Distributed Mobile Computing for Deep Learning Applications

Seunghyun Lee, Haesung Jo, Jihyeon Yun and Changhee Joo*
Department of Computer Science and Engineering, Korea University
Seoul, Korea
{seunghyunlee, atregy, jihyeonyun, changhee}@korea.ac.kr

***Abstract*—Distributed computation is the widely used methodology to overcome challenges that the application covering multiple mobile devices mostly experiences, such as the high complexity of the computation and the resource limitation. By splitting the required computation and distribute the computation across the multiple devices, it can achieve lowered computation time and resource required per device and effective utilization in terms of the total resource management. This is risen as an appropriate approach to manage problems that recent applications with deep learning process have. Followed by the generalization of Internet of Things (IoT) and the development of data collecting technology, the deep learning process has to handle much larger dataset which makes it hard to be transferred through the network. This also leads to more complex computation that a single device may not be able to operate itself. In this paper, we consider the distributed computation applied in various fields, and how it is applied to distribute the deep learning process through observing researches studying about it.**

I. Introduction

The rapid growth of the artificial intelligence have brought dramatic changes in a diverse field, which includes autonomous driving, personalized recommendation, and image recognition/detection. Especially, recent years shows remarkable advances based on the progress of deep learning.

Deep learning is a subset of machine learning, which uses artificial neural networks. Through multiple layers of neural networks consisting of an input layer, an output layer, and hidden layers, deep learning can learn from enormous sets of data and use them to model and solve complex problems at high speed that the human cannot follows. The subject that how the deep learning can be applied to improve our real life has been continuously studied [1]. Microsoft applied deep learning technique to audio recognition, enabling Microsoft Audio Video Indexing Service (MAVIS) to search audios and video files through human voices and speeches. Similarly, Google uses deep learning on big data environment for providing image search service [2]. Recently chatbot services like ChatGPT and Bing Chat which are trained through deep learning based on tremendous data appears. In medical area, deep learning technique is used for analyzing the microscopy image [3], reducing human intervention in generating features in health informatics [4], and in other various ways.

* C. Joo is the corresponding author.

This work was supported by the NRF grant funded by the Korea government (MSIT) (No. 2022R1A5A1027646).

The development of deep learning technique enables various applications for mobile devices. For example, for recommending applications, for optimizing web browsing, monitoring activity or health information, etc [5]. Such applications in mobile environment face more problems due to the restriction that mobile devices commonly have. Compared to the edge server that usually has performed the overall computation, it is common that the mobile device suffers from the limited hardware resources like the battery or CPU. As the recent applications require the ability to handle more complex tasks, it becomes the main challenge that the deep learning on mobile device should overcome.

In this survey, we study how the distributed computation can be a solution for overcoming such limitations and the literature studying about distributing learning computation over multiple devices. In Section II, we introduce the distributed computation in general use. In Section III, we study how the distributed computation can be adapted in deep learning process, then introduce researches that has been studied in two categories by providing the literature in Section IV.

II. Distributed Computation

As Internet of Things (IoT) be generalized, tremendous number of devices start to operate as nodes consisting the network environment, communicating each other by transferring or receiving data. Each node still has the limitation of computation and energy resource, thus a single node is usually not capable of executing tasks required for centralized process. It is hard to increase the computational resource or energy capacity for the mobile device due to the realistic limitation, so the methodology of distributing the required computation over multiple devices to utilize more resources is considered as a reasonable approach.

In case of vehicular network, each automobiles becomes a communicating node consisting the network and researcher needs to consider their mobility and the connectivity among them. In [6], authors model a two-dimensional network topology regarding each vehicle as a single node, and offload the task depending on each node's status including capability, connectivity, and etc.

Distributed computation is often handled when transmitting traffic end-to-end. To reduce the transmission delay, packets might be compressed in the middle of the process or optimal

route should be found considering each node's status like energy cost and channel state. Research [7] proposes the methodology that can be applied in multi-hop transmission in IoT networks. In this study, authors consider the situation that IoT devices transmit short packets to multiple destinations, while harvesting energy in the middle of the process from multiple power beacons. According to the study, proposed scheme shows better performance in terms of throughput, execution time, reliability, and so on.

Research [8] analyzes the mobile edge computing under the assumption that the devices are powered wireless. In this study, authors achieve decentralization of the algorithm through online learning, and consider both the distributed computation and energy harvesting through offloading the computation.

Literature [9] suggests a neural-network-based framework considering two-dimensional IoT mesh network. The framework considers the energy that each node consumes and the routing problem, then distribute the data processing over multiple nodes or operate the data aggregation. From the evaluation through the numerical simulations, the authors succeed to show the meaningful latency reduction is achieved even in the realistic environments.

III. Distributed Computing in Deep Learning

As the digital technologies are improved, the larger size of data collection and storing it are possible, which the modern application actively use to provide better quality of service. Therefore, the deep learning model which will be applied to such applications needs to consider of handling complex data sets unlike the conventional deep learning models [10]. Moreover, for some kinds of services have to take mobile environment into consideration, i.e., applications applied to IoT devices, automobiles, or mobile phones. In these cases, solving a few restrictions followed by using mobile devices becomes one of major challenge.

When the deep-learning-based service is provided through a mobile device, the model takes the dataset collected from the device as the input. To start the process with the dataset, there exist two options: proceeding the learning on the mobile deivce, or transmitting the dataset to the server to let it do computation. The former faces the challenge of the restricted computation resource that mobile devices usually have. The latter is free from such problem, but another challenge occurred by the size of dataset stands. Handing over huge dataset to the server through unstable wireless channel may result in severe degradation in the channel condition [11].

Distributed computing can be one solution for this problem. By offloading some parts of computation process from mobile device to the server, the mobile device is able to compute the partial operation which does not exceed its capacity. Also, because not the entire dataset but the partially computed load is transmitted, the load that the wireless channel should be capable of is much smaller.

Except the local computing that all the computation is processed on the mobile device and the edge computing that all the computation is processed on the edge server, the distributed computing methodology used in deep learning can be classified into two categories: split computing and early exiting [12]. Split computing indicates that the mobile device does the partial computation and offload the rest to the edge server [13]. Early exiting is similar to the split computing, but there exist several exits in the layers that makes the computation be terminated when the target confidence is achieved [14].

IV. Application of distributed computing

To design the deep learning model which takes distributed computing into consideration, there exist several points that need to be considered. Depending on the targeted work and the environment that the model will be applied, the model may experience different computation complexity, the network channel condition, available computation resource, and other factors. Targeted values for various objectives that the model should achieve are also different.

In this section, we introduce various researches proposing distributed-computing-adopted deep learning models, each considers different target achievement, network topology, objective of the model, and etc. According to the methodology achieving the distribution, we categorized the literature into two groups: Task distribution and data/model parallelism.

A. Task Distribution

Distributing the learning process is the basic method for accomplishing the distributed computing. By handing over the partial task to other devices and utilizing resources from multiple devices, the challenge coming from the practical limitation to add the hardware for increasing the computational resource can be solved. There have been many studies taking this methodology, analyzing the possible challenges and designing the model.

Reference [15] studies about deep neural networks (DNNs) for large models, targeting for solving the throughput bottleneck occurred by the limited computational resource of mobile devices. Authors propose a framework named DEFER (Distributed Edge inFERence) which divides DNNs into several layers so each can be deployed in each node consisting the network.

The authors in [16] operates the distributed computing in order to run a large CNN on a collection of concurrent IoT sensors. In this research, the proposed model split the model into multiple small models, and each model is targeted to specific kind of tasks like voice or speech. Using the model proposed in the research, authors succeed to reduce the model size and the inference time without the accuracy loss.

Graph neural networks (GNNs) treat very large sized dataset, and known as an effective model for graph structured dataset. Research [17] studies the framework which partition the large graph into multiple small sub graphs, and computes them in a distributed manner.

Reference [18] analyzes the methodology that partitioning DNN into different edge devices. Authors adopt genetic algo-

rithm in the partitioning process, achieving the lowered energy cost and shorter inferencing time.

The authors in [19] propose a DISSEC, a distributed scheduling strategy for DNN inference on IoT edge clusters. They describe the approaches that deploy complete DNNs on resource-constrained edge devices and the search algorithms used to generate the optimal distributed scheduling strategy. DISSEC reduces communication overhead by 20% and execution latency by 9%.

The authors in [20] introduce Distributed Inference with Sparse Communications which is called DISCO. They train DNN model using within-layer model parallelism that distributes the inference of each layer into multiple nodes. However, since the dependency of each layer can be a bottleneck to parallel inference, they employ sparse communication to reduce the data transfer between nodes while parallel inference. They achieved a 3.5x latency reduction on 20 models.

B. Data/Model Parallelism

There are several approaches to parallelizing the training of deep learning model. First, data parallelism divides the training data into multiple small batches. In addition, it maintains copies of the DNN model to process each batch at the same time. Each copy computes the gradient of its batch and periodically updates the parameters of the model using gradients. Next, model parallelism split the DNN model and places it on multiple different devices. A mini-batch of training data. In this case, forward and backward passes are performed in a coordinated manner across devices. It induces low device utilization. Pipeline parallelism has been proposed to utilize mini-batch more efficiently by combining data and model parallelism. Similar to model parallelism, it split the model into stages and assigned it to different devices. Also training data divide into multiple mini-batches. Different from them, however, different devices can process different mini-batches at the same time. Hence it can be used for large models and large datasets.

The authors in [21] propose effective multi-level model parallelism optimization for distributed inference of CNN, which is denoted as DeCNN. They address the inherent problem of CNN that tightly-coupled structures. The proposed DeCNN split the original CNN model into several sub-models. They achieved higher performance improvement and lower memory footprint than single-device experiments by using three-level optimization for model parallelism. Their approaches outperformed existing the other methods and can apply to existing CNN models such as ResNet-50.

The authors in [22] develop an efficient pipeline parallelism framework that comprises two components. First, is a pipeline partition and device mapping algorithm that splits a DNN as partitions over available GPUs. Another one is a pipeline scheduler that processing over of microbatches over the partitions. They demonstrate the benefits of synchronous pipelining over earlier pipeline designs and show their approach accelerates the training of DNN model up to 157%.

The authors in [23] introduce a direct-connect DNN training system that co-optimizes the computation, communication, and network topology which is called TopoOpt. They analyze DNN training jobs from production clusters of Meta and address the communication overhead and traffic pattern of datacenter. They also discuss the challenges of finding the best topology and switching techniques for DNN models.

The authors in [24] introduced an open-source library, called Horovod, which provides a simple, efficient, and flexible approach to distributed training in TensorFlow. They proposed ring-allreduce algorithm for efficient gradient aggregation and optimizing communication overhead. They achieved high-performance multiple GPUs learning with minimal code changes.

V. Conclusion

The development of the deep learning attracted a lot of interests from various fields due to its possible application, and it has been accelerated steeply in recent years. With the improvement in data collection technology, this trend require larger-sized dataset in the learning process, which in turn requires more complex computation to be operated. As an approach to solve this challenge, it has been researched that distributing the computation needed across the edge devices.

In this work, we overview the studies about adopting the distributed computing methodology in various fields of research. We discuss about why how this methodology operates and why this is important to solve specific challenges. We observe the studies to consult the examples of the construction of the distributed computing, then introduce how researchers adopt the distributed computing to the deep learning which should be different in implementation because of the diverse in environment of the application.

References

[1] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2018, pp. 1–6.

[2] M. Gheisari, G. Wang, and M. Z. A. Bhuiyan, "A survey on deep learning in big data," in *2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC)*, vol. 2. IEEE, 2017, pp. 173–180.

[3] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, "Deep learning in microscopy image analysis: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4550–4568, 2017.

[4] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[5] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digital Communications and Networks*, vol. 8, no. 1, pp. 1–17, 2022.

[6] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[7] T.-V. Nguyen, V.-D. Nguyen, D. B. da Costa, and B. An, "Short-packet communications in multi-hop wpins: Performance analysis and deep learning design," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[8] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao, and G. Wang, "Online learning for distributed computation offloading in wireless powered mobile edge computing networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1841–1855, 2021.

[9] E. Di Pascale, I. Macaluso, A. Nag, M. Kelly, and L. Doyle, "The network as a computer: A framework for distributed computing over iot mesh networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2107–2119, 2018.

[10] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 747–750, 2020.

[11] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li, "Communication-efficient distributed deep learning: A comprehensive survey," *arXiv preprint arXiv:2003.06307*, 2020.

[12] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–30, 2022.

[13] A. Bakhtiarnia, N. Milošević, Q. Zhang, D. Bajović, and A. Iosifidis, "Dynamic split computing for efficient deep edge intelligence," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[14] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[15] A. Parthasarathy and B. Krishnamachari, "Defer: Distributed edge inference for deep neural networks," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022, pp. 749–753.

[16] J. Chen, D. Van Le, R. Tan, and D. Ho, "Split convolutional neural networks for distributed inference on concurrent iot sensors," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 66–73.

[17] Z. Jia, S. Lin, M. Gao, M. Zaharia, and A. Aiken, "Improving the accuracy, scalability, and performance of graph neural networks with roc," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 187–198, 2020.

[18] J. Na, H. Zhang, J. Lian, and B. Zhang, "Partitioning dnns for optimizing distributed inference performance on cooperative edge devices: A genetic algorithm approach," *Applied Sciences*, vol. 12, no. 20, p. 10619, 2022.

[19] Q. Li, L. Huang, Z. Tong, T.-T. Du, J. Zhang, and S.-C. Wang, "Dissec: A distributed deep neural network inference scheduling strategy for edge clusters," *Neurocomputing*, vol. 500, pp. 449–460, 2022.

[20] M. Qin, C. Sun, J. Hofmann, and D. Vucinic, "Disco: Distributed inference with sparse communications," *arXiv preprint arXiv:2302.11180*, 2023.

[21] J. Du, X. Zhu, M. Shen, Y. Du, Y. Lu, N. Xiao, and X. Liao, "Model parallelism optimization for distributed inference via decoupled cnn structure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1665–1676, 2020.

[22] Z. Luo, X. Yi, G. Long, S. Fan, C. Wu, J. Yang, and W. Lin, "Efficient pipeline planning for expedited distributed dnn training," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 340–349.

[23] W. Wang, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, and A. Kewitsch, "TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 739–767. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/wang-weiyang

[24] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.