchronize with each other to exchange the parameters of the task-identical model structures. After sending and receiving these parameters, individual edge node will aggregate the parameters with its local ones and resume training with local data. Thus, neighbor edge nodes' CNN models can achieve consensus on their parameters of the shared tasks. Considering the highly interconnected structures, all parameters for shared tasks will reach consensus after decentralized communication. After all the sub-models are well-trained, each individual sub-model can fulfill its specific inference tasks independently. Meanwhile, the original target CNN model can be also recomposed if necessary, by collecting parameters from all edge nodes. As shown in in Figure. 1 (b), the new shared layers will be generated through the average of all the shared layer parameters. The new functionality structure for one specific task will be generated from the average of all corresponding functionality structures.

III. Experiment

a) ***Experiment Setup****:* We adopt two CNNs for evaluation: AlexNet and VGG16. Two image classification datasets CIFAR10, CIFAR100 and one keyword spotting (KWS) speech dataset [10] are evaluated to demonstrate the generality of our proposed decentralized training method.

For the AlexNet model implementation, the last 3 convolutional layers are decoupled, and each independent functionality structure preserves 10% of total number of neurons (*e.g.* filters). For the VGG-16 model, the last 6 layers are decoupled, and the preserved neurons are optimized for different dataset, *i.e.*, we preserve 10% (52), 10% (52), 1% (5) of total number of neurons for the CIFAR10, KWS, and CIFAR100 datasets.

In our decentralized training, one requirement is the collaborative topology has to be one graph. As shown in Fig. 2, Ring topology requires the minimum connection, which should achieve our lower-bound convergence performance. By contrast, Mesh topology enables a fully-connected connection scheme which should achieve the upper-bound performance. In practice, the fully-connected topology can hardly be established with high scalability. In this preliminary work, we mainly implemented the Ring collaborative topology.

b) ***Convergence Speed Evaluation****:* In this part, we demonstrate the local and global convergence of our collaborative training method. Figure. 3 (a) and (b) show the training accuracy of local sub-models and global model versus communication rounds. The training configuration for the Ring topology on CIFAR10 is "4N_5Cls", which means that each local model learns 5 classification targets. For example, the classification target index of "9,0,1,2,3", "2,3,4,5,6", "5,6,7,8,9", and "8,9,0,1,2" in the CIFAR10, are deployed on 4 edge nodes, respectively. The parameters of task-identical model structures corresponding overlapped classification targets are averaged during the training.

As shown in the Figure. 3 (a) and (b), both local model and global model will converge as collaborative training proceeds. Each local model reaches very good accuracy alone, which means that our parameter sharing scheme is safe to local convergence. Also, we can observe that, the accuracy of our method reached that of centralized training methods, which proves the effectiveness of our method under this setting.

c) ***Collaborative Topology Evaluation****:* In this part, we analyze the performance of Ring topology under

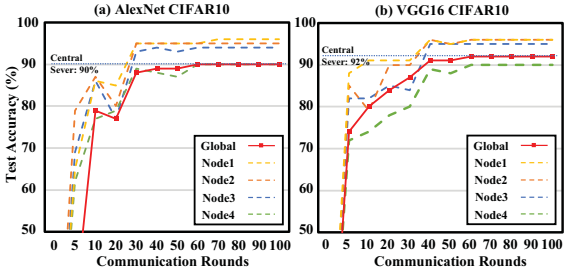

Fig. 3. Decentralized Convergence Evaluation on CIFAR10.

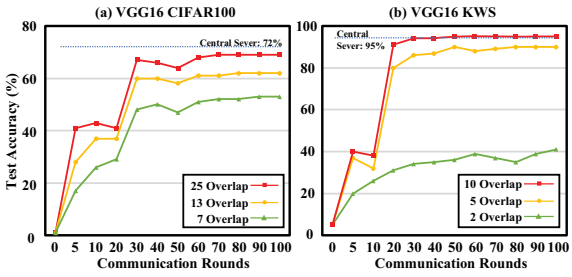

Fig. 4. Ring Collaborative Topology with Different Task Overlap.

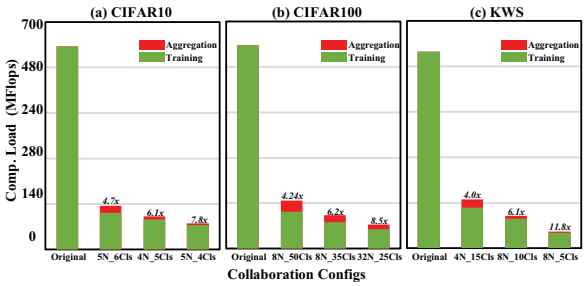

Fig. 5. Local Computation Cost Evaluation.

different task-overlap settings. Noted that, in our experiments with Ring topology, the global task set is repeated twice across the whole system. As shown in the Figure. 4 (a) and (b), the "n Overlap" means that each edge node shares n tasks with either neighbor. If residue exists in task allocation, it will be allocated to the last device as extra work load, which doesn't impact the performance of collaboration very much.

We can see that the accuracy of our global model drops when task-overlap decreases. For KWS dataset, the global model trained under the vanilla setting of "10 Overlap" reached the accuracy of centrally trained model, which proved the effectiveness of our method. The model trained with "2 Overlap" showed poor performance, since each sub-model can only reach consensus with very few sub-models on neighbor devices. On CIFAR100 dataset, the accuracy of model trained with setting "25 Overlap" approached that of centrally trained models. From these experiment with Ring topology, we can find that the number of tasks each device has a significant impact on the accuracy of global model. Allocating half of global tasks on each device and working with 4 device is suggested when we want to train a model with better accuracy.

d) ***Computation Cost Evaluation****:* In this part, we evaluate the computation cost reduction when the VGG model is decoupled into sub-models for local training. As shown in the Figure. 5, the red portion in each collaboration configuration indicates the aggregation overhead caused by aggregating the parameters of "shared layers" and the task-identical sub-models between local nodes. We can see that, with less learning tasks deployed, the computation load could be reduced by 7.8× in CIFAR10, 8.5× in CIFAR100 and 11.8× in KWS. Such results indicate that the proposed model decoupling can effectively resolve the edge training workload with optimal model training accuracy.

IV. Conclusion

In this paper, we explored the decentralized collaboration in heterogeneous edge training. Given a predefined CNN model structure, it can be configured into any specialized sub-models for dedicated edge tasks. The computation cost of the specialized sub-models can be significantly reduced. Based on this, we construct a collaborative learning method where global consensus is reached through exchanging the overlapped sub-models parameters for shared tasks between edge nodes. Experiments demonstrate that, our decentralized training method can save up to 11.8× less computation load while achieve central sever test accuracy.

References

[1] S. Park, J. Lee, and H. Kim, "Hardware resource analysis in distributed training with edge devices," *Electronics*, vol. 9, no. 1, p. 28, 2020.

[2] H. B. McMahan and *et al*, "Communication-efficient learning of deep networks from decentralized data," *arXiv:1602.05629*, 2016.

[3] Y. Zhao and *et al*, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[4] K. Chaudhuri and *et al*, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, 2011.

[5] H. Khelifi and *et al*, "Bringing deep learning at the edge of information-centric internet of things," *IEEE Communications Letters*, 2018.

[6] S. Chang and *et al*, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[7] A. Bellet and *et al*, "Personalized and private peer-to-peer machine learning," *arXiv:1705.08435*, 2017.

[8] Z. Qin, F. Yu, C. Liu, and X. Chen, "Functionality-oriented convolutional filter pruning," *arXiv preprint arXiv:1810.07322*, 2018.

[9] F. Yu and *et al*, "Dc-cnn: computational flow redefinition for efficient cnn through structural decoupling," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, 2020.

[10] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209*, 2018.