修 士 論 文 概 要 書

Master's Thesis Summary

Date of submission: 01/25/2021

専攻名 （専門分野） Department	Computer Science and Communications Engineering	氏 名 Name	Tao Wang	指 導 教 員 Advisor	渡辺 裕 印 Seal
研究指導名 Research guidance	Audiovisual Information Processing	学籍番号 Student ID number	CD 5119FG02-7		
研究題目 Title	A Lightweight Network Applying in The Edge-Cloud System for Real-Time Object Detection				

1. Introduction

The pervasiveness of "Internet-of-Things" in daily life has led to a recent surge in fog computing, encompassing a collaboration of cloud computing and edge intelligence. As a significant field of IoT, real-time detection and classification have a huge demand. Due to the gap of hardware performance between mobile devices and cloud servers and the increment of internet bandwidth and speed, the cooperative approach of edge devices and cloud servers would be an accessible orientation for real-time tasks.

Based on the thought of mobile edge computing and proposed edge and cloud cooperative approach, we create an edge-cloud system named ECNet [1]. And in the ECNet system, the network at edge side is the most vital component. This thesis is mainly designing a lightweight CNN network for edge side. By introducing the residual unit and other fine-tuning procedure, the network reach to an equilibrium which means apart from single network performance, the designed network can make the connection between edge side and cloud side smoothly. Besides, we set a series of offload determination in the system and make further analysis.

2. Related Technologies

2.1 YOLO

YOLO [2] is one-stage object detection algorithm. YOLO firstly divides the image into S×S grid units, and each grid predicts the bounding box and confidence score of the object, and then filters the redundant bounding box based on the probability distribution of the category. Compared with the R-CNN series of two-stage object detection algorithms, YOLO does not require intensive calculations before and after ROI warping, which makes the algorithm parameters smaller and make the processing of calculation faster. The network structure of YOLOv2 is similar to GoogleNet [3] which is contained with convolutional layers, max-pooling layers and fully connected layer. The convolutional layer and max-pooling layer in the network are used for feature extraction, and the usage of fully connected layer is to output category scores and the location of the target.

2.2 ResNet

In order to solve the optimization problem, a residual network is proposed. In the residual network, instead of letting the network directly fit the original mapping, it fits the residual mapping. The residual network adds some shortcut connections to the forward network so that these connections will skip some layers and pass the original data directly to the subsequent layers. The newly added shortcut connection will not increase the parameters and complexity of the model.

There are two connection grogram of residual block [4]. The one is the BasicBlock, which does not do the upgradation, so the output dimension of the residual structure is the same as the input dimension. Another one is Bottleneck, which is used to reduce the number of channel dimensions and increase speed.

3. Proposed Approach

3.1 ECNet

Based on the following guideline and the thought of BranchyNet [5], we design a new edge-cloud system aiming for objects detection task, named ECNet. As shown in Figure 1, the general framework of ECNet is mainly consists of edge operation and cloud operation. Feature maps extracted from edge-side will be transferred to cloud-side determined by offload controller. ECNet combine a light weight neural network on edge devices with a high-performance network on cloud servers and compress input sensor data, then offload it from edge side to cloud side depending on the metric for the result of edge side.

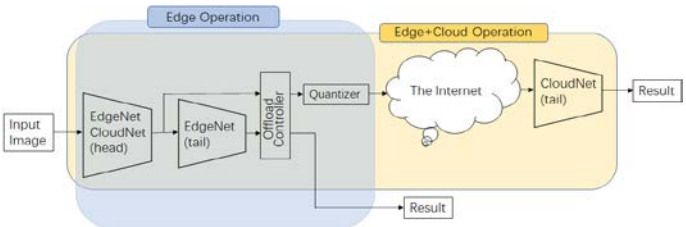

Figure 1. General framework of ECNet

3.2 Network at edge side

We intend to design front part of both network being same so that sensor data extracted from edge side can directly apply in cloud side. So we have to change the structure of the network at edge side.
However, this kind of distributed approach is challenging for a number of considerations, including:

- The structure of DarkNet53 is built on numerous residual block, and each of residual block contains successive 3 × 3 and 1 × 1 convolutional layers connected by one shortcut connection. This structure is aiming to solve the degradation problem on deep networks. Reconstructed front part of edge-side network should avoid dividing residual block to ensure its integrity.
- In detection tasks, YOLOv3 predicts boxes at 3 different scales. The cloud-side of ECNet extracts features from those scales using a similar concept as feature pyramid networks. It has good performance on small objects that are to be recognized by the detector. The location of offloading feature map to cloud-side should be before the layer where starting extracting features.
- To limit computing cost and processing time at edge-side, the depth of edge-side should not be too large.

Guided by aforementioned considerations, the structure of edge-cloud network is designed after several times of trials and simulations.

4. Experiments and results

We set the receptive field as indicator that help us abandon some low performance model quickly which make our work more efficiently.

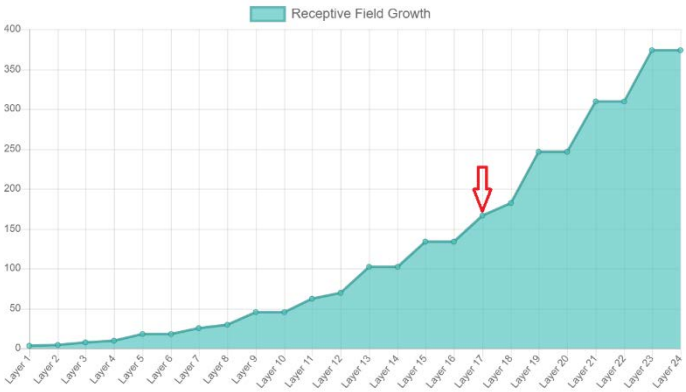

Figure 2. Receptive field growth of our network

Figure 2 shows the receptive field growth in our finally decided network structure. The blue part represents the amount of the receptive field after every layers and the red arrow points the layer where the system will do the operating of sensor data compression and transmission.

Table 1. Performance on the 10 classes dataset

	Rank-1(%)	*Rank-5(%)*	*Processing Time(s/frame)*
Our Network	68.5	81.8	0.013
Darknet19 (YOLO v2)	64.3	76.4	0.006
Darknet53 (YOLO v3)	81.2	98.2	0.023

By testing on our 10 classes dataset, we can find our designed network can improve the Top-1 accuracy about 4% comparing with the DarkNet19 as Table 1 shows. But the price is that we have to scarify some performance on processing time. By running on our platform, the FPS our network can reach is about 77, which is enable for our real applying scenario.

To add the algorithm of early-exit in the ECNet, we consider the confidence score to measure how confident the result we get at edge side network. By setting different threshold, the performance of network is significantly changing.

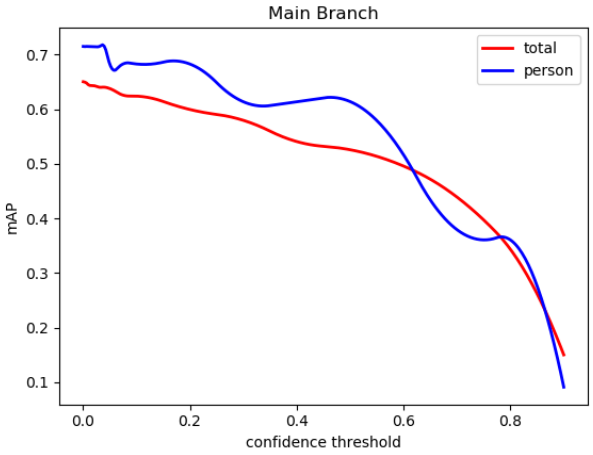

Figure 3. Accuracy of edge side network for varying confidence threshold

5. Conclusion

We proposed ECNet, which is an edge-cloud network system to make combination and connection between lightweight network on mobile devices and high-performance network on cloud servers, dealing with the balance between performance and time cost in the real-time detection tasks. Our main focus is on the designation of the lightweight network and offload algorithm and methods in system.

6. Reference

[1] L. Hu, T. Wang, H. Watanabe, S. Enomoto, X. Shi, A. Sakamoto and T. Eda: "ECNet: A Fast, Accurate, and Lightweight Edge-Cloud Network System Based on Cascading Structure," IEEE Global Conference on Consumer Electronics (GCCE) 2020, pp.259-262, Sep. 2020.
[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," CVPR, 2016.
[3].K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv: 1512. 03385, 2015.
[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.
[5] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," 2016 23rd ICPR. IEEE, 2016, pp. 2464–2469.

A Lightweight Network Applying in The Edge-Cloud System for Real-Time Object Detection

A Thesis Submitted to the Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: January 25th, 2021

Tao WANG

(5119FG02-7)

Advisor: Prof. Hiroshi Watanabe

Research guidance: Research on Audiovisual Information Processing

Acknowledgements

First of all, I am very grateful to my supervisor Prof. Hiroshi Watanabe for his enthusiastic guidance and the continuous support not only in the research aspect but also in my daily life and future career. Without Prof. Hiroshi Watanabe's encouragement and help, it will be impossible for me to smoothly complete master course and my research.

Then, I appreciate the joint research group members of NTT Software Innovation Center, Mr. Takeharu Eda, Mr. Shohei Enomoto, Ms. Xu Shi, and Mr. Akira Sakamoto for the professional research advices and inspirations in academic field they provided.

I would also want to express my gratitude to every members in Watanabe Lab who create great atmosphere in our lab that let me feel relax and comfortable.

Finally, I am very thankful that my family support me constantly in my life and energetic encouragements in my oversea years.

Abstract

The pervasiveness of “Internet-of-Things” in daily life has led to a recent surge in fog computing, encompassing a collaboration of cloud computing and edge intelligence. As a significant field of IoT, real-time detection and classification have a huge demand. Due to the gap of hardware performance between mobile devices and cloud servers and the increment of internet bandwidth and speed, combination of edge devices and cloud servers would be an accessible orientation for real-time tasks. We create an edge-cloud system named ECNet.

The network at edge side is the most vital component in ECNet, as the edge side part is usually deployed on embedded computing boards or smartphones. We should consider not only the balance between processing time and accuracy performance but also the versatility with the network at cloud side. Our works focus on addressing these issues so that we designed a lightweight CNN network for edge side. By introducing the residual unit and other fine-tuning procedure, the network reach to an equilibrium which means apart from single network performance, the designed network can make the connection between edge side and cloud side smoothly. Besides, we set a series of offload determination in the system and make further analysis.

Keywords: Real-time detection, edge-cloud system, ECNet, lightweight network, edge computing

List of contents

List of Figures

List of Tables

1. Introduction

1.1. Current demand of real-time detection

With the deepening of deep learning research, the application fields of deep learning have also been expanding in recent years, and it has become a series of powerful machine learning models. Object detection [1] is also a research hotspot in the scientific research field in recent years. The main work of this task is an important fusion in the field of artificial intelligence (AI), which realizes the intelligence of robots or other platforms through various things and tasks. Besides, object detection is related to computer vision and image processing, which processes the detection of semantic target instances of a certain type (such as people, buildings or cars) in digital images. And the detection fields mainly include face detection and pedestrian detection. Robots and AI programs select and recognize targets from input information such as video and camera images which can be used in multiple scenarios including component recognition, edge detection, and appearance analysis from different angles. In a word, object detection has huge needs in many fields of computer vision, including image retrieval and video surveillance to unmanned driving fields.

Detection task can separate into 2 fields, real-time and off-time. The performance on off-time person detection is pretty well and now is applying in real situation like the Takumi Eyes system [2] which is developed by NTT Company, applying for detection and person identification. Video recording will upload to cloud from the camera or transfer to the server and then start to detect by the trained network.

The pervasiveness of “Internet-of-Things” [3] in daily life has led to a recent surge in fog computing, encompassing a collaboration of cloud computing and edge intelligence. As a significant field of IoT, real-time detection and classification have a huge demand. The range of usage of real-time object recognition is including automatic driving, surveillance in public area, big data collection, etc.

In this thesis, we mainly propose investigating system-level solution for object recognition by combining edge and cloud network, which mainly focus on lightweight CNN network in the edge side.

1.2. Problem statement

With the development of data transmission, edge and cloud cooperative approach for object detection has been proposed [4]. Object recognition can be performed by many cloud vision API services using deep learning. In this case, images are provided to cloud on the Internet. On the other hand, object recognition at an edge becomes possible because of the improvement of computation power on edge devices. And new neural network architecture such as MobileNet [5], YOLO-tiny [6] for light hardware has been developed. The current state of deep learning systems on edge devices still leaves an unsatisfactory result comparing with cloud server mainly because of the gap of calculation power between edge devices and cloud servers. It is prone to sacrifice either processing time or accuracy. Besides, the step of offloading input sensor data to large models in the cloud will easily lead to associated communication costs, latency issues and privacy concerns [7].

To solve insufficient calculation power of edge side, the thought of mobile edge computing [8] has been applied. Mobile edge computing is a cloud server running at the edge of a mobile network and complete some tasks that could not be achieved by traditional network on edge side. Besides, the edge side can preprocess data and extract feature that we need. Edge computing allows more computing tasks to take place on the decentralized nodes at the edge of networks. Many applications which are delay sensitive and mission-critical can leverage edge devices to reduce the time delay or even to meet the need of real-time detection and online decision.

1.3. Thesis outline

The outline of this thesis is organized as follows:

Chapter 1: We describe the background of real-time objects detection and with deep learning and the problem that needed to be solved in this work. Besides, some challenges and problems existing in the field of real-time objects detection at this stage have been pointed out, and we propose the innovation points of specific application scenarios based on the edge-cloud system.

Chapter 2: We introduce the technologies related to this work, ranging from the current research status in the field of target detection, the summary of the research on several types of target detection algorithms, the principle knowledge of CNN and the residual network algorithms. Through analyzing the contributions, focuses and the limitations of previous detection networks, the potential benefits of our work have been shown.

Chapter 3: We demonstrate the whole edge-cloud system which is designed for real-time tasks. To satisfy the demand of performance and other details in whole system, we design a new network which is going to be applied in edge side and introduce the framework of the new network. Besides, the inference design in the system has been discussed in this chapter.

Chapter 4: The experimental environment is introduced in this chapter. By training the designed network on test dataset, the evaluation results be analyzed and be compared with other network, along with the illustrated results, we demonstrate the superiority of our proposed method in the field of accuracy and time effectiveness.

Chapter 5: In this chapter we conclude this thesis.

2. Related Technologies

2.1. Classification and detection methods

As an important branch of computer vision, objects detection has developed rapidly in the fields of video and image recognition. In recent years, due to the substantial increase in hardware CPU and GPU computing power, deep learning has developed rapidly, the results obtained by traditional target detection methods are slowly overtaken by target detection based on deep learning algorithms, and the detection based on deep leaning algorithms achieve better results. So nowadays, the mainstream objects detection methods can be divided into the following two categories: traditional objects detection algorithms and objects detection algorithms based on deep learning. Figure 2.1 shows a brief overview of objects detection algorithms.

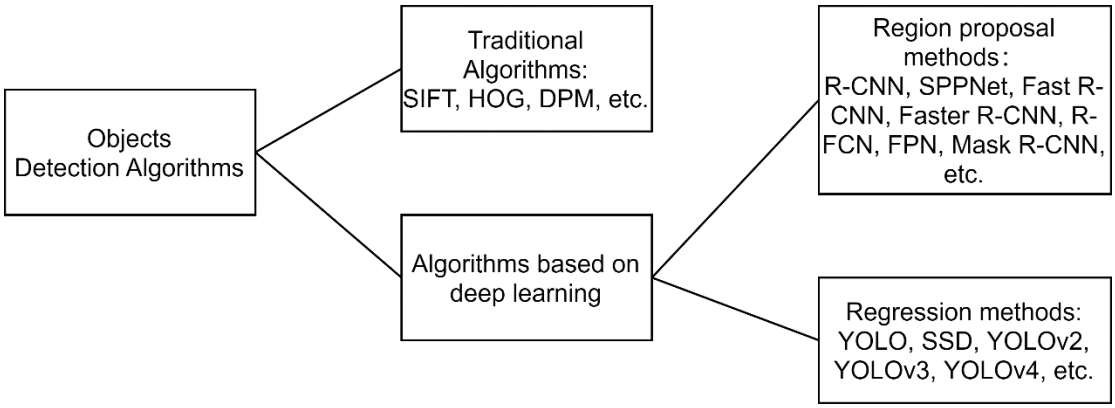

Figure 2.1 Classification of Object Detection Algorithms

2.1.1. Traditional objects detection algorithms

Traditional target detection methods [9, 10] mostly use the following steps: 1) Identifying the target on the image and filter out the target area; 2) Extracting the target features of the candidate area; 3) Using the classifier such as support vector machines [11] to classify the candidate target in the last step. Following is the detailed operations and characteristics of

each steps.

1) Region selection

In a given picture, the position of the target object in the picture appears randomly, and the size of the target is undeterminable. Therefore, the selection of candidate regions requires the use of sliding windows with different aspect ratios to scan the image, resulting in many irrelevant sliding windows which cause that in high time complexity for the region selection process, and subsequent effects on the recognition efficiency of the entire model. To solve this problem, a sliding window with a fixed aspect ratio is usually used, but the detection effect of multiple categories in the image is reduced, and the target position cannot be selected much accurately.

2) Feature extraction

Feature extraction uses SIFT [12], HOG [9] or DPM [13, 14] operators. However, in actual images, the shapes of detection targets are diverse. Besides, the lighting conditions are complex and the background is ever-changing. These complex conditions cause that it is difficult for traditional operators to adapt to the real scenario, resulting in poor performance on feature extraction. If the feature is not extracted well, the classification effect of the subsequent classifier will be unqualified.

3) Classifiers

In this step, every classes in the graph will have their own corresponding classifier and will be trained separately. The classifier can be Linear SVM or Adaboost [15]. The features extracted from each candidate region will pass to each classifier, and comprehensively judge for classification.

Generally, the main problems affecting the efficiency of traditional target detection algorithms are region selection and feature extraction. The region selection strategy based on sliding windows is not targeted, leading to window redundancy. Besides, traditional operators are difficult to adapt to changes in diversity of targets' size. These problems lead to the traditional target detection algorithm need more time on region selection with low accuracy.

What's more, the system has poor recognition robustness for multi-category object features.

2.1.2. Objects detection based on deep learning

Objects detection algorithms based on deep learning can be divided into two parts. One is the R-CNN series of algorithms which is based on candidate target regions like R-CNN[16]，Fast R-CNN[17] and Faster R-CNN[18]. They need to use selective search [19] or EdgeBoxes [20] to determine region proposal, and then do classification and regression. The continuous evolution of objects detection algorithms has shifted from dense sliding window-based methods like DPM to region proposal methods. The method of region proposal can effectively reduce the number of candidate bounding box. By achieving a more complex learning mechanism than sliding windows, the algorithm can improve the performance of accuracy on objects detection. Since CNNs won the championship in ILSVRC2012 [21], convolutional neural networks have been widely used in target detection models in recent years.

R-CNN applied CNNs to the bottom-up region filtering generated by selective search. R-CNN generates 2000 region of interest (ROI) through selective search, and extract features separately through CNN so that the network significantly improved the detection accuracy. In the final stage, R-CNN uses a support vector machine classifier to classify and predict the target. In order to obtain better performance, linear regression is also used to fine-tune the position and size of the detection bounding box. Since the R-CNN model has achieved amazing results, many new ideas have been implemented on CNN, such as the SPP-Net [22], Fast R-CNN and Faster R-CNN. The accuracy of objects recognition and processing speed of the above methods have been rapidly improved, and the fastest recognition speed can reach 15fps. The continuous improvement of the Fast R-CNN makes the detection task more accurate and faster. Although these methods use thousands of region of interest to reduce the space of the target area that the image needs to search, the detection speed still cannot meet the requirements of real-time target detection.

Compared with the two-stage detection model, the one-stage detection model of the YOLO

[23] series and the SSD [24] series has realized real-time detection due to its simple structure and faster detection algorithm as Figure 2.2 shows.

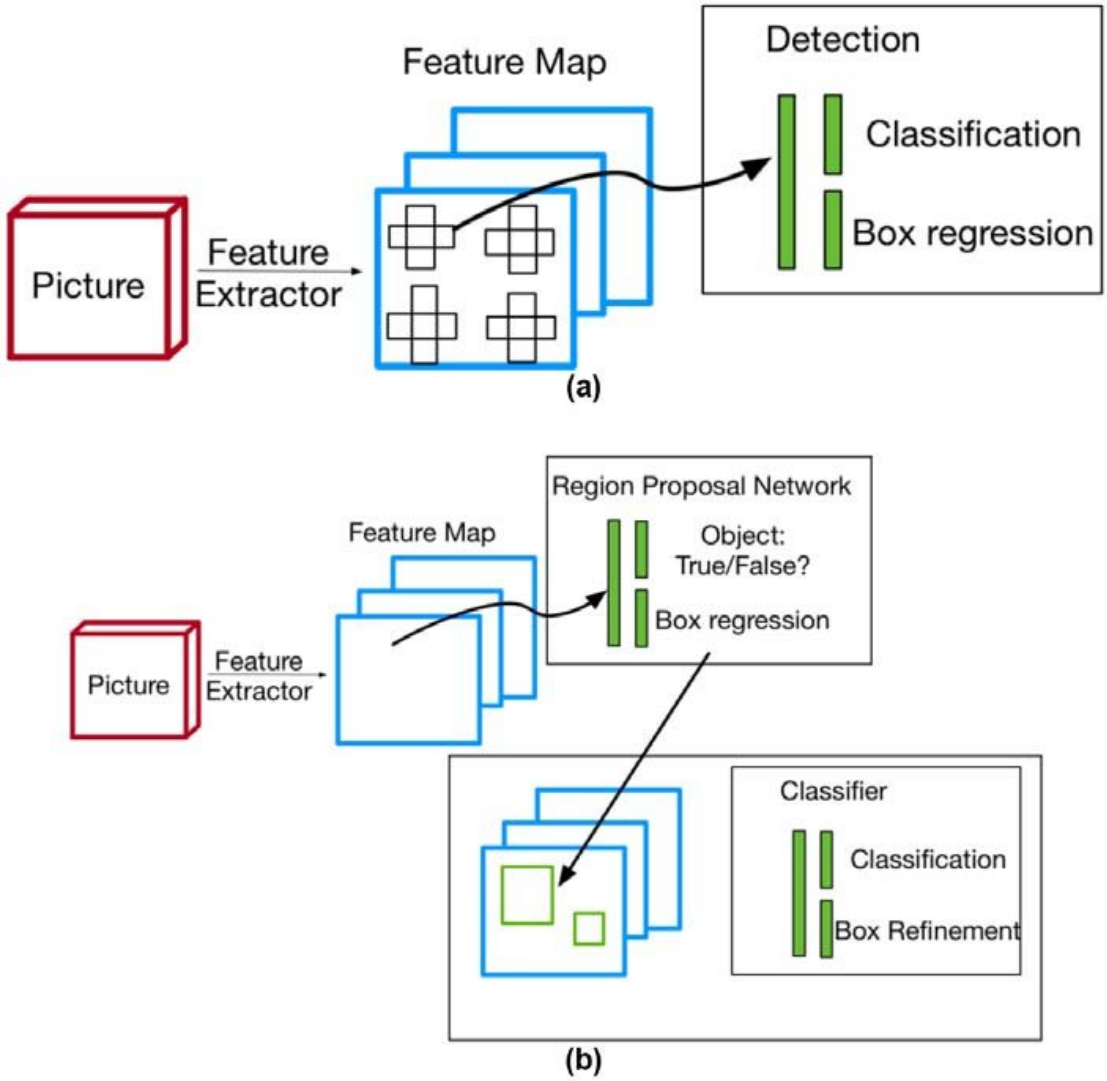

Figure 2.2 Schematic plot for (a) one-stage detector and (b) two-stage detector

2.2. You Only Look Once

In 2016, one-stage detection algorithm YOLO with a simple network structure has been proposed. YOLO firstly divides the image into S×S grid units, and each grid predicts the bounding box and confidence score of the object, and then filters the redundant bounding box based on the probability distribution of the category. Compared with the R-CNN series of two-stage detection algorithms, YOLO does not require intensive calculations before and after ROI

warping, which makes the algorithm parameters smaller and make the processing of calculation faster. Based on YOLO, YOLOv2[25] proposed a new backbone network Darknet-19 that reduced the computational cost by 80% compared to VGG16 network, and added several batch normalization layers after each convolutional layer to speed up the operation of convergence aiming to improving the speed of network training. Besides, the anchor mechanism is applied in the YOLOv2, and the K-means clustering algorithm is used to determine the number and size of anchors, which significantly improves the recall rate, but its positioning of the bounding box is still inaccurate. Compared with YOLO, YOLOv2 achieves 21.6% mAP in the MS COCO [26] dataset with faster speed.

YOLOv3 [27] proposed less floating-point operations and new backbone network Darknet-53 which use convolutional layers to achieve image scale changes and introduces a residual structure to improve detection accuracy. When predicting the anchor box, the confidence and the coordinates are predicted separately, instead of directly predicting the bounding box coordinates and confidence through the network regression in YOLOv2. In addition, drawing on the idea of feature pyramid and predicting on three scales, the detection rate of small targets can be improved to a certain extent. YOLOv3 has improved 11.4% mAP on the MS COCO data set with fewer parameters.

In 2020, YOLOv4 [28] was proposed as an efficient and powerful target detection model. CSP Darknet53 [29], which can better balance the input network resolution, number of convolutional layers, and parameters, was selected as the backbone network. SPP[22] module is added in the network, which uses four pooling layers of different scales to operate on features, thereby significantly increasing the receptive field without affecting the running speed and getting more context features. Thanks to the SPP module, the detection performance is improved by fusing three features of different scales. In addition, YOLOv4 uses the Mosaic data enhancement method, self-confrontation training method, cross mini-batch normalization method, point-oriented attention module and other improved tuning methods to further improve the detection accuracy. YOLOv4 obtained 43.5% mAP at 65 frames per second on the MS COCO data set, which is 10.5% higher than YOLOv3 mAP.

2.2.1. Algorithm of YOLO

YOLO divides the picture into S×S areas. Note that the concept of this area is different from the area where the picture is divided into N areas mentioned above and thrown into the detector. The area mentioned above is really cropping the picture, or cutting a certain part of the picture into the detector, and the division area here is only a logical division. The division is reflected in the last fully connected layer of YOLO, which is the prediction made by YOLO for each picture.

The predicted vector is a vector of length S×S× (B×5+C). Where S is the number of grids divided, generally S=7; B is the number of frames predicted by each grid, generally B=2; C is the number of categories related to your actual problem, but it should be noted that we should use the background as one category is considered.

S×S×C category information indicates what category each grid may belong to; S×S×B confidence levels indicate the confidence level of B boxes in each grid. After YOLO predicts, generally only the confidence level is above 0.5 the boxes will be retained. Of course, this threshold can also be adjusted manually. S×S×B×4 pieces of position information, the 4 pieces of position information are (x, y, w, h), where x, y are the center points of the box. We multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i \mid \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2.1}$$

which gives each candidate box a certain confidence scores on class perspective. These scores contain the meaning not only degree of how well the class we predicted in box fits to the real object in the image but also the probability of this class appearing in the box.

In YOLOV2, the model introduces the K-means clustering algorithm to filter out the most suitable candidate bounding box, and the K-means algorithm can be used to predict the length and width of the detection boxes. In order to reduce the error of the Euclidean distance in K-means on the position of the target candidate boxes, the intersection and union ratio (IOU) is set as the measurement standard:

$$d(\text{box, centroids}) = 1 - \text{IOU}(\text{box, centroids}) \tag{2.2}$$

YOLO will output of 7×7×30 features at last, and each cell corresponds to 1×1×30. In one cell, the first 10 features mainly contain coordinates of 2 bounding boxes, and the last 20 represent the probability that the cell belongs to 20 categories under the assumption that it contains objects. Figure 2.3 is the diagram of feature dimension.

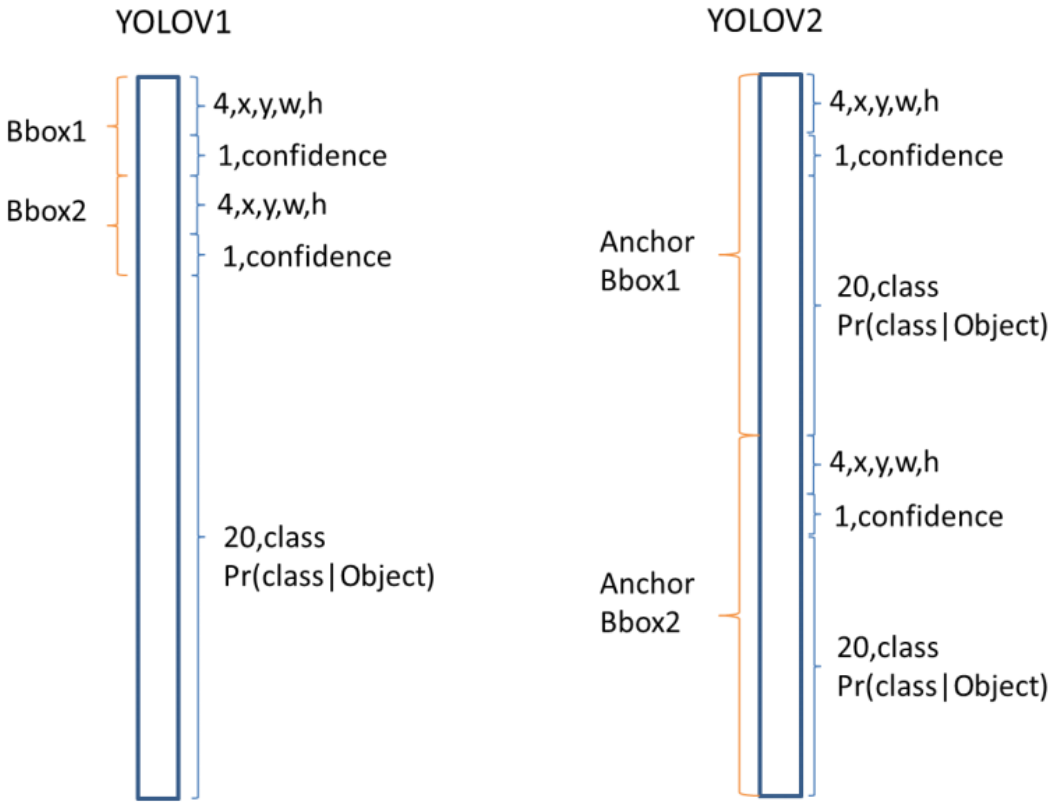

Figure 2.3 Characteristic dimension difference between YOLOv1 and YOLOv2

2.2.2. Architecture of YOLOv2

The network structure of YOLOv2 is similar to GoogleNet [30] which is contained with convolutional layers, max-pooling layers and fully connected layer. The convolutional layer and max-pooling layer in the network are used for feature extraction, and the usage of fully connected layer is to output category scores and the location of the target. The network structure is shown in Figure 2.4. The network system can read images of any size and feed

them back to the network. The input image is divided into 7×7 grids and for each grid, and the network will do prediction and output 3 bounding boxes with object category classifications.

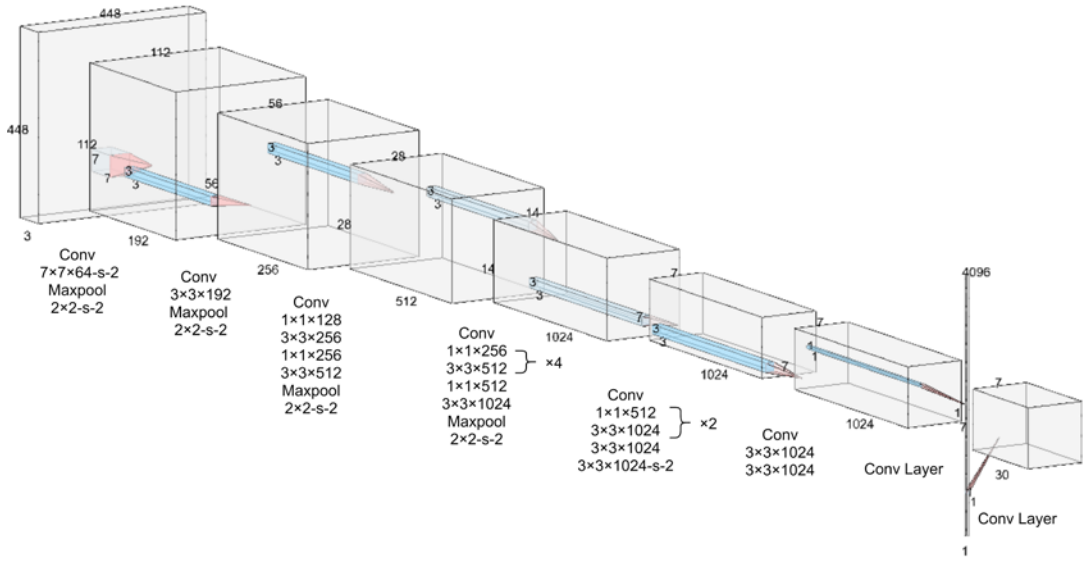

Figure 2.4 YOLO Network Structure

2.2.3. Loss function

In YOLO, we use multi-part loss function for training. There are 3 parts in the multi-part loss function: the loss for calculating the confidence error of the background, the loss for calculating the coordinate error of anchor boxes and prediction boxes and the total loss of each part of the prediction box with the ground truth.

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\
&+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] \\
&+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2
\end{aligned} \tag{2.3}$$

In the third part, the loss between ground truth and prediction boxes can further separate into 3 parts. One is coordinate loss, firstly we determine which cell the center point falls on,

and then calculate the IOU value of the 5 a prior boxes and the ground truth of this cell. Second part is confidence loss. We add a weight coefficient in the processing of calculating loss, when it is 1, the loss is the true IOU value of the prediction frame and ground truth. And the final one is the classification loss.

2.2.4. Non-Maximum Suppression

Non-Max Suppression [31] means that the detection results of each candidate frame are compared, the maximum value is retained, and other repeated regions are screened and removed, thereby leaving the target candidate region with the best effect. In the target detection model, many rough candidate results can be obtained through detection, but it is obviously unrealistic to adjust these rough results one by one. Therefore, these results need to be filtered out, and the most accurate case can be selected from them, and then the screening subsequent results are adjusted to improve model efficiency.

Algorithm1 Non-Max Suppression

```
1: procedure NMS(B, c)
2:     Bnms ← Ø
3:     for b_i ∈ B do
4:         discard ← False
5:         for b_j ∈ B do
6:             if same(b_i, b_j) > λnms then
7:                 if score(c, b_j) > score(c, b_i) then
8:                     discard ← False
9:         if not discard then
10:            Bnms ← Bnms ∪ b_i
11:    return Bnms
```

As shown in the Figure 2.5, sort according to the classification probability of the classifiers

of these boxes for the area where is a dog : A<B<C<D<E.

Then we take out the E detection box with the highest probability and calculate the intersection ratio of A, B, C and E respectively. The algorithm will compare with the previously set threshold after calculation. If the IOU is greater than the set threshold, keep the current corresponding, if the detection frame is less than the set threshold, remove the corresponding detection frame, such as the B frame and the maximum probability E frame for storage.

Second round we select D from the remaining detection boxes of A, C, D, and then repeat the above steps, respectively use the A, C box and the detection box D to calculate the intersection ratio, keep the greater than the threshold. We can remove the bounding box less than threshold and save D as the second reserved detection frame, and continue to iterate to find all remaining detection box.

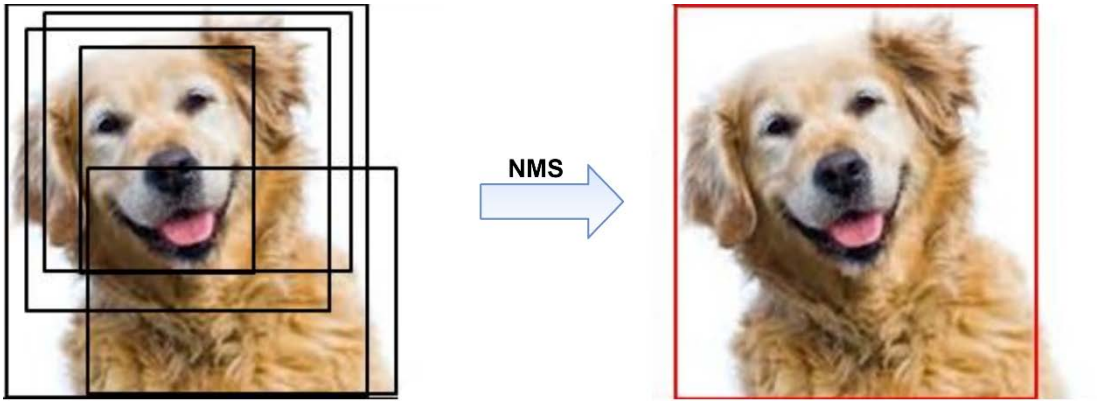

Figure 2.5 Comparison of NMS algorithm before and after screening

2.2.5. Bounding-box regression

We can find that the IOU intersection ratio between the red box (proposal) and the target marker box (ground truth) detected by the algorithm in the model is less than 0.5, which is lower than the set threshold of 0.5 as shown in Figure 2.7. Therefore, this detection frame cannot be used as the target detection result and we need to fine-tune the proposal and make it closer to the annotation window of ground truth.

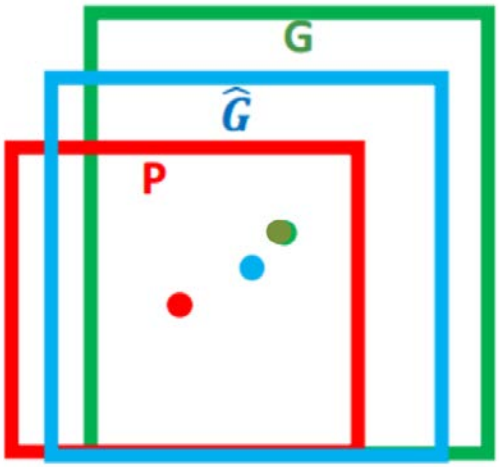

Figure 2.6 Bounding-box regression diagram

In Figure 2.6, red box represents the original proposal, green box represents the ground truth. Our ultimate goal is to make proposal and ground truth as close as possible. The optimization goal can be expressed as formula 2.4.

$$W_* = \arg\min_{\hat{W}_*} \sum_i^N \left(t_*^i - \hat{W}_*^T \phi_5(P^i)\right)^2 + \lambda \|\hat{W}_*\|^2 \quad (2.4)$$

To achieve this goal, the normal method is translating the center point firstly, and then doing scale scaling.

2.3. ResNet

Deep convolutional networks have made a series of breakthroughs in image classification tasks and it integrates three-level features and classifiers of low, medium and high through a multi-layer end-to-end approach, and the number of these features can also be increased by stacking the number of layers. As the number of network layers increases, problems in the procedure of training become more prominent. The more significant problem is the disappearance/exploding of gradients, which will affect convergence at the beginning network training. On the premise that the deep network can converge, as the network depth increases, the accuracy rate begins to saturate or even drop, which is called the network degradation. Because of this problem, increasing the number of layers on a given network will increase the

training error.

The degradation of the network shows that not all systems are easy to optimize. Under extreme conditions, if all the added layers are direct copies of the previous layer, the training error of the deep network should be the same as that of the shallow network. Therefore, the main problem that cause the network degradation is still an optimization problem.

In order to solve the optimization problem, a residual network is proposed. In the residual network, instead of letting the network directly fit the original mapping, it fits the residual mapping. The residual network adds some shortcut connections to the forward network so that these connections will skip some layers and pass the original data directly to the subsequent layers. The newly added shortcut connection will not increase the parameters and complexity of the model.

Kaiming He proposed a brand-new network called Deep Residual Network [32], which allows the network to deepen as much as possible, and introduced a new structure as Figure 2.8 shows.

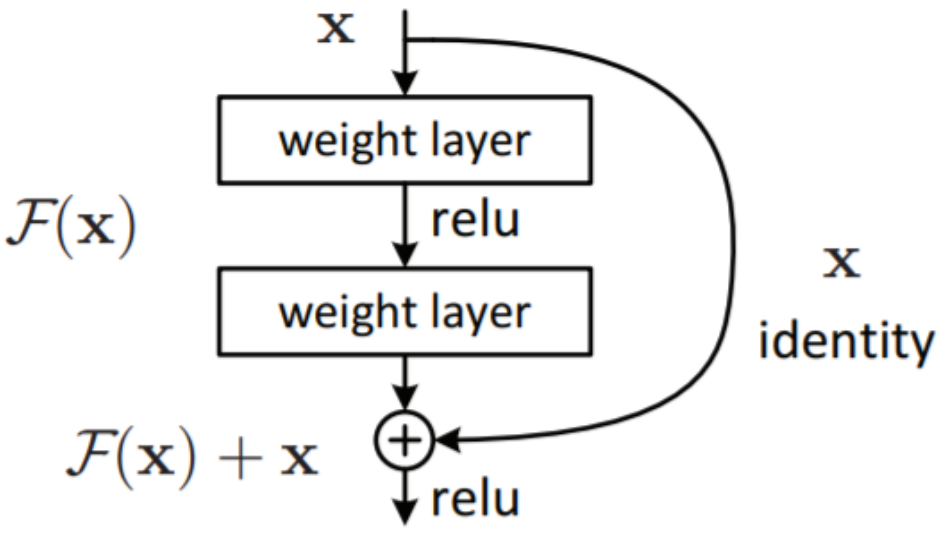

Figure 2.7 Residual Block Diagram

ResNet proposes two types of mappings: one is identity mapping, which refers to the curve in Figure 2.7, and the other is residual mapping, which refers to the part except the curve.

Based on the structure, we know the final output is y = *F*(x) + x. To solve the problem that accuracy is decreasing as the network deepens, ResNet provides two options. If the network has given acceptable output, the residual mapping will be decreased to 0 when we continue to deepen the network, which means that the network is always in the optimal state in theory, and the performance of the network will not decrease as the depth of network increases.

There are two connection grogram of ResNet which are shown in Figure 2.8. Left one is the BasicBlock, which is designed for ResNet-18 and ResNet-34. BasicBlock does not do the upgradation, so the output dimension of the residual structure is the same as the input dimension. And right one is Bottleneck, which is used on ResNet-50/101/152. The purpose of using Bottleneck is to reduce the number of channel dimensions and increase speed.

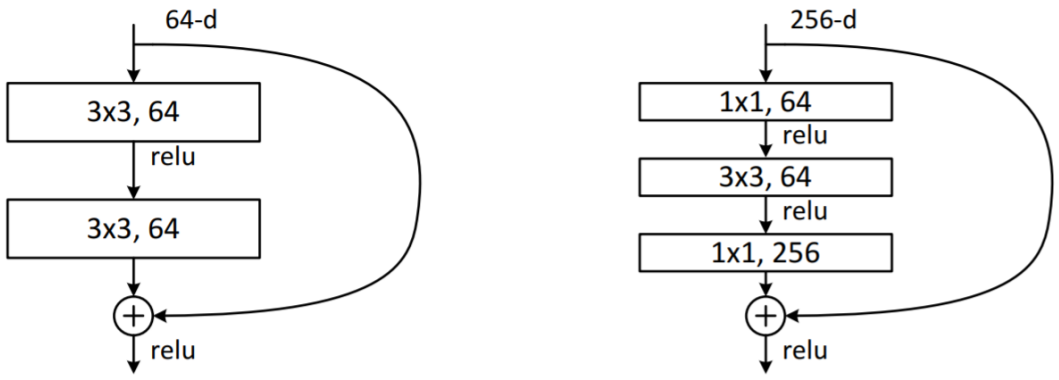

Figure 2.8 Structure of BasicBlock and Bottleneck

2.4. Metric

2.4.1. Precision and recall

Normally, a good classifier requires a combination of precision and recall to evaluate it. We usually use the degree of precision, recall and F1 score as the basic data to judge the performance of a network.

Table 2.1 Confusion matrix example

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

True Positive (TP): detected that the target is this category and matches the real category classification;

False Positive (FP): detected that the target is this category and the classification situation is different from the real category;

False Negative (FN): detected that the target is not this category and the classification situation is different from the real category;

True Negative (TN): detected that the target is not this category and matches the real category classification.

The following parameters are usually used in detection task:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{num of predicted Positive}} \tag{2.5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{num of actual Positive}} \tag{2.6}$$

$$\text{F1 score} = \frac{2 \cdot \text{P} \cdot \text{R}}{\text{P+R}} \tag{2.7}$$

2.4.2. mAP

mAP (mean average precision) is used for measuring the recognition accuracy in target detection. In multiple categories of object detection, each category can draw a curve based on recall and precision, AP is the area under the curve, and mAP is the average of multiple

categories of AP

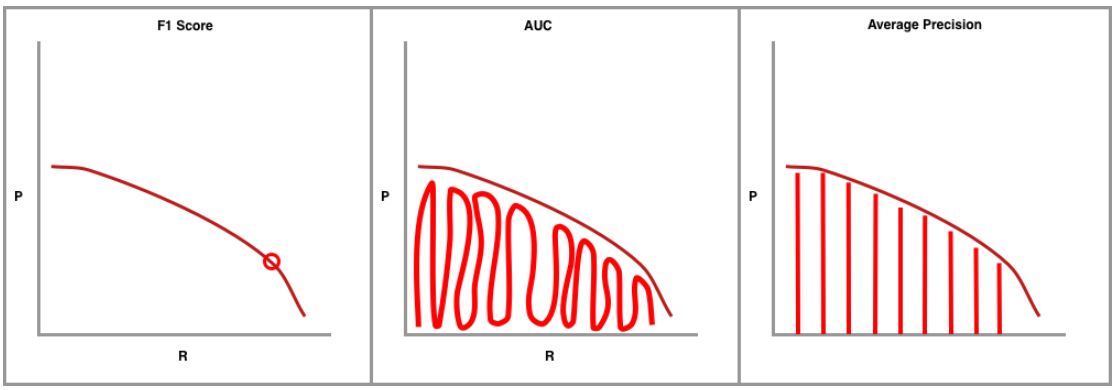

Figure 2.9 Evaluation of F1 Score, AUC and AP

2.4.3. FPS

In addition to the performance of accuracy, another important performance indicator of the target detection algorithm is speed. Only high speed of detection can achieve the goal of real-time detection, which is extremely important for some application scenarios. A common indicator for evaluating speed is frame per second (FPS), which is the number of pictures that can be processed per second. The time required to process a picture can also be used to evaluate the detection speed, which means that the shorter the processing time, the faster the speed.

3. Proposed Approach

Taking the thought of BranchyNet [33] and other edge computing system as reference, we incorporate the proposed residual block into network at edge side. So there are several architectural improvements need to be applied. We will introduce them in detail and further discuss the framework of real-time detection system and the design principle behind it.

3.1. ECNet

Edge and cloud cooperative approach for object detection has been proposed. The current state of deep learning systems on edge devices leaves an unsatisfactory result mainly because of the gap of calculation power between edge devices and cloud servers. It is prone to sacrifice either processing time or accuracy. Besides, the step of offloading input sensor data to large models in the cloud will easily lead to associated communication costs, latency issues and privacy concerns.

To address these problems, it is nature to consider an edge-cloud system which combine a light weight neural network on edge devices with a high-performance network on cloud servers and compress input sensor data, then offload it from edge side to cloud side depending on the metric for the result of edge side. The light weight model at an edge device can quickly output feature extraction, and also complete the judgement if the model is confident. The sensor data which are not fulfilling the metric will send to cloud side to do further processing and final classification or detection. This approach has the benefit of low communication costs compared to continuous offloading input to the cloud and can achieve higher accuracy compared to a simple model on device. Additionally, since sensor data which has been feature extracted and compression from the edge device model are sent instead of raw image data, the system could provide lower need of network bandwidth and better privacy protection.

Based on the following guideline and the thought of BranchyNet, we design a new edge-cloud system aiming for objects detection task, named ECNet [36]. As shown in Figure 3.1, the

general framework of ECNet is mainly consists of edge operation and cloud operation. Feature maps extracted from edge-side will be transferred to cloud-side determined by offload controller.

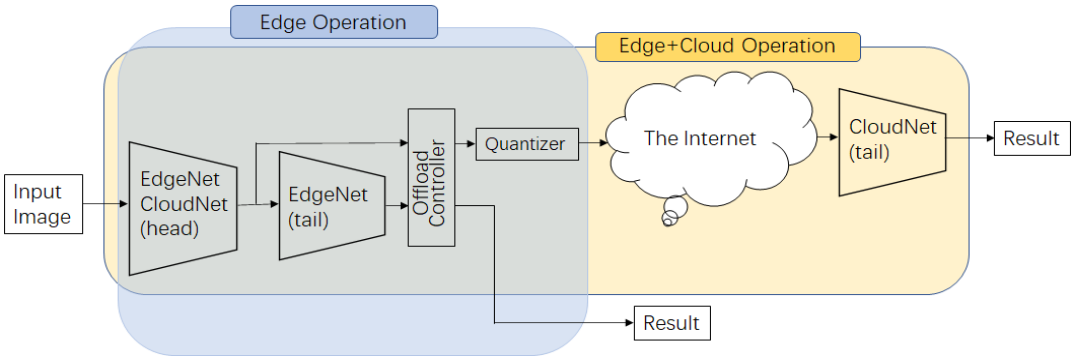

Figure 3.1 General framework of ECNet [36]

3.2. Network at edge side

3.2.1. Architecture

In the initial construction of the edge-cloud system. The edge side is applied Darknet19 and the cloud side will leverage DarkNet53 as backbone, planning setting exit point at edge side. After the compression the sensor data will transfer to cloud side. Owing to the different structure of both side network, sensor data extracted at the exit point of edge side could not be directly employed in further processing. Therefore we intend to reconstruct front part of both network being same so that sensor data extracted from edge side can directly apply in cloud side. So we have to change the structure of network. However, this kind of distributed approach is challenging for a number of considerations, including:

- Residual block in Darknet53. In structure of DarkNet53 is built on numerous residual block and each of residual block contains successive 3 × 3 and 1 × 1 convolutional layers one shortcut connection. This structure is aiming to solve the degradation problem on network which has deep structure. Reconstructed front part of network should avoid dividing residual block to ensure its integrity.

- In detection tasks, YOLOv3 predicts boxes at 3 different scales. Cloud network extracts features from those scales using a similar concept to feature pyramid networks. It has good performance on small objects that are to be recognized by the detector. The location of offloading sensor data to cloud side should before the layer where extracting features.
- Simplify the structure to limit computing cost and processing time at edge side.

Table 3.1 Architecture of designed network

	Type	Filters	Size/Stride
	Convolutional	32	3 × 3
	Convolutional	64	3 × 3 / 2
1×	Convolutional	32	1 × 1
	Convolutional	64	3 × 3
	Residual		
	Convolutional	128	3 × 3 / 2
2×	Convolutional	64	1 × 1
	Convolutional	128	3 × 3
	Residual		
	Maxpool		2 × 2 / 2
	Convolutional	256	3 × 3
	Convolutional	128	1 × 1
	Convolutional	256	3 × 3
	Maxpool		2 × 2 / 2
	Convolutional	512	3 × 3
	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
	Maxpool		2 × 2 / 2
	Convolutional	1024	3 × 3
	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
	Convolutional	1000	1 × 1
	Avgpool		Global
	Softmax		

Guided by aforementioned considerations, the structure of edge-cloud network is decided as Table 3.1 shows after several times trials and comparison.

3.2.2. Loss function

In the target detection task, the network needs to predict the class score of the object and the position of the bounding box. Therefore, the total loss $\text{loss}_{\text{total}}$ is composed of category cost $\text{loss}_{\text{class}}$ and bounding box expenditure loss_{IOU}, as shown in formula 3.1. The category loss is the difference between the category probability that network classified and the ground truth. The loss of the bounding box is calculated by the intersection set (IOU) between the predicted box and the ground truth box.

$$\text{loss}_{\text{total}} = \text{loss}_{\text{class}} + \text{loss}_{\text{IOU}} \tag{3.1}$$

The influence of these parts on the overall detection effect depends on the degree of their contribution, so a weight parameter needs to be added for fine-tuning. The position error in target detection has a relatively large influence factor, so the position error weight is generally set to 5. Formula 3.2 is the loss of target detection in the thesis. The scale parameter is used to adjust the weight of different costs. The total loss is the sum of the S × S grid loss. The loss of each grid is composed of the object existence loss of N grids, the cost of class C, and the IOU loss of the best prediction box. Note that only when the object target exists in the grid, the loss function will penalize the class classification error by adding the class loss and the best IOU loss.

$$\begin{aligned}\text{Cost} = & \sum_{m=0}^{S*S} \{\sum_{n=0}^{N} \text{noobject}_{\text{scale}} * P_{\text{object},n}^{2} \\ & + is\text{Obj} * \{-\text{noobject}_{\text{scale}} * P_{\text{object,best}_{\text{prediction}}}^{2} \\ & + \text{object}_{\text{object}} * \left(1 - P_{\text{object, best}_{\text{predict}}}\right)^{2} \\ & + \left(1 - \text{IOU}_{\text{best}_{\text{predict}}}\right)^{2} \\ & + \sum_{i=0}^{c} \left(\text{class\_scale} * \left(P_{\text{class},i} - P_{\text{truth},i}\right)\right)\}\}\end{aligned} \tag{3.2}$$

In formula 3.2, S is the grid size, and N is the number of bounding boxes predicted in one grid. C is the number of target categories. The parameters of non-target existence ratio, target

ratio and category ratio are used to adjust the weight of each part of the cost. $P_{\text{object},n}$ is the probability that the target exists in n networks and m bounding boxes. The isObj indicates whether the target object exists in the current bounding box. $P_{\text{object,best prediction}}$ is the best intersection and union ratio (IOU) of $P_{\text{object},n}$ in n grid boxes. $\text{IOU}_{\text{best}_{\text{predict}}}$ is the best intersection ratio between the true bounding box and the predicted bounding box in m grid. It is the probability parameter that predicts whether the target category i exists, and the probability that there is a target object of category i in the image, that is, 0 or 1. $P_{\text{class},i}$ is the probability parameter for predicting whether the target category i exists, and $P_{\text{truth},i}$ is the probability of whether there is a target object of category i in the image or not .

3.3. Inference design

We add metric to judge whether the feature map should transfer from exit point or not and we call it confidence score. Confidence score is mainly based on entropy of the output at the end of edge network. If the score high than the threshold we set, which means the output is not acceptable. Then the feature map will transfer to the cloud side and do further operation. If the confidence score is under the line of threshold, which means the output is good enough for the whole system. So the system will directly output the result without enabling the cloud side.

Besides, we apply the technology of quantization and compression into the system for loosing the offload burden. Firstly the system will saving feature map as binary file which can greatly reduce the storage cost. After transfer the binary file the system will read the file and do the dequantization on feature map data.

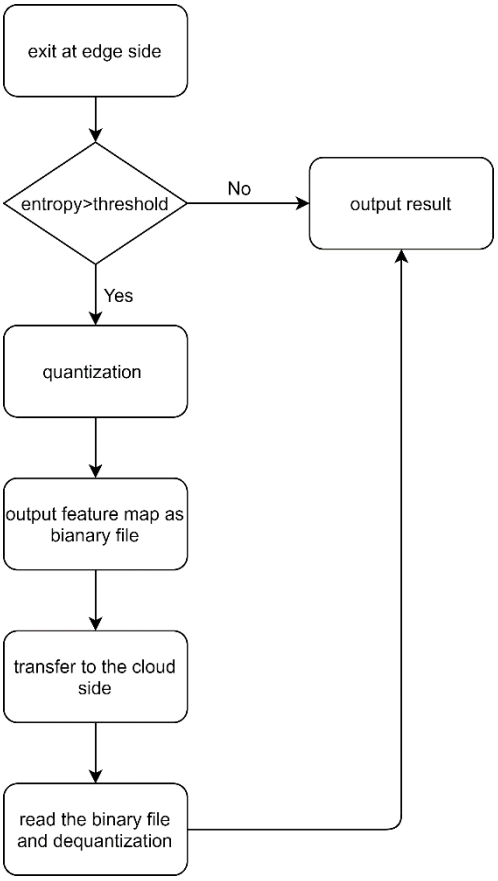

Figure 3.2 Flow of system inference

4. Experiment and results

4.1. Implementation details

4.1.1. Experiment environments

The ECNet edge-cloud system for real-time objects detection is implemented in PyTorch 1.3.1. The platform we train and evaluate the models is Nvidia GeForce GTX 2070 SUPER GPU with 8G memory and AMD 3600 CPU. And the experiments are conducted under the OS of Windows.

4.1.2. Dataset for training and testing

In order to verify the effectiveness of the model and the ability of target detection, we use two datasets in this experiment. The datasets we used are ImageNet [34] dataset and Pascal VOC 2007/2012 [35].

① ImageNet

The ImageNet image dataset started in 2009, followed by the 7th ImageNet Challenge based on the ImageNet dataset. From 2017, ImageNet dataset has been maintained by Kaggle. It is a large-scale visualization database for the research of visual object recognition task. More than 14 million image URLs are manually annotated by ImageNet to indicate objects in the picture and more than one million images is providing bounding boxes. Besides, ImageNet contains more than 20,000 categories.

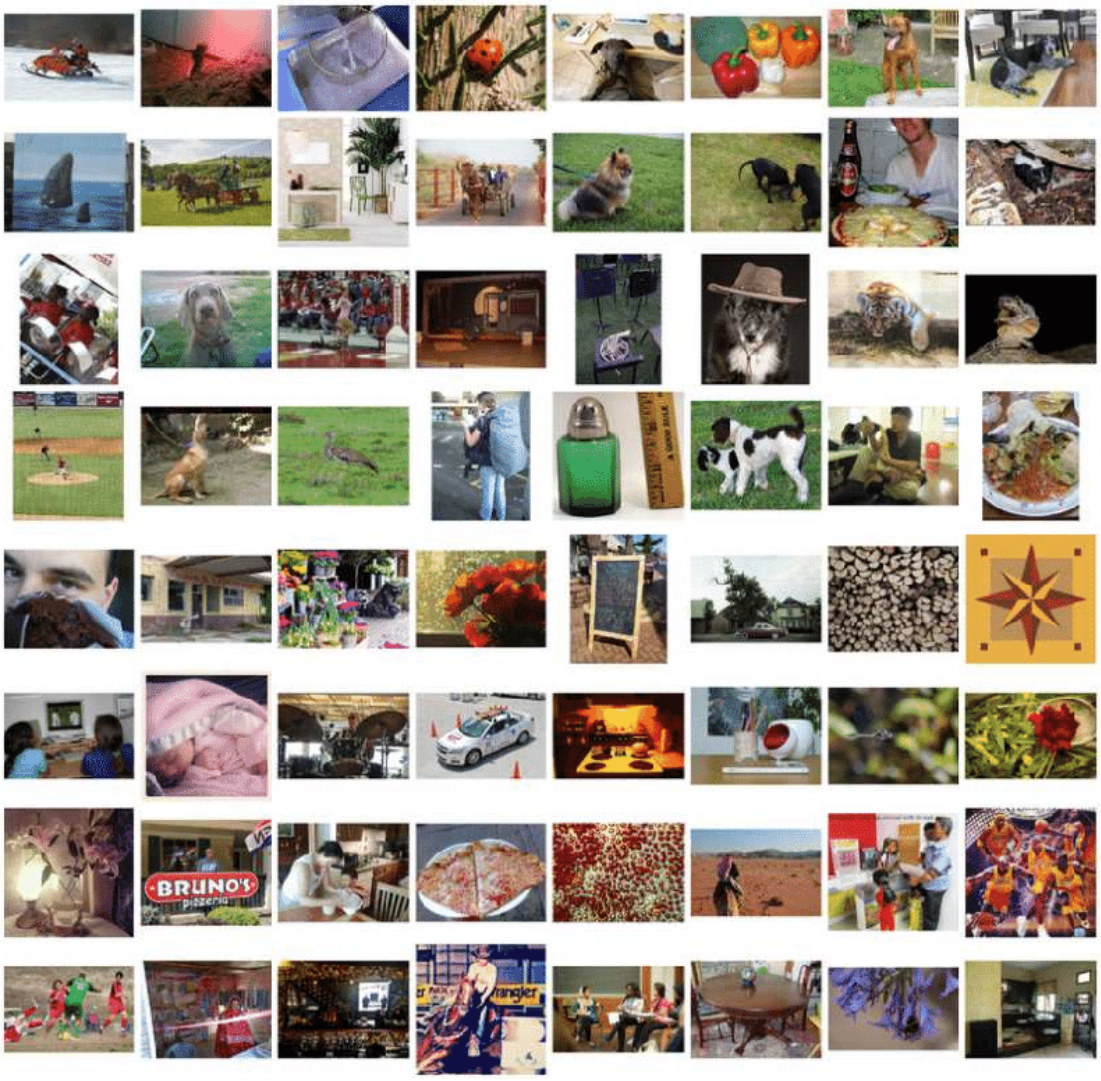

Figure 4.1 Samples from ImageNet Dataset [34]

② Pascal VOC 2007/2012

The Pascal VOC dataset is an important foundation of the Pascal VOC Challenge, which promotes the development of image classification and object detection. The dataset used in the thesis is from the Pascal VOC 2007 and 2012 Challenges, which contain 9963 images and 11125 images in 20 categories.

Figure 4.2 20 Classes in Pascal VOC 2007/2012 Dataset [35]

4.2. Experiments and results analysis

4.2.1. Design of network and theoretical analysis

Because of the total algorithm and framework of the edge-cloud system, the design of network at edge side guideline which means the network should not only meet the need of the compatibility with the cloud side and early exist mechanism but also have a relatively satisfactory performance on processing time and accuracy.

Based on the above guideline, we choose the darknet19 as the backbone, the structure of darknet19 is following. And then we add residual block into the layer and make a huge change at top layers.

Table 4.1 Architecture of Original DrakNet19

Type	Filters	Size/Stride
Convolutional	32	3 × 3
Maxpool		2 × 2 / 2
Convolutional	64	3 × 3
Maxpool		2 × 2 / 2
Convolutional	128	3 × 3
Convolutional	64	1 × 1
Convolutional	128	3 × 3
Maxpool		2 × 2 / 2
Convolutional	256	3 × 3
Convolutional	128	1 × 1
Convolutional	256	3 × 3
Maxpool		2 × 2 / 2
Convolutional	512	3 × 3
Convolutional	256	1 × 1
Convolutional	512	3 × 3
Convolutional	256	1 × 1
Convolutional	512	3 × 3
Maxpool		2 × 2 / 2
Convolutional	1024	3 × 3
Convolutional	512	1 × 1
Convolutional	1024	3 × 3
Convolutional	512	1 × 1
Convolutional	1024	3 × 3
Convolutional	1000	1 × 1
Avgpool		Global
Softmax		

During the trial on reconstruction based on DarkNet19, we see the growth of receptive field as an indicator for our design. We firstly to watch the change receptive field to determine where to add the residual block and how to set parameter. After that, we will train the designed model and to see the result. Setting the receptive field as indicator can help us abandon some low performance model quickly which make our work more efficiently.

Figure 4.3 shows the receptive field growth in our finally decided network structure. The blue part represents the amount of the receptive field after every layers and the red arrow

points the layer where the system will do the operating of sensor data compression and transmission.

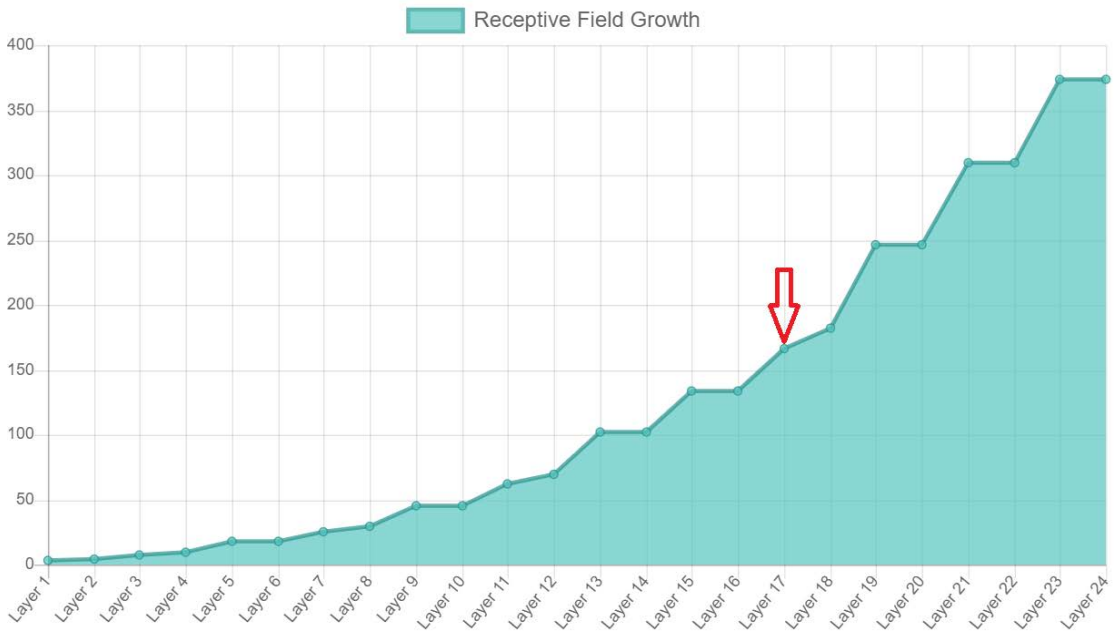

Figure 4.3 Receptive field growth of our network

4.2.2. Performance evaluation and analysis

After determining the network structure, we firstly pre-train the network with ImageNet dataset in 1000 classes. And then collect several classes aiming for the need of our task and do the further training. We use 10 classes dataset for experiment (10000 images for training and 3000 images for testing) and the result is shown in the following table.

Table 4.2 Performance comparison on the 10 classes dataset

	Rank-1(%)	Rank-5(%)	Processing Time(s/frame)
Our Network	68.5	81.8	0.013
Darknet19	64.3	76.4	0.006
Darknet53	81.2	98.2	0.023

The data set we used is under complex environment so the detection task is hard for the network. Based on such preconditions, we can find our designed network can improve the Top-1 accuracy about 4% comparing with the DarkNet19. But the price is that we have to scarify some performance on processing time. By running on our platform, the FPS our network can reach is about 77, which is enable for our real applying scenario.

Besides, our network has another merit is that the top layers before the first maxpooling layer keep the same structure with the DarkNet53 which is the backbone of the cloud side in the whole system. Using same structure means that they not only can share the weight of the top layers with each other but also simplify the procedure of sensor data transmission between edge side and cloud side in the whole system.

4.2.3. Adaptive adjustment of network

By validating and evaluating the performance of our designed network, we confirming that edge side network can meet our need. While the edge side network is serving for the whole system, so we should consider the algorithm of early-exit and the adaptive adjustment for different scenario.

The main component of the early-exit point is the confidence score which can measure how confident the result we get at edge side network. How calculate the confidence score for the detection result as shown in formula 4.1.

$$Pr(\text{class}_i \mid \text{object}) * Pr(\text{object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = Pr(\text{class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \qquad (4.1)$$

The confidence score includes two parts. One is the probability that the bounding box contains the target, and the other is the accuracy of the bounding box. Each detection cell will give the predicted category probability value, which represents the probability that the bounding box responsible for the cell belongs to which category. No matter how many bounding boxes a cell predicts, it only predicts one set of class probability values.

By setting different threshold of confidence score, we can not only decide the amount of

offload in the system but also can influence the performance of the network. Figure 4.4 shows how confidence threshold affects performance of edge side network in task of object detection. We use Pascal VOC 2007/2012 as the train and validate dataset. Accuracy will smoothly descend with threshold of confidence score until threshold about 0.5 and then the accuracy will drop fast.

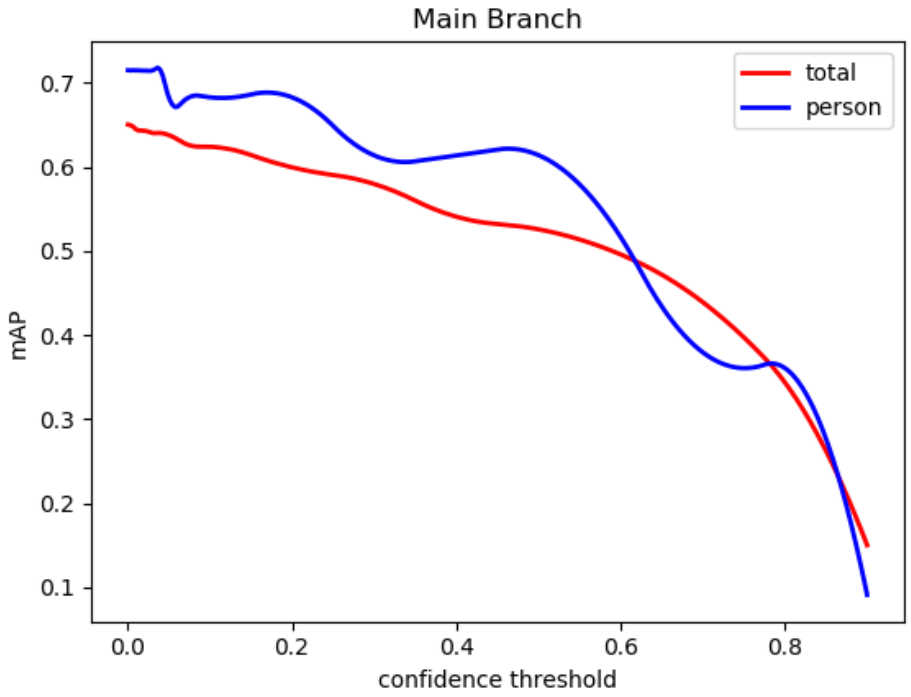

Figure 4.4 Accuracy of edge side network for varying confidence threshold

Through further analysis on different scenes, we find that setting of threshold of confidence score could have completely different result according to the task we are facing. Figure 4.5 show the performance of detection under different scene. The images under different scenario is selected from Pascal VOC 2007/2012, we can divide it into simple scene and complex scene. In simple scene, setting threshold higher will avoid unsuited bounding box. Meanwhile, in complex scene we can recognize that setting threshold lower will remain many bounding box with low probability value but there are also some repeated bounding box.

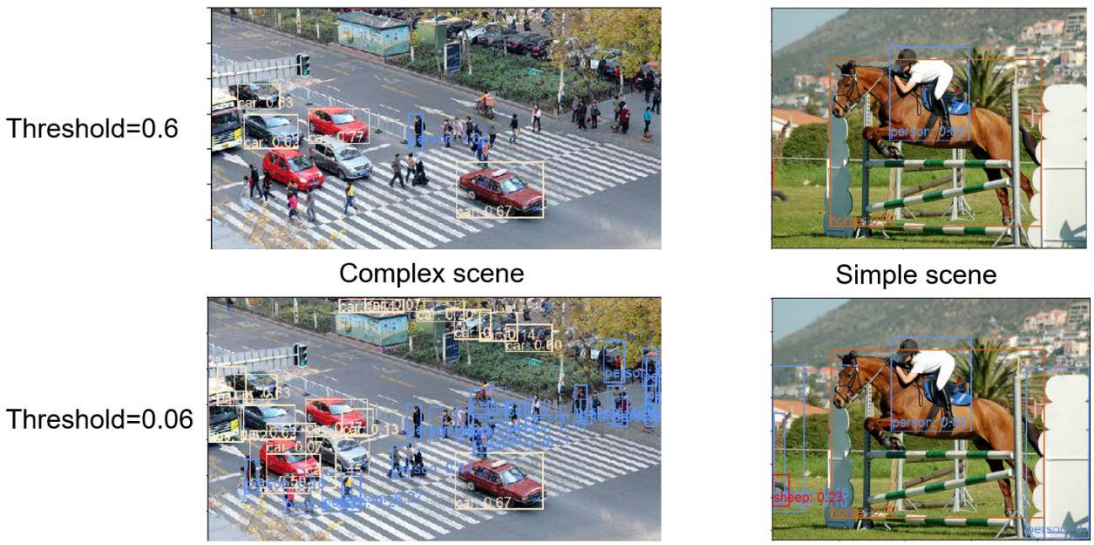

Figure 4.5 Performance with changing threshold under different scene

From the above result and analysis, we learn the characteristic of the confidence score and its threshold. The changing performance with the different scenes matches with our task in real scenario very well. No matter in parking lot or market, the changes of cars or people flow is periodic. So the threshold of confidence score also can be set periodic following the variety of cars or people flow.

5. Conclusion

We proposed ECNet, which is an edge-cloud network system to make combination and connection between lightweight network on mobile devices and high-performance network on cloud servers, dealing with the balance between performance and time cost in the real-time detection tasks. Our main focus is on the designation of the lightweight network and offload algorithm and methods in system.

Based on the constraint of computation power and the intercommunity between edge side and cloud side, we designed the network applying at edge devices and set an exit point for sensor data offloading. The result analysis and comparison on 10 classes' dataset testify the performance of our network. Furthermore, the algorithm of early-exit also shown in the thesis and the potential of adaptive adjustment for different scenario. The proposed method is demonstrated to be able to fulfill the needs of real-time tasks at the edge side. And as a part of ECNet system, the early-exit algorithm can connect two sides successfully.

6. Appendix

6.1. List of academic achievements

International conference:

L. Hu, T. Wang, H. Watanabe, S. Enomoto, X. Shi, A. Sakamoto and T. Eda: “ECNet: A Fast, Accurate, and Lightweight Edge-Cloud Network System Based on Cascading Structure,” IEEE Global Conference on Consumer Electronics (GCCE) 2020, pp.259-262, Sep. 2020.

Domestic conference:

T. Wang, L. Hu, H. Watanabe, S. Enomoto, X. Shi, A. Sakamoto, and T. Eda: “Exit-Point Setting in Edge-Cloud Solution for Object Detection,“ IEICE General Conference D-12-14, Mar. 2020

L. Hu, T. Wang, Y. Zhou, H. Watanabe, S. Enomoto, X. Shi, A. Sakamoto, and T. Eda: “Transfer Rate Estimation in Edge-Cloud Neural Network Solution for Object Detection,“ IEICE General Conference D-11-20, Mar. 2020

Bibliography

[1] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” International Journal of Computer Vision (IJCV), 2013.

[2] Takumi eye, https://www.ntt.com/content/dam/nttcom/hq/jp/about-us/press-releases/pdf/2017/0712.pdf.

[3] P.P. Ray, “Internet of Robotic Things: Concept, Technologies, and Challenges,” IEEE Access, 4 (2016) 9489-9500.

[4] H. Choi, and I.V. Bajic: “Deep Feature Compression for Collaborative Object Detection,” IEEE International Conference on Image Processing (ICIP2018) WP. P6.8, Oct. 2018.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[6] I. Khokhlov, E. Davydenko, I. Osokin, I. Ryakin, A. Babaev, V. Litvinenko, R. Gorbachev. “Tiny-YOLO object detection supplemented with geometrical data,” 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 2020.

[7] S. P. Chinchali, E. Cidon, E. Pergament, T. Chu, and S. Katti, “Neural networks meet physical networks: Distributed inference between edge devices and the cloud,” in Proc. ACM Workshop on Hot Topics in Networks, pp. 50–56, ACM, 2018.

[8] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.

[9] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection,” CVPR, pages I: 886–893, 2005.

[10] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004

[11] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” IEEE Intelligent Systems and their applications, vol. 13, no. 4, pp. 18–28, 1998.

[12] Lowe D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. “International Journal of Computer Vision,” 2004, 60(2):91---110.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 32 (9) (2010) 1627–1645.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” IJCV, 2010.

[15] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting,” Annals of Statistics, 28(2):337–407, 2000.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” CVPR, 2014, pp. 580–587

[17] R. Girshick, “Fast R-CNN,” in IEEE International Conference on Computer Vision (ICCV), 2015.

[18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015.

[19] J.R. Uijlings, K.E. van de Sande, T. Gevers, and A. Smeulders, “Selective search for object recognition,” IJCV, vol. 104, no. 2, pp. 154–171, 2013.

[20] C. Zitnick and P. Dollar, “Edge boxes: Locating object proposals from edges,” ECCV, 2014, pp. 391–405.

[21] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “Imagenet classification with deep convolutional neural networks,” NIPS, 2012, pp. 1097–1105.

[22] K. He, Zhang X, Ren S, et al., “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015, 37(9): 1904–1916.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” CVPR, 2016.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and Reed S.E., “SSD: Single shot multibox detector, [C]” CoRR, abs/1512.02325, 2016.

[25] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” IEEE Conference on Computer Vision and Pattern Recognition, 2017: 7263–7271.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft coco: Common objects in context,” European conference on computer vision, pages 740–755. Springer, 2014.

[27] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv preprint arXiv: 1804. 02767, 2018.

[28] A. Bochkovskiy, C. Wang, and H. Liao, “Yolov4: Optimal speed and accuracy of object detection,” arXiv preprint arXiv: 2004. 10934, 2020.

[29] C. Wang, H. Liao, I. Yeh, Y. Wu, P. Chen, and J. Hsieh, “CSPNet: A new backbone that can enhance learning capability of CNN,” CVPR Workshop, 2020. 2, 7.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[31] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, “Improving Object Detection with One Line of Code,” arXiv e-prints, Apr. 2017.

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv: 1512. 03385, 2015.

[33] S. Teerapittayanon, B. McDanel, and H.-T. Kug, “Branchynet: Fast inference via early exiting from deep neural networks,” 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016, pp. 2464–2469.

[34] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and F. Li, “Imagenet large scale visual recognition competition 2012,” ILSVRC2012, 2012.

[35] M. Everingham and J. Winn, “The pascal visual object classes challenge 2011 (voc 2011) development kit. Pattern Analysis, Statistical Model ling and Computational Learning, “Tech. Rep (2011)

[36] L. Hu, T. Wang, H. Watanabe, S. Enomoto, X. Shi, A. Sakamoto and T. Eda: “ECNet: A Fast, Accurate, and Lightweight Edge-Cloud Network System Based on Cascading Structure,” IEEE Global Conference on Consumer Electronics (GCCE) 2020, pp.259-262, Sep. 2020.