An adaptive DNN Partition Algorithm Greedy Algorithm（ADPA）

zhanglei
School of Computer and Information
Hohai University
Nanjing, Jiangsu Province, China
191307040033@hhu.edu.cn

2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS) | 978-1-6654-1506-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSESS52187.2021.9522172

***Abstract*—To solve the problem of DNN running too slowly on mobile terminals. The traditional approach is to use the edge server to cooperate with deep reasoning. However, this approach cannot be adapted to the partition according to the characteristics of each layer of the network and DNN, and can not achieve satisfactory reasoning acceleration effect. In this paper, an adaptive DNN partition algorithm is proposed to reduce the running time of DNN by considering the characteristics of each DNN layer and adaptively selecting the computing devices that should be divided and deployed in each layer. Experimental results show that the algorithm can effectively reduce the running time of DNN in different network environments and meet the delay requirements.**

Keywords- deep learning; edge computing; computation offloading; edge intelligence

I. INTRODUCTION

In recent years, as mobile devices become more and more intelligent and deep learning [1] has made great progress in speech recognition [2], computer vision [3] and natural language processing [4], etc. Running deep neural network (DNN) applications on intelligent mobile terminal devices has gradually become the focus of people's attention [5]. However, the computing power of intelligent mobile devices is limited after all, and for some DNN applications, mobile devices cannot meet their needs. In particular, some delay-sensitive DNN applications require higher computational time. With the further research on Mobile Edge Computing (MEC) [6], Mobile Edge Computing (MEC) is a concept that integrates Edge Computing into Mobile network architecture. By lowering cloud and business platforms to the edge of the network, mobile edge computing enables users to be closer to computing services, thus achieving the purpose of reducing application computing time. By dividing the computing tasks on the device to the mobile edge computing server, the user can use the higher computing and storage resources on the mobile edge computing server to meet the computing requirements of the computation-intensive tasks and the delay-sensitive tasks. In this way. The emergence of mobile edge computing enables computation-intensive and delay-sensitive DNN applications to realize edge intelligence in a near-real-time response [7].

For the DNN partition algorithm, we hope that the algorithm can choose the partition method adaptively according to the actual situation under different network conditions and devices, and segment the DNN to the edge server to minimize the running time of the DNN and meet the needs of users.

The traditional method[8,9,10] adopts dichotomy, that is, some layers in DNN are divided into terminal devices or edge servers as a whole. We found that we could get better results with a finer-grained approach, where each layer could be adapted to run on an end device or edge server, rather than the traditional two-part approach.

II. PROBLEM MODELING

We assume that every mobile device is able to connect to an edge server for computing. Partitioning is done as soon as the mobile device makes a partition request. Next, we will combine the workload of the edge server with the task dependency requirements to develop the appropriate partition scheme. Similar to literature [8,9], trained DNN is installed in advance on terminal equipment and edge server. During DNN partition, only intermediate data calculated between layers is transmitted, and DNN is not transmitted

A. DNN Dependency

Similar to reference [12], we modeled DNN as a directed acyclic graph (DAG). A layer in a neural network can be represented as a node in a DAG, numbered sequentially from 1.The data dependencies between layers i and k are described by a directed connection between nodes i and k. $pred\ (i)$ is used to represent the data dependencies of layer i, that is, all precursors of node i in DAG.

B. Calculate Time Model

Each DNN computation task can be computed on the terminal or split to the edge server for processing. For each DNN , we use $p_i^d, p_i^e \in \{0,1\}$ to represent the partition decision of layer i, respectively, whether layer i is processed locally or on the edge server.

The partition decisionp_i can be constrained by Equation (1)

$$p_i = p_i^d + p_i^e = 1 \quad (1)$$

only one of the values of $p_i^d$and p_i^e can be equal to 1.

If I don't split layer i, $p_i^d = 1$. The time required for local computation is given by Equation (2), which decreases as the CPU frequency f_i increases

978-1-6654-1506-4/21/$31.00©2021 IEEE

$$t_i^d = \frac{v_i}{f_i} \quad (2)$$

Where v_i represents the number of CPU clocks required to complete the computation task of layer i.

If layer i is split to the edge server, $p_i^e = 1$. When layer i successfully reaches the edge server, the edge server starts to calculate task I, the whole computation time is

$$t_i^{es} = \frac{v_i}{f^{es}} \quad (3)$$

f^{es} is the processing rate of the edge server.

Therefore, the calculation time of layer i is given by formula (4)

$$t_i^c = p_i^d t_i^d + p_i^e t_i^{es} \quad (4)$$

C. Transmission Time Model

When layer I is divided into edge servers, in addition to edge server computation time, there is also transmission time. Transmission time is

$$t_i^{tr} = \frac{d_i}{\mathrm{B}_i} \quad (5)$$

Where B_i is the uplink rate of communication middle layer i, and d_i is the amount of data transmitted to the edge server. B_i can be calculated by formula (6), where W represents the network bandwidth, S represents the signal power, and N represents the noise power

$$B_i = W \cdot log_2\left(1 + \frac{S}{N}\right) \quad (6)$$

D. Time Model

We assume parallel processing on local execution and on the edge server. Therefore, the latency of the DNN application will be equal to the maximum time it takes to complete the task in the task dependency chain. Let K_i be the completion time of the task-dependent chain ending with layer i. K_i can be calculated using the following formula

$$\mathrm{K}_i = \max\{\mathrm{K}_j | j \in pred(i)\} + t_i^c + \sum_{j \in pred(i)} |p_i^e - p_j^e| \cdot t_i^{tr} \quad (7)$$

$|p_i^e - p_j^e| \cdot t_i^{tr}$ represents the transfer time of layer i, but if the precursor of layer i is in the same computing device, the transfer time can be ignored, when $|p_i^e - p_j^e| = 0$,represents that layer i is on the same device as the precursor of layer i, and no additional transfer time is required, when $|p_i^e - p_j^e| = 1$, It means that the precursors of layer i and layer i are located in different devices, and the transmission time needs to be calculated. If layer i has no precursor, we have$\max\{\mathrm{K}_i\} = 0$. At this point, the cumulative execution delay of DNN at layer i is

$$\mathrm{T}_i = \max\{K_j | j \in pred(i)\} \quad (8)$$

When all layers have completed the computation, the total execution delay of DNN can be calculated by

$$\mathrm{T} = \max\{\mathrm{K}_i | i \in \xi\} \quad (9)$$

,$\xi = \{1,2, \dots\}$。

E. Problem definition

The goal of this article is to minimize the running time of the DNN. Under the constraint of data dependence, the problem is expressed as

$$\min \mathrm{T} \quad (10)$$

$$\text{s.t. } (1),(7)$$

III. Partition Decisions Based on Greed Algorithm

We use greedy algorithm to solve the problem, and finally obtain the global optimal partition decision by selecting the local optimal partition decision of each layer.

We define that the optimal partition decision for each layer is made up of three parts, namely state, decision and result, which are described by a tuple (s, a, r).

A. Algorithm Framework

State: We define the state at layer i is s_i divided into $s_i^1, s_i^2, \dots, s_i^j$, and $s_i = \{s_i^1, s_i^2, \dots, s_i^j\}$, s_i^j represents the possible transition of the precursor of layer i, and $s_i^j = (\mathrm{pred}_i^j(\mathrm{i}), T_{i,j}^{pred})$consists of the precursor transition $\mathrm{pred}_i^j(\mathrm{i})$ and the accumulated delay $T_{i,j}^{pred}$.

Action: The decision a_i at the i layer is the appropriate partition decision p_i selected according to the state of the I layer, that is $a_i = p_i$。

Result : When at layer i, the decision a_i is executed in states_i and a result $r(s_i, a_i)$ is obtained. In order to comply with the constraints of (10), the result should be closely related to the running time. Specifically, the goal of our optimization problem, which is to get the minimum running time of DNN, is exactly the goal of greedy algorithms. So we use the delay T_i when we run to layer i to represent the result

$$r(s_i, a_i) = T_i \quad (11)$$

B. Greedy Algorithm

Algorithm **1** Greedy algorithm

1. model DNN as DAG;
2. $i \leftarrow 1$;
3. **while** $i \leq M$ do
4. obtain states_i;
5. Select the optimal decision based on $a_i = \mathrm{argmin}_a T_i$
6. according to the decisiona_i, the state s_i moves to the next state s_{i+1};
7. $i \leftarrow (i + 1)$;
8. **end while**

The greedy algorithm is shown in algorithm 1.As shown in lines 5 and 6, we will select the optimal partition decision according to the state of each layer, and the partition decision will be made according to the current state $s_i$to transfer the state to the next state s_{i+1}.With this information, we can get the final partition decision step by step

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

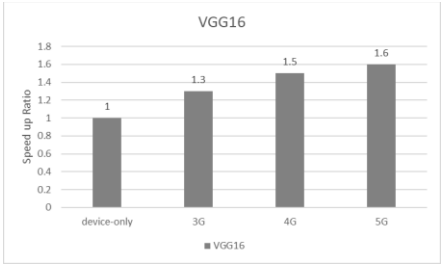
We use Raspberry Pi as mobile smart device, desktop PC as edge server and Ari cloud as remote cloud to verify the feasibility and effectiveness of ADPA. The Raspberry Pi 4B is a mobile smart device with a 1.5GHz processor and 2GB of RAM. PC serves as edge server, 8-core Intel Core server i7-9700k@3.60GHz processor.

In the experiment, we used four different DNNs, including a chain-like model and three DAG models. One chained mode is VGG16, and the three DAG models are Resnet50, MobileNet, and Xception. We also compared the operation effect of DNN after partition in different network states. We used 3G, 4G and 5G as the default transmission technology, and the maximum upload rate was 1.1Mbps, 5.58Mbps and 76.1Mbps respectively.

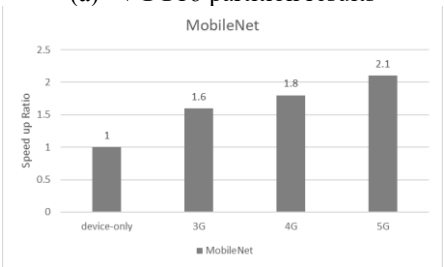
B. Comparison of Partition Effects of Different Models

We tested the partition effects of different types of DNN under different network conditions.

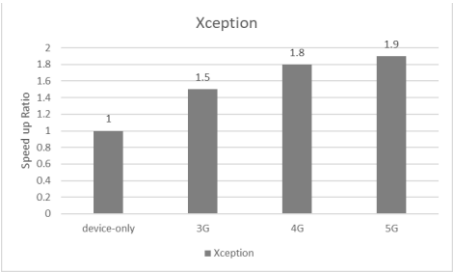
As shown in Figure 1, it can be seen that under the conditions of 3G, 4G and 5G, with the improvement of network conditions, ADPA partition effect is getting better and better. This is because with the increase of network bandwidth, the data transmission time has also been correspondingly reduced. Due to its simple chainlike structure and fewer layers, VGG16 is less affected by network changes than other DNNs no matter what the network conditions are. In DNN of DAG topology, the model elapsed time of MobileNet, Xception, and Resnet50 was also greatly optimized after ADPA partition. It can be seen that the partition effect of ADPA can achieve a higher partition effect no matter in the chain topology or DAG topology, and the partition effect of different DNN is not only affected by the network bandwidth.



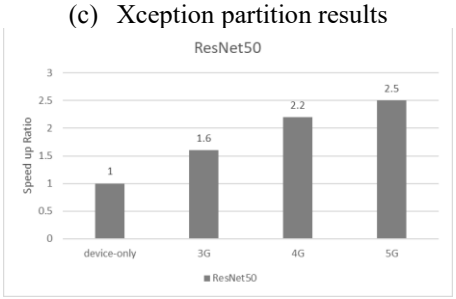
(a) VGG16 partition results



(b) MobileNet partition results



(c) Xception partition results



(d) ResNet50 partition results

Fig. 1 Partition effects of each model under 3G, 4G and 5G

C. Partition Effects Comparison under Dynamic Networks

In this section, we discuss the partition effect under dynamic network conditions.

As shown in Fig. 2, the partition effect changes of ADPA under different bandwidth conditions are plotted. When the network bandwidth is 0, that is, the device can only run DNN locally, the ADPA algorithm will not perform partition, and the acceleration ratio will be 1.With the improvement of network conditions, ADPA will select the final appropriate partition method for DNN partition according to the change of bandwidth. It can be found that with the improvement of network conditions, the partition effect of the four models has been improved. The results show that ADPA can adjust the partition strategy in time with the changes of the network, choose an optimal partition strategy to reduce the running time of DNN, and the final acceleration ratio will gradually tend to a stable value.

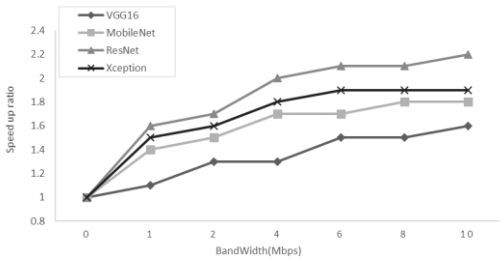


Fig. 2 Partition effect of ADPA under dynamic network

V. RELATED WORK

Neurosurgeon [8] was the first literature to study DNN partition. Neurosurgeon builds a regression model for each layer type to predict the layer delay and power. Using the layer performance prediction model, the Neurosurgeon can dynamically select the best DNN partitioning points, but its method is only suitable for DNNs with chained topologies. For a directed acyclic graph (DAG) topology, the Neurosurgeon

does not find the optimal partition. On the basis of Neurosurgeon,[9]proposed an adaptive distributed DNN reasoning acceleration framework, Edgent, which is suitable for edge computing environment, to further reduce the DNN computing time by early exit reasoning on the appropriate intermediate DNN layer. Although this method reduces the DNN run time, the accuracy of the model is reduced.

[10] proposed a concept of optimal partitioning ratio. By dividing the workload of DNN tasks, the tasks were executed in parallel on mobile devices and edge servers, so that the completion time of the workload unloaded on the edge was the same as that of the workload on mobile devices, so as to further reduce the computing time. DINA [11] proposed a technique to divide DNN into multiple partitions, which can be locally processed or unloaded by terminal devices to one or more powerful computing nodes. This scheme includes an adaptive DNN partitioning scheme and a distributed algorithm based on matching game method to unload computation.

In some studies, the DNN was modeled as DAG and segmented by graph algorithm, taking into account the data dependence between layers of DNN.DADS [12] conducted a comprehensive study on DNN partition under light and heavy load conditions, and proposed an approximate algorithm based on graph minimum cut method by transforming it into a minimum cut problem.

REFERENCES

[1] Lecun Y , Bengio Y , Hinton G . Deep learning[J]. Nature, 2015, 521(7553):436.

[2] Sak H , Senior A , Beaufays F . Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition[J]. Computer Science, 2014:338-342..

[3] Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[C]// Advances in neural information processing systems. Curran Associates Inc. 2012.

[4] Mikolov T , Sutskever I , Chen K , et al. Distributed Representations of Words and Phrases and their Compositionality arXiv : 1310. 4546v1 [cs. CL] 16 Oct 2013. 2013.

[5] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in Proc. Int. Conf. Telecommun. (ICT), Sydney, NSW, Australia, 2015, pp. 313–318.

[6] Mao Y , You C , Zhang J , et al. A Survey on Mobile Edge Computing: The Communication Perspective[J]. IEEE Communications Surveys & Tutorials, 2017, PP(99):1-1.

[7] Zhou Z , Chen X , Li E , et al. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing[J]. Proceedings of the IEEE, 2019.

[8] Kang Y , Hauswald J , Cao G , et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[J]. ACM SIGOPS Operating Systems Review, 2017.

[9] Li E, Zeng L, Zhou Z, et al. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing[J]. IEEE Transactions on Wireless Communications, 2020, 19(1): 447-457

[10] Cao J, Yang L, Cao J, et al. Revisiting Computation Partitioning in Future 5G-Based Edge Computing Environments[J]. IEEE Internet of Things Journal, 2019, 6(2): 2427-2438

[11] Mohammed T , Joe-Wong C , Babbar R , et al. Distributed Inference Acceleration with Adaptive DNN Partitioning and Offloading[C]// IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. IEEE, 2020

[12] Hu C, Bao W, Wang D, et al. Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge[C]. international conference on computer communications, 2019: 1423-143