Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 3842738, 9 pages
https://doi.org/10.1155/2023/3842738

Research Article

Data Exchange Mechanism for Real-Time Object Detection in Cloud-Edge IoT System

Weiwei Miao,[1] Zeng Zeng,[1] Jin Huang,[1] Shihao Li,[1] Yuanyi Xia,[1] Sibo Bi,[1] and Xilong Wang[2]

[1]*Ltd. Information & Telecommunication Branch, State Grid Jiangsu Electric Power Co., 210000, China*
[2]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 211106, China*

Correspondence should be addressed to Zeng Zeng; zengking913@126.com and Xilong Wang; cloudwxl@nuaa.edu.cn

Received 28 June 2022; Revised 26 September 2022; Accepted 25 April 2023; Published 25 July 2023

Academic Editor: Xu Zheng

Real-time target detection based on massive surveillance video data plays a key role in the smart grid, especially in power vision applications. With the indepth construction of power Internet of Things, centralized cloud applications pose challenges to real-time applications due to high bandwidth costs, resulting in delays and loads. Edge computing attempts to perform all the tasks on the edge. However, limited resources on the edge generally cannot guarantee real-time tasks, while complex intelligence algorithms are difficult to fully deploy. Given that both cloud computing solution and edge-only computing solution cannot achieve a good effect, this paper proposes a data exchange mechanism for real-time object detection in a cloud-edge IoT system, which achieves a balance of object detection task executing between delay and accuracy by utilizing the designed intelligent task scheduler. Experimental results performed in real environments using real surveillance video datasets show that by using our cloud-edge collaboration mechanism, the task response time of the system is only about 12% of cloud computing solution while ensuring query accuracy within the practical scope. The system also improves the query accuracy by about 59% compared to edge computing solutions, while the task response time was reduced to its 72%.

1. Introduction

With the rapid development of Internet of Things (IoT) technology, more and more sensor devices are being installed and utilized. It also brings a large wave of smart applications, such as smart grid [1]. Computer vision with the development of artificial intelligence (AI) technology is also to a new height and most of the smart applications are based on computing vision. Take the power vision applications in the power IoTs as examples: the monitoring of personnel trespass in the substation (with the help of an electronic fence and local image recognition, it is found that the personnel in the station have entered the area that does not belong to the scope of this operation) and the violation of personnel dress (the personnel do not wear safety helmet, no work clothes). However, these computer vision applications face the biggest challenge that how to use the IoT technology to collect data for real-time processing, namely, real-time visual application. With the extensive deployment of surveillance cameras, the number of monitoring video data has exploded. Information Handling Services' (IHS) report in 2019 predicted that there would be over 180 million surveillance cameras worldwide in the next few years. Automatic and intelligent video analysis has attracted much academic attention due to the high cost and low effect of extracting information from continuous surveillance video in a traditional manual manner. An important application of video analysis is to conduct real-time statistical analysis of objects in monitoring video streams, which is crucial in intelligent visual application, such as object detection, target tracking, and image classification [2–4].

The traditional solution to upload computing tasks to the cloud becomes no longer applicable with high real-time requirements, strong privacy protection requirements, and huge data transmission. Edge computing [1] is proposed to solve such problems. Edge computing places a partial

computational task on the edge side to relieve the load on the core network. However, edge computing still faces several technical difficulties in solving the requirements of real-time object detection tasks. The first point, the edge computing power is insufficient. Most of the methods of object detection are combined with deep learning technology, which requires a large number of computations through a complex structural neural network, which requires pretty high performance of graphics card and CPU of edge devices, so that the program of object detection cannot be run normally on the edge side and needs to use a lightweight network or be offloaded to the cloud for computing. Second, these application tasks require extremely high real-time performance. For the application scenarios of object detection, the data sources are mostly the video streams of the camera. The general camera acquisition frame rate is about 25 frames per second. In general, an edge device needs to process more than 3 video streams. It requires some delay for handing by the edge devices with limited resources, which could lead to application errors and cause immeasurable losses in some fields. Third, the detection result's accuracy in Edge is often too low to use. In the case of insufficient edge computing power, to achieve the goal of rapid response task, the most current edge computing approaches are to run lightweight neural network model to complete the application task with low resource. While lightweight neural network models (e.g., MobileNet [5]) reduce recognition accuracy by 20%-30% compared to structurally complex neural network models (e.g., ResNet152 [6]), which is sometimes intolerable.

In the existing research for the application of edge video analysis, some scholars carry out a reasonable allocation of resources starting from the dynamic scheduling of resources. Chameleon [7] dynamically selects the optimal resource allocation for the existing deep neural network based video analysis pipeline. The paper found that the optimal parameter combination of the video analysis pipeline tends to stabilize over time and that similar camera video streams can often share the same optimal parameter combination. By making full use of the spatiotemporal correlation of these two video contents, the authors propose to periodically change the configuration, the joint search of similar shooting scenes, and the accuracy of combining the known configuration to solve the problem of dynamic selection resource allocation. This system targets the analysis of offline video and does not handle the online situation, so the dynamic situation of video uploading can be considered. Similarly, VideoStorm [8] was proposed to process thousands of live video streams. The VideoStorm system also only considers offline queries and analyzes and tests for offline video data, which has a big gap between real application scenarios. Another part of scholars started from the video query system [3], mainly developed a large-scale video data query system with low latency and low cost and divided the query into two stages, coarse classification and fine classification, respectively.

In summary, both Chameleon and VideoStorm consider the trade-offs of resource quality and attempt to find the optimal configuration for delay-sensitive video analysis applications. However, they both simply considered regular video analysis without focusing on real-time query optimization of surveillance videos. Current video query systems, including Noscope [2], Focus, and Blazeit [9], specifically optimize large video queries and use neural networks to significantly improve query performance [10–12]. The above studies mainly focus on the rational allocation of resources, the optimization of video stream query, and so on. Part of the problem is solved while also providing some directions for subsequent research work. For example, in real scenes, most of the video streams collected at the edge end have spatiotemporal characteristics, such as the similar proportion of objects in similar scenes and the query results are periodic. Analysis can be performed for spatiotemporal features with different strategies to reduce the workload of video analysis.

The above video analysis system is the most mainstream video analysis system based on edge computing. And most of them conduct optimal resource allocation for offline video set, query acceleration for offline video flow, and do not optimize the query for real-time video flow. Security and privacy are also key concerns for visual applications. In recent years, there have been many priorities related to privacy and security in mobile crowdsourcing [13–15]. However, it is still rare to use cloud-edge cooperation to solve IoT environment, especially power vision application. At the same time, the core of cloud-edge collaboration is the data exchange mechanism. Therefore, this paper proposes a data exchange mechanism for real-time object detection in cloud-edge IoT system to improve the delay of image target detection task response in the application process and realize a balance between detection delay and detection accuracy. The main contributions of this paper are as follows:

(i) A scheduling algorithm for data exchange for mechanism is designed. The algorithm targets the network situation between cloud and edges, the load situation of edge nodes, and the accuracy of the AI models to determine whether the task is performed at the edge devices or at the cloud, aiming to improve the accuracy and delay of target detection. In this paper, through a large-scale stochastic data experiment and small-scale real experiment, the proposed cloud edge collaboration algorithm improves the detection accuracy compared with the edge computing resolution and reduces the detection time compared with the cloud computing resolution

(ii) Based on practical application, considering the complex edge network situation, we select a suitable scheme for cloud-edge communication, reduce the time of cloud-edge message transmission, and realize a practical application system by using edge devices and cloud resources

(iii) This paper performs systematic experiments based on our own implementation prototype system and a real surveillance video data. The experimental results show that the proposed method significantly outperforms traditional cloud computing and edge computing solutions in terms of accuracy and delay

2. Materials and Methods

2.1. Cloud-Edge Collaborative System for Object Detection. This subsection mainly describes the cloud-edge collaboration image object detection system studied in this paper, including the introduction of the cloud-edge collaboration system and the overall layout description of the cloud-edge collaboration mechanism.

2.1.1. The Cloud-Edge Collaborative System. Cloud-edge collaboration, that is, collaborative utilization of resources at the cloud center and edge side, accomplish more complex tasks. As an abstract cloud-edge collaboration system is shown in Figure 1, what the cloud-edge collaboration does is to complete the task request generated by the edge device using the resources of the cloud and edge. In this paper, image target detection and analysis of the video flow are performed.

For the application of edge image object detection in the proposed system, the cloud servers deploy neural network models with complex structure, large computation volume and high result accuracy, while edge nodes deploy neural network models with simple structure, small computation volume, but low result accuracy. In order to give full play to the respective advantages of the cloud-edge collaboration and achieve the accuracy-delay balance of the image object detection task processing, it is necessary to select the cloud center or the edge node as the final position of the response task according to the current network delay and the load of the edge node. Briefly, this paper determines the location of the task response based on the cloud-edge network situation, the edge nodes' load and the cloud's and edge's neural network model accuracy.

In the video object query application proposed in this paper, the user sends the query instruction to the system, and the system detects and analyzes the real-time video flow. The query task of a video stream requires a real-time response to a video frame that contains a user-defined query object, which requires processing of each video frame immediately after receiving it, which is different from traditional video processing. Therefore, this paper simplifies the query task of the video flow to the image object detection task on the edge nodes, which performs the analysis and detection of every frame of the camera-uploaded video in real time. After the user initiates the query, the task result is returned to the user by the response node. Both the cloud center and the edge nodes store their respective task results for subsequent inference model optimization.

2.1.2. Practice System Architecture. The entire system consists of a cloud center server, a number of edge nodes, and multiple cameras. Referring to the structure of Figure 1, the edge device is embodied as a camera. The cloud server serves all the edge nodes, and one edge node can serve multiple cameras. The edge node is responsible for receiving the camera data and communicating with the cloud center in response to user requests. Each edge node has a local edge task scheduler, and the user needs to specify an edge node when initiating a query task. After the edge node receives the user task request, it starts the corresponding task, and the edge task scheduler schedules the execution process specifically. Finally, the edge node returns the result to the user. After receiving the results, the user needs to feedback on the accuracy of the task, and the edge node receives and saves the feedback information, thus promoting the edge task scheduler to perform more reasonable decisions. If the user does not return any feedback, the edge node default that the query task's result is correct.

The specific object detection application system architecture is shown in Figure 2. The gate in the figure indicates that the results of the final task execution will be from the neural network model in the cloud or the neural network model at the edge. Specific decision results are determined by the scheduling algorithm. The whole system is divided into training stages and image recognition stages. The neural network model at the edge adopts the core of MobileNet, which is a deep, separable convolution. This technology can not only reduce the computational complexity of the neural network model but also greatly reduce the running memory size required by the neural network model itself, but the accuracy of image recognition is relatively low. In the cloud, ResNet152, with more complex neural network structure, requires more computational power and larger running memory than MobileNet, but the identification results are more accurate to make full use of the resources of the cloud server. ResNet and MobileNet are used as experimental models because, in the current production environment, most visual DNN applications use neural-network architectures based on them to fine-tune according to the application scenarios. The effect of our scheduling mechanism applied to the two models would have a general applicability. In order to make full use of the advantages of cloud and edge and make the image identification results more accurate and faster, this paper designs a task scheduling algorithm to make decisions according to the current network status and the accuracy of edge nodes' neural network model and choose the optimal scheme to realize the load balance of the whole system and the accuracy-delay tradeoff of the query task.

2.2. Design of Data Exchange Mechanism. This subsection details the edge real-time task scheduler designed in this paper and the data exchange mechanisms and tools to use for cloud-edge collaboration practical systems.

A data exchange mechanism for cloud-edge collaboration needs to consider two aspects. The first aspect is the task scheduler. In the cloud-edge collaboration mechanism, it needs to play to their respective advantages through the cloud and the edge so as to complete the task together. And how to split the task and find the optimal terminal that completes the task is determined by the task scheduler. This paper is aimed at the task of image object detection. The usual practice is to preprocess images on the edge and run the first few layers of the neural network. However, the amount of data transmitted between the layers of the neural network is too large which caused a heavy bandwidth cost. Therefore, our system adopts the complete picture transmission, regardless of the problem of task splitting, simplifying

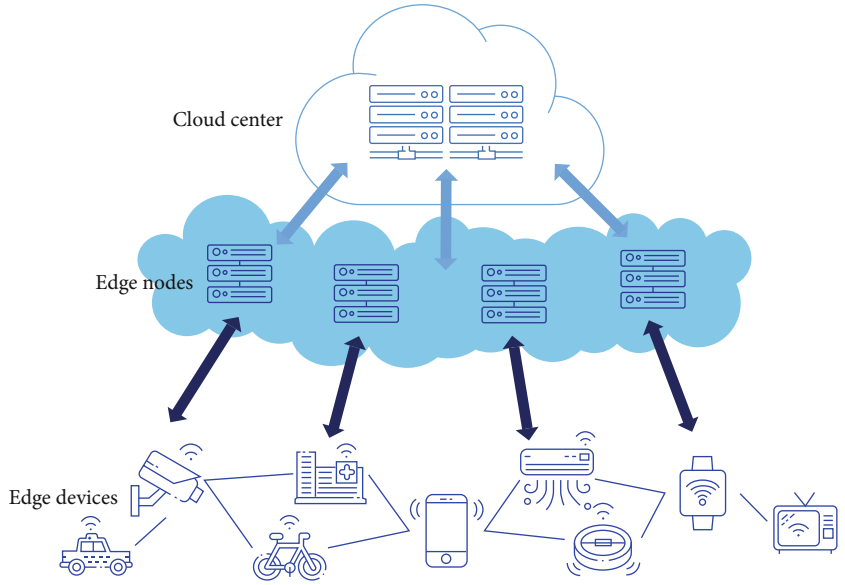

Figure 1: Cloud-Edge collaborative system architecture.

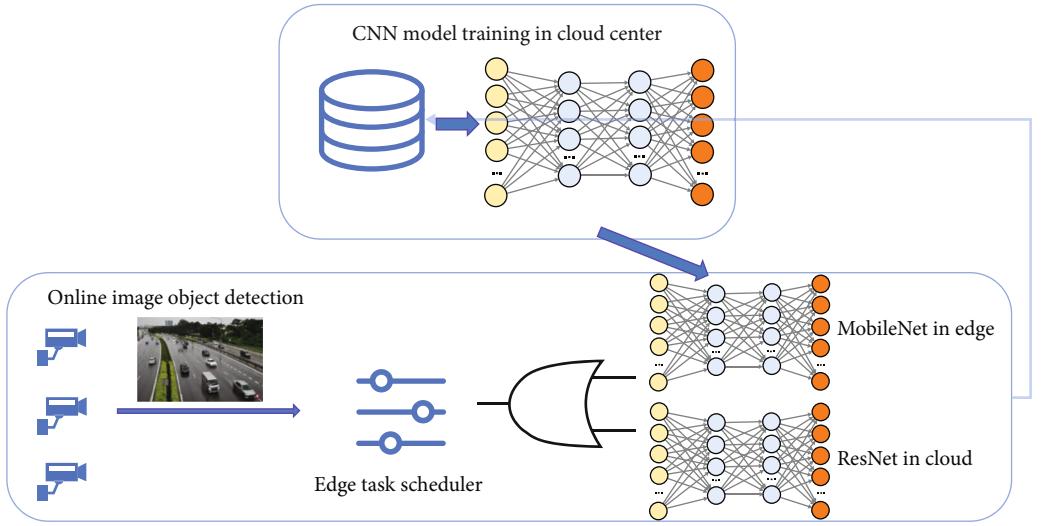

Figure 2: Image object detection system architecture.

the problem while reducing the network transmission pressure. Therefore, the task scheduler only needs to simply determine the running location of the image target detection task. The scene faced by this paper is the video flow detection in the edge environment, while large edge side with high load and low detection delay are easy to appear as task timeout. And the detection speed of the cloud is fast with an uncertain transmission time, fully offloading the task to the cloud is not suitable for time-insensitive query tasks. Then, the task scheduler in this paper comprehensively considers the cloud-edge network status, neural network model accuracy, and edge-side load status. The second aspect is the consideration of the cloud-edge transmission mechanism. In the edge environment, the network is unstable, and situations of offline and low response delay often occur. So, the system explores the cloud-edge transmission mechanism of the actual architecture. For high frequency picture transmission, this paper adopts TCP long connections and changes the image format to ensure the picture quality and compress the picture size. Meanwhile, the acknowledgment confirmation mechanism is adopted to ensure that all kinds of messages on the edge can be transmitted to the cloud center.

Finally, based on the above technique, we propose a data exchange mechanism to balance task accuracy and delay. First, when a task reaches the edge node, the scheduler adds it to two queues at the same time: a transmission queue and a local execution queue. So the task is transmitted to the cloud center and executed locally at the same time. Then our scheduler determines which results as a response according to an algorithm; the specific algorithm is given in the next chapter. In this way, if the network is good, the cloud execution result is adopted with high accuracy. In

the case of a poor network, the edge execution result with unstable accuracy is adopted, but it can compensate for the delay problem. In this way, the average accuracy and time delay of the tasks are well balanced.

2.2.1. The Edge Real-Time Task Scheduler. The whole image target detection process is shown in Figure 3, mainly divided into four stages, and the specific practice of each step are detailed as follows.

The first step is the image data transmission, where the edge camera divides the collected video stream and sends each frame of the segmented image to the edge node. Assuming there are n edge nodes, edge node i is represented by E_i, and i values range from 1 to n. Each edge node maintains a task queue with N_i images waiting for detection and identification. Using t_i, it represents the average time that edge node E_i recognizes an image. The final location of the task response is determined by real-time task scheduler decisions. The real-time task scheduler calculates the optimal edge node with the smallest expected waiting time for processing the pictures. The calculation formula is

$$d = \underset{1 \le i \le n}{\operatorname{argmin}} N_i * t_i, \tag{1}$$

where d represents the final task execution location.

In the actual system of this paper, each edge node E_i stores the task queue N_x and t_x values of the other edge nodes E_x near it in a local database, SQLite, while calculating the task scheduling location d in real time based on all the N_x and t_x value changes.

Optimization scheduling is the core work of this paper. Its decision logic is detailed below:

(1) For each task request receives by each E_i, the request is placed in a send queue Q_{send} waiting for sending and a local recognition queue Q_{work} waiting for recognition

(2) For each task request taken from Q_{work}, the same task in Q_{send} is sent to the cloud for processing at the same time. If the cloud's reply is received before the local processing is completed, the cloud detection results are adopted directly, and the current local task is completed to handle the next task

(3) If the cloud fails to calculate the results in advance, the local task ends with a result confidence f. Here, a minimum tolerance value is set up for m. When the confidence f is less than or equal m, the recognition result of the edge node is not credible; that is, the task result is no corresponding object to be detected in this image. When f is greater than m, the recognition result is trusted. m has the formula of

$$m = \underset{1 \le i \le n}{\operatorname{argmax}} \frac{x_i}{N_i}, \tag{2}$$

where N_i is the number of images of E_i to be detected in the test set and x_i is the number correctly identified; that is, m is the maximal number of images correctly recognized by each edge node, ensuring a better overall recognition result.

(4) When the task processing result f of the edge node is greater than m, the scheduler further balances the task execution time and result accuracy. The scheduler comprehensively determines through the cloud task processing duration and the accuracy of its improved results to obtain a reference score s. The specific calculation formula is

$$s = p * \alpha - (1 - p) * \frac{t_{\text{cloud}}}{t_i}, \tag{3}$$

where p is a weight parameter customized by the user, namely, the trade-off between accuracy and task execution time, and it takes values between 0 and 1. α represents the improved accuracy values obtained by subtracting the accuracy values of the cloud neural network modelfrom the accuracy values of the edge nodes. t_{cloud} indicates the time elapsed from sending to the cloud to receiving the result, which is determined by the last ten communication delays between the cloud center and the edge node through least squares fitting to obtain the curve and calculate the estimated delay.

(5) Due to the unstable network situation, it may occur that the edge node still does not receive the result returned by the cloud after waiting for a long time when f is less than m, and then the detection result of the edge node is directly returned

In the object detection step, the real-time task scheduler gives the scheduling results, and the scheduled node begins to perform the task. In the actual system of this paper, the node first uses the Yolov3 network to detect the common objects in the video stream, followed by ResNet152 or MobileNet to classify the detected objects and obtain the identification results.

Finally, the whole task is completed, and the identification results are returned to the user and deposited in the cloud database to optimize training the neural network model.

2.2.2. Data Exchange Mechanism Analysis and Tool Selection. To ensure the reliability of picture transmission, this paper adopts the TCP transmission control protocol. Then, during task requests peak hours, dozens of query requests are initiated per second. In this situation, establishing a complete connection for each query request causes a huge communication overhead. Under the same number of communication conditions, TCP-long connections can save a fraction of the time compared to the regular TCP connection mode. Therefore, this paper chooses to establish a TCP long connection in the actual system, reducing the time cost of each handshake. Secondly, the WebP data transfer format is adapted to our system. The advantages of WebP can be reflected in its more excellent image compression algorithm,

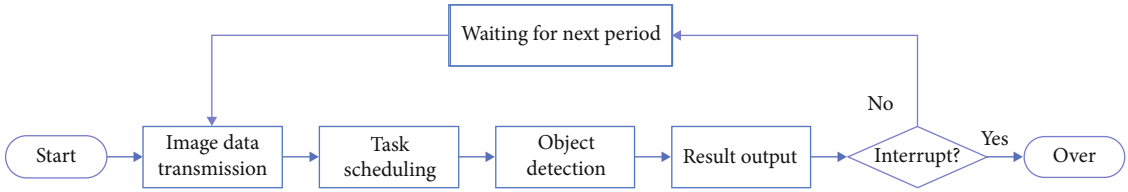

Figure 3: Image object detection flow chart.

so that the picture has a smaller volume and does not change the picture quality. WebP transmission can greatly reduce the transmission time without affecting the picture quality. Through these two transmission mechanisms, the system greatly reduces the time of cloud-edge data transmission.

In the construction of a practical system, we finally realized the picture transmission with the NetCat tool through analysis and exploration. Compared with several mainstream file transfer tools such as Rsync and Scp under the Linux operating system, the main advantages of NetCat are reflected in fast transmission speed and strong adaptive ability. The transmission tools, such as Rsync and Scp, more or less encrypt the transmission content itself, block, and other operation to ensure the security of the transmission or facilitate break-point retransmission, and NetCat directly connects with the TCP/UDP protocol of the next layer, enabling completely transparent transmission.

As shown in Table 1, this paper compares the three tools of transmission, with five transfers of a total of 61 MB of fragmented data on two nodes with the same configuration of A and B. Test results yield the fastest and least time transfer, followed by Rsync and Scp. Strong self-adaptation is reflected in the NetCat tool can choose the optimal transmission mode according to the network quality, and the specific details are no longer redescribed.

2.3. Experimental Settings. In terms of hardware, we select some edge nodes with chip with AI-computing acceleration capability in the design of the actual system. The cloud center uses multiple servers with graphic cards, which ensure absolutely sufficient computing resources.

In the software configuration, the entire system is deployed above the KubeEdge [16] architecture and implemented with Python. Also, we use the Docker container to simplify the application environment deployment migration. For communication between edge nodes and the cloud center, we use the lightweight communication protocol MQTT. Finally, each edge node deploys a small database in SQLite to store the required historical data.

On the configuration of the transmission mechanism, we change the TCP configuration parameter for all machines in the operating system, which are *ipv4.tcp_keepalive_intvl*, *net.ipv4.tcp_keepalive_probes*, and *net.ipv4.tcp_keepalive_time*. For instance, the three parameters were, respectively, configured as 75, 9, and 7200. The first parameter indicates how long the probe packet is sent after the *keepalive* packet starts. The second represents the number of *keepalive* packet sent before deciding the connection is disconnected and notifying the application layer. The third represents the last sent data interval and the first *keepalive* packet after the connection is marked as requiring *keepalive*. Then, if a client sends a package labeled *keepalive* and disconnects, the server needs $7200\,s + 9^{*}\,75\,s = 7875\,s$ before the connection is released, achieving the effect of the TCP long connection.

2.4. Experimental Data Set. To better simulate the real situation, we select the open source data set of YouTube-8 M [17], which has the following features:

(i) Videos are publicly available, downloadable, and have at least 1,000 frames each

(ii) The length of the videos' time is all between 120 s and 500 s

(iii) Videos are all associated with at least one knowledge graph entity

We analyze each video, ultimately extracting tens of thousands of images from 25 videos with a resolution of 1080 p and 30 fps. These videos cover representative locations at intersections, streets, schools, parks, and more. In our experiment, 80% of the images were selected for training, and the rest of the data were used to validate the accuracy of the model.

Since the accuracy of query systems based on object detection is highly dependent on the training data, an intuitive approach is to use images captured by a video camera to train a model. However, this leads to excessive training costs and waiting times. It is feasible to jointly train neural models at the edge to simultaneously serve a set of similar cameras. Clearly, cooperation between similar scene cameras also increases the amount of training data for each edge model, effectively preventing overfitting due to the small training volume. Therefore, this paper generates a configuration file by calculating the proportion of the item appearing for each camera based on the video frames generated by each camera and uploaded to the cloud. The camera profile can roughly represent the scene where the camera is deployed. On the one hand, if the number of cars in the two surveillance videos is large but the proportion of people is low, both cameras tend to approach the main road. On the other hand, if there are more people and fewer cars, they tend to be close to crowded squares or walking trails. So, in our system, the profiles for each camera are calculated and clustered (based on their profiles) using the K-Means algorithm [18]. Cameras within the same cluster were considered as in the same context, thus sharing the same context-specific training data set to optimize training quality.

Table 1: Time cost comparison of the three transmission tools.

Transmission tool	1	2	3	4	5	Average value
Scp	253 ms	263 ms	288 ms	229 ms	254 ms	257.4 ms
Rsync	139 ms	158 ms	161 ms	150 ms	141 ms	149.8 ms
NetCat	124 ms	135 ms	123 ms	145 ms	141 ms	133.6 ms

3. Results and Discussion

This section provides a concise and precise description of the experimental results.

To evaluate the effect of the real-time task scheduling mechanism designed in this paper, we analyze the experimental results with different task scheduling mechanisms under different solutions, namely, traditional cloud and edge computing solutions. The comparative different task scheduling mechanisms are as follows:

(i) *Edge only*: all images are identified at the edge, with no help from the scheduling system and cloud

(ii) *Cloud only*: all pictures are sent to the cloud for recognition, and the edge only collects data

(iii) *Thin system*: a cloud edge system without task scheduling, the minimum tolerance value m is set to 0.5, and the default cloud processing time t_{cloud} is 0.6 s

We evaluate the performance of the above three schemes and the task scheduling mechanism we design in terms of accuracy and average delay, respectively.

We use the F -score to measure the accuracy of the query results. The F-score considers both the precision p of the query and the recall rate r, as

$$F_\lambda = \left(1 + \lambda^2\right) * \frac{p * r}{\lambda^2 + r}. \tag{4}$$

Since the recall rate calculation wastes a significant amount of time in the detection on the edge, the value of λ is set to 2 to ensure a low recall rate.

3.1. Results in the Case of Cloud-Edge Isomorphism. Consider scenarios with isomorphic devices first. In this experiment, simulation was constructed on multiple identical devices.

Results in single edge node and cloud situations are shown in Table 2. Compared with the cloud-only query method, the average task delay of the task scheduling mechanism proposed in this paper is only about 12% of the original, and the query accuracy is also within the scope of application tolerance. The mechanism proposed in this paper improves the accuracy by about 59% compared to the edge-only method, while only the task delay is only 72% of the time delay.

3.2. Results in the Case of Cloud-Edge Heterogeneity. Then, we consider scenarios with heterogeneous devices. In this experiment, simulation was constructed on multiple heterogeneous devices. As the results shown in Table 3, the average delay decreases by nearly 3.3 times compared with all transfers to the cloud center. The mechanism in this paper, compared to mechanisms of edge only and thin system, improves the accuracy by 55% and 7%, respectively, with the average delay of only 73% and 42% of the two methods.

Table 2: Results in the case of cloud-edge isomorphism.

Deployment scheme	Accuracy	Average delay
Thin system	83.22%	2.372 s
Our system	91.43%	1.321 s
Edge only	32.40%	1.83 s
Cloud only	99.6%	15.232 s

Table 3: Results in the case of cloud-edge heterogeneity.

Deployment scheme	Accuracy	Average delay
Thin system	79.21%	23.21 s
Our system	86.77%	9.82 s
Edge only	31.22%	13.29 s
Cloud only	99.8%	36.25 s

4. Discussion

As for the results in the case of cloud-edge isomorphism, the cloud-edge collaborative mechanism proposed in this paper performs a good trade-off between accuracy and task response delay, with real-time task scheduling for time delays in edge nodes and reducing a large amount of waiting time. At the same time, the appropriate task processing of edge nodes also greatly improves the accuracy of object detection. And for the results in the case of cloud-edge heterogeneity, in terms of overall execution delay, each deployment scheme is greatly increased, but the overall system performance ranking is not much different. This is because heterogeneous devices produce some intermediate overhead when processing the same computing task, so the increase in execution delay is expected. Our system achieves the best results in both homogeneous and heterogeneous hardware environments, which shows that the implementation effect of the cloud-edge collaborative mechanism proposed in this paper is not affected by the physical devices.

In addition, we calculate the waiting time for each frame of the input video, with the probability density function shown in Figure 4. Figure 4 reflects the probability distribution of the image recognition delay under different scheduling mechanisms. As can be seen from Figure 4, the average time of image recognition using the cloud-only mechanism

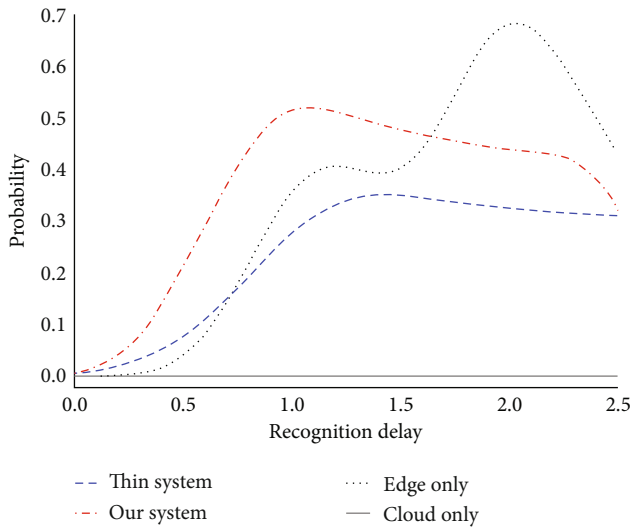

Figure 4: Recognition delay probability distribution plots.

is much more than several other mechanisms, so the probability of the recognition time before 2.5 s is always 0. Our mechanism is marked by red dot lines, and the recognition delay is basically distributed between 1 s and 2.5 s, and more than 70% of the image recognition delay is lower than the other mechanisms. That is to say, the scheduling strategy of this paper can effectively control the average delay in a low range, and it greatly reduces the average latency of query requests. In fact, when the query delay of a certain frame is too large, the query results of that frame are meaningless. The edge real-time task scheduler in this paper enables a cloud high-precision classifier in nonbusy time periods to identify more images, significantly reducing the user waiting time, which adaptively balances the system load and query performance.

There are still some shortcomings in this study. We only consider the images after the video stream was cut and did not consider full video stream detection. This article makes the following outlook for future work:

(i) Consider the situation where the actual scene is input into the video stream. This situation requires target detection of a large number of images and requires better algorithms

(ii) Better scheduling strategies can be used to optimize the scheduler designed in this paper, for example, by making online decisions by reinforcement learning

(iii) Some basic scheduling algorithms can be implanted into the cloud-edge collaborative framework, so that the cloud-edge collaboration can be truly implemented

5. Conclusions

The maturity of target detection technology has brought great convenience to the applications of various intelligent visual application, such as intelligent transportation testing and public safety. At the same time, the high hardware requirements of target detection technology also bring new challenges, such as insufficient computing power on edge devices, too long cloud transmission time. In this situation, this paper mainly performs the following two points:

(i) A data exchange mechanism for real-time object detection in cloud-edge IoT system is proposed. The system can significantly improve identification accuracy and reduce query delay while ensuring system stability

(ii) We experimentally implement the proposed system separately on edge homogeneous/heterogeneous devices and experimentally demonstrate the advantages and effectiveness of the proposed cloud-edge collaborative mechanism

Data Availability

We selected the open-source data set of YouTube-8M from http://research.google.com/youtube8m.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Science and Technology Project of State Grid Jiangsu Electric Power Company (No. J2021028).

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "NoScope," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, 2017.

[3] K. Hsieh, G. Ananthanarayanan, P. Bodik et al., "Focus: querying large video datasets with low latency and low cost," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad CA, USA, 2018.

[4] G. Ananthanarayanan, P. Bahl, P. Bodik et al., "Real-time video analytics: the killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[5] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: efficient convolutional neural networks for mobile vision applications," 2017, https://arxiv.org/abs/1704.04861.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, 2016.

[7] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 253–266, Budapest, Hungary, 2018.

[8] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, Boston, MA, USA, 2017.

[9] D. Kang, P. Bailis, and M. Zaharia, "BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics," 2018, https://arxiv.org/abs/1805.01046.

[10] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[11] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 144–153, Dallas, TX, USA, 2019.

[12] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[13] Y. Wang, Z. Cai, Z.-H. Zhan, B. Zhao, X. Tong, and L. Qi, "Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1033–1046, 2020.

[14] Z. Sun, Y. Wang, Z. Cai, T. Liu, X. Tong, and N. Jiang, "A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing," *International Journal of Intelligent Systems*, vol. 36, no. 5, pp. 2058–2080, 2021.

[15] Y. Wang, Y. Gao, Y. Li, and X. Tong, "A worker-selection incentive mechanism for optimizing platform-centric mobile crowdsourcing systems," *Computer Networks*, vol. 171, article 107144, 2020.

[16] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 373–377, Seattle, WA, USA, 2018.

[17] S. Abu-El-Haija, N. Kothari, J. Lee et al., "Youtube-8m: a large-scale video classification benchmark," 2016, https://arxiv.org/abs/1609.08675.

[18] M. A. Wong and J. Hartigan, "Algorithm AS 136: a K-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.