Online Task Scheduling for DNN-Based Applications over Cloud, Edge and End Devices

Lixiang Zhong, Jiugen Shi, Lei Shi(✉), Juan Xu, Yuqi Fan, and Zhigang Xu

School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei 230009, China
shilei@hfut.edu.cn

Abstract. As a combination of artificial intelligence (AI) and edge computing, edge intelligence has made great contributions in pushing AI applications to the edge of the network, especially in reducing delay, saving energy and improving privacy. However, most of researchers only considered the computation approach of end device to edge server and ignored the scheduling of multi-task. In this paper, we study DNN model partition and online task scheduling over cloud, edge and devices for deadline-aware DNN inference tasks. We first establish our mathematical model and find the model can not be solved directly because the solution space is too large. Therefore, we propose the partition point filtering algorithm to reduce the solution space. Then by jointly considering management of the networking bandwidth and computing resources, we propose our online scheduling algorithm to meet the maximum number of deadlines. Experiments and simulations show that our online algorithm reduces deadline miss ratio by up to 51% compared with other four typical computation approaches.

Keywords: Edge computing · Edge intelligence · Model partition · Task scheduling

1 Introduction

In recent years, machine learning, especially deep learning, has attracted significant attention from both industry and academia. As the key component of deep learning, Deep Neural Networks (DNNs) is widely used in various fields, such as natural language [1], computer vision [2], speech recognition [3] and so on. With the growth of the number of Internet of Things (IoT) devices, the traditional cloud-centric computating approach will inevitably lead to network congestion, high transmission delay and privacy disclosure [4–6]. Therefore, it is a trend to offload the computation to the end device. However, Internet of Things devices with limited energy and computing resources cannot afford computing-intensive tasks.

Z. Xu—Supported by the National Natural Science Foundation of China (Grant No. 61806067), the Anhui Provincial Key R&D Program of China (202004a05020040).

Z. Liu et al. (Eds.): WASA 2021, LNCS 12939, pp. 183–191, 2021.
https://doi.org/10.1007/978-3-030-86137-7_20

Edge computing (EC) is proposed as a promising computing model for solving these problems by deploying servers at the network edge close to the end devices [7]. In [8], authors define a query processing problem in an Edge Assisted IoT Data Monitoring System which aims to deriving a distributed query plan with the minimum query response latency. In [9,10], the authors studied task scheduling and resource allocation in edge computing environment for different scenarios, and the experimental results have been significantly improved. In [11], authors set the weighted value of the delay sensitivity of the task, and optimized the offloading strategy of the task with the goal of minimizing the total weighted corresponding time of all tasks. In addition, edge computing has great potential in smart cities [12] and smart homes [13].

Indeed, the combination of artificial intelligence and edge computing has given rise to a new research area, namely edge intelligence, which was proposed in [14]. Instead of entirely relying on the cloud, edge intelligence makes the most of the widespread edge resources to perform tasks such as DNN inference. On this basis, offloading for DNN-based applications in EC has been broadly studied. In [15], authors proposed the DeepWear model, which uses DNN model partition to offload tasks to wearable applications, and achieves a good acceleration effect. In [16], authors realized the task offloading through model partition, and combined with the model early exit technology to further reduce the delay of DNN-based applications. In [17], authors proposed the partition and offloading strategy which can make the optimal tradeoff between performance and privacy for mobile devices. In [18], authors proposed a technique to divide a DNN in multiple partitions that can be processed locally by end devices or offloaded to one or multiple powerful nodes.

Previous work has made some contributions in the field of edge intelligence. However, most of them only considered the computation approach of end device to edge server and ignored the scheduling of multi-task. In this paper, for deadline-aware DNN inference tasks, we study DNN model partition and online task scheduling over cloud, edge and devices. We first establish our mathematical model, which can hybridly exploit all the available resources from the cloud, the edge and the devices to unleash all the full power of the deep learning networks therein. Since the model contains a large number of variables, which make the model can not be solved directly because the solution space is too large, we propose the partition point filtering algorithm to reduce the solution space. Then by jointly considering management of the networking bandwidth and computing resources, we propose our online scheduling algorithm to meet the maximum number of deadlines. Experiments and simulations show that our online algorithm achieves better performance than other typical computing approaches.

The rest of this paper is organized as follows: In Sect. 2, we introduce our system model and define our problem. In Sect. 3, we give the model partition filtering algorithm and online scheduling algorithm respectively. In Sect. 4, we give the simulation results and analyze them. In Sect. 5, we summarize this paper.

2 System Model and Problem Definition

2.1 DNN Model Partition and Network

Consider a two-dimensional network consists of n servers, m end devices and one cloud center. Denote C as the could center, s_j $(s_j \in S, j = 0, \ldots n)$ as one of the edge servers and d_i $(d_i \in D, i = 0 \ldots m)$ as one of the end devices. As shown in Fig. 1, we consider that edge servers are heterogeneous and have different computing power, end devices are all of the same specifications and have the same computing power, and cloud center has the strongest computing power.

For the whole task scheduling time T, it is divided into h time slots and each time slot is expressed as t_τ $(\tau = 0 \ldots h)$. The length of t_τ is 1 ms. The task may be generated at any time slot, and the transmission time and calculation time of the task are counted in time slot. For one end device, the next task can only be generated after the current task is completed. Each end device or edge server can only compute one task at a time and there is no waiting queue, while the cloud center can compute multiple tasks at the same time. We denote m_i^τ as one of the task, where i means the task is generated by device d_i, and τ means the task is generated in the τ-th time slot.

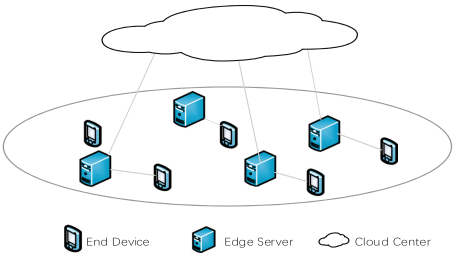

Fig. 1. System model.

We suppose that all tasks are inference tasks of the same DNN model. The model has r layers, and we denote $l_k (k = 0, 1 \ldots r)$ as one of the layer. We denote $G_k (k = 0, 1 \ldots r)$ as output data size of the lay l_k. It is obvious that G_0 means the size of the input data and G_r means the size of the result of the model.

As shown in Fig. 2, each task can be divided into up to three parts, which can be computed in turn on the end device, the edge server, and the cloud center. Therefore, by comprehensively considering all the current tasks, our formulated problem should not only determine the partition point, but also consider the appropriate scheduling of the partitioned DNN inference tasks, so that as many tasks as possible can be completed before the deadline in the scheduling time.

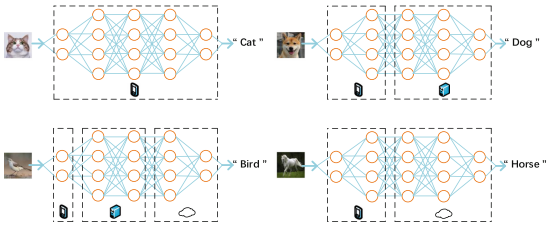

Fig. 2. Four typical DNN partition strategies.

2.2 Problem Formulation

For each task m_i^τ, its total completion time $t\left(m_i^\tau\right)$ can be expressed as

$$t\left(m_i^\tau\right) = t_d\left(m_i^\tau\right) + t_e^{\uparrow}\left(m_i^\tau\right) + t_e\left(m_i^\tau\right) + t_c^{\uparrow}\left(m_i^\tau\right) + t_c\left(m_i^\tau\right). \tag{1}$$

The total completion time is made up of five parts. The first part $t_d\left(m_i^\tau\right)$ is the computing time of the task in the end device. The second part $t_e^{\uparrow}\left(m_i^\tau\right)$ is the transmission time of the intermediate data from the device to the edge server. The third part $t_e\left(m_i^\tau\right)$ is the computing time of the task in the edge server. The fourth part $t_c^{\uparrow}\left(m_i^\tau\right)$ is the transmission time of the intermediate data from the edge server to the cloud center, and the fifth part $t_c\left(m_i^\tau\right)$ is the computing time of the task in the cloud center.

For the $t_d\left(m_i^\tau\right)$, we use binary variable $x_u\left(m_i^\tau\right)$ to indicate whether the first partition point is the layer l_u. If the first partition point is the layer l_u, $x_u\left(m_i^\tau\right) = 1$, which means after computing layer l_u, the device will upload the intermediate data to the edge server and no longer participates in the computation of the task m_i^τ. Otherwise, $x_u\left(m_i^\tau\right) = 0$.

$$x_u\left(m_i^\tau\right) = \begin{cases} 1: & \text{the first partition point is the layer } l_u; \\ 0: & \text{otherwise.} \end{cases} \tag{2}$$

It satisfies

$$\sum_{u=0}^{r} x_u\left(m_i^\tau\right) = 1. \tag{3}$$

We express $t_d\left(m_i^\tau\right)$ as the time cost for computation from l_0 to l_u for d_i. It can be calculated by

$$t_d\left(m_i^\tau\right) = \sum_{u=0}^{r}\left(x_u\left(m_i^\tau\right) \cdot \sum_{k=0}^{u} \alpha_i^k\right), \tag{4}$$

where α_i^k represents the time cost for d_i to compute the lay l_k of DNN model.

For the $t_e^{\uparrow}\left(m_i^\tau\right)$, we use binary variable $y_j\left(m_i^\tau\right)$ to indicate whether the task m_i^τ is uploaded to the edge server s_j. If the task is uploaded to the server s_j, $y_j\left(m_i^\tau\right) = 1$. Otherwise $y_j\left(m_i^\tau\right) = 0$.

$$y_j\left(m_i^\tau\right) = \begin{cases} 1: & \text{the task is uploaded to the edge server } s_j; \\ 0: & \text{otherwise.} \end{cases} \tag{5}$$

It satisfies

$$\sum_{j=0}^{n} y_j\left(m_i^\tau\right) = 1. \tag{6}$$

Then we express $t_e^{\uparrow}\left(m_i^\tau\right)$ as the time cost for transmission from d_i to s_j. It can be calculated by

$$t_e^{\uparrow}\left(m_i^\tau\right) = \sum_{j=0}^{n}\left(y_j\left(m_i^\tau\right) \cdot \sum_{u=0}^{r}\left(x_u\left(m_i^\tau\right) \cdot \frac{G_u}{b_j\left(m_i^\tau\right)}\right)\right), \tag{7}$$

where $b_j\left(m_i^\tau\right)$ represents the bandwidth allocated between d_i and s_j for m_i^τ. The bandwidth $b_j\left(m_i^\tau\right)$ satisfies

$$0 < b_j\left(m_i^\tau\right) \leq b, \tag{8}$$

$$\sum_{j=0}^{n} b_j\left(m_i^\tau\right) \leq b, \tag{9}$$

where b represents the total bandwidth between all edge servers and devices.

For the $t_e\left(m_i^\tau\right)$, we use binary variable $y_j\left(m_i^\tau\right)$ to indicate whether the second partition point is l_v. If the second partition point is l_v, $z_v\left(m_i^\tau\right) = 1$, which means after computing layer l_v, the edge server will upload the intermediate data to cloud center. Otherwise, $z_v\left(m_i^\tau\right) = 0$.

$$z_v\left(m_i^\tau\right) = \begin{cases} 1: & \text{the second partition point is the layer } l_v; \\ 0: & \text{otherwise.} \end{cases} \tag{10}$$

It satisfies

$$\sum_{v=0}^{r} z_v\left(m_i^\tau\right) = 1. \tag{11}$$

It should be noted that when $v = u$, the edge server does not participate in the computation of the task and uploads the task directly to cloud center. Then we express $t_e\left(m_i^\tau\right)$ as the time cost for computation from l_{u+1} to l_v of DNN model for s_j. It can be calculated by

$$t_s\left(m_i^\tau\right) = \sum_{v=0}^{r}\left(z_v\left(m_i^\tau\right) \cdot \sum_{u=0}^{r}\left(x_u\left(m_i^\tau\right) \cdot \sum_{k=u+1}^{v} \beta_i^k\right)\right), \tag{12}$$

where β_i^k represents the time cost for s_j to compute the lay l_k of DNN model.

For the $t_c^{\uparrow}\left(m_i^\tau\right)$, we express it as the time cost for transmission from s_j to C. It can be calculated by

$$t_c^{\uparrow}\left(m_i^\tau\right) = \sum_{v=0}^{r}\left(z_v\left(m_i^\tau\right) \cdot \frac{G_v}{b_c}\right), \tag{13}$$

where b_c represents the bandwidth between edge servers and cloud center.

For the $t_c\left(m_i^\tau\right)$, we express it as the time cost for computation from l_{v+1} to l_r of DNN model for cloud center. It can be calculated by

$$t_c\left(m_i^\tau\right) = \sum_{v=0}^{r} \left(z_v\left(m_i^\tau\right) \cdot \sum_{k=v+1}^{r} \gamma_i^k \right), \tag{14}$$

where γ_i^k represents the time cost for C to compute the lay l_k of DNN model.

For each task m_i^τ, there is a deadline $D\left(m_i^\tau\right)$. If $t\left(m_i^\tau\right) > D\left(m_i^\tau\right)$, the task misses its deadline. We denote N as the total number of tasks generated during the scheduling time and N_{miss} as the total number of tasks that missed deadline. Then the deadline miss rate for tasks is

$$\eta_{miss} = \frac{N_{miss}}{N}. \tag{15}$$

Our problem can be formulated as

$$\begin{aligned} &\min \quad \eta_{miss} \\ &\text{s.t. } (3)(4)(6)(7)(8)(9)(11)(12)(13)(14)(15). \end{aligned} \tag{16}$$

In (16), α_i^k, G_u, b, β_i^k, b_c, γ_i^k are are all constants or determined values for specific network. $x_u\left(m_i^\tau\right)$, $y_j\left(m_i^\tau\right)$, $b_j\left(m_i^\tau\right)$ and $b_j\left(m_i^\tau\right)$ are variables. However, these variables appear in almost different forms in all formulas, which makes the original problem model complex and difficult to solve directly. Therefore, for the problem to be solved in polynomial time, we need to make further analysis and find some ways to reduce the complexity of the original problem.

3 Algorithm

In this section, we introduce the algorithm to solve the optimization problem. In the optimization problem, the variables are $x_u\left(m_i^\tau\right)$, $y_j\left(m_i^\tau\right)$, $b_j\left(m_i^\tau\right)$ and $z_v\left(m_i^\tau\right)$, which are not independent but restrict each other. The variables $z_v\left(m_i^\tau\right)$ and $z_v\left(m_i^\tau\right)$ which control the partition point will influence the variables $y_j\left(m_i^\tau\right)$ and $b_j\left(m_i^\tau\right)$ which correspond to the scheduling strategy. In order to solve this problem, we propose online scheduling algorithm for multi-task computation.

3.1 Partition Point Filtering

As we all know, the layered structure of the neural network determines it can be divided at any layer. However, through the analysis of the structural characteristics and the number of parameters of each layer, we find that some neural network layers have no potential to become partition point. Therefore, we propose a DNN partition point filtering algorithm, which can effectively help us to get a suitable set of partition points κ. We first calculate the corresponding total delay for different partition points. Then we give a delay baseline L and take the partition point where the total delay is less than the baseline as the

optional partition point. Since the number of parameters in each layer in the DNN model is structural characteristics, these partition points will perform well on heterogeneous edge servers. At the same time, due to the great reduction of the number of optional partition points, the solution space of the later algorithm can be further reduced.

3.2 Online Strategy

For the online strategy, we propose a scheduling solution for multi-task. We dynamically allocate the bandwidth of the end device and the edge server, and try to calculate the total task time under the hybrid computing approach. While minimizing the deadline miss rate, we choose the computation strategy with the minimum total time. We schedule tasks when there is no idle server, and try to maximize the computing resources of edge servers by means of preemption.

We first determine the number of tasks n_τ generated by the current time slot, the deadline of the task, and the status s^j_{sta} of each edge server. When s_j is working, $s^j_{sta} = 1$. Othwewise $s^j_{sta} = 0$. We denote b_τ as the total bandwidth that can be allocated for the τ-th time slot and n^τ_b as the number of servers that need to allocate bandwidth. We determine the bandwidth between device and server based on the deadline. We select the optimal partition strategy with the minimum total computing time, and calculate the time t^{up}_j for the task to be uploaded to the edge server. When all the partition strategies fail to meet the deadline requirements of the task, we will upload the task to S_0, where S_0 does not participate in the computation of the task.

It should be noted that after the end device completes the computation, the edge server starts to compute at time t^{up}_j. The computing resources of the edge server are available during this period of time. When all edge servers are scheduled but there are still tasks, we first try to use the partition strategy to schedule these tasks. If there is no edge server that meets the requirements, we let the task compute on the edge server as much as possible and choose the partition strategy with the minimum total time. If the task still cannot be completed by the deadline, we upload the task to S_0.

4 Simulation and Experiment

In this section, we present experiments to demonstrate the performance of our algorithms. The DNN model in experiments is VGG16 and the input data is a set of RGB images with each of the size is $320 \times 320 \times 3$. All tasks in our system are inference task and the programming backend is pytorch. The devices are HP notebook computers with Inter Core i5-6200U 2.3 GHz quad core CPU. The edge servers are computers with different processors. The cloud center is the Google Cloud Platform with NVIDIA Ampere A100 GPU.

We first get the values of α^k_i, β^k_i, γ^k_i and G_k and then calculate the total computing time under different partition points and get the optional partition point set κ. We deploy several end devices, 8 edge servers, and a cloud center in

the network. The total bandwidth b is set to 800 $Mbps$, where $s_0 = 100\,Mbps$, $b_c = 35\,Mbps$. h is set to 1000. We get the deadline miss rate η_{miss} under five different approaches and the result of each approach is the average of 20 repeated experiments. According to the analysis of the structural characteristics of VGG16, we find that there are five points have the smallest amount of intermediate data. Therefore, we believe that whether these five partition points can perform well in any case.

In 1000 time slots, for a different number of tasks, we get the deadline miss rate η_{miss} under five different computation approaches. The result is shown in Fig. 3.

(a) Deadline miss ratio (b) Average time cost

Fig. 3. Simulation result. (a) Deadline miss ratio under different number of tasks. (b) Average time cost under different number of tasks

As shown in the Fig. 3(a), when the number of tasks increases gradually, the performance online scheduling algorithm is obviously better. In Fig. 3(b), we show the average completion time of a task. Obviously, our online algorithm has an advantage in any number of tasks.

5 Conclusion

In this paper, we study DNN model partition and online task scheduling over cloud, edge and devices for deadline-aware DNN inference tasks. We first establish our mathematical model and find that the model contains a large number of variables. Therefore, we propose the partition point filtering algorithm to reduce the solution space. Then we propose our online scheduling algorithm to meet the maximum number of deadlines. Experiments and simulations show that our online algorithm reduces deadline miss ratio by up to 51% compared with other four typical computation approaches.

References

1. Li, H.: Deep learning for natural language processing: advantages and challenges. Nat. Sci. Rev. **5**, 24–26 (2018)
2. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition, pp. 41.1–41.12. BMVA Press (2015)

3. Nassif, A.B., Shahin, I., Attili, I.B., Azzeh, M., Shaalan, K.: Speech recognition using deep neural networks: a systematic review. IEEE Access **7**, 19143–19165 (2019)
4. Li, S., Xu, L., Zhao, S.: The internet of things: a survey. Inf. Syst. Front. **17**, 243–259 (2015)
5. Chettri, L., Bera, R.: A comprehensive survey on internet of things (iot) toward 5G wireless systems. IEEE Internet Things J. **7**, 16–32 (2020)
6. Cai, Z., Zheng, X.: A private and efficient mechanism for data uploading in smart cyber-physical systems. IEEE Trans. Netw. Sci. Eng. **7**, 766–775 (2020)
7. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**, 637–646 (2016)
8. Cai, Z., Shi, T.: Distributed query processing in the edge assisted IoT data monitoring system. IEEE Internet Things J. **8**, 12679–12693 (2020)
9. Zhu, T., Shi, T., Li, J., Cai, Z., Zhou, X.: Task scheduling in deadline-aware mobile edge computing systems. IEEE Internet Things J. **6**, 4854–4866 (2019)
10. Duan, Z., Li, W., Cai, Z.: Distributed auctions for task assignment and scheduling in mobile crowdsensing systems. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 635–644 (2017)
11. Han, Z., Tan, H., Li, X., Jiang, S.H., Li, Y., Lau, F.C.M.: Ondisc: online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds. IEEE/ACM Trans. Netw. **27**, 2472–2485 (2019)
12. Khan, L.U., Yaqoob, I., Tran, N.H., Kazmi, S.M.A., Tri, N.D., Hong, C.: Edge-computing-enabled smart cities: a comprehensive survey. IEEE Internet Things J. **7**, 10200–10232 (2020)
13. Alam, M.R., Reaz, M., Ali, M.A.: A review of smart homes: past, present, and future. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**, 1190–1203 (2012)
14. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**, 1738–1762 (2019)
15. Xu, M., Qian, F., Zhu, M., Huang, F., Pushp, S., Liu, X.: Deepwear: adaptive local offloading for on-wearable deep learning. IEEE Trans. Mobile Comput. **19**, 314–330 (2020)
16. Li, E., Zeng, L., Zhou, Z., Chen, X.: Edge AI: on-demand accelerating deep neural network inference via edge computing. IEEE Trans. Wirel. Commun. **19**, 447–457 (2020)
17. Shi, C., Chen, L., Shen, C., Song, L., Xu, J.: Privacy-aware edge computing based on adaptive DNN partitioning. In: 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2019)
18. Mohammed, T., Joe-Wong, C., Babbar, R., Francesco, M.D.: Distributed inference acceleration with adaptive DNN partitioning and offloading. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, pp. 854–863 (2020)