Distributed and Efficient Object Detection via Interactions Among Devices, Edge, and Cloud

Yundi Guo, *Student Member, IEEE*, Beiji Zou, Ju Ren, *Member, IEEE*, Qingqing Liu, Deyu Zhang, *Member, IEEE*, and Yaoxue Zhang, *Senior Member, IEEE*

***Abstract*—With the rapid development of Internet-of-Things and communication techniques, media transmission in surveillance applications is gradually relying on wireless networks. Meanwhile, the emergence of edge computing has pushed the media data analysis from the cloud to the edge of the network to achieve fast response for delay-sensitive media processing tasks. Object detection is a representative delay-sensitive image processing task in surveillance applications, but faces significant challenges in this context. For example, how to compress images for transmission in wireless environment without compromising the detection accuracy, and how to integrate and update local inference models online in an edge computing-based object detection system. In this paper, we propose an object detection architecture based on edge computing to achieve distributed and efficient object detection for surveillance applications. Under this architecture, we develop an adaptive Region-of-Interest-based image compression scheme for end devices to efficiently compress their captured images for wireless transmission but not to sacrifice the object detection accuracy of edge servers. Furthermore, we carefully design distributed and communication-efficient interactions among end devices, edge servers, and the cloud to dynamically optimize the object detection accuracy online. Extensive simulation results demonstrate that our proposed architecture not only achieves a competitive detection accuracy to traditional cloud-based objective detection solution with reduced response delay but also significantly improves the image transmission efficiency with adaptive image compression ratio.**

***Index Terms*—Object detection, edge computing, image compression, RoI, wireless surveillance applications.**

I. Introduction

SURVEILLANCE applications are widely applied in traffic monitoring, national security and many other fields. These applications mainly consist of numerous high-resolution cameras to constantly capture the real-time pictures or videos of interested areas. Since the captured real-time data are usually generated in a high speed, deploying dedicated fiber networks is a common practice for supporting this high-throughput data communications. Meanwhile, with the advance of machine learning technology in recent years, surveillance applications have gradually relied on more complex algorithms and larger computing resources to improve performance. In order to concentrate resources on processing huge amounts of media data, cloud computing becomes a dominant solution for surveillance applications. In the traditional cloud-based architecture, massive real-time media data is sent directly to the cloud for data processing and knowledge mining via Internet or dedicated high-speed fiber networks, leading to unstable media processing delay or high network deployment cost [1].

As we are stepping into the era of Internet-of-Things, the dramatically increasing mobile devices and their generated data pose significant challenges on the response delay and communication burden of cloud computing. Instead of transmitting all the data to the cloud for analysis, processing the data at the edge of the network, known as edge computing, becomes a promising complement of cloud computing. By fully exploiting the computing and storing capabilities of nearby devices and infrastructures, edge computing can process and analyze the real-time media data at the edge of the Internet in a distributed way [2], [3]. Some researchers have already tried to utilize edge caching scheme to improve the quality of mobile media services while effectively alleviate the backhaul burden and reduce response time [4], [5]. Meanwhile, with rapid development of 5G technology, transmitting the massive amounts of real-time data over the wireless network becomes a promising solution, which can greatly reduce the cost of dedicated network construction for surveillance applications.

Object detection is one of the typical delay-sensitive and computation-extensive functionalities of surveillance applications, which can significantly benefit from the computing paradigm shift. However, when we apply edge computing for object detection, it also faces some new challenges arisen by the new computing paradigm and network environment. First, although 5G will enable wireless transmission to be feasible for real-time media data transmission, the wireless networks with limited bandwidth and insufficient stability are still the bottleneck for media data transmission. In this case, IoT devices need to compress the data sufficiently without compromising

Manuscript received November 2, 2018; revised March 8, 2019 and April 11, 2019; accepted April 16, 2019. Date of publication April 22, 2019; date of current version October 24, 2019. This work was supported in part by the Natural Science Foundation of China under Grants 61702562, 61702561 and 61573380, in part by the 111 Project under Grant B18059, in part by the International Science and and Technology Cooperation Program of China under Grant 2013DFB10070, and in part by the China Hunan Provincial Science and Technology Program under Grant 2012GK4106. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shiwen Mao. *(Corresponding author: Ju Ren.)*

The authors are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China. Yundi Guo and Beiji Zou are also with the Hunan Province Engineering Technology Research Center of Computer Vision and Intelligent Medical Treatment, Central South University, Changsha 410083, China (e-mail: csgrandeur@csu.edu.cn; bjzou@csu.edu.cn; renju@csu.edu.cn; liuqingqing@csu.edu.cn; zdy876@csu.edu.cn; zyx@csu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2912703

the accuracy of object detection in the edge servers. Second, both the data compression algorithm for IoT devices and the object detection algorithm for edge servers need to be sufficiently lightweight and able to process large amounts of real-time media data. Third, compared to cloud computing, distributed edge servers with locally trained models may lack global knowledge for object detection [6]. Thus, the last but not least challenge is how to use the cloud to integrate the local models of different edge servers and efficiently update them through interactions between the edge and the cloud.

In this paper, we propose an object detection architecture based on edge computing for surveillance applications. In this architecture, real-time captured image data is highly compressed by a Region-of-Interest (RoI) based image compression algorithm before transmitted to the edge server through wireless network. The compressed image can well preserve the original data of the regions containing interested objects for object detection. The end devices are responsible for data compression, the edge servers are responsible for object detection, while the cloud is responsible for data and model integration. Moreover, the end devices, the edge servers, and the cloud can provide distributed and stable object detection services through efficient communication and cooperation. The key contributions of this paper are summarized as follows:

- We propose an end-to-end framework that leverages end-edge-cloud orchestrated computing to achieve distributed and timely object detection. To the best of our knowledge, this is the first work to provide such a systematic solution.
- Under the proposed framework, we develop a tailored BING based RoI Extraction method, named BRE, which can adaptively compress the images to achieve the trade-off between detection accuracy and compression ratio.
- We design the collaborations and interactions among end devices, edge servers and the cloud to dynamically update the local object detection models on edge servers and BRE methods on end devices for detecting new objects.

The remainder of this paper is organized as follows. Section II provides a brief review to the related work, while Section III introduces the proposed edge computing based object detection system. We present the tailored RoI based image compression algorithm and the detailed interactions among end devices, edge servers and the cloud in Section IV. Finally, Section V evaluates the performance of our proposed solution, followed by a conclusion in Section VI.

II. Related Work

With the rapid development of artificial intelligence during recent years, object detection has been a hot topic in the research area of image processing. In this section, we briefly review the related works from three perspectives, including image compression and deep neural networks for object detection, as well as edge computing systems for image processing.

A. Image Compression for Object Detection

In the era of IoT, end devices usually work in wireless networking environments, leading to great burden on the high-frequent and large-scale multimedia transmission for cloud computing or edge computing. Multimedia data need to be sufficiently compressed to alleviate the pressure of wireless transmission. For image compression, Joint Photographic Experts Group (JPEG) [7] is a commonly used lossy compression protocol for digital images. Compression ratio can be adjusted to accommodate the trade-off between storage size and image quality. Typically, JPEG can achieve a compression ratio of 1/10 with little perceptible loss in image quality. JPEG2000 [8] is an image compression and encoding protocol designed to improve image compression quality relative to JPEG and add new features such as scalability and editability. However, difficulties and challenges may arise when applying these traditional image compression methods in specific computer vision tasks, like object detection. Slightly compressing the image does not effectively reduce the pressure on the wireless network, while high-intensity compressed images tend to significantly reduce the accuracy of object detection.

In order to compress images sufficiently and meanwhile keep the details of the key areas containing specific targets, region of interest (RoI) coding is proposed [8]. It highly preserves RoI area details and greatly compresses other areas. But how to identify RoI areas is still a challenging problem, since RoI is variously defined for different tasks. There are plenty of research works extracting saliency as RoI [9] which is outstanding in color or texture. This kind of methods is usually based on bottom-up models [10]. However, most of object detection tasks aim to identify the areas with latent objects which may not be conspicuous, which needs goal-driven methods. Object proposal generation is such kind of solutions, which can find latent objects with bounding boxes with high recall. Some representative solutions include objectness scoring [11]–[14], similarity grouping [15]–[17], supervised learning [18]–[20], and hybrid/part-based proposal generation [21], [22]. Nevertheless, most of the existing solutions bring a large computation burden and consequently are not suitable for lightweight end devices. To achieve the trade-off between performance and computation efficiency, a lightweight model, named binarized normed gradients (BING) model [13] is proposed, which can provide fast proposal generation with a speed of 300 fps.

B. Deep Neural Networks for Object Detection

In recent years, the fast development of deep learning technology has significantly improved the performance of object detection [23]. Faster R-CNN [24] is a representative solution that adopts a single, unified network for object detection. It is composed of two modules, i.e., a deep fully convolutional network that proposes regions, and a Fast R-CNN detector [25] that uses the proposed regions for object detection. Faster R-CNN also brings a Region Proposal Network (RPN) for efficient and accurate region proposal generation. Using the popular terminology of neural networks with "attention" [26] mechanisms,

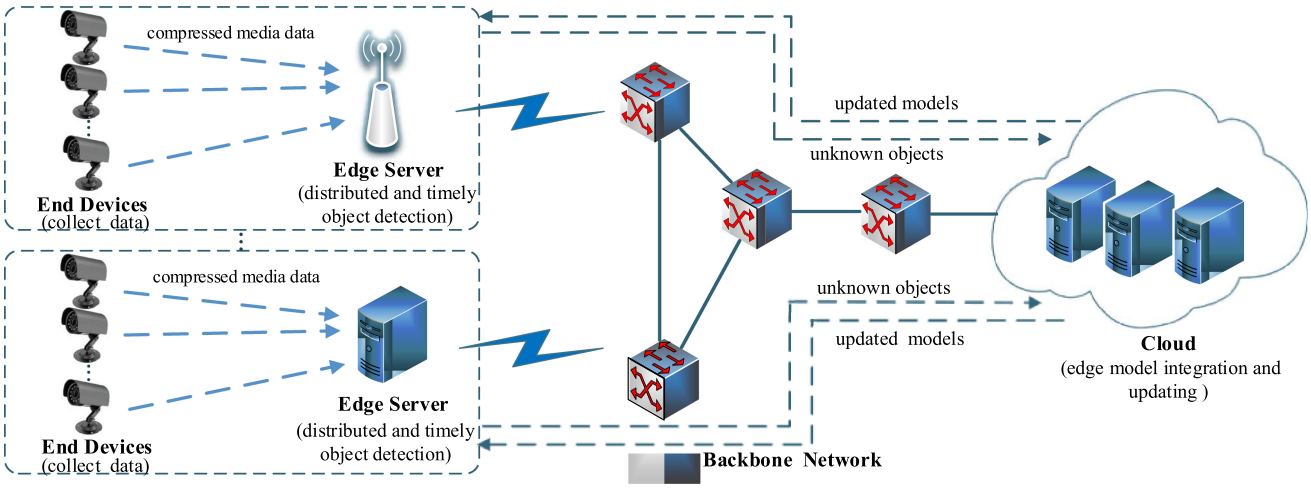


Fig. 1. Framework of the proposed edge computing based object detection system.

the RPN module can tell the Fast R-CNN detector where to look. After that, some similar models like Single Shot MultiBox Detector (SSD) [27], You Only Look Once (YOLO) [28] and YOLOv2 [29], have been proposed successively. A shortcoming of such kind of methods is that using a CNN-based model with one training for permanent use cannot adapt to dynamic environment, since there is always concept drift [30] in real-world surveillance applications. It means that the statistical properties of the targets, which the deployed model is trying to predict, may change over time in an unpredictable way. Thus, how to dynamically update the object detection models for accurately detecting objects during environment and target objects changing should be devoted more research efforts.

C. Edge Computing for Image Processing

Since edge computing has emerged as an important complement for cloud computing to support real-time and context-aware computing tasks [1], several research works have been proposed to leverage edge computing for real-time image processing applications. In [31], the authors develop a face recognition application, where a smartphone's photos are proceeded by the edge server instead of the cloud, to achieve reduced response time. S. Teerapittayanon *et al.* [32] propose distributed deep neural networks (DDNNs) over a hierarchical architecture, consisting of the cloud, edge and end devices. They extract image features on end devices with a lightweight neural network called binary neural network (BNN) [33] and make classification on edge servers. However, these research works mainly aim at recognition or classification on the whole image. For object detection, it is hard to be implemented by only extracting features on end devices without transmitting images to edge servers. The main reason is that object detection is still a relative computation-heavy work, where end devices lack sufficient computation capability to extract enough information. Thus, it is crucial to design a dedicated edge computing architecture for object detection in order to make full use of the computing ability of both the end devices and the edge servers.

III. Framework

In this section, we briefly introduce the framework of edge computing based object detection system, which consists of three parts, including end devices, edge servers, and the cloud. Fig. 1 shows the proposed framework of the edge computing based object detection system. Each of the three parts is introduced as follows.

A. End Devices

The end devices consist of the widely deployed monitoring devices for collecting media data in real time. Since the computing capability of end devices is usually insufficient for complex computation, it is not compatible to perform object detection directly on end device. Meanwhile, as these devices are connected to the edge server by unstable wireless networks with limited bandwidth, the end device layer is responsible for compressing the real-time captured pictures before transmitting them to edge servers. We deploy a lightweight region of interest (RoI) based image compression method on end devices. The RoI is found with a data driven method which is improved from binarized normed gradients (BING [13]). The RoI output by this method could most probably cover the latent objects. JPEG2000 [8] is utilized for the RoI compression. RoI compression means the background of the image is highly compressed and the accuracy of the RoI is kept. We name this approach as BING based RoI Extraction (BRE).

B. Edge Servers

The edge servers are the local servers and base stations which are distributed at the edge of the Internet. These servers conduct object detection algorithms on the compressed images from end devices to provide real-time detection services. Most of the captured images are processed on the edge servers without uploaded to the cloud through the backbone network [34]. Compared to the cloud, edge servers are usually small in scale and have limited computing power. The deployed algorithms should be relatively lightweight while meet the accuracy of the application scenario.

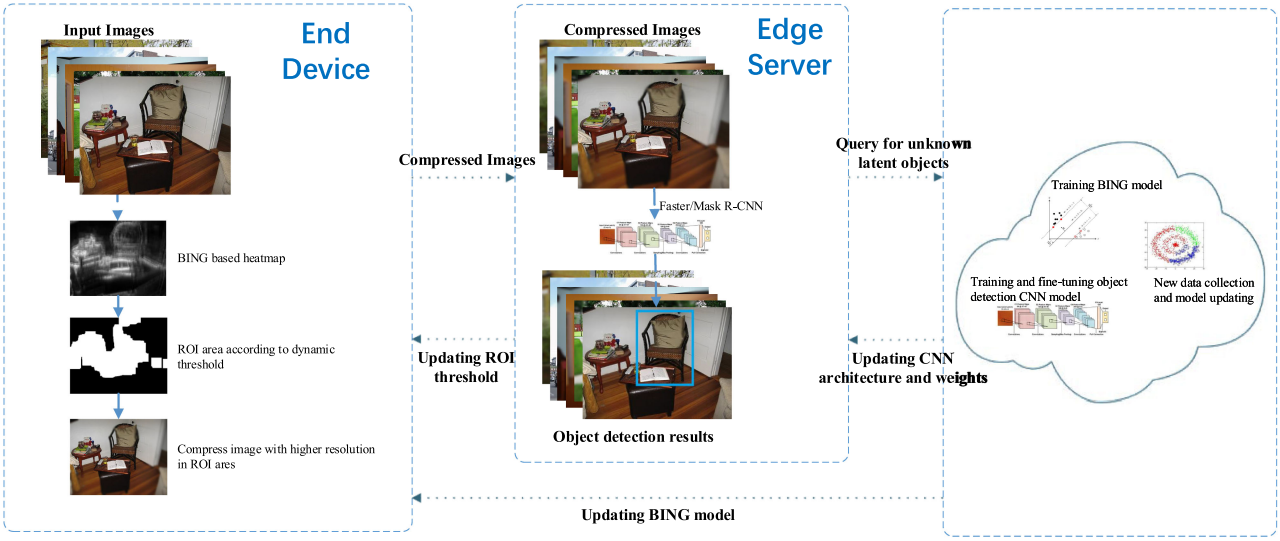


Fig. 2. Implementation overview.

Moreover, each edge server is only responsible for a certain number of adjacent end devices, which means that the edge servers work in a distributed manner. We deploy state-of-the-art object detection methods on edge server. Recently, CNN-based methods, such as Faster R-CNN [35], Mask R-CNN [36] and YOLO [37], are the dominant solutions with prominent performance for object detection. Since Faster R-CNN is the most famous architecture in recent years [38], we use it in our implementation and experiments. The deployed object detection algorithm can be updated to any newly proposed CNN models according to the application requirements or specific scenarios.

C. Cloud

The cloud is to collect data from the edge servers and incorporate global knowledge to update the model of the edge servers and end devices. Since the local model of the edge server is limited by the initial training samples, the accuracy may degrade over time. The cloud can collect labelled data from different edge servers and build an integrated model for object detection. When an edge server detects a potentially unknown object, the cloud can assist in detecting unknown objects. Furthermore, The cloud can trigger the model update operation under certain conditions, such as the detection request frequency of the same object, and update the edge server with its global knowledge. The RoI based image compression model, i.e., BRE model, is trained in the cloud. We also design a model monitoring and updating algorithm in the cloud for the integration of different object detection models and the model updating for end devices and edge servers.

IV. Edge Computing based Object Detection System

In this section, we introduce the details of our design, including the RoI based image compression method and the strategies for the interactions among end devices, edge servers and the cloud. Fig. 2 shows the implementation overview of the proposed framework and the interactions among different layers.

A. RoI-Based Image Compression Method for Distributed Object Detection

When designing an edge computing based artificial intelligent system, there are usually two kinds of data transfer strategies between the end and the edge. One is to transfer the raw data [31], while the other is to transfer extracted features (e.g., feature maps in deep learning model) [32]. For example, for the common classification tasks, we can deploy simplified network models on the lightweight end devices to extract compact data features [33] and transmit them to edge servers. In this way, the amount of the transmitted data can be significantly reduced. However, this strategy cannot work well for object detection tasks. Generally, object detection models need full-size feature maps to evaluate proposals throughout the image, and the sizes of the feature maps among the network layers may be even larger than the original raw image. Moreover, it is still an open challenge to design an object detection model for lightweight end devices for extracting feature maps. So we use the compressed raw image instead of feature maps for object detection at the edge side.

To sufficiently compress the captured images for supporting wireless transmission while providing acceptable detection accuracy, an intuitive idea is to keep the color and texture information of latent objects and significantly compress the residual background areas. RoI coding is a typical image compression strategy that JPEG2000 already provides a standard. The main challenges lies on how to deploy RoI extraction on end devices for efficient object detection. The first challenge is how to extract appropriate RoIs to effectively support object detection after image compression. Most RoI extraction methods are based on top-down models [10], which mainly consider the visual attention of people. Differently, we need to develop a bottom-up (goal-driven) RoI extraction method that can extract RoIs

covering latent objects with high recall. Second, existing methods are generally designed for computers with high computing performance, which are difficult to be deployed on lightweight end devices. We need to simplify the RoI extraction model to make it applicable for lightweight end devices.

We find that binarized normed gradients (BING) model [13] could be a suitable underlying algorithm to design an RoI extraction method. BING is an object proposal generation algorithm which provides object candidates for object detection. With a fast speed at 300 fps to find latent object bounding-boxes on a lightweight terminal, it is quite suitable to be deployed in end devices. However, the raw output of BING is a large number of object proposals that nearly cover the whole image. It makes the image difficult to be compressed without losing the potential proposals. We propose an RoI Extraction algorithm based on BING, named BRE, for the RoI extraction task. In the proposed BRE method, we utilize the generated proposals to create a probability heat map, which is further used for controlling the sizes of RoI areas based on an adaptive threshold. Only the pixels with a heat score higher than the threshold will be included in the RoI areas. Such that, we can accurately identify more compact RoI areas with potential objects for image compression. In the following we introduce the proposed BRE method in detail.

The image gradient map is firstly calculated to be seen as image features. Then, we resize the gradient map to a series of predefined window sizes. Each 8×8 area of the resized gradient maps is defined as a 64D normed gradient (NG) feature. Since the gradient maps are resized to different sizes from the original gradient map, 8×8 area in each resized gradient map denotes a different shape of bounding box in the original image.

With a two stage model training that utilizes binary approximation for acceleration [13], [39], we are able to get the score maps corresponding to each gradient map efficiently. Each pixel of the score maps indicates the object probability score of its left top 8×8 region. The object probability score denotes if the corresponding box in the original image contains an object. We reshape these feature maps to the size of the original image with bilinear interpolation and integrate them to a single probability heat map.

$$o_i' = ResizeToOriginal(o_i, i), \tag{1}$$

where o_i is object score map with window size i, and o_i' is the corresponding reshaped object score map with the shape as the original input image. Then, the heatmap can be calculated by

$$heatmap = \sum_{i \in \boldsymbol{S}} o_i', \tag{2}$$

where $\boldsymbol{S}$ denotes the set of all the predefined feature map sizes.

According to the created heatmap, an adaptive threshold is utilized to control the size of the RoI area. The pixels with heat score higher than the threshold could be included in the RoI area. Then, we have

$$roimap(x, y) = \begin{cases} 1 & heatmap(x, y) > th \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where (x, y) is the position of a window. A threshold th is calculated with a binary search algorithm to fit an RoI ratio rt. rt is manually defined at the beginning, but will be further fine-tuned by the interactions of edge server and end devices for the best performance of object detection. With a binary search for th, rt will finally satisfy

$$rt = \frac{\sum_{(x,y) \in \boldsymbol{C}} roimap(x, y)}{W \times H}, \tag{4}$$

where (x, y) is the position of the window and $\boldsymbol{C}$ is the set of the coordinates, W and H means the width and height of the original image. With the post-processing of erode and dilate, we can obtain a well-defined RoI map which covers all the latent objects.

In traditional image compression protocols, there are mainly three steps: data transformation (e.g., cosine transform and wavelet transform), quantization, and coding. RoI coding only changes the last two steps. After wavelet transformation, quantization and coding with higher resolution are conducted on the regions of interests while the other parts are quantized and coded with lower resolution.

JPEG 2000 standard [8] offers a well defined RoI coding method based on Maxshift [40]. The Maxshift method allows the RoI to be irregularly shaped. The image is firstly processed with multistage wavelet transform. Then, the coefficients are processed with a quantization phase and a coding phase. With Maxshift, the RoI and background are processed with different parameters during the quantization and coding phase. Finally, we use the RoI map made by BRE as the RoI mask of the Maxshift based RoI coding method in JPEG 2000 to compress the images.

B. Interaction Between End Devices and Edge Servers

With the tailored BRE based RoI compression method, we can significantly compress the images without losing the important information for object detection. However, a larger compression ratio may degrade the accuracy of object detection. Since different application scenarios may have different characteristics (such as the following aspects), the compression ratio is nonlinearly related to the detection accuracy. This motivates us to design interaction schemes that allow the end device to adjust the compression parameters to compress the picture as much as possible while keeping the edge server detection accuracy within the controllable range.

1) Different Ratio: The proportion of the target occupying the image may be different in different scenes. For example, when the application is monitoring pedestrians at intersections, the target (i.e., pedestrian) is relatively small in the image. The total area of pedestrian may only take less than 10% of the whole image. In the scene of identity ID recognition, the contour of the person is the RoI, which usually occupies 80% to 90% of the entire image. We can set a higher compression ratio for the former to perform high-intensity compression on non-RoI regions. But for the latter, we can only set a smaller compression ratio because there is a larger proportion of RoI regions.

2) Different Degrees of Significance: In some application scenarios, the objects may be very outstanding in color or texture, while some others are not. For example, yellow tigers in a green grassland are conspicuous, but a green parrot is not. With

outstanding appearance, the objects could be highly compressed and will not seriously degrade the detection accuracy.

3) Different Model Quality: If the model generalization performance in the edge server is very good and there are no subsequent tasks such as recognition, we can give up more image details for a higher compression ratio. Conversely, if the annotation data of the application scene is difficult to obtain and the model has limited precision, it is necessary to retain more image details and result in a lower compression ratio.

Since end devices are responsible for compressing the image while the edge server is responsible for object detection, we can dynamically adjust the compression ratio of end devices during communicating with the edge server. In this way, we could compress the data as much as possible while ensuring the detection accuracy of the edge server. In BRE, we control the size of the RoI region with an adaptive threshold. The compression ratio is dynamically modified by the specific scene, the type of objects, and the robustness of the edge server model. To avoid ambiguity, we define the compression result ratio cr as:

$$cr = \frac{S_{after}}{S_{before}} \tag{5}$$

where S_{after} and S_{before} are the size of data after and before compression respectively. Smaller cr means higher compression ratio.

Specifically, after the trained model is deployed on the edge server, cr would be automatically reduced until the detection accuracy achieves a predefined level.

$$cr = \arg\min_{cr} \frac{mAP_{cr}}{mAP_{100\%}}, \frac{mAP_{cr}}{mAP_{100\%}} > \sigma, \tag{6}$$

where AP means the average precision (AP). This is the most important indicator of object detection task. mAP_{cr} means the AP under this cr, $mAP_{100\%}$ means the AP without compressing the image. Our goal is to find the lowest cr to make the mAP_{cr} no less than $mAP_{100\%} \times \sigma$. The cr would be continually modified with a descending Δ till the target is satisfied.

To dynamically modify the cr, a dynamic AP/CR chart would be made by the edge server. The edge server periodically requests an image without compression from the end device. This image would be compressed with a series of sampling cr value. Since there's no ground truth of a new coming image, the detection result of the non-compressed image is seen as an approximation ground truth. The AP is calculated by:

$$AP = \sum_{k=1}^{N} P(k)\Delta r(k), \tag{7}$$

where k is the ID of each image, N is the number of images, $P(k)$ is the precision of image k, and $\Delta r(k)$ is the change in recall from image $k-1$ to k.

So we keep N non-compressed images $\{I_{t-N}, I_{t-N+1}, ..., I_t\}$ in the edge server for the dynamic modification of cr. Here t is time stamp, I_t is the non-compressed image coming in t. When the edge receives image I_t, the oldest image I_{t-N-1} would be discarded or ignored and a new AP/CR chart would be updated for the cr updating.

C. Interaction Between Cloud and End Devices

Since end devices usually have limited computing resources, complex algorithms like training models are not applicable. As a result, the BRE model training is completed and deployed offline. As the cloud continues to get new data and update the model, the judgement on the RoI area of end devices may not be able to adapt to the constantly rich scene and target set. Therefore, the cloud needs to use the continuously updated global knowledge to incrementally retrain the BRE model and update the end devices online.

To update the BRE model deployed on end devices online, we initialize with the parameters of the previous model and conduct the incremental training. The model is trained on the cloud, and distributed to end devices. Due to the large number of end devices, network congestion can be prevented by asynchronous update.

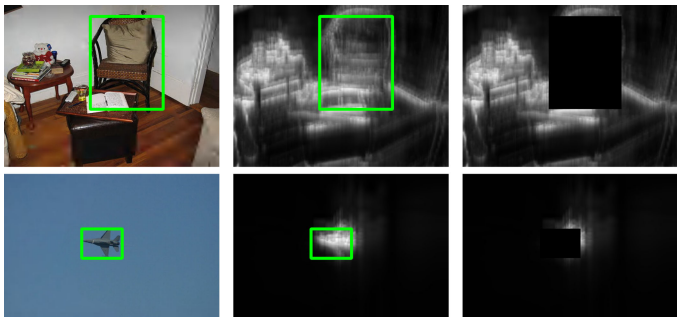
D. Interaction Between Edge Servers and Cloud

In the proposed framework, each edge server independently deploys its initial model according to its own application scenario, while the cloud has global information. Since edge servers are usually not as powerful and rich in computing resources as the cloud, it is impossible to directly deploy a huge global model like the cloud to detect very wide variety of objects. Moreover, each edge server only serves adjacent end devices, while the number of types of objects that need to be detected in a particular scenario is quite limited. Therefore, it is sufficient to maintain high detection accuracy with a local inference model.

The model deployed on edge servers needs to be updated periodically for two reasons. The first one is that there is always conceptual drift [30] during system running. Concept drift means that the statistical properties learned by the predictive model change in an uncertain manner over time. As a result, models that have been well-behaved may degrade over time. The second one is that on the edge server, the model could only recognize a small set of objects for specific tasks. When the scene changes and one or more new kind of objects appears stably and chronically, the model needs to be updated with the global knowledge from the cloud.

To handle the concept drift and the emergence of new targets in long-term deployments, we design an automatic triggering update mechanism to update the edge server model through the cloud. The updating procedure contains three parts: latent unknown object discovery, updating trigger and updating scheme.

Because there is no prior knowledge to identify a new kind of object, it is difficult to decide whether to send an image to cloud for further detection. As we can get a well defined heatmap from BRE model with global knowledge, latent objects could be reflected on the scores of the heatmap. After object detection on the edge server, objects with high confidence are found as bounding boxes. If we remove the boxes in the corresponding heatmap matrix, the remaining residual heatmap would reflect the probability whether there are other latent objects. We define a "heat threshold" ht. If the average score of the residual heatmap is higher than ht, the image would be considered to contain latent



(a) Origin Detection (b) Heatmap (c) Residual Heatmap

Fig. 3. Residual heatmap example. (a) Original detection results. (b) Heatmap with detected box. (c) Residual heatmap.

new objects and would be sent to cloud for further detection. The residual heat score is defined as:

$$rhs = \frac{\sum_{x,y}^{C} heatmap'(x,y)}{W \times H}, \tag{8}$$

where rhs is the residual heat score, (x, y) is position on the heatmap, C is all the coordinates, $heatmap'$ is the residual heatmap with the detected boxes removed, W and H is the width and height of the heatmap. An example of residual heatmap is shown in Fig. 3. The first line is the detection result of a chair. The small table is mis-detected because it is not in the dataset. After the detected box removed from the heatmap, the residual heat score is high. The original image in the second line only contains a plane. The residual heat score is relatively low.

1) Updating Trigger: The cloud server can detect objects with the global knowledge. If there are objects that are not included in the edge server model in the uploaded image, the cloud would simply send the detection results to the edge server as a helper. When the same kind of object appears rapidly in a limited time window, the cloud would trigger an updating. We define the updating frequency as: a kind of object that is not included in the edge server model appears more than a times in b hours.

2) Updating Scheme: To update the model deployed on the edge, a naive solution is to directly train a new model with all the related labeled data. However, training new model for each edge server would increase the computing pressure of the cloud. In order to alleviate the burden of the cloud, we use an improved fine-tuning scheme. In Faster R-CNN, the detection output is usually composed of two parts. One is the softmax layer for classification, and the other is full connection layer and box regression for object bounding box coordinates. In order to enable the model to detect new kinds of objects, we first adjust the structure of the output layer to make it correspond to the number of object kinds. Since the old model is already well trained for recorded objects, we only use a subset of the labeled data of the recorded objects and all the data of the new object to fine-tune this modified network. Without fully training a new model, the training time could be reduced. The new model would be sent from cloud to edge server for updating after structure modification and fine-tuning.

TABLE I
SETTINGS OF THE EXPERIMENTAL PLATFORM

	Device	Hardware	Software
End device	Xiaomi 6	Snapdrago 835 (2.45GHz), 6GB, Andreno 540	BRE algorithm
Edge server	PC	i7 6700 (4GHz), 32GB, GTX 980Ti	Faster R-CNN
Cloud server	work station	E5-2683 V3 (2.00GHz), 128GB, GTX TitanXp*4	Faster R-CNN and BRE training

TABLE II
COMPARISON OF THE TWO STRATEGIES ON THE EXECUTION TIME OF THE END DEVICE AND THE EDGE SERVER, AS WELL AS THE COMPRESSED DATA SIZE

	BRE-based Compression	Feature Map Compression
Time on End Device	0.17s	6.83s
Time on Edge Server	0.12s	0.12s
Data Transmission	0.066MB	30MB

V. PERFORMANCE EVALUATION

In this section, we evaluate the proposed edge computing based object detection system from different aspects. Table I shows our experimental platform. For the training and testing of BRE, Faster R-CNN and the evaluation of the interaction schemes, we use Pascal VOC2007 dataset [41].

A. Compression Strategy Evaluation

We compare our solution with the feature map compression strategy. We process 100 images from VOC2007 dataset with the two strategies separately under the same hardware environment. For the BRE method, we set $cr = 50\%$ since cr only affects the object detection accuracy while does not affect the execution time of end device. Object detection algorithms, such as Faster R-CNN, usually start with feature extraction networks, e.g., ResNet and VGGNet. To relief the computation burden of the end device, we use the first two convolutional layers of VGGNet to calculate the feature maps on the end device while the edge server is responsible for the computation of the rest layers. Since the feature maps contain important feature information of the input image and it is difficult to determine the importance of different features, we cannot conduct lossy compression on the feature maps as we do on raw images. Therefore, we compress the feature maps with *gzip* which is a common lossless file compression protocol.

As shown in Table II, our solution outperforms the feature map compression strategy in terms of the average execution time and the size of transmitted data. This is because current object detection models cannot use simplified networks such as BNN [33] to extract compact feature maps, meanwhile the computation is relatively heavy for lightweight end devices without powerful GPUs.

Although we can compress raw data into smaller feature vectors for classification tasks, object detection models require full-scale feature maps to analyze proposals in different regions. As a result, the size of the compressed feature maps is much larger

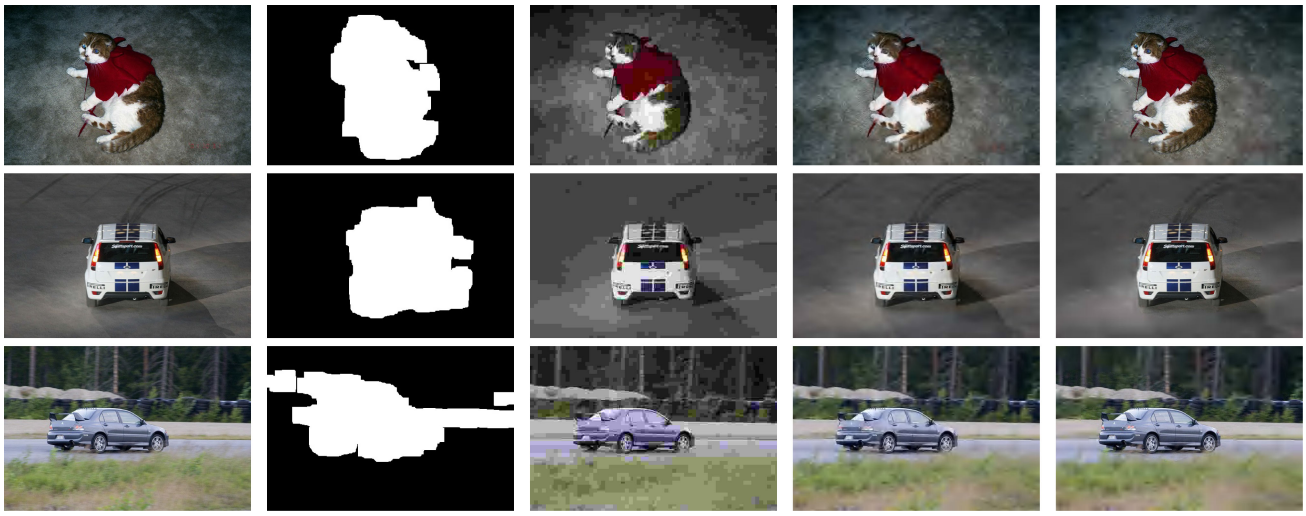


Fig. 4. Visual comparison with the classical compression methods under the same compression ratio. (a) Original images. (b) RoI Area by BRE. (c) Image compressed with JPEG. (d) Image compressed with non-RoI version of JPEG2000. (e) BRE based RoI compression.

than the size of the compressed raw image. For example, a 3-channel image with the size of $512 * 512$ can be seen as a feature matrix with the shape of $512 * 512 * 3$. However, the shape of the output feature map matrix by the first two convolutional layers of VGGNet is $512 * 512 * 64$. With more data to be compressed, the computation burden of the end device will be significantly increased. Furthermore, small changes in the network model of the edge server have few impacts on the execution time, since it has relatively stronger computing power.

B. RoI Compression and Detection

We use the traditional and our proposed BRE-based compression methods to compress the images separately. Fig. 4 first shows the RoI area achieved by BRE and then the compressed images under different image compression algorithms with the image compressed by about 95%. Benefit from the BRE based RoI extraction algorithm, the areas containing latent objects are treated differently in the compression algorithm. Under the same high compression ratio, the proposed method preserves more details of potential objects than JPEG and JPEG2000.

Since the proposed BRE algorithm can accurately identify the areas with latent objects, the proposed compression solution can keep more details of the latent objects for object detection than JPEG and JPEG 2000 under the same compression ratio.

As most object detection works do, we use mean Average Precision (mAP) [42] as the performance evaluation metric. Fig. 5 shows the correspondence between different cr and edge server object detection accuracy under the test data set used in our experiment. The cr ranges between the original image and the image with the non-RoI areas removed. We can see that the detection accuracy is still not bad with only about 5% reduction even at a cr of 20%. It indicates that our solution can effectively reduce the network transmission pressure of object detection

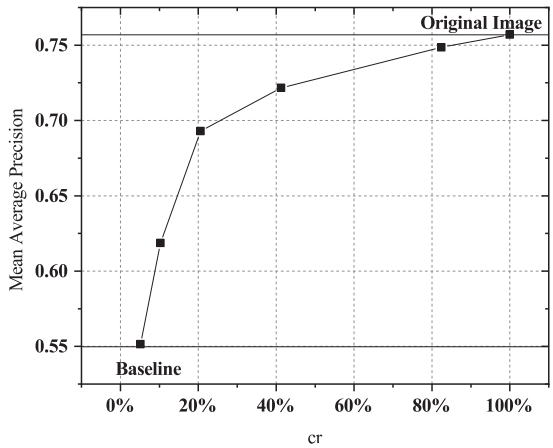


Fig. 5. Object Detection Accuracy v.s. cr.

tasks with a reasonable impact on the accuracy in surveillance applications. In addition, according to the sharp decrease of the detection accuracy of the near-baseline part in the figure, it can be known that the background information can help the edge server to perform object detection and cannot be removed simply.

C. Detection Results

As shown in Fig. 6, our BRE compression method could well support the detection task. It can be noticed that the JPEG2000 compression method maintains image quality better than JPEG and achieves similar object detection result to BRE. However, since there is no object-based region of interest extraction, the detected target region also loses image quality. This will be detrimental to potential follow-up tasks such as license plate recognition. In contrast, images compressed based on the BRE method have lower overall quality, but the detected object area retains better details.

(a) Ground Truth (b) Original Version (c) JPEG (d) JPEG2000 (e) RoI Based

Fig. 6. Detection results. (a) Ground truth. (b) Detection results of Faster R-CNN on uncompressed images. (c) Detection results on images compressed with JPEG. (d) Detection results on images compressed with non-RoI version of JPEG2000. (e) Detection results on images compressed with BRE based RoI compression.

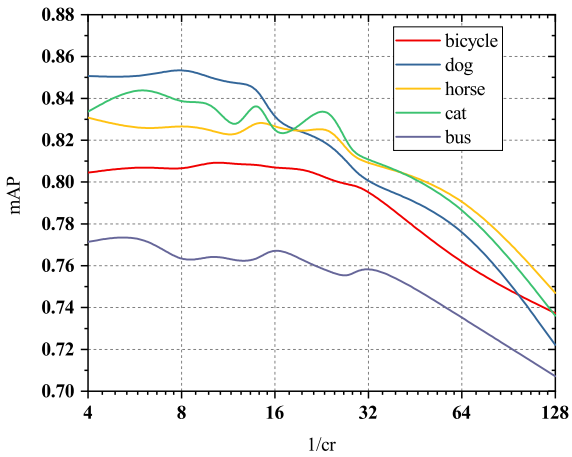


Fig. 7. Dynamic modification cr on different classes.

D. Dynamic Modification of Compression Ratio

In the interaction between the end devices and the edge server introduced in IV-B, we dynamically adjust the compression result ratio cr to reduce the data transmission as much as possible while maintaining the detection accuracy. The scheme of adjustment is based on the AP/CR chart drawn by asynchronously obtaining uncompressed data.

As shown in Fig. 7, different objects/scenes have different changing curves on the relationship between cr and mAP. With dynamic modification of cr between edge servers and end devices, the detection task could be optimized. Because the amount of data collected by the edge server for each target is limited in the experiment, the detection accuracy under different compression result ratios has some fluctuations. But the figure still shows the change trend of the detection precision along with the cr, which helps the end device adjust the cr. For example, The detection accuracy of the "dog" begins to decrease sharply around 16 times compression, while the detection accuracy of the "bus" is relatively stable before 32 times. The accuracy of the AR/CR change would continue to increase as new data is continuously accumulated at the edge server.

E. Residual Heatmap and New Kind of Objects

In the interaction between the cloud and the edge, we need the edge to perceive the potential new emerging objects and report them to the cloud. Without global knowledge, the edge server cannot know exactly if new objects are present. We use the residual heatmap strategy to determine whether to request cloud help by cumulative probability. It only interacts with the cloud when the residual heat score is significantly increased, thereby saving the bandwidth of the backbone network. As shown in Fig. 8, when new kind of objects appears, the residual heat score rhs roughly rises. The trend of rhs could reflect the appearance of new kinds of objects. Uploading images to the cloud when the rhs is higher than a threshold rather than every time could significantly decrease the pressure of the cloud.

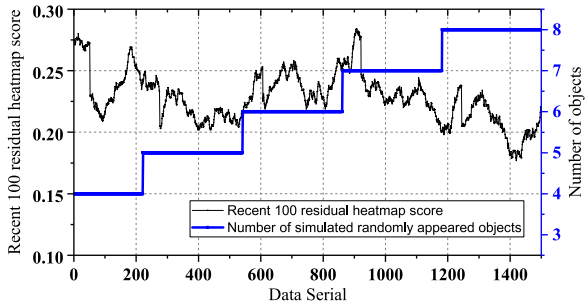


Fig. 8. Relation between residual heat score and the appearance of new kind of objects.

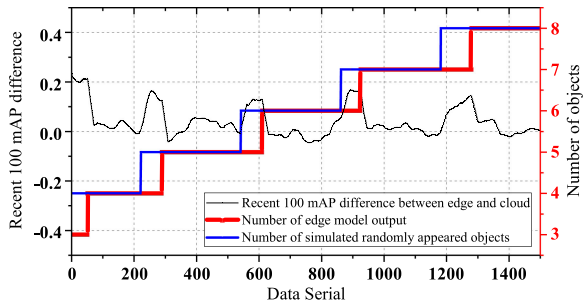


Fig. 9. Interaction between the edge server and the cloud, the appearance of simulated new objects and the performance variation of the edge server.

TABLE III
EFFECT OF FINE-TUNING METHOD TO UPDATE EDGE MODEL

	Old model	20%	30%	40%	100%
Bird	0.716	0.694	0.701	0.701	0.716
Cat	0.813	0.811	0.821	0.828	0.852
Dog	0.850	0.800	0.836	0.822	0.865
Horse	0.788	0.833	0.851	0.863	0.835
Chair (new)	0	0.492	0.495	0.507	0.565
mAP	0.791	0.726	0.741	0.744	0.767
Training Time	-	32min	40min	55min	120min

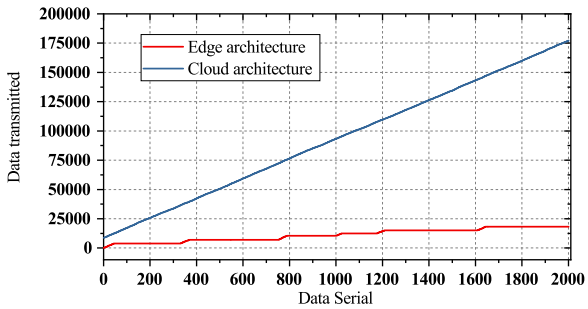


Fig. 10. Comparison of bandwidth occupancy between edge computing architecture and cloud computing architecture.

F. Updating of Edge Server Model

With the interaction of the cloud with the edge server, the model of the edge server can be automatically updated when new objects appear or the accuracy drops due to concept drift. This experiment tests the automatic update effect of the model by artificially providing new objects to the edge server. Fig. 9 illustrates the interaction between the edge server and the cloud for the update of the edge model. The blue line means the number of objects frequently appear in the monitoring areas of the edge server, while the red line indicates the number of objects that the edge server can recognize. In order to view the effect of the edge server model update, we also calculated the detection result of each image in the cloud. The difference of detection accuracy of recent 100 images in the data serial between the edge server and the cloud is also reflected in the figure. When a new kind of objects appears, the gap of mAP between the edge server and the cloud increases. After a short delay, i.e., the edge server frequently sends images with latent new objects to the cloud, the cloud triggers the behavior of updating the model of the edge server. After that, the gap of mAP between the edge server and the cloud finally returns to normal level. This process shows that the interaction scheme between the edge server and the cloud can well solve the problem of concept drift in edge servers effectively and timely, thus improving the stability of this online system.

G. Model Updating Scheme for Edge Server

If there are a large number of edge servers, frequent model updates can put tremendous pressure on cloud servers. In order to alleviate the computational pressure of the cloud to continuously train new models for the edge server update, we use a modifying and fine-tuning strategy which is introduced in Section IV-D. In this experiment, we test the validity of the fine-tuning strategy with different data ratios. As shown in Table III, without any of the training data corresponding to the previous model to fine-tune the modified network would still achieve a good accuracy. With 40% of those data, the result is already quite close to the retrained result. It can be seen that there exists a trade-off between the retraining time and detection accuracy when we choose different portions of original data in model update. However, the speed of model retraining can be significantly accelerated by slightly sacrificing the detection accuracy.

H. Bandwidth Saving Compared with Cloud Architecture

Our solution could save most of the backbone network traffic. We show the comparison result of accumulative data transmitted by the backbone network in a simulation. The images are all compressed under the same scheme.

As shown in Fig. 10, our solution could save almost 90% bandwidth of the backbone network when compared with the cloud computing architecture. Traditional cloud computing architecture needs to transmit all the images to the cloud. As the number of end devices continues to increase, more edge servers can be deployed to meet service needs. The edge server only needs to send a small number of images with uncertain or potential new objects to the cloud through the backbone network. Edge systems in the same deployment area only need to be updated once by the cloud when specific new kind of objects occurs frequently. In this way, a large number of service requests are solved in a regional network with low stability and bandwidth requirements, thereby greatly saving the bandwidth of the backbone network.

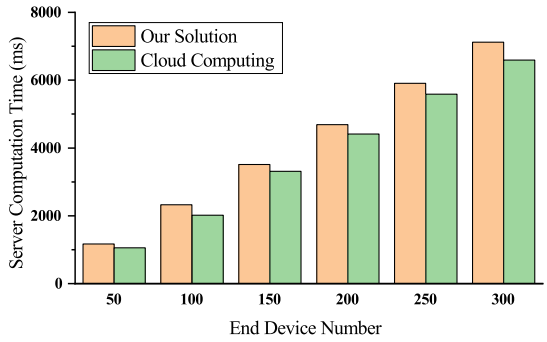


Fig. 11. Computation delay comparison under different number of end devices.

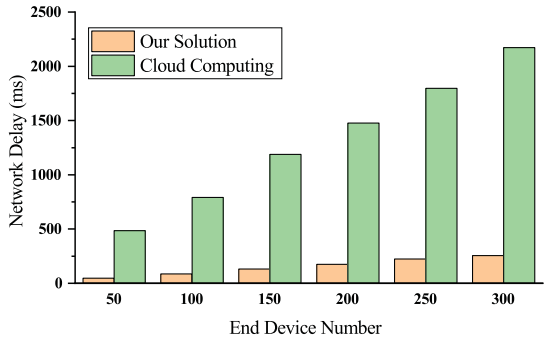


Fig. 12. Network delay comparison under different number of end devices.

I. Processing Delay Testing

In this subsection, we compare the computation delay and the network delay of the proposed solution and cloud-based solution under different network scales. In our experiments, each camera generates an object detection request every 5 seconds. We deploy 5 edge servers for processing the captured images of cameras. The wireless data transmission rate between the cameras and edge server is set as 100 Mbits/s. Meanwhile, the delay between the cameras and the cloud is measured by sending data to the Alibaba Cloud (a public cloud platform in China).

Fig. 11 shows the comparison result of the computation delay by the server. Although the performance of a single edge server is not as strong as that of the cloud server, by properly distributing the edge server nodes, the computation time consuming of our solution to provide object detection services is similar to cloud-based solution. It should be noted that, in practical applications, Faster R-CNN can be replaced with a faster SSD300 [27] or YOLOv2[29], which could promote the object detection speed 3˜5 times as needed. This can greatly increase the data processing speed of the server. Fig. 12 shows the comparison result of the network delay. It can be seen that the proposed solution can greatly reduce the network delay of object detection service. Moreover, as the network scale increases, the improvement is getting larger.

Moreover, we also compare the processing delay of different object detection schemes under limited network bandwidth. The simulation results are shown in Fig. 13, where the network bandwidth is set to 10 Mbits/s and 300 end devices are connected to 5 edge servers. It can be seen from the figure that our solution outperforms the other methods in terms of the total processing delay. Compared to cloud computing based methods, edge servers can timely detect the objects without the long transmission delay from end device to the cloud via core network. Compared to the edge computing based solution without BRE, our solution can significantly reduce the data transmission delay between end devices and edge servers, although there exists some additional time cost on the end devices for image compression.

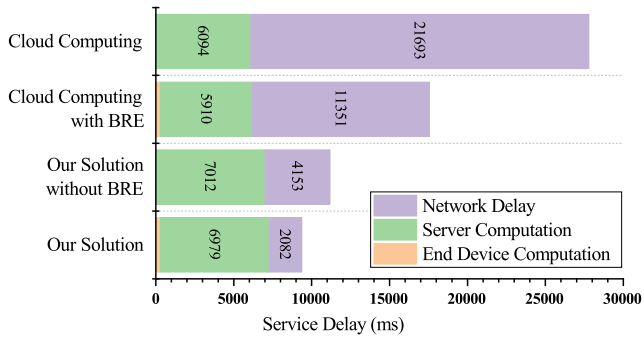


Fig. 13. Object detection delay comparison under different methods.

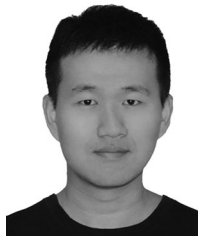
VI. Conclusion

In this paper, we propose an edge computing based object detection system and its implementation details for surveillance applications. An RoI based image compression method is designed to compress the images for wireless transmission without significantly reducing the object detection accuracy. We have also presented how end devices, edge servers and the cloud interact to dynamically optimize the model and system online. A large number of simulations on the dataset of Pascal VOC2007 have demonstrated that the proposed architecture can achieve reduced response delay without sacrificing detection accuracy when compared to the traditional cloud-based object detection method. In our future work, we will study the joint learning method which allows the edge computing system to be trained online and learns new knowledge in a semi-supervised or unsupervised way.

References

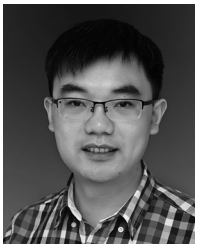
[1] Y. Zhang *et al.*, "A survey on emerging computing paradigms for big data," *Chin. J. Electron.*, vol. 26, no. 1, pp. 1–12, 2017.
[2] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, 2017.
[3] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Netw.*, vol. 32, no. 6, pp. 137–143, Nov./Dec. 2018.
[4] P. Yang *et al.*, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.
[5] L. Sun, H. Pang, and L. Gao, "Joint sponsor scheduling in cellular and edge caching networks for mobile video delivery," *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3414–3427, Dec. 2018.
[6] Z. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2432–2455, Oct./Dec. 2017, doi: 10.1109/COMST.2017.2707140.

[7] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.
[8] M. Rabbani, "JPEG2000: Image compression fundamentals, standards and practice," *J. Electron. Imag.*, vol. 11, no. 2, p. 286, 2002.
[9] A. Borji and L. Itti, "State-of-the-art in visual attention modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 185–207, Jan. 2013.
[10] H.-C. Nothdurft, "Salience of feature contrast," in *Proc. Neurobiol. Attention*, 2005, pp. 233–239. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123757319500422
[11] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2189–2202, 2012.
[12] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool, "Online video seeds for temporal window objectness," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 377–384.
[13] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *IEEE Conf. Comput. Vis Pattern Recognit.*, 2014, pp. 3286–3293.
[14] C. Lu, S. Liu, J. Jia, and C.-K. Tang, "Contour box: Rejecting object proposals without explicit closed contours," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2021–2029.
[15] I. Endres and D. Hoiem, "Category-independent object proposals with diverse ranking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 222–234, Feb. 2014.
[16] P. Krähenbühl and V. Koltun, "Geodesic object proposals," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 725–739.
[17] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 128–140, Jan. 2017.
[18] X. Chen *et al.*, "Monocular 3d object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 2147–2156.
[19] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Craft objects from images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 6043–6051.
[20] Z. Jie *et al.*, "Scale-aware pixelwise object proposal networks," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4525–4539, Oct. 2016.
[21] M. Cho, S. Kwak, C. Schmid, and J. Ponce, "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1201–1210.
[22] Z. Fang, Z. Cao, Y. Xiao, L. Zhu, and J. Yuan, "Adobe boxes: Locating object proposals using object adobes," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4116–4128, Sep. 2016.
[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[25] R. Girshick, "Fast R-CNN," in *IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[26] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.,* 2015, pp. 577–585.
[27] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 779–788.
[29] J. Redmon and A. Farhadi, "YoLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6517–6525.
[30] A. Tsymbal, "The problem of concept drift: Definitions and related work," Comput. Sci. Dept., Trinity College Dublin, Ireland, Tech. Rep. 2, vol. 106, no. 2, p. 58, 2004.
[31] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol.*, Nov. 2015, pp. 73–78.
[32] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 328–339.
[33] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[34] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.
[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. 28th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[36] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *IEEE Int. Conf. Comput. Vis.*, Oct. 2017.
[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[38] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *CoRR*, vol. abs/1809.02165, 2018. [Online]. Available: http://arxiv.org/abs/1809.02165
[39] S. Hare, A. Saffari, and P. H. Torr, "Efficient online structured output learning for keypoint-based object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1894–1901.
[40] P. G. Tahoces, J. R. Varela, M. J. Lado, and M. Souto, "Image compression: Maxshift ROI encoding options in JPEG2000," *Comput. Vis. Image Understanding*, vol. 109, no. 2, pp. 139–145, 2008.
[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," 2007. [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html
[42] M. Zhu, *Recall, Precision and Average Precision*, vol. 2. Waterloo, ON, Canada: Dept. Stat. Actuarial Sci., Univ. Waterloo, 2004, p. 30.

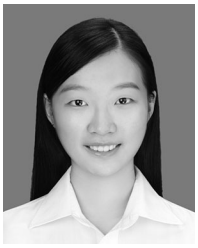


Yundi Guo received the B.S. degree in computer science from Central South University, Changsha, China, in 2012. He is currently working toward the Ph.D. degree in computer science at Central South University, China. His Research interests include deep learning and image processing.

Beiji Zou received the B.S. degree in computer software from Zhejiang University, Hangzhou, China, in 1982, the M.S. and Ph.D. degrees in computer science and technology from Tsinghua University, Beijing, China, in 1984, and Hunan University, Changsha, China, in 2001, respectively. He joined the School of Computer and Communication, Hunan University, Changsha, China, in 1984, where he became an Associate Professor in 1997, and a Professor in 2001. Since 1997, he has been the Vice Dean of Hunan University, Changsha, China. He is currently a Professor with the School of Computer Science and Engineering, Central South University, Changsha, China. Until now, he has authored/co-authored more than 100 papers in journals. His research interests include computer graphics, image processing, and virtual reality technology.



Ju Ren (S'13–M'16) received the B.Sc., M.Sc., Ph.D. degrees in computer science, from Central South University, Changsha, China, in 2009, 2012 and 2016, respectively. During 2013–2015, he was a visiting Ph.D. student with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada. He is currently a Professor with the School of Computer Science and Engineering, Central South University, China. His research interests include Internet-of-Things, wireless communication, network computing, and cloud computing. He is a co-recipient of the Best Paper Award of IEEE IoP'18 and the most popular paper award (2015–2018) of Chinese Journal of Electronics. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and *Peer-to-Peer Networking and Applications*, and a TPC member of many international conferences, including IEEE INFOCOM'19/18, Globecom'17, WCNC'17, WCSP'16, etc. He also served as a TPC Chair for IEEE BigDataSE'19, a poster Co-Chair for IEEE MASS'18, a track Co-Chair for IEEE ICCC'19, VTC17 Fall, and I-SPAN18, and an active reviewer for more than 20 international journals. He is a member of ACM.



Qingqing Liu received the B.Sc. degree in automation from Central South University, Changsha, China, in 2016, where she is currently working toward the M.Sc. degree in control science and engineering. Her research interests include machine learning, edge computing, and Internet-of-Things.

Deyu Zhang (S'14–M'17) received the B.Sc. degree in communication engineering from PLA Information Engineering University, China, in 2005, and the M.Sc. degree in communication engineering and the Ph.D. degree in computer science from Central South University, Changsha, China, in 2012 and 2016, respectively. He is currently an Assistant Professor with the School of Software and a Postdoctoral Fellow with the Transparent Computing Lab in the School of Information Science and Engineering, Central South University, China. From 2014 to 2016, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada. His research interests include stochastic resource allocation transparent computing, edge computing and Internet-of-Things. He is a member of CCF.

Yaoxue Zhang (M'17–SM'18) received the B.Sc. degree from Northwest Institute of Telecommunication Engineering, Xi'an, China, in 1982, and the Ph.D. degree in computer networking from Tohoku University, Sendai, Japan, in 1989. He is currently a Professor with the School of Computer Science and Engineering, Central South University, Changsha, China, and also a Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computer networking, operating systems, ubiquitous/pervasive computing, transparent computing, and big data. He has authored/co-authored more than 200 technical papers in international journals and conferences, as well as 9 monographs, and text-books. He is currently serving as the Editor-in-Chief of *Chinese Journal of Electronics*. He is a Fellow of the Chinese Academy of Engineering.