Cloud-Edge Joint Inference Algorithm for Target Recognition in Cloud-Edge Collaborative Intelligence

Gongyi Xiao[1], Jing Chen[2*], Wen Li[3], Hao Sun[4], Chuanfu Zhang[5], Yudong Geng[6]

Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), China

Email: [1]lmnnxhh@163.com; [2]jingchen94@163.com; [3]17854117620@163.com; [4]1148065598@qq.com; [5]1374582138@qq.com; [6]2725554502@qq.com
* Correspondence: jingchen94@163.com

Abstract. Cloud-edge co-computing has been widely used for real-time processing of image detection scenarios in IoT. The functional limitations of edge devices and how to effectively determine whether an edge device initiates cloud mode is a new challenge. By proposing a joint inference algorithm. The goal of this algorithm is to automatically determine the trigger conditions for activating cloud mode and improve the effectiveness of cloud-edge collaborative reasoning. The algorithm considers the resource utilization of the edge device, the image recognition accuracy of the edge device, and the network conditions between the edge device and the cloud platform to decide whether to trigger the cloud-edge collaborative inference. Experiments are also conducted on a pathology image recognition project, and the results show that the edge devices show significant improvements in end-to-end reliability and workload reduction.

Keywords: cloud-edge computing, joint reasoning, image detection

1 Introduction

Edge devices have been widely used in IoT target detection scenarios. As the industrial target detection task environment becomes more and more complex, but the computing resources of edge devices are limited, therefore, establishing an efficient cloud-edge collaborative joint inference mechanism has become a research hotspot at home and abroad.

Over the years, models that have been compressed have typically been deployed due to resource constraints of edge devices [1], compact DNN architectures [2]. Based on the Deep Joint Source-Channel Coding (DeepJSCC) scheme and the filter pruning strategy, a joint feature compression and transmission scheme is proposed to achieve efficient and reliable inference when the computational resources of edge devices are limited [3]. Based on the study of traditional cloud-based inference DNNs, the impact of network latency is a huge obstacle to inference, so a collaborative approach to cloud-based edge DNNs is proposed [4]. By investigating channel pruning techniques in deep neural networks (DNNs) to remove these inapplicable channels, the accuracy of the model is not compromised [5].

However, using small models or using models that have been compressed, it must sacrifice accuracy and lose more information. Therefore, the cloud-edge collaborative inference architecture, which allows edge devices to handle simple instances and cloud servers to handle difficult instances, not only maintains high model accuracy but also has better practicality. Considering the computational requirements of user terminals, the latency constraints of computational tasks, and the construction of a collaborative offloading system for UAV tasks at the edge, it allows UAVs and cloud servers to collaborate on complex tasks and achieve parallel processing, thus reducing the computational power consumption and durability pressure on terminals [6]. To address the problems of real-time response, dynamism and multimodal data fusion of intelligent robots, a CMfg cloud-edge device collaboration framework is first proposed to support intelligent collaborative decision making of intelligent robots by considering data sources, data processing locations, and data sharing. [7]. A new cloud-edge collaboration architecture (AppealNet) is proposed to predict whether DL models deployed on resource-constrained edge devices can successfully proceed and, if not, to attract more powerful DL models deployed on the cloud. [8].

Therefore, a joint inference algorithm is proposed to determine whether a task is a hard example and enable detection by a large cloud-based model. According to the experimental results, this joint inference algorithm is able to mine hard examples well and maintain a high level of performance in complex target detection scenarios.

2 Problem

Neither model cutting nor model pruning techniques enable the cloud and edge devices to work together, and each detection requires uploading data to the cloud, which wastes a lot of resources; only the detection accuracy of the edge-side model is used as a kind of threshold for whether to enable the cloud model, which cannot achieve the efficacy of cloud-edge collaboration.

Target detection scenarios are usually complex and must use cloud servers for joint inference. The resource utilization, detection accuracy, and network conditions of the edge devices need to be considered to allow simple examples to be detected on the edge devices and difficult examples to be de-emphasized on the cloud servers.

Whether a certain input can be reasonably determined to require support from the cloud model becomes a critical issue.

(1) How to determine the maximum stress load on the edge device when the model deployed on the edge device and the size of the data volume of the detection task are uncertain.

(2) Whether the job re-quires secondary validation of the model on the cloud server even if the stress load on the edge device resources is low.

(3) Whether the network conditions between the edge device and the cloud server are suitable for offloading of the job.

3 A collaborative cloud-edge joint inference method

3.1 Impact factor of the joint inference approach

The joint inference algorithm is determined by the following factors. CPU residual rate of the edge device, memory residual rate of the edge device, data size, average time to start the model on the edge device, accuracy of the edge device model detection, and network latency between the edge to the cloud.

(1) The image queue is called the work queue and its expression is $S=\{S_1,S_2,\cdots S_n\}$. The size of each image is denoted as h, $h=\{h_1,h_2,\cdots h_n\}$. Images with a large amount of data increase the resource occupation of the edge device and reduce the detection performance.

(2) CPU residual rate. It can be expressed as R_a, $R_a=\{R_{a_1},R_{a_2},\cdots R_{a_n}\}$. CPU remaining rate can represent the edge device resource occupancy rate.

(3) Memory Remaining Rate. The memory remaining rate can be expressed by R_b, $R_b=\{R_{b_1},R_{b_2},\cdots R_{b_n}\}$. The memory remaining rate can indicate the edge device resource usage.

(4) Average start-up time model. Set the average startup time of the model to constant a. Models with longer startup times require more resources to support them.

(5) Edge device model detection accuracy. It can be expressed as P. The value of P affects whether the job requires a second test

(6) Network latency between the edge and the cloud. This can be expressed as T. It is used to determine whether the image can be uploaded to the cloud at this time.

3.2 Design and implementation of the Joint inference algorithm

The joint inference algorithm is based on the cloud-edge system framework, and the main steps of this algorithm are as follows:

(1) The edge devices capture the images to be detected through the end devices. These pictures form the job queue S. Each element of the job queue is represented as S_i.

(2) Obtain the value of image data size h_i and model loading time a. Reading the current resource status of edge device. Obtain the values of edge device CPU remaining rate $R_{a_i}$and edge device memory remaining rate R_{b_i}.

(3) Calculating the relationship value between edge device model load and resource utilization $D(r,h)$. Based on the value of $D(r,h)=\frac{R_{a_i}+R_{b_i}}{\log(a+h_i)}-0.5$, obtain the resource pressure load parameter N_1:

(a) When $D(r,h)\geq 0$, $N_1=1$. It means that the edge devices have sufficient resources and the remaining resources can support this detection task. Proceed to step (4).

(b) When $D(r,h)<0$, $N_1=0$. The edge devices resource occupancy is high and the remaining resources are not sufficient to support this detection task. Proceed to step (5).

(4) When $N_1=1$. The edge device has sufficient resources. Detection is performed on the edge device and the actual detection accuracy P_i is output. the value of the relationship between the actual accuracy and the minimum accuracy, $B(p)$ is calculated.

Based on the value of $\mathrm{B(p)}=\frac{\mathrm{P_i}}{\mathrm{C}}$-1, the qualified parameter $\mathrm{N_2}$ of the detection accuracy is obtained.

The value of P_i is the actual accuracy of the output. the value of C is the minimum acceptable accuracy.

(a) When B(p)<0, $\mathrm{N_2} = 1$. This means that the actual accuracy is higher than the lowest acceptable accuracy. The detection result L_i of job $\mathrm{S_i}$ is output by the edge device. The edge device will read the next job S_{i+1} and jump back to the beginning of step (1).

(b) When B(p)<0, $\mathrm{N_2} = 0$. The detection accuracy is unacceptable. This image will be uploaded to the cloud for large model detection in the cloud server. Continue with step (5).

(5) When $N_1 = 0$ or $N_2 = 0$. There are not enough resources or the accuracy is low. Obtain the network latency value $\mathrm{T_i}$ between edge and cloud. Evaluate this value along with the minimum network requirement value m to obtain the network state parameter $\mathrm{W(t)=T_i\text{-}m}$.

(a) When $W(t) < 0$, $N_3 = 0$. The network conditions are not suitable for data transmission. Mark this image as "waiting". The edge device will read the next job S_{i+1} and jump back to the beginning of step (1). When job S_j reaches step (5) and $W(t) \geq 0$, the image will be uploaded to the cloud along with the image that was marked as "waiting".

(b) When $W(t) \geq 0$, $N_3 = 1$. Network conditions are good. The image will be uploaded to the cloud along with the image labeled "waiting". The edge device will read the next job S_{i+1} and jump back to the beginning of step (1).

4 Experiment and result analysis

4.1 Experimental setup

A cluster is created. It consists of one cloud node and two edge nodes. The edge nodes deploy the same small model and the cloud node deploys the large model. Edge node 1 has a joint inference algorithm and node 2 does not. The cloud node is configured with an 8-core CPU and 16G of RAM; the edge node has a 2-core CPU and 16G of RAM. General parameters: $a = 4.2s; m = 150ms; C = 50\%$.

The experimental scenario is set for image detection of pathology, using a dataset of pathological sections of esophageal cancer. The control node will re-identify all images. Images with accuracy below the threshold will be uploaded to the cloud server for detection. The experimental node will first determine whether the amount of resources is sufficient, if not, it will judge the current network conditions, if good, it will upload to the cloud server; if not, it will wait for good conditions before uploading. If the resources are sufficient, it judges whether it needs to be uploaded to the cloud for secondary confirmation according to the recognition accuracy, and whether it can be uploaded according to the network conditions.

4.2 Experimental results

Table 1 shows the average recognition accuracy and time spent for 5 groups of medical pathology image detection experiments (400 images). A total of 302 images were detected by the experimental nodes. 98 images were considered by the joint inference algorithm as exceeding the resource load. 72 images were secondarily confirmed by the cloud due to low accuracy. A total of 400 images were detected by the control node. 87 images with low accuracy were uploaded to the cloud for secondary detection.

Table 1. Experimental results of the control group (Drawing by the author)

Experimental nodes							
Nodes	Detect the number of images			Average Accuracy (%)			Total time
Cloud	170(72+98)			0.7602			156s
Edge	302	Simple examples	Hard examples	0.5699	Simple examples	Hard examples	
		230	72		0.6518	0.3086	
Comparison Node							
Nodes	Detect the number of images			Average Accuracy			Total time
Cloud	87			0.7548			356s
Edge	400	Simple examples	Hard examples	0.5709	Simple examples	Hard examples	
		313	87		0.6386	0.3105	

The experimental nodes using the joint inference algorithm detected 21% fewer total images and used 54% less time. The detection accuracy was 69.79% for the experimental group and only 66.39% for the control group, improvement of 5.12%. By recording the total detection time per 50 images for the experimental node and comparison nodes, as shown in Table 2.

Table 2. Average detection time per 50 images (Drawing by the author)

Experimental node								
range	1~50	51~100	101~150	151~200	201~250	251~300		301~302
Time	22	23	24	21	22	23		1
Comparison Node								
range	1~50	51~100	101~150	151~200	201~250	251~300	301~350	351~400
Time	25s	31s	43s	45s	43s	37s	41s	41s

As can be seen from Table 3, the experimental nodes use the joint inference algorithm, and the resources spent on detection per 50 images is significantly reduced. By recording the resource variation per 50 images detected by the edge device, as shown in Table 3.

Table 3. Resource remaining rate (CPU + Memory) per 50 images detected (Drawing by the author)

Experimental node								
range	1~50	51~100	101~150	151~200	201~250	251~300	301~302	
Resources	46%	40%	40%	38%	41%	39%	40%	
Comparison Node								
range	1~50	51~100	101~150	151~200	201~250	251~300	301~350	351~400
Resources	46	29	21	22	19	18	20	21

According to Table 3: the average value of the sum of CPU and memory remaining rate of the experimental nodes from the beginning to the end of the experiment for the edge devices is 39.75%, comparison nodes is 24.5%.

5 Conclusion

Edge devices suffer from high resource consumption, slow operation, and low resource utilization when processing target detection tasks due to resource constraints. To solve these problems, it is necessary to use joint inference. How to determine whether a task needs to be offloaded to a cloud server. Two factors that affect the efficiency of edge devices are considered from the hardware layer, CPU residual rate and memory residual rate. Then, in terms of task requirements, the recognition accuracy of small models of edge devices is considered. Then, the network link between the edge device and the cloud server is considered from the network layer. The approach is applied to a cancer pathology detection scenario and compared with a system framework that processes only simple instances at the edge and difficult instances in the cloud. The experimental results show that the nodes with the joint inference algorithm improve the detection speed and detection accuracy, reduce the load and energy consumption of the edge devices, and improve the resource utilization of the system.

Funding

This work was supported by Project ZR2020MF034 by Shandong Provincial Natural Science Foundation.

References

1. Zhang Wanpeng, Wang Nuo, Li Liying, Wei Tongquan. Joint compressing and partitioning of CNNs for fast edge-cloud collaborative intelligence for IoT [J]. Journal of Systems Architecture, 2022,125.
2. Li Guangli, Ma Xiu, Wang Xueying, Yue Hengshan, Li Jiansong, Liu Lei, Feng Xiaobing, Xue Jingling. Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning [J]. Journal of Systems Architecture, 2022(prepublish).
3. M. Jankowski, D. Gündüz and K. Mikolajczyk, "Joint Device-Edge Inference over Wireless Links with Pruning," *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Atlanta, GA, USA, 2020, pp. 1-5
4. Li En, Zeng Liekang, Zhou Zhi, Chen Xu. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing[J]. IEEE Transactions on Wireless Communications, 2020,19(1).
5. Liu Mengran, Fang Weiwei, Ma Xiaodong, Xu Wenyuan, Xiong Naixue, Ding Yi. Channel pruning guided by spatial and channel attention for DNNs in intelligent edge computing[J]. Applied Soft Computing Journal,2021,110.
6. Xyya B, Wjna B, Ye Z, et al. UAV-assisted cooperative offloading energy efficiency system for mobile edge computing. 2022.

7. Chen Yang, Yingchao Wang, Shulin Lan, Lihui Wang, Weiming Shen, George Q. Huang, Cloud-edge-device collaboration mechanisms of deep learning models for smart robots in mass personalization, Robotics and Computer-Integrated Manufacturing.
8. M. Li, Y. Li, Y. Tian, L. Jiang and Q. Xu, "AppealNet: An Efficient and Highly-Accurate Edge/Cloud Collaborative Architecture for DNN Inference," *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2021, pp. 409-414.