Adaptive DNN Model Partition and Deployment in Edge Computing-enabled Metro Optical Interconnection Network

Mingzhe Liu, Yajie Li, Yongli Zhao, Hui Yang, Jie Zhang*
Beijing University of Posts and Telecommunications, Beijing 100876, China
**lgr24@bupt.edu.cn*

Abstract: A DNN model partition and deployment algorithm is proposed between edge nodes and cloud in metro optical network. Simulation results show that the algorithm can deploy more DNN tasks with the same network resource.

1. Introduction

Artificial Intelligence (AI) serves as a promising technology to address complex issues in optical transmission and networks [1], such as OSNR monitoring, traffic prediction, and failure location. In the existing schemes, AI algorithms are deployed in the logically centralized control layer, which is usually located in cloud nodes. However, massive training data sets need to be transmitted from/to the remote cloud node, which incurs high traffic and computational load, and high latency. The emergence of edge computing enables AI to be deployed closer to the nodes generating the data (edge nodes) as well as in the centralized controller (cloud). Therefore, some AI tasks (e.g., data pretreatment and model pre-training) can be realized locally to prevent overhead and guarantee the Quality of Service (QoS). With edge computing, a Deep Neural Network (DNN) model with multiple layers can have parts of the layers offloaded to edge nodes and then transfer intermediate data to the centralized cloud node. This model partition method can not only reduce the data size and transmission delays but also ease the computational load of cloud [2].

A typical deep learning model usually has many layers in the learning network, and the intermediate data size can be quickly scaled down by each network layer. Different partition schemes of DNN model between edge nodes and cloud correspond to different usage of network resources. Usually, the more layers deployed in edge nodes, the lower the bandwidth required for transmission due to data size reduction. However, more Computing Units (CUs) are required to enable data processing in edge nodes. In contrast, the fewer layers running in edge nodes need the fewer CUs, with the cost of higher transport bandwidth. In dynamic network scenario, the arrival and depart of services will change the usage of network resources. A fixed DNN model partition scheme cannot guarantee the deployment of DNN tasks due to limited network resources. Therefore, it is necessary to investigate the flexible model partition based on network resources in dynamic network scenarios.

Some related research has been conducted on flexible model partition between edge nodes and cloud. The authors in [2] introduced deep learning for IoT into the edge computing environment and proposed model partition algorithms to maximize the number of tasks, which optimized network performance and protected user privacy in uploading data. In the study of [3], the DNN model was partitioned between the mobile device and the edge node. The authors proposed a regression-based method to return an optimal partition point which made the model inference meet latency and energy requirements. Nevertheless, these studies are based on fixed bandwidth and quasi-static scenario, while flexible model partition based on network resource is still unsolved in dynamic network scenario.

In this paper, we propose an adaptive model partition scheme to meet the requirements of latency and network resources of DNN tasks in dynamic network scenario. The algorithm can adaptively select model partition points according to network status by jointly considering CUs in edge nodes and bandwidth resources in optical transport networks. Simulation results show that the strategy can effectively deploy 6.4% more DNN tasks with the same network resource in different traffic loads.

2. Network scenario and problem description

A DNN task has specific bandwidth, CUs, and latency requirements, and the edge computing-enable Elastic Optical Network (EON) is the promising candidate to meet these demands in metro optical interconnection networks. Fig. 1 shows the architecture of DNN model partition between edge nodes and cloud in EON. DNN model is divided into two parts. Lower layers which cost less calculation is running in edge nodes. The intermediate data size can be quickly scaled down by each DNN layer [2]. Then the intermediate data is transferred by optical fiber and processed by remaining layers in cloud. As is shown in Fig. 1(a), there is a four-layer DNN and three model partition schemes. Partition scheme 1 determines to run only one layer in the edge node, and it costs a few CUs and four Frequency Slots (FSs). In partition 3, more CUs are required for operations, and more FSs are saved. In comparison, partition 2 costs suitable CUs for calculation and three FSs for transferring intermediate data. In the real-time network scene, different model partition schemes will cause a different impact on network and QoS.

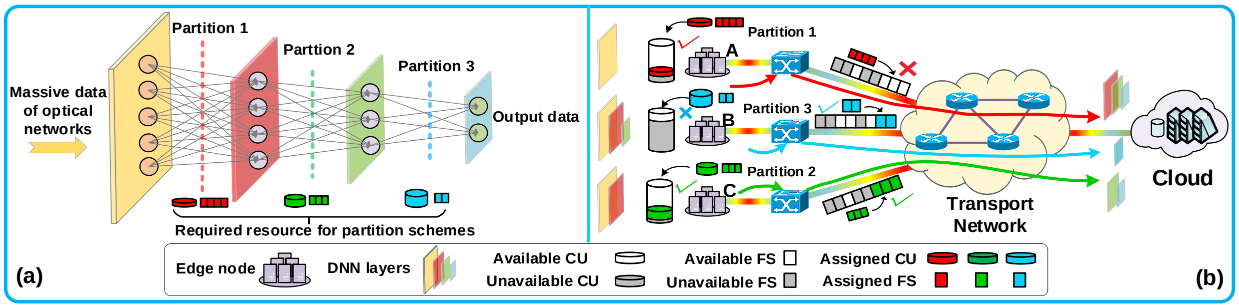

Fig. 1 (a) Different model partition schemes (b) DNN tasks implemented by different model partition schemes in EON

Fig.1(b) shows three DNN tasks with model partitions in the edge computing-enabled EON. For edge node A, the DNN task is blocked if using partition 1, due to the lack of FS resources. Similarly, for edge node B, the DNN task cannot be deployed with partition 3 for lack of CU resources. By contrast, the DNN task is successfully implemented in edge node C with partition 2. Therefore, in dynamic network scenario, bandwidth resources in optical links and CUs in edge nodes should be considered jointly. In order to guarantee the FSs, CUs, and latency requirements of DNN tasks, DNN models need to be divided flexibly.

3. Adaptive Model Partition and Deployment Algorithm

In this section, we employ an adaptive model partition and deployment (AMPD) algorithm to solve the DNN model flexible partition problem in dynamic network scenarios. In algorithm 1, we traverse all layers of the DNN and perform resource pre-allocation, then jointly consider CU and FS resources to select the optimal partition point.

Algorithm 1: AMPD algorithm

```
Input: topology $G$ and DNN task $t_i\{m, e_i, d_i, L_{\max,i}\}$
Output: model partition point $k$, CU and FS assignment
1.  for each layer $k$ in DNN $m$ do
2.      if $c_{e_i} > c_k$ then
3.          pre-calculate $LC_i$ and $K$ shortest paths Set $P_{KSP}$ from edge
        node to cloud;
4.          for each path $p$ in $P_{KSP}$ do
5.              calculate $LT_i$ and $L_i$ ;
6.              if $L_i < L_{\max,i}$ then
7.                  add $p$ to Set $P_{candidate}$ ;
8.              end if
9.          end for
10.         select the path $p_i^k$ according to FS maximum;
11.         add $k$ , $c_k$ , $p_i^k$ to Set $S_i$ ;
12.     else break;
13. end for
14. if $S_i$ is empty then
15.     $t_i\{m, e_i, d_i, L_{\max,i}\}$ is blocked
16. else select the best scheduling scheme $s$ in $S_i$ according to the
        principle of $\theta_i$ minimum
17. end
```

Notations

m :	DNN model for a DNN task
e :	Edge computing node
d :	Original data size of DNN task (GB)
$L_{\max}$:	Latency tolerance (in ms)
LC :	Latency of calculating
LT :	Latency of optical transmission
L :	Total latency of a DNN task
c_k :	Required edge CU by the k th layer (units/s)
c_e :	Available CU in edge node (units/s)
r_k :	Ratio of the intermediate data to the input data
b_k :	FS cost by the k th layer (slots)
o_k :	CU overhead of a unit of input data after k layers (units/GB)
S_i :	Candidate scheduling scheme set
$b_{available}$:	Average free FSs of a path

Formulas

$$LC_i = \frac{o_k^i \times d_i}{c_k^i} \qquad (1) \qquad\qquad LT_i = \frac{r_k^i \times d_i}{b_k^i \times B \times l_i^p} \qquad (2)$$

$$L_i = LT_i + LC_i \qquad (3) \qquad\qquad \theta_{CU} = \frac{c_k}{c_e} \qquad (4)$$

$$\theta_{FS} = \frac{b_k}{b_{available}} \qquad (5) \qquad\qquad \theta_i = \theta_{CU} + \theta_{FS} \qquad (6)$$

The AMPD algorithm consists of two main parts, i.e., pre-scheduling for each layer and best partition selection. **Step1:** a DNN task t_i occurs at the edge node e_i . For each DNN layer, we first calculate LC_i according to Eq. (1). For simplicity, we assume that occupied CUs are proportional to the number of layers as $c_k = k \times \alpha$, where α is a constant. Total latency is composed of two parts—computing latency and transmission latency. Then we traverse K shortest paths from edge to cloud and obtain LT_i by Eq. (2), where l_i^p is the modulation level adaptively according to the distance of p , B is the capacity of a FS. Hereafter, we get the pre-scheduling scheme of the kth layer according to FS maximum from all the paths which L_i can satisfy the latency $L_{\max,i}$. **Step2:** In order to accommodate as many tasks as possible, CU metrics θ_{CU} and FS metrics θ_{FS} are jointly considered by Eqs. (4) and (5). Finally, we balance θ_{FS} and θ_{CU} as balanced metrics θ_i by Eq. (6), then select the best scheme according to the minimum θ_i .

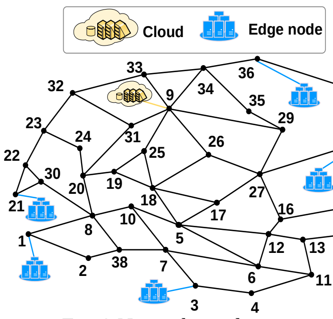

Fig. 2 Network topology

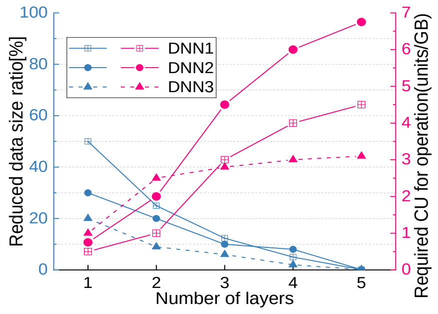

Fig. 3 Reduced data and complexity of DNN

Table1 Simulation parameters

FSs per fiber link	80 slots
CUs in edge nodes	1000 units
Latency tolerance	[50,200] ms
Data size per task	[1,5] GB
Distance per link	20km
K, paths of KSP	8
Task number	50000
Simulation times	20
Cloud/Edge nodes	1/6

4. Simulation and Results Analysis

Fig. 2 shows a metro network topology used as the simulation topology [5]. We assume that the EON can support distance adaptive modulation levels. The modulation level of BPSK, QPSK and 8QAM, with the longest transmission distance 240km, 120km and 60km is, respectively, 1, 2 and 3. Parameter α and B are set as 20 (units/s) and 6.25(GB/slot). For simplicity, three five-layer DNN are considered and the occupied FSs for transferring intermediate data generated by each layer are 5, 4, 3, 2, and 1, respectively. Referring to [2], the DNN models have different data ratio reduction and complexity for each layer, as shown in Fig. 3. Table 1 summarizes other simulation parameters. Note that the latency tolerance of a task is related to the data size. To verify the performance of the proposed AMPD, we also implement three DNN model partition strategies as reference algorithms, which are CU-based, FS-based, and fixed partition (baseline, DNN1, DNN2, and DNN3 are divided at 3th, 2rd and 1st layer, respectively).

Fig. 4(a) shows the DNN task blocking probability of four algorithms in different traffic loads. Note that the traffic load is characterized by the task arrivals per unit of time in the Poisson distribution traffic model. We can see that with traffic load from 400 to 650 Erlangs, blocking probability of all algorithms increases and, AMPD achieves the lowest blocking probability among the four algorithms. As the traffic load increases, the difference between AMPD and baseline blocking probability increases gradually. When the traffic load is up to 650 Erlangs, AMPD can deploy 6.4% more DNN tasks than the baseline. This is because AMPD can divide DNN models adaptively and better cope with changes in traffic intensity. In Fig. 4(b), we fix the traffic load at 500 Erlangs and set the CUs from 800 to 1200 units. Except by the CU-based algorithm, blocking probability of the other three algorithms decrease in degrees, especially FS-based. This is because the CU-based algorithm prefers to choose the scheme that occupies less CUs and more FSs, resulting in blocking due to lack of FS resources instead of CUs. In Fig. 4(c) and (d), we contrast the CU and bandwidth utilization of AMPD against the other three algorithms in different traffic loads. We observe that with the increase of traffic load, more CUs and more FSs are needed to meet the requirements of DNN tasks. Because of balanced metrics, AMPD can balance CU and FS resources compared to CU-based and FS-based algorithms.

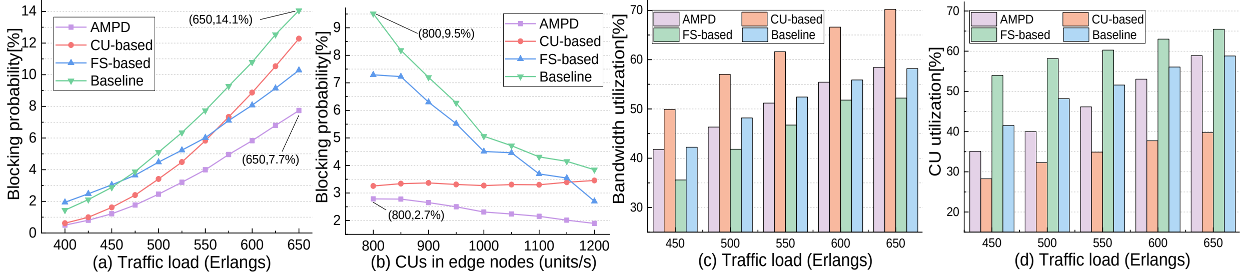

Fig.4 Blocking probability and resource utilization for various traffic load and CUs

5. Conclusion

This paper proposes a heuristic algorithm that adaptively selects the appropriate DNN model partition between edge nodes and cloud nodes based on network resources in dynamic optical network scenarios. Simulation results show the proposed algorithm could reduce the DNN task blocking probability and deploy 6.4% more DNN tasks than baseline.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (NSFC) Project (Grant No. 61901053, 61822105) and China Postdoctoral Science Foundation (2019M650588).

References

[1] Y Zhao, et al., “Coordination between control layer AI and on-board AI in optical transport networks”, IEEE/OSA Journal of Optical Communications and Networking, 12 (1), pp. A49-A57 (2020)
[2] H Li, et al., “Learning IoT in edge: Deep learning for the Internet of Things with edge computing.” IEEE Network 32 (1), pp. 96-101 (2018)
[3] Y Kang, et al., “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ”ACM SIGARCH Computer Architecture News. 45 (1), pp.615-629 (2017)
[4] Z Liu, et al., “Joint Jobs Scheduling and Routing for Metro-Scaled Micro Datacenters over Elastic Optical Networks”, Proc. ECOC 2019
[5] M Fiorani, et al., “Flexible architecture & control strategy for metro-scale networking of geographically distributed DCs”, Proc. ECOC 2016