Real-time inference improvement on ECNet: A Lightweight Edge-Cloud Network System based on Cascading Structure

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

Libo HU
(5119FG04-4)
Advisor: Prof. Hiroshi Watanabe
Research guidance: Research on Audiovisual Information Processing

Acknowledgments

To begin with, I would like to express my greatly thanks to my research supervisor Professor Hiroshi Watanabe for his continuously overall support on all the stages of my research. He has provided an excellent laboratory circumstances and he has always been there to give us his constructive ideas when we face any problems.

I would like to thank the joint research group members of NTT Software Innovation Center, Mr.Takaharu Eda, Mr.Shohei Enomoto, Mr.Akira Sakamoto, and Ms.Xu Shi for their remarkable research advices and academic inspirations they provided.

I also want to give my appreciation to all members of the Advanced Multimedia Systems Lab for their support and encouragement through my two years study.

Lastly, my gratitude to my parent's constant overall support, and my girlfriend Miss.Guerrero Estrada Itzel Arianna, who has given me the warmest help through my life in Japan

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 insufficiency of computing power in mobile devices and the increment of network bandwidth, combination of edge devices and cloud servers would be an accessible orientation for real-time tasks. In this work, we present ECNet—an edge-cloud network system dealing with the balance between inference performance and time cost.

We propose to transmit the output feature map of an exit point from the edge-side network to the cloud-side network with an offload controller and quantizer deployed to minimize the transmission cost. ECNet is tested to reach a balance between processing time and accuracy performance with reducing transmission cost down to 25%. We also consider implementing an integrated feature map encoder to further reduce the bandwidth demand and meanwhile minimize the loss of accuracy. Additional achievements could be expected in our future work.

Keywords—Edge-cloud system, Quantization, ECNet, model cascading, classification, encoding.

Contents

List of Tables

Chapter 3

Chapter 4

List of Figures

Chapter 2

Chapter 3

Chapter 4

Chapter 1

Introduction

1.1. Background and Challenge

As the construction of smart cities is in full swing, technical requirements for intelligent surveillance and other related computer vision technology are also increasing. As a basic task in the field of computer vision, pedestrian detection has attracted most attention from experts in the field of computer vision. The traditional pedestrian detection method is to use a manually designed feature extractor to train the classifier by extracting features such as Histogram of Oriented Gradients (HOG) [1], Local Binary Pattern (LBP) [2], etc., to realize the detection of pedestrians. However, the artificially designed pedestrian characteristics are difficult to adapt to the large changes in pedestrian behavior. Essentially, pedestrian detection is nothing but a special object detection. Thus, it can be achieved by learning from the general object detection method. At present, the mainstream object detection methods are mainly divided into two types: one is two-stage, and the other is one-stage. The two-stage method mainly uses Fast Region-based with Convolutional Neural Network feature (Fast R-CNN) [3]and Towards Real-Time Object Detection with Region Proposal Networks (Faster R-CNN) [4]. The one-stage method includes unified real-time object detection (You Only Look Once, YOLO) [5], and its upgraded version (YOLOv2) [6], and single shot multibox detector (SSD) [7]. Compared with the two-stage method, the one-stage method has faster detection speed, but the detection quality is slightly lower. YOLOv2 has an outstanding performance in real-time detection and is faster, but when directly using YOLOv2 to detect pedestrians, since the pixels of pedestrians are relatively low, the detection effect of YOLOv2 is poor, and the position of pedestrians is not accurate enough. In addition, under the

condition of a high degree of overlap (Intersection Over Union, IOU) threshold, the performance of YOLOv2 is not ideal.

CNN (Convolutional Neural Network) methods such as HyperNet [8] use feature fusion to improve the detection quality of small objects, while Feature Pyramid Network (FPN) [9] uses multi-layer feature prediction to improve the detection quality. Pedestrian detection methods can also fuse CNN features and traditional pedestrian features. The advantage of feature fusion is that the low-level feature semantic information is relatively small, but the target location is accurate; the high-level feature semantic information is relatively rich, but the target location is relatively rough, and the context semantic features are integrated, the high-level low-level features are easy to detect. However, these methods always utilized very deep backbone network, which causes the network parameters to be too large, resulting in a very slow detection speed, which affects the applicability in actual detection.

In recent years, edge and cloud cooperative approach for object detection has been proposed [10]. Thanks to advancements in both hardware and deep learning technology, even deeper networks which further improve the classification performance have been emerged. The integration of deep neural networks can greatly enhance the functions of edge device, however, the rapid increase in runtime and power for gains in accuracy may deepen the neural network which become less tractable in many real-time situations where latency and energy costs are important factors ·

The current state of deep learning systems on edge devices leaves an unsatisfactory result mainly because of the gap of computation power between edge devices and cloud servers. It is prone to sacrifice either processing time or inference accuracy. Besides, the step of offloading image data to a large model in the cloud will easily lead to associated communication costs, latency issues and privacy concerns [11]. When meeting a real-time task with a very high data rate, the

challenge lies in the demand to achieve high image throughput with a limited transmission bandwidth.

1.2. Research Objective

To address the problems that is claimed above, we consider an edge-cloud system based on cascade structure, which combines a lightweight neural network on edge devices with a high-accuracy network on cloud servers. The lightweight model at the edge-side can quickly output feature map extraction, and also complete the inference. The offload controller takes charge of determine whether the inference result from the edge-side network is satisfactory or not. The feature map that is barely satisfactory should be transmitted to the cloud-side network for further processing with more powerful DNN model and relatively sufficient computational resources. The initial idea of our network designment is to achieve lower computing cost than that in a DNN model, and higher accuracy performance compared with a simple lightweight model on edge devices. Further improvement of edge-side and cloud-side network is fulfilled for compatibility. Additionally, data extraction and compression module are deployed to reduce communication cost and achieve real-time nature of our proposed network system [12].

The major contributions of this thesis are:

- **Designation and implementation of the edge-cloud architecture:** Specifically modified edge-side network processes majority of inference, and exits controllable parts of the feature maps. The whole system is mean to reduce computational costs, resulting in running time saving with achieving substantial overall performance on deep learning tasks.
- **System Regulation via offload-controller:** Entropy of classification result is set as threshold that operated by users to control the offload rate of the feature map data, thus ensure ECNet to meet customized accuracy demand

- **Feature data compressing and encoding:** We characterize the accuracy impact of different quantization while introducing several feature space encoding method with time cost analysis.

1.3. Thesis Outline

The outline of this thesis is organized as follows:

Chapter 1: We describe the background of the real-time detection and classification tasks. We briefly introduced the advantage and disadvantage of some mainstream network and pointed out the motivation we need to propose an edge-cloud network system.

Chapter 2: We introduced some related technology.

Chapter 3: We demonstrated the framework of the proposed ECNet system and explained the working flow. We specifically introduced the essential idea of the designment of the edge-side and cloud-side network with the model structure figure. We made some simulation experiment to evaluate the classification power of the edge-side network

Chapter 4: We designed an offloading controller to determine the task distribution of the edge-cloud network system. We used evaluated quantization approach to save the computing resource and proposed a regulation of the offload operation. Further data compression ademption is described at the last part of this chapter.

Chapter 5: The conclusion of this thesis

Chapter 2

Related Technologies

2.1. Efficient classification and detection network

The one-stage network-based target detection and recognition architecture has attracted much attention due to its high computational efficiency while ensuring the inference performance, such as the YOLO and YOLO9000 network models proposed by Redmon in 2016 and 2017. Both of these network models input the entire image into an effective backbone network, and then output the corresponding detection results, and realize real-time target detection and recognition. In 2016, Liu Wei, etc., proposed the single-shot detector (SSD) algorithm, which uses multi-scale feature maps to detect and recognize target objects of different sizes in a similar manner to a regional proposal network (RPN). In 2019, Wang Ming, etc., set Resnetv2-50 network and YOLOv2 network [13], removed the last pooling layer and full connected layer of the Resnet network, fused the shallow and deep features of the image, and realized the feature extraction by adding convolutional layers Dimensionality reduction processing. Although these one-stage network-based target detection and recognition models can quickly complete target detection and recognition tasks, their main limitation is that their positioning accuracy and recognition accuracy are generally lower than those of two-level detection networks. Because the target detection idea based on the two-level network is coarse positioning + fine classification, and the target detection idea based on the single-level network is direct positioning + classification, and there is no process of coarse positioning and screening.

Two-stage network target detection and recognition architecture also plays an important role on image processing, such as the Fast R-CNN network model

proposed by Ren, etc. In 2015, based on Fast R-CNN, the Region Proposal Network (RPN) is introduced to generate candidate frames. The introduction of RPN has achieved further acceleration, and real-time detection and recognition effects have been achieved on the GPU. In 2017, Lin et al. introduced the Feature Pyramid Network (FPN) structure into the basic Fast R-CNN and proposed the Fast R-CNN + FPN network structure. The fusion of FPN into Fast R-CNN greatly increases awareness of the full image information of the detector. In 2017, based on Fast R-CNN, He Kaiming, etc. added a branch for segmentation tasks to support instance segmentation and named it as Mask R-CNN [14] network model. In Mask R-CNN, ROI Align is used instead of ROI Pooling in Fast R-CNN. After ROI Align is used, the accuracy of the mask has been significantly improved. In addition, segmentation tasks, positioning and classification tasks are performed simultaneously. In the training process of the target detection and recognition model based on the two-stage network, the detection performance of the detector is often limited by imbalance. In order to solve this problem, in 2019, Pang Jiangmiao, etc. just proposed a balanced learning target detection framework-Libra R-CNN [15]. Libra R-CNN uses IOU-balanced Sampling to select representative candidate frames, uses Balanced Feature Pyramid to integrate feature pyramids of different sizes, and uses Balanced L1 Loss to balance positioning and recognition tasks.

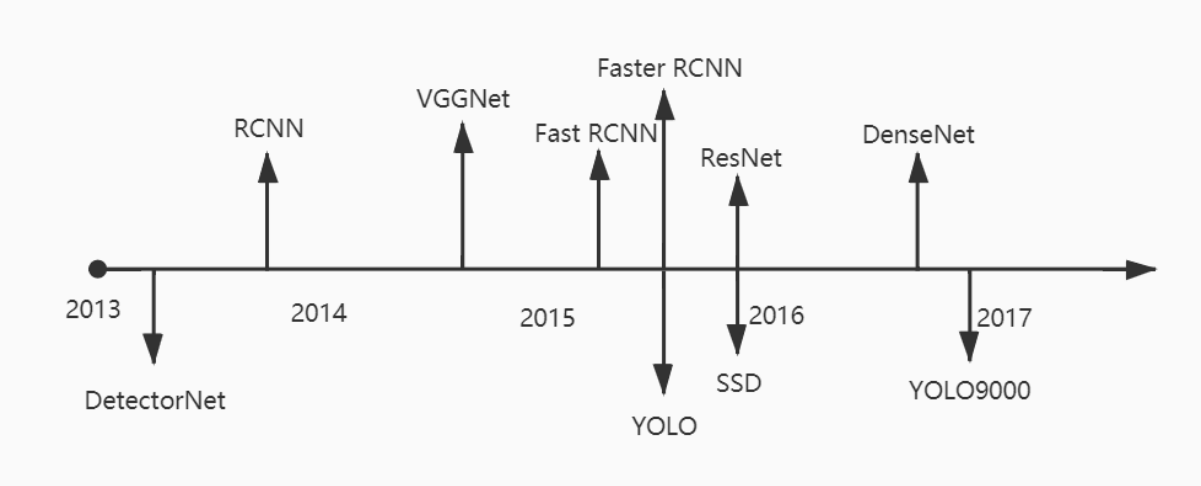

Fig. 1. Milestones in generic object detection based on deep learning

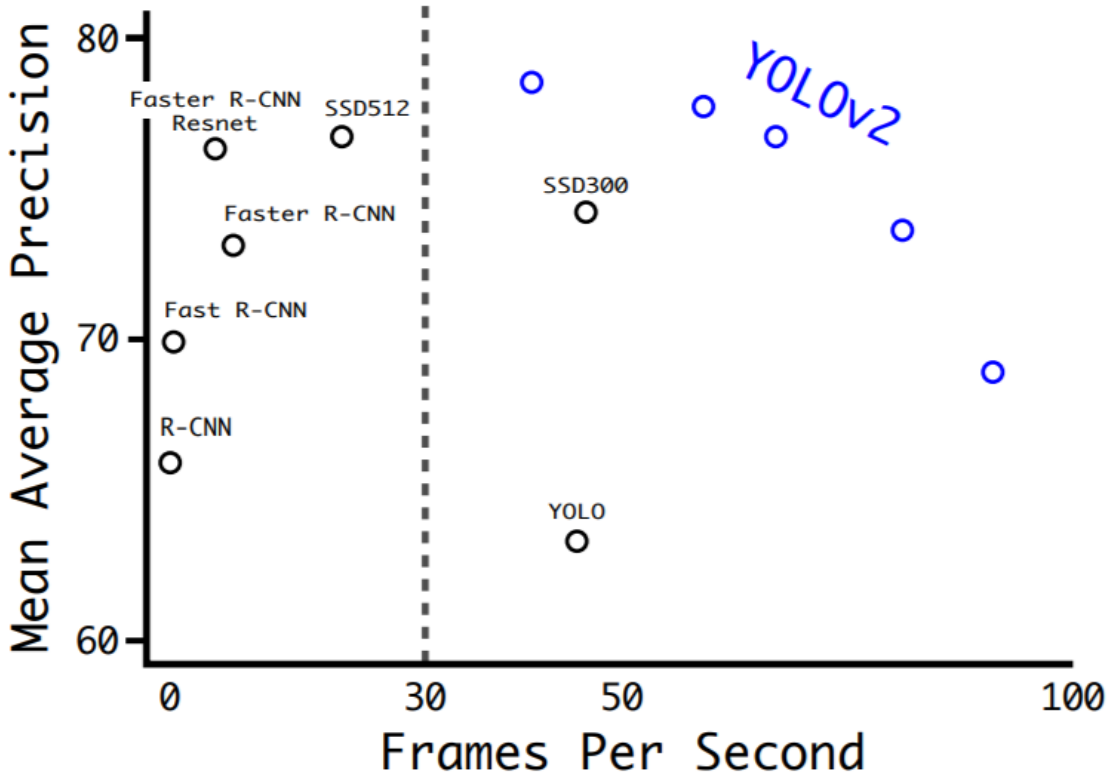

Fig. 2. Performance preview of several detection models on VOC 2007

2.1.1. BranchyNet

On September 2017, Surat Teerapittayanon, etc. from Harvard University designed a tool for students and researchers to use on their DNN model structures for fast inference called BranchyNet [16]. This is a greatly creative network architecture that use early branch exits to promote fast inference comparing with the running the original model.

The essential idea of BranchyNet is that the features learnt at the earlier layers of most DNN network model is already sufficient for some task, which indicates that the later layers might have little effects on performance improvement but only bring unnecessary burden of the time cost and computing resources. Through proper branching structures and exit points, BranchyNet shows how to balance the test

samples to be correctly processed and reach an earlier output. It should be typically pointed out that Branchy net also finish the whole network layers for those test data

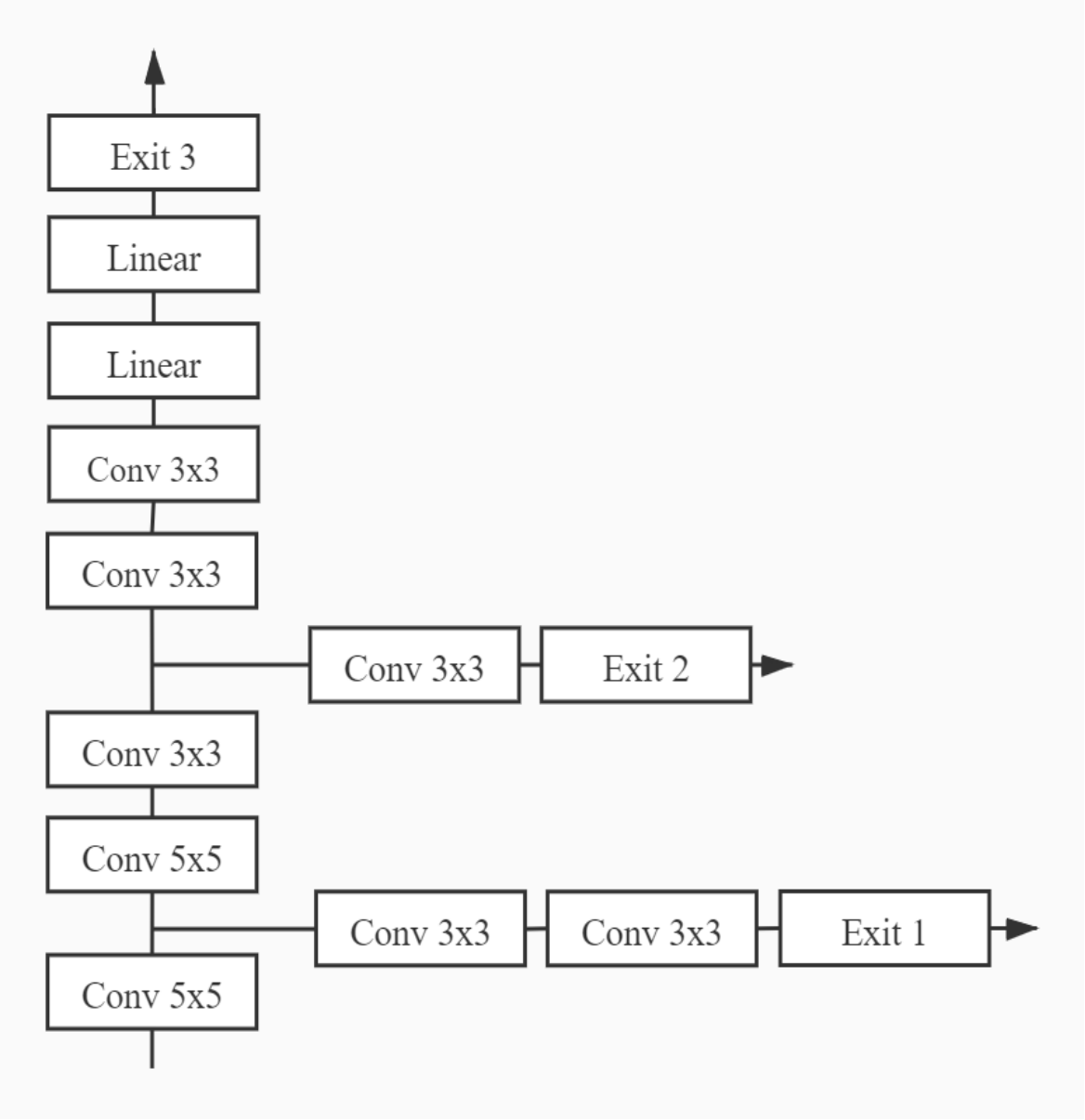

Fig. 3. A simple BranchyNet with two branches added to the baseline (original) AlexNet.

that requires high confidence prediction results and less frequency.

Branchynet executed a joint optimization with the weighted loss of all exit points to prevent overfit and the earlier branch exits can provide additional gradient back propagation in terms of improving confidence. Researcher can set one or multi exit points on different positions to fit the challenges of different training data. The

measure of confidence in classification prediction is softmax entropy at all branch exit, which also works as an easy adjustable threshold in order to handle classification tasks with different test data.

2.2. Model Cascading

The cascading structure [17] basically infers with a lightweight model and offload a part to a high-accuracy model to balance the trade-off between accuracy and processing speed. By performing most of the inference at the lightweight network, it can reach a high throughput, low latency, and reduce data transfer volume.

For example, when running a classification task, the image data will be processed by a fast lightweight network model and output a confidence score,

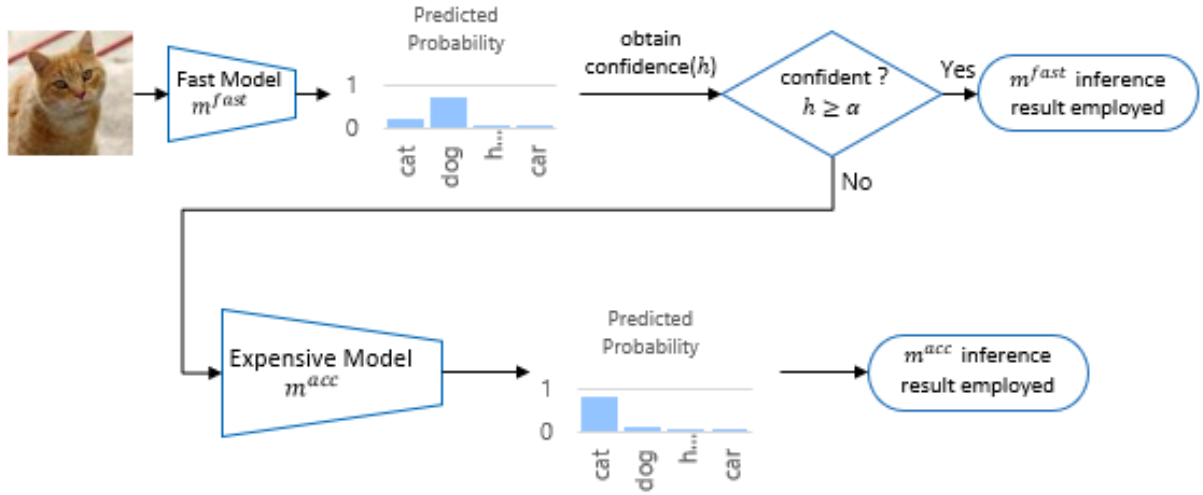

Fig. 4. An overview of a cascading structure system

when the result is not confident enough, the image data will be further transferred to an expensive model for better classification performance. The cascading structure can combine two model to reduce processing time significantly with the least accuracy loss.

2.3. Quantization

Over the years, the outstanding performance of deep learning in many fields has made it the mainstream direction of machine learning today, but its huge amount of calculation is still criticized. Especially in recent years, as the computing power of end devices has increased, more and more intelligent applications based on deep neural networks have emerged in the industry. In order to bridge the gap between computing power demand and supply, model compression has become one of the hot spots in the industry in recent years [18] [19].

Quantification can bring many benefits:

- **Increased computational efficiency**: Many processor integer calculation instructions are more efficient than corresponding floating-point calculations. Taking CPU as an example, the latency of floating-point arithmetic instructions will be longer on average than the corresponding integer arithmetic instructions, especially in early CPUs. For FPGAs, floating-point operations are even more troublesome. Moreover, the complexity of multiplication is usually proportional to the square of the operand bit-width, so reducing the representation accuracy can effectively reduce the complexity.
- **Memory and storage occupancy reduction**: quantization of the "slimming" effect on the model can be said to be immediate. The benefits it brings are twofold: First, it reduces the memory footprint. In many cases, the bottleneck of inference performance is not the calculation but the memory access. In this case, increasing the calculation density will have a significant optimization effect on time-consuming; second, saving storage space, reducing the size of the application, and facilitating software upgrade.
- **Reduce energy consumption**: Power consumption mainly comes from two parts: calculation and memory access. On the one hand, taking multiplication and addition operations as an example, the energy

consumption of 8-bit integers and 32 floating-point types can be orders of magnitude different. On the other hand, memory access is a major power hungry. Assuming that the model that can only be placed in DRAM can be quantized and placed in SRAM, it will not only improve performance, but also reduce energy consumption.

For typical floating-point to integer quantization, it essentially maps a certain segment in the real number domain to an integer. Noticed that different quantization function should be applied depending on the specific tasks. If linear quantization is used, its general form can be:

$$\boldsymbol{q = round(s * x + z)} \tag{3.1}$$

Among them, $\boldsymbol{x}$ and $\boldsymbol{q}$ are the numbers before and after quantization, $\boldsymbol{s}$ is called scaling factor, and $\boldsymbol{z}$ is called zero point which is the quantized value of 0 in the original value range. There will be a lot of 0 in weight or activation (such as padding, or after ReLU), so we need to make the real number 0 can be accurately represented after quantization.

The quantization approach is also divided into uniform (Uniform) quantization and non-uniform (Non-uniform) quantization. The simplest one is to make the distance between the quantized levels equal. This type of quantization is called uniform quantization. But there will be more information loss, because in general, there must be some areas in the dynamic range that are dense, and some are sparse. Correspondingly, there is non-uniform quantization which refers to the unequal length between quantization levels, such as log quantization. Also, through learning to get a greater degree of freedom mapping (which can be represented by a lookup table) is an advance approach. Intuitively, non-uniform quantization seems to be able to achieve higher accuracy, but its disadvantage is that it is not conducive to hardware acceleration [20].

There are many options for the number of quantization bits. It can be roughly divided into several categories:

- Float16 quantization is a safer approach. In most cases, there is a significant performance improvement without losing too much precision. Because they are all floating point, it is relatively easy.
- Int8 is relatively common and relatively mature. There are many related studies, and various mainstream frameworks are basically supported.
- There are relatively more academic circles about int8 quantization, and little industrial support. Below 8 bits are mainly 4, 2 and 1 bit. If the accuracy is as low as 1 bit, that is, binarization, it can be calculated by bit arithmetic. This is very friendly to the processor. [21]

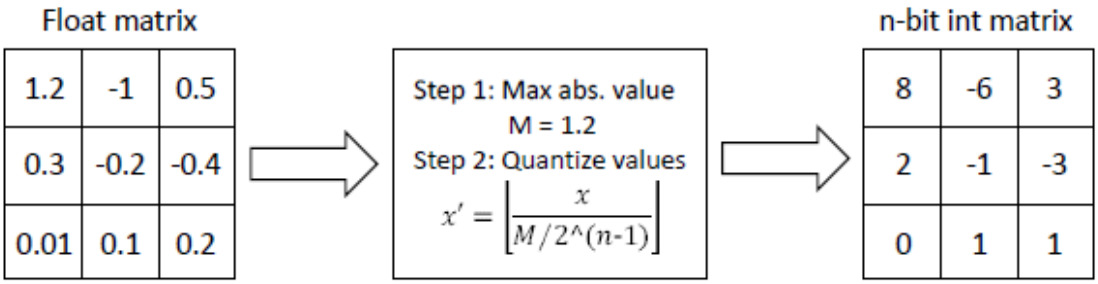

Fig. 5. An example of n-bit quantization proposed by SeerNet (e.g. n=4)

Chapter 3

ECNet system and model designment

3.1. Overall framework

When facing some real-time classification and detection tasks, some mainstream DNN model that is able to fit the accuracy requirement always have problems to be embedded on IoT devices. On the other hand, when we only rely on light-weighted neural network, the inference performance is difficult to meet the accuracy requirement. To solve this problem, especially when running a real-time video data processing such as human detection on a surveillance camera video data, we need to design a network system that can not only ensure the inference performance, but also have low time or computing requirement. We proposed a network system that can process relatively easy tasks on the edge-side network which is deployed on some IoT devices, and link to a cloud-side DNN network to solve the difficult tasks of detection and classification.

With the essential idea of cascade structure, we present a novel edge-cloud network system：ECNet, which is augmented with additional branch exit that can output feature maps from specific layers to be transferred to the cloud-side network. It consists of a relatively lightweight CNN network as the edge-side network, and a deep network as the cloud-side network for further inferring. We also designed an offload controller to achieve a real-time offload rate optimization with the specific threshold that can be adjusted on demand of the real-time tasks. Due to the limitation of the network bandwidth, the feature map should be compressed before transmission. We take the usage of quantization to accelerate our edge-cloud network system with a dequantization module deployed.

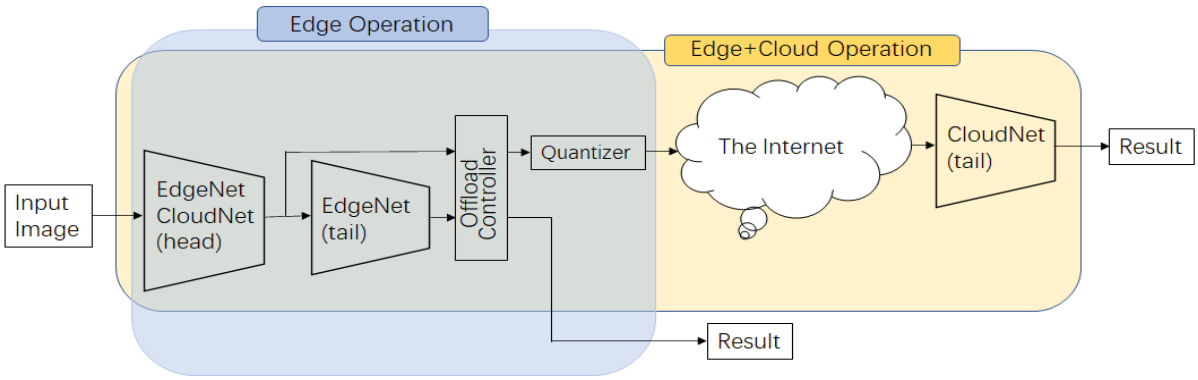

Fig. 6. General framework of ECNet

3.2. Edge-side and cloud-side structure

To fit the interest of the edge-side and cloud-side network, we must make our choice with a big number of candidate networks. The edge-side network should be a lightweight CNN model that can be deployed on an IoT device for real-time inference. The cloud-side network should be mainstream DNN model that can handle most of the detection and classification tasks. Based on these essential conditions, we have several candidates network for edge-side and cloud-side as shown in the chart.

Method	mAP (%)	FPS
Faster R-CNN	72.70	11.23
YOLO	62.52	42.34
YOLOv2	73.82	64.65
SSD300	74.18	58.78
SSD512	76.83	27.75
DSSD(ResNet-101)	76.03	8.36
R-SSD300	75.02	43.78
R-SSD512	77.73	24.19
RefineDet320	76.97	46.83
RefineDet512	77.68	29.45
SIN	77.26	10.79
YOLOV3	88.09	51.26

Table. 1. COMPARISON WITH ALTERNATIVE NETWORKS UNDER UA-DETRAC DATASET

As the most well used network for object detection tasks, YOLOv3 [22] shows the best accuracy performance, and the earlier version YOLOv2, shows the fastest

processing speed. According to the comparison result of all these candidate network, we attempt to use Darknet19 (the backbone network of YOLOv2) as the edge-side network, and Darknet53 (the backbone network of YOLOv3) as the cloud-side network. As the principle of our network system, the further issue is the feature map transmission from the edge-side to the cloud, which requires the cloud-side network to fully accept the extracted feature maps from the edge-side. Thus, we propose to cut the Darknet53 into two parts: the head part and the tail part. The head part network ensures the tensor size of the output feature which needs to fit the size that the tail part network can flatly accept.

Type	Filters	Size	
Convolutional	32	3×3	
Convolutional	64	3×3/2	
Convolutional	32	1×1	
Convolutional	64	3×3	**×1**
Residual			
Convolutional	128	3×3/2	
Convolutional	64	1×1	
Convolutional	128	3×3	**×2**
Residual			**Branch exit**
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			

Table. 2. EDGE-SIDE NETWORK STRUCTURE

Type	Filters	Size	
Convolutional	32	3×3	
Convolutional	64	3×3/2	
Convolutional	32	1×1	
Convolutional	64	3×3	×1
Residual			
Convolutional	128	3×3/2	
Convolutional	64	1×1	
Convolutional	128	3×3	×2
Residual			
Convolutional	256	3×3/2	
Convolutional	128	1×1	
Convolutional	256	3×3	×8
Residual			
Convolutional	512	3×3/2	
Convolutional	256	1×1	
Convolutional	512	3×3	×8
Residual			
Convolutional	1024	3×3/2	
Convolutional	512	1×1	
Convolutional	1024	3×3	×4
Residual			
Avgpool		Global	
Connected		1000	
SoftMax			

Table. 3. CLOUD-SIDE NETWORK STRUCTURE

As is shown in the Table .2, we set up the branch exit at the layer after the third residual block, and the part before the branch exit is chosen to be the head part of the proposed edge-side and cloud side network. This kind of distributed structure is challenging for several consideration including:

- The structure of DarkNet53 is built on numerous residual blocks, and each of residual block contains successive 3 × 3 and 1 × 1 convolutional layer connected by one shortcut connection [23]. 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 network 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.

We provide additional simulation experiments on key aspects of the edge-side and cloud-side of ECNet. We use 10 classes ImageNet dataset for experiment (10000 images for training and 3000 images for testing). For updating the weight parameters of the full-connected layer of CNN, image data reshape is necessary. In our experiment, we center crop the image data into 224×224.

ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). Images of each concept are quality-controlled and human-annotated.

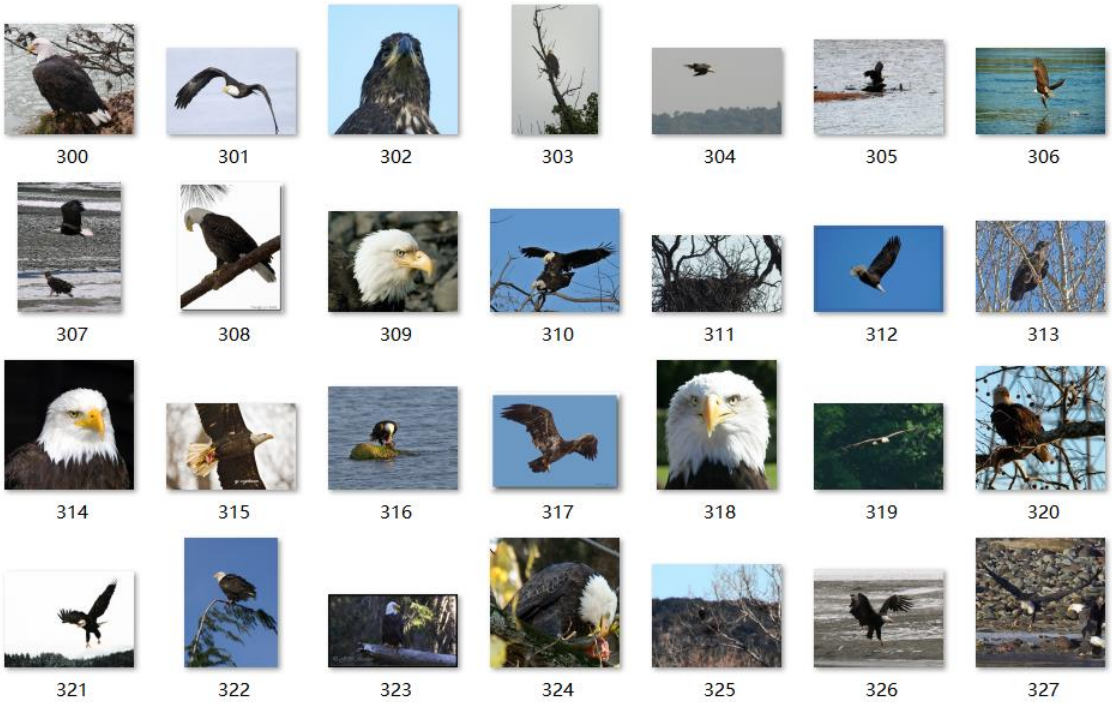

Fig. 7. Preview of the image data (class Haliaeetus leucocephalus) [24]

The training environment is NVIDIA GeForce RTX 2070 SUPER 8G (GPU), and AMD Ryzen 3600 6Core (CPU).

	Rank-1(%)	***Rank-5(%)***	***Processing Time(s/frame)***
Edge-side	68.5	81.8	0.013
Darknet19 (YOLOv2)	64.3	76.4	0.006
Darknet53 (YOLOv3)	81.2	98.2	0.023

Table. 4. CLASSIFICATION PERFORMANCE COMPARATION

The result shows that the designed Edge-side network has considerable classification performance, which is better than the original Darknet 19, and the less processing time per frame than Darknet53. We can frankly lead to the conclusion that our edge-side and cloud-side network makes good contribution for the proposed ECNet edge-cloud system. Guided by aforementioned considerations, the structure of edge-cloud network is designed after several times of trials and simulations.

Chapter 4

Offloading control & feature map compression

4.1. Measure of confidence score and offloading control

The essential concept or the ECNet is to let the edge-side network handle the inference that is relatively easy and transfer the hard tasks to the cloud-side network for exacter calculation. Specifically, the edge-side work will offload the feature maps output from the head part network to the cloud, according to the offloading regulation. Setting up the offload controller remains to be the second stage of the proposed ECNet system. In the task of classification, we use entropy of a classification result (e.g., by SoftMax) as measure [25]. The entropy can work as a confidence score in the simulation experiment to evaluate the performance of the offload controller. When the classification result stands with high confidence, the value of entropy will be small, which indicate that the image is not hard to be classify. Otherwise, the value of entropy will be large and shows that the corresponding image data is hard to process, that should be transferred to the cloud-side network. Users can set up a threshold for determine the part of feature maps to be transferred to the cloud-side network. Entropy is defined as

$$\mathbf{entropy}(\boldsymbol{y}) = -\sum_{c \in \mathcal{C}} \boldsymbol{y}_c \mathbf{log}\, \boldsymbol{y}_c \tag{4.1}$$

where $\boldsymbol{y}$ is a vector containing computed probabilities for all possible class labels and $\mathcal{C}$ is a set of all possible labels.

As a simulation experiment, we counted the entropy distribution of the sample feature map data output from the head part network branch exit.

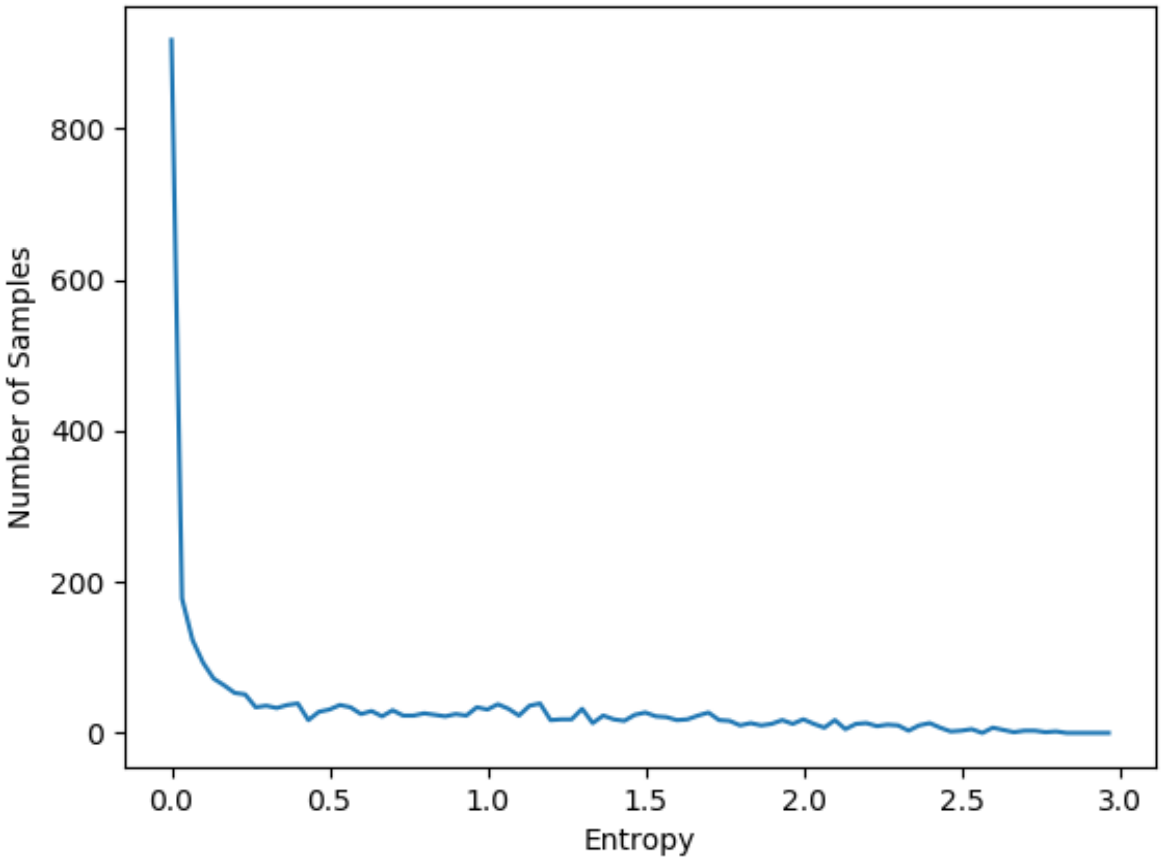

Fig. 8. Entropy distribution of feature maps

Entropy of each sample from our test dataset is counted to confirm the entropy distribution as shown in Fig.9. most of feature maps outputted from edge-side has less entropy than 0.5, which confirmed the relatively reliable classification performance of our designed edge-side network. The part above 0.5 tends to be sparse distributed, which shows better classification ability is expected on the cloud-side network.

4.2. Feature map quantization

Since the feature map data is the result after convolution, the data size of the output feature w_{out} can be calculate as:

$$\boldsymbol{w_{out}} = \frac{\boldsymbol{w_{in}} - \boldsymbol{F} + \boldsymbol{1}}{\boldsymbol{stride}} \tag{4.2}$$

where w_{in} is the size of the input feature map, F is the size of the convolution kernel and $stride$ is the convolution stride. For decreasing the complexity of the

convolution, we tend to choose small size of the convolution kernel [26], which will expand the size of the output feature map. As a result of our simulation experiment, the size of the original image data from the 10-class classification ImageNet dataset is 147KB, but the size of the output feature map has expanded to 2.16MB. In the case, when the internet bandwidth is not that satisfactory, it is a huge burden to transfer the output feature map in such a big size to the cloud-side network. The unacceptable time cost will become the primary factor that affects the real-time nature of the proposed ECNet system. Thus, when the internet bandwidth is determined, we need to compress the output feature map to reduce the time cost on feature map transmission. On the other hand, the data compression method should have less affection on the inference performance. Taking all these factors into consideration, we propose to set up a quantization and dequantization module between the edge-side and cloud-side network.

Quantization is the process of constraining an input from a continuous or otherwise large set of values (such as the real numbers) to a small set of finite discrete set (such as the integers). In the context of simulation and embedded computing, it is about approximating real-world values with a digital representation that introduces limits on the precision and range of a value. As an extended use quantization method for CNN, int8 quantization [27] has become the first choice that we can use here in the ECNet system. For figuring out the best way of quantization on our 10-class classification task, we have made experiments on all the candidate quantization bit, of which the formula comes:

$$\boldsymbol{Q} = \begin{cases} \mathbf{0} & \boldsymbol{Q} \leq \mathbf{0} \\ \frac{\boldsymbol{O}}{\boldsymbol{C}} * \mathbf{2}^{\boldsymbol{x}} & \mathbf{0} \leq \boldsymbol{Q} \leq \mathbf{2}^{\boldsymbol{x}} - \mathbf{1} \\ \mathbf{2}^{\boldsymbol{x}} - \mathbf{1} & \mathbf{2}^{\boldsymbol{x}} - \mathbf{1} \leq \boldsymbol{Q} \end{cases} \tag{4.3}$$

Where $\boldsymbol{Q}$ is the quantized value, $\boldsymbol{O}$ is the original value, $\boldsymbol{C}$ is the sample constant depending on the dataset (for our 10-class ImageNet dataset is 30), and $\boldsymbol{x}$ is the chosen quantization bit.

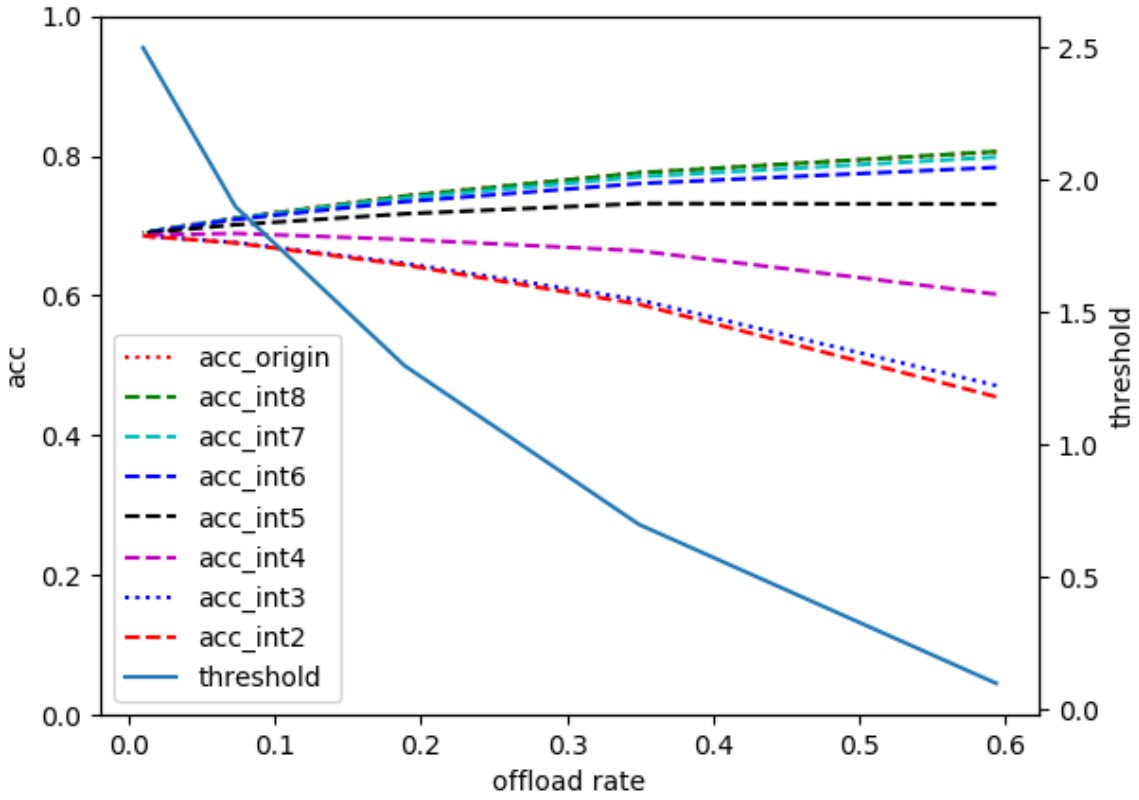

Fig. 9. Total accuracy with different quantization bits and offload control by threshold operating

We evaluated the overall accuracy of ECNet for varying entropy threshold with different quantization bits. As shown in Fig.10. (overlap existed between the green dotted line and the original result), the int8 quantization has the least accuracy drop, which shows that int8 quantization almost doesn't affect the classification performance, for our simulation dataset. The accuracy drops with the quantization bit decreasing, which is reasonable because when less quantization bit means more lose of data information. After the int5 quantization, the compression leads to a serious accuracy drop, which shows that it cannot be a choice to use quantization bit less than 5 for our test dataset. The solid line in blue shows how operating of threshold affects offload rate. The threshold and offload rate satisfy negative correlation, more feature maps will be transferred if we set the threshold of entropy less. The accuracy tends to have little improvement when the offload rate reaches to

0.5. Thresholds should be chosen when it satisfies the latency requirement while maintaining accuracy requirement.

To simulate making the best choice of the quantization bit when facing different classification tasks, we have also counted the entropy of the feature map data after different quantization approach.

Quantization bit	Entropy (bit)
2	0.569
3	1.435
4	2.001
5	2.388
6	4.292
7	5.213
8	6.113

Table. 5. ENTROPY OF FEATURE MAP DATA BY ALL QUANTIZATION APPROACH

According to the Fig.11. and the Table.5., the entropy of the feature map shows a lift, and the accuracy value shows an unacceptable drop after int5 quantization bit. This shows that the average information amount of the feature maps under int5 quantization is relatively less. On the other hand, it does not have a huge effect on the accuracy score. Thus, we should consider int5 quantization approach under this simulation environment

4.3. Further compression approach assessment

Thanks to the quantization operation and offload controlling based on threshold operating, we can limit the transmission burden with little accuracy loss comparing with the original network. Since most real-time tasks have strict transmission demand, we still consider the offload performance could be unsatisfactory for real scenario, which lead to the idea of compression on feature maps after quantization to further improve energy-efficiency and throughput.

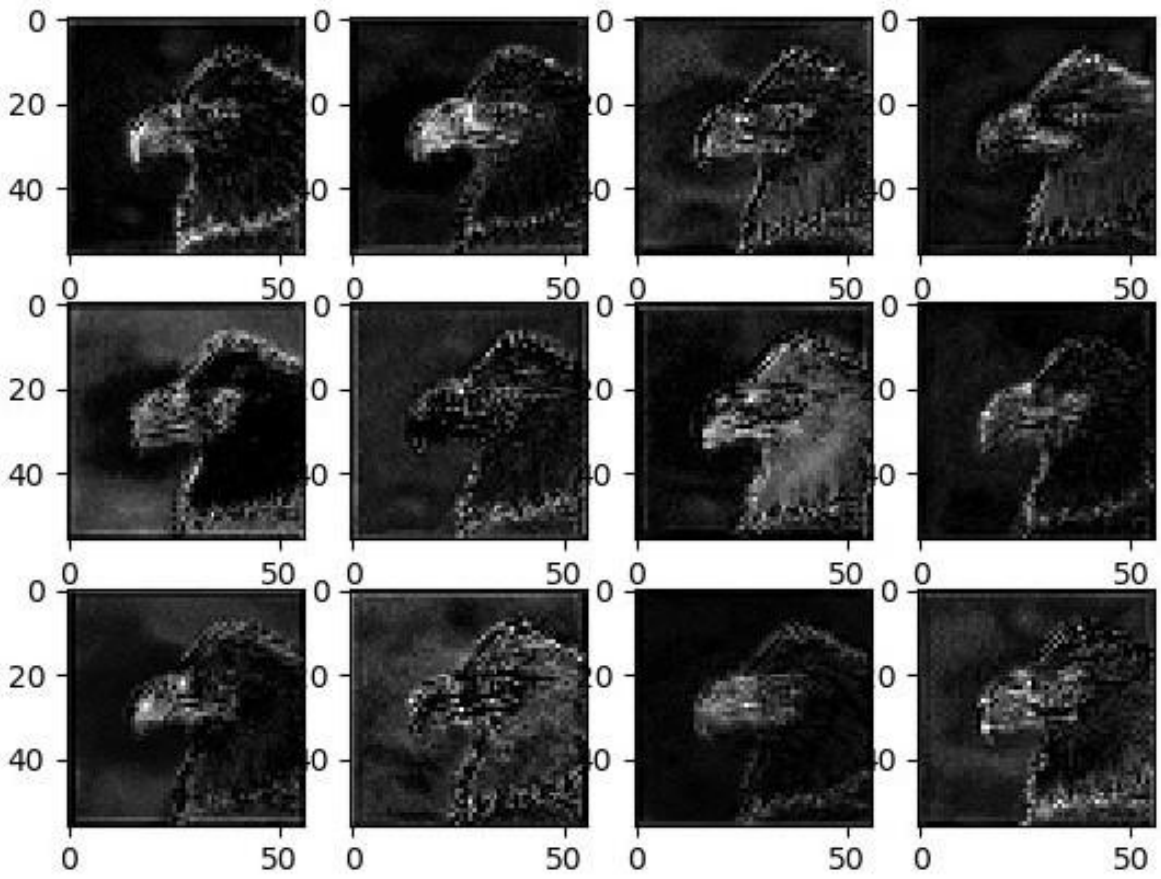

Fig. 10. Visualization of feature maps at exit point by channels

According to the visualization of the feature maps that output from the edge-side branch exit as Fig.11. shows, it seems to be less likely to apply differential pulse-code modulation (DPCM) [28] solution in our task, since the degree of sparsity and similarity of feature data hardly meet our need, not only between channels but also between raw or column in channel. Besides, we analyze the data variation of the quantized feature maps.

	Raw data	***DPCM between channels***	***DPCM in channel***
Number of symbol	98.4	171.8	151.4
Entropy	4.85	5.73	4.90

Table. 6. COMPARISON OF INFORMATION VOLUME

Table.6. gives the result of data analysis of quantized feature map, feature map after DPCM applied between channels and feature map after DPCM applied in channel. We use 1200 images for calculation and applying DPCM in channel for the element of each column. The result shows that DPCM operation cannot lead to a reduction of the entropy which means it cannot help reduce the amount of information in feature map. We start our attemption to find out other compression approach.

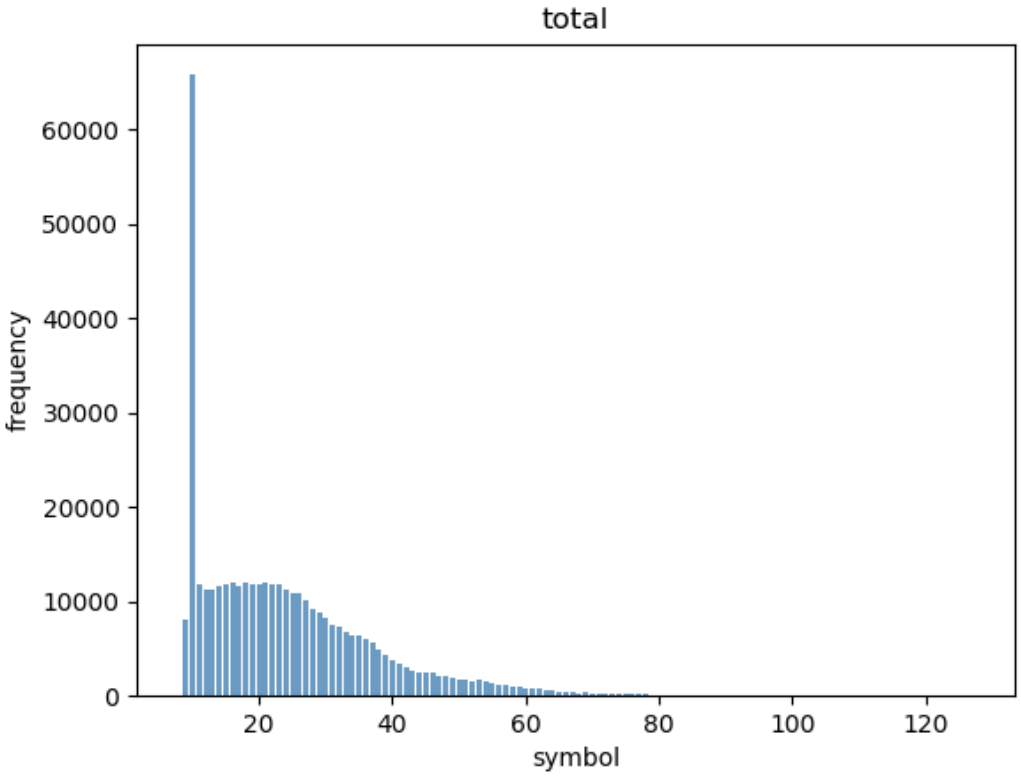

Fig. 11. Data variation of the feature map in single channel

Fig.12. shows the analysis of data variation of quantized feature maps on 100 images. Due to the int8 quantization function and the standard attributes of the ImageNet datasets we use during the simulation, the symbol of '10' has the highest amount of frequency while other symbol satisfying a Gaussian-like distribution as shown. According to this data distribution factor, we propose to apply lossless compression method on whole channels like Huffman coding and ZIP compression after quantization operation[29] [30].

	Origin	***Huffman coding***	***Optimized Huffman coding***	***ZIP***
Compression ratio	1	1.62	1.50	1.68
Time cost(s/frame)	0.001	0.98	0.35	0.009

Table. 7. COMPRESSION APPROACH EVALUATION

The improvement of optimized Huffman coding [31] is using prior Huffman tree which calculate sufficient samples instead of setting Huffman tree for every feature map. Since the operation of Huffman coding is not based on matrix operation which means the encoder and decoder have to run on the whole dataset at least once a time, the time cost is increasing dramatically. As shown in Table.7., ZIP compression method [32] has better performance both on compression ratio (compression ratio is defined as the ratio between uncompressed size and compressed size) and time cost factors. The ZIP compression could reach to 1.68 compression ratio with 0.009 second each frame, however, the time cost is still a major concern.

Besides the method we have mentioned, we also continuing making research on other compression method such as principal component analysis (PCA) [33]before coding, compression in neural network and JPEG compression [34] for monochrome images. We expect further improvement on feature map compression task and completely settle the problem of time cost limitation.

Chapter 5

Conclusion and future work

5.1. Conclusion

In this paper, we proposed ECNet edge-cloud network system with designed edge-side and cloud-side network.

The edge-side network is lightweight CNN network that is able to process most of the inferring tasks. An branch exit is set for transmitting quantized feature maps to the cloud, administrated by the offload controller with entropy as threshold. The improvement of ECNet is leveraged by reaching a balance between processing time and accuracy performance with reducing transmission cost down to 25% when using int8 quantization approach. This system has been evaluated on classification tasks and chose proper quantization bit based on experiments. We also evaluated several data compression method for further improving on saving the time cost and internet bandwidth.

5.2. Future work

For further improving the accuracy performance, we believed that applying in transfer learning on the cloud-side network is a possible solution. Based on the 10-class ImageNet dataset, we collected the quantized feature maps and build a new dataset for retraining. We are looking forward on getting a better result after retraining on the feature map dataset. Also we are planning to adapt the ECNet to detection tasks, and make evaluation in actual scenarios

Chapter 6

Appendix

6.1. List of academic achievements

International conference:

Libo Hu, Tao Wang, Hiroshi Watanabe, Shohei Enomoto, Xu Shi, Akira Sakamoto and Takeharu 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:

Libo Hu, Tao Wang, Yucheng Zhou, Hiroshi Watanabe, Shohei Enomoto, Xu Shi, Akira Sakamoto, and Takeharu Eda: "Transfer Rate Estimation in Edge-Cloud Neural Network Solution for Object Detection", IEICE General Conference D-11-20, Mar. 2020

Libo Hu, Tao Wang, Hiroshi Watanabe: "Two-side Network for Person Detection and Person Re-identification", 2019 Picture Coding Symposium・2019 Image Media Processing Symposium (PCSJ/IMPS2019), P-4-08, Nov. 2019

Bibliography

[1]. Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), pp. 886-893, Jan.2005

[2]. Wang X, Han T X, Yan S. An HOG-LBP human detector with partial occlusion handling[C]//2009 IEEE 12th international conference on computer vision. IEEE pp. 32-39, 2009

[3]. Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision.pp. 1440-1448, 2015

[4]. Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. arXiv preprint arXiv:1506.01497, 2015.

[5]. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition.pp. 779-788, 2016

[6]. Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. pp.7263-7271, 2017

[7]. Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, pp. 21-37, 2016

[8]. Kong T, Yao A, Chen Y, et al. Hypernet: Towards accurate region proposal generation and joint object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 845-853, 2016

[9]. Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117-2125, 2017

[10]. Choi, H., & Bajić, I. V. Deep feature compression for collaborative object detection. In 2018 25th IEEE International Conference on Image Processing (ICIP) pp. 3743-3747, Oct. 2018

[11]. 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,” ACM Workshop on Hot Topics in Networks (HotNets2018), pp.50-56, Nov. 2018

[12]. Ko, J. H., Na, T., Amir, M. F., & Mukhopadhyay, S. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. In 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) pp. 1-6, Nov. 2018

[13]. Wang L, Yang S, Yang S, et al. Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network[J]. World journal of surgical oncology, pp. 1-9, 2019

[14]. He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. pp. 2961-2969, 2017

[15]. Pang J, Chen K, Shi J, et al. Libra r-cnn: Towards balanced learning for object detection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 821-830, 2019

[16]. Teerapittayanon S, McDanel B, Kung H T. Branchynet: Fast inference via early exiting from deep neural networks[C]//2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, pp. 2464-2469, 2016

[17]. Li H, Lin Z, Shen X, et al. A convolutional neural network cascade for face detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5325-5334, 2015

[18]. Gray R M, Neuhoff D L. Quantization[J]. IEEE transactions on information theory, pp. 2325-2383, 1998

[19]. Gray R. Vector quantization[J]. IEEE Assp Magazine, pp. 4-29, 1984

[20]. Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[21]. Cao S, Ma L, Xiao W, et al. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11216-11225, 2019

[22]. Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[23]. He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition pp. 770-

778, 2016

[24]. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, pp. 1097-1105, 2012

[25]. Rényi A. On measures of entropy and information[C]//Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics. The Regents of the University of California, 1961.

[26]. Haussler D. Convolution kernels on discrete structures[R]. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

[27]. Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2704-2713, 2018

[28]. Schniter P. An Introduction to Source-Coding: Quantization, DPCM, Transform Coding, and Sub-band Coding[M]. Connexions, Rice University, 2009.

[29]. Chmiel, B., Baskin, C., Banner, R., Zheltonozhskii, E., Yermolin, Y., Karbachevsky, A., ... & Mendelson, A. Feature map transform coding for energy-efficient cnn inference. arXiv preprint arXiv:1905.10830. 2019

[30]. Cavigelli, L., Rutishauser, G., & Benini, L. EBPC: Extended bit-plane compression for deep neural network inference and training accelerators. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9(4), pp. 723-734, 2019

[31]. Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[32]. Boopathi G, Arockiasamy S. An image compression approach using wavelet transform and modified self organizing map[J]. International Journal of Computer Science Issues (IJCSI), pp. 323, 2011

[33]. Abdi H, Williams L J. Principal component analysis[J]. Wiley interdisciplinary reviews: computational statistics, pp. 433-459, 2010

[34]. Rabbani M, Joshi R. An overview of the JPEG 2000 still image compression standard[J]. Signal processing: Image communication, pp. 3-48, 2002