Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему EFFECTIVE NEURAL NETWORKS IMPLEMENTATION FOR LOW POWER DEVICES

Виконав: студент 2 курсу, групи КНТ-612м_ін
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Спеціалізовані комп'ютерні системи»

YANG ZHAO
(ПРІЗВИЩЕ та ініціали)
Керівник ILYASHENKO M.B.
(ПРІЗВИЩЕ та ініціали)
Рецензент KAPLIENKO T.I.
(ПРІЗВИЩЕ та ініціали)

2023

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти магістерський

Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.
_____
"____" _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

YANG ZHAO
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) 3D models development for virtual reality computer system

керівник проєкту (роботи) PhD, associate professor, ILYASHENKO Matviy
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від "24" жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) single Board Computer

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Overview of the subject area;
2) Overview of technologies used on low-power equipment;
3) Development of algorithms and training strategies for low-power equipment;
4) Implementation of algorithms on low-power equipment.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ILYASHENKO M., associate professor		
нормоконтроль	SHCHERBAK N., senior lecturer		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Subject Area Review and Requirements Statement	10.10.2023 р.	
	For low-power equipment		
2	Designing algorithms and training methods Neural	25.10.2023 р.	
	networks on low-power equipment		
3	Implementing the system on a single-board computer	10.11.2023 р.	
4	System Research	15.11.2023 р.	
5	Drawing conclusions from the results obtained	27.11.2023 р.	
6	Registration of the obtained results in the software	02.12.2023 р.	
7	Design of auxiliary material	10.12.2023 р.	

Студент Yang Zhao YANG ZHAO
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проєкту (роботи) Matviy ILYASHENKO
(підпис) (ім'я, ПРІЗВИЩЕ)

ABSTRACT

EN: 72 p., 38 pictures, 11 tables, 69 references

LIGHTWEIGHT NETWORK, LOW-POWER EMBEDDED DEVICES, MODEL PRUNING, MODEL QUANTIZATION, TARGET DETECTION

In recent years, deep learning algorithms have been widely used in low-power devices, such as unmanned driving, security monitoring and other scenarios. Therefore, how to improve the existing network models of low-power devices and optimize the forward computing framework of efficient neural network is a hot field of research on low-power devices. This dissertation is a related study on lightweight target detection algorithm based on low power and low computing equipment. The main research contents are to improve the lightweight feature extraction network, model compression (parameter pruning, model quantification) and the deployment of detection algorithm on embedded devices.

First, the optimal lightweight classification network is selected, and the feature extraction network in the target detection algorithm is replaced. The new detection network is annotated and trained, and the parameter pruning is completed to shorten the forward inference time under the premise of minimizing the accuracy loss. Then, the pruned target detection model is transplanted to the embedded computation platform. Finally, by using an improved lightweight network model, parameter pruning, and model quantification, multiple objects are detected on low-power embedded mobile devices. The lightweight objective detection algorithm in this paper is improved by the lightweight MobileNet V3 Large network and the YOLO V3 algorithm. After the model was obtained by network training, random nonlinear pruning and int8 quantification processing were performed to accelerate the forward inference to transform the model, and to flexibly select according to the device environment. The final average size of the final model was 8.2MB.

The innovation point of this study is the redesign of the deep learning-based object detection optimization method and the application of three low-power embedded computing platforms, namely Raspberry Pi 3B +, Jetson Nano and KendryteKD233. First, the search engine and the existing public data sets are used to collect and expand a variety of data, such as VOC data set, Oxford-17 class data set, concrete crack data set, Marine biology data set, etc. Then, using the LabelImg software on the computer to complete data annotation, then complete the model training and model pruning, finally use forward reasoning to accelerate the model to three embedded computing platform and test, the results show that the average single frame prediction time on NVIDIA JetsonNano, Raspberry Pi 3B+ and KD233 development suite is 165ms, 40ms, 35ms, respectively, based on the average accuracy on the four data sets is 84.02%. The optimized target detection algorithm basically achieves the effect of real-time operation on low-power embedded devices.

CONTENTS

INTRODUCTION

The 21st century is not only the era of big data, but also the era of artificial intelligence. The explosive growth of data and the huge increase in hardware computer computing power have accelerated the rise of artificial intelligence, especially in computer vision. In addition, many excellent convolutional neural network algorithms introduce complex hierarchical structure to obtain the semantic information of features. While improving the accuracy, the depth information and width information of the convolutional neural network structure become more complex, and the number of parameters of the model is larger, which is accompanied by huge consumption of hardware resources. The progress of deep learning algorithms has also promoted the development of the field of computer vision. Computer vision technology simulates the cognitive process of human vision, and processes the input images to extract useful information, so as to judge the objects and scenes in the image.

The reason why object detection becomes a widely studied problem in the field of computer vision is because it plays an important application value. The task of target detection is to determine the category and position of each object in the input image. With the performance improvement of CNN [1] (Convolutional Neural Networks), VGG [2] (Visual Geometry Group Network), Inception [3], ResNet [4], GoogleNet [5] and other network algorithms, For the two-stage algorithms represented by Faster R-CNN [6] and the classic YOLO series algorithms in single-stage algorithms, a large number of excellent target detection algorithm models have been generated. For example, the Cascade R-CNN algorithm, which introduces the cascade structure to alleviate the overfitting problem, and the single-stage target detection algorithm without prior box proposed by Baidu. In addition, the application of target detection algorithms has been very common in daily life, especially in the smart home Internet of Things, unmanned driving, unmanned aerial vehicle control and other fields. Target detection technology to promote the development of

automatic driving technology in a variety of vehicles, especially in unmanned cars, target detection technology is used to sense, obstacles, pedestrians and a variety of complex road environment information, unmanned vehicles through the camera using laser radar for environmental images, send it back to the master program and help cars understand the environment, finally achieve driving.

1 ANALYSIS OF THE STUDY

1.1 Significance and motivation of the study

Intelligent Internet of Things appears more and more frequently in our lives, and more and more users have felt the convenience of life in some Internet of Things application systems composed of low-power embedded devices, such as fall detection, fire warning and agricultural monitoring. The addition of deep learning algorithm makes low-power embedded devices more intelligent, and there have been baidu whole-house intelligent Internet of Things products with Xiaodu students as the main line. However, the existing deep learning-based target detection models, although the actual detection accuracy is very high, reaching more than 90%. However, due to too many internal parameters of the model and too large structure, forward reasoning cannot be directly realized on the embedded platform, and remote servers often need to be used in practical applications. That is, the image first collected in the embedded device; then sent to the server through the network to detect the image; and the final detection result is transmitted back to the embedded device. This working mode for a wide range of applications, the use cost will increase, and due to network reasons, it will cause the delay of the results, and the poor stability.

With the development of low-power chips and related computing power integrated devices, deep learning has become a general trend in some convenient low-power devices. But low-power devices, especially some embedded development devices with AI chips, have very limited hardware computing power and storage space. If you can avoid the server and run the deep learning detection algorithm directly on the convenient low-power devices, it can not only reduce the use cost, but also increase the possibility of the device to be used in various complex environments. In addition, localization detection also reduces the data security of users by reducing the frequent transmission of data. Therefore, how to achieve the removal of the redundant structure of the existing detection algorithms, while

reducing the performance loss caused by the deredundancy. It is a very meaningful topic to be perfectly deployed on low-power devices and achieve high-precision detection.

1.2 Research questions

This paper studies the deep learning optimization algorithm based on low-power devices. The main problem of this paper is to design and realize a target detection algorithm with strong real-time operation and high reliability on a variety of low-power devices. It includes the following aspects:

Questions1. Study several lightweight deep learning classification network algorithms, analyze the structural characteristics of each network and its performance in practical application, and then analyze and pair the characteristics of the object detection algorithm based on deep learning. Finally, introduce the mainstream algorithm implementation mode of low-power devices in different application scenarios.

Questions2. Analyze and compare the characteristics of the existing lightweight network structure, and select an appropriate target detection algorithm to build a lightweight neural network model. The network model was used to train the model on the existing datasets. Then, the existing model pruning and model quantification methods are analyzed in detail, and the advantages and disadvantages of each method are compared, and finally the appropriate model compression method is selected. Through the current mainstream target detection algorithm YOLO V3 lightweight improvement and optimization for low power equipment.

Questions3. Lightweight target detection algorithm based on three embedded computing platforms of raspberry PI 3B +, Jetson Nano and Kendryte KD233 was designed and implemented. First, we based on YOLO V3 algorithm to optimize a suitable for low power equipment on lightweight target detection framework - YOLO

V3 + MobileNet V3, and then use the model pruning, model quantification model compression method to reduce the number of network model, then will train on the server good deep learning detection model deployed in raspberry pie 3B +, Jetson Nano, Kendryte KD233 three pieces of low power embedded devices. Finally, we transplanted the compressed YOLOV3 + MobileNetV3 model to the low-power device separately, and compared the frame rate, time, and accuracy respectively.

1.3 Overview of the thesis structure

The structure of this paper is divided into five parts, and the specific content is shown as follows:

In the first chapter, introduce the research problems and objectives of this topic, and briefly explain the research significance and motivation of this paper. In addition, the main research questions are highlighted and the full text structure is combed.

The second chapter introduces the low-power computing equipment in detail, mainly from the current technology needed to implement the deep learning algorithm for low-power devices. It mainly includes lightweight classification network, the implementation mode of mainstream target detection algorithm for low-power devices in different application scenarios, and common parameter pruning and model quantification technology, and analyzes the advantages and disadvantages of each technology.

The third chapter details the improved target detection algorithm for low-power devices. This paper mainly introduces the characteristics of the overall network structure and how to replace the original feature extraction network in YOLO V3 with Mobilenet V3 network for improvement. In addition, the optimization strategy and model compression method of target detection model training are also introduced, mainly including random nonlinear parameter pruning strategy, int8

fixed-point quantization model method and synchronous mode multiple GPU parallel training method.

In the fourth chapter, a variety of low-power devices and practical effect test experiments are conducted. Firstly, the characteristics of the three low-power devices and the environment construction process are introduced. Then, the data sets required for the experiment are compared and the characteristics of each data set are analyzed. Then briefly introduce the overall process of the experiment and show different scenarios to detect different objects (concrete cracks, different kinds of flowers, different kinds of Marine life and other objects in the VOC data set) and mark the object position and confidence, finally get four kinds of data set in the raspberry PI 3B +, Jetson Nano, KendryteKD233 three low power embedded equipment terminal on the running speed, accuracy, and the performance of the frame rate.

The fifth chapter summarizes, analyzes the shortcomings and further optimization, and discusses the future development direction of computer vision for low-power embedded devices based on deep learning.

2 RELATED TECHNOLOGIES BASED ON LOW-POWER EQUIPMENT APPLICATIONS

2.1 Research overview of target detection algorithm and lightweight network model

Different from the conventional classification network algorithm, the target detection algorithm can effectively frame the position of the object from the image while detecting the object category. Target detection algorithm is the most used in the existing computer vision task algorithm, mainly because the fall detection, face recognition, unmanned driving and other algorithms are targeted optimization based on the target detection algorithm. In addition, the technological improvement in China's security field has also promoted the application of more excellent target

detection algorithms in the civil and military fields, such as smart home monitoring system, common UAV obstacle avoidance, military UAV precision strike, etc. In the past two years, more and more research scholars have shifted their research focus to computer vision, and more and more high-precision and high-performance algorithms have been proposed.

At present, the existing target detection algorithms are roughly divided into feature-based target detection algorithms and regression position-based target detection algorithms. The former is mainly represented by Faster R-CNN [6], and the latter is mainly represented by YOLOV3 [28]. Many subsequent studies were optimized based on these two algorithms, and with the introduction of new structures such as residual network [4] and dense network [10], the gradient vanishing problem was solved. However, the introduction of most of the new structures has led to the greater complexity of the deep information of the convolutional neural networks, which further makes their computational complexity higher. The resulting problem is that the detection speed is slow, and the requirements for hardware resources are higher, that is, the detection speed of single input images on the server containing high-performance GPU has reached the level of seconds, such an algorithm is difficult to play a timely and effective role in practical application scenarios.

Lightweight networks are a collection of all networks with a small number of parameters, low computational complexity and short inference time. They are usually deployed on mobile and edge devices with low power consumption and low computing power. In addition, lightweight is a relative concept, and it is a heavyweight network, there is no clear boundary between the two. Backbone networks are mainly composed of deep learning-based object detection algorithms. Therefore, the depth of the backbone network also determines the complexity of the target detection algorithm. In order to enable the algorithm to detect in real time on low-power devices, the efficient MobileNetV1 network [11] was proposed in 2017, and the network model meets the real-time requirements of low-power embedded vision application engineering. Subsequently, Xception network [14] proposed the idea of extending extension depth to improve the model structure, realize network

lightweight, and be applied in Inception V3 network [15]. Subsequently, the MobileNetV2 [12] and MobileNetV3 [13] networks with a lower complexity of the model parameters were also proposed by the Google team.

In the same year, Megvii found that Xception [14] and ResNext classification network [16], and introduced 11 convolution, which led to the poor performance of the model in small networks. In order to reduce the impact of 11 convolution on the performance of lightweight network, a new lightweight network model ShuffleNet [17,18] came into being. In addition, in order to further improve the running speed of neural network and the accuracy loss, SqueezeNet [19], PeleeNet [20]and GhostNet [21] have been successively proposed. Among them, PeleeNet is a lightweight network variant based on DenseNet [10], which is mainly deployed to the embedded end. In addition, the Google team recently proposed the MnasNet model [22], which can be run directly on Pixel phones. The accuracy of ImageNet [1] classification task on pixel 1 phone reached 74.0 %, and it took only 76 milliseconds to detect a single image, which is 1.5 times faster than MobileNetV2.

Therefore, if you want to in driverless cars, intelligent Internet of things, robot areas such as deep learning algorithm implementation engineering application, must be in the power and hardware performance limited low power consumption embedded platform, to optimize the performance better lightweight detection algorithm, it deployed on the actual equipment and complete the test. The research goal of this paper is to study based on the above questions.

2.2 Lightweight classification network algorithm

In recent years, due to the rapid development of the Internet of Things, many low-power intelligent devices have been widely used in our lives. As one of the driving forces for the progress of animal Internet, deep learning lightweight network has become a research hotspot in the field of deep learning vision. Lightweight

network refers to those convolutional neural network with a small number of parameters and a fast running speed. The methods to reduce the model size and parameter complexity mainly include the following four categories：

Optimize network structure to realize lightweight: for example, packet convolution (group convolution for GConv) and separable convolution are introduced in Mobilenet V3 classification network to realize the balance optimization between network model accuracy and model complexity.

Model pruning: Large networks often have a certain redundancy structure, which reduces the number of network model parameters by cutting out the redundancy.

Quantification: Using Tensor RT/NCNN library to achieve quantification (generally using multi-value, int8 fixed-point quantification method, etc.), the model size is not only reduced by several times, but also can improve the operation rate on low power consumption equipment.

Knowledge distillation: Use large models to help small models learn and improve the accuracy of small models. Several common lightweight network structures are introduced next.

2.2.1 Mobile Net series algorithm

MobileNetV1 Network [11] is proposed to build a small and fast model to run perfectly on devices with far less computing power (smart home embedded development devices, drone low-power devices, etc.). Mobile Net V1 Differs from the traditional convolutional neural network in that the former uses deep separable convolution instead of the traditional convolution implementation model optimization to further improve the operation efficiency. In object recognition, ImageNet recognition, expression recognition, attitude detection and other problems, MobileNetV1 algorithm realizes model scale reduction without much loss of accuracy. However, the single structure of Mobile Net V1 network leads to the accuracy of the network not reaching the requirements of practical application. If the above optimization method can be integrated into ResNet, DenseNet and other high-precision networks, it will certainly be able to achieve the balance of network speed

and accuracy optimization, and the cost performance is higher.

Inspired by the idea of residual error, MobileNetV2 [12] network introduces improved residual structure based on MobileNetV1,which can flexibly realize dimension rise and fall conversion and gradient propagation, and greatly reduce the use of memory resources required for forward reasoning. Then the ReLU activation function after the Narrow layer (low dimension or deep layer) is removed, retaining the feature diversity while also improving the network performance. In addition, MobileNetV2 is a full convolutional network that can accommodate multi-size images. Under the condition of low accuracy calculation, the MobileNetV2 network uses the Relu6 activation function (the ReLU function with the maximum output value of 6) to improve the generalization performance of the model. ReLU, the output domain of the function is [0, +). If the activation range is wide, the low precision model cannot accurately describe the values within this range, resulting in a performance decline. MobileNetV2 The network structure is shown in Figure 2.1, where down sampling at step 2 can be used for deeply separable ones if down sampling is required.

The MobileNetV3 [13] network integrates the characteristics of various networks to realize the further reduction of the model and the control of accuracy loss. The main fusion methods include the deep separable convolution of MobileNetV1, the reverse residual idea of Mobile Net V2 network, and the lightweight attention model of Mnas Net. MobileNetV3 Is optimized for the first two versions of the Mobile Net network, first introducing the SE structure into the bottlenet structure and putting it after the channel-by-channel filter. The number of channels in the extended layer with SE structure reduces improved accuracy and lower computational complexity. Then, modify the tail structure of the Mobile Net V2, first move the 11 before the average pooling to the average pooling; before the feature input, feed the average pooling layer, reduce the size of the feature graph from 77 to 11, then feed the previous convolution to 11 to achieve the dimension increase, the computation required by the network decreases by 49 times. Both 33 and 11 of the previous spindle convolution are also directly deleted to become the

structure shown in the second row of Figure 2.2. Finally, the experiment shows that there is no loss of accuracy, and the calculation scale is indeed reduced.

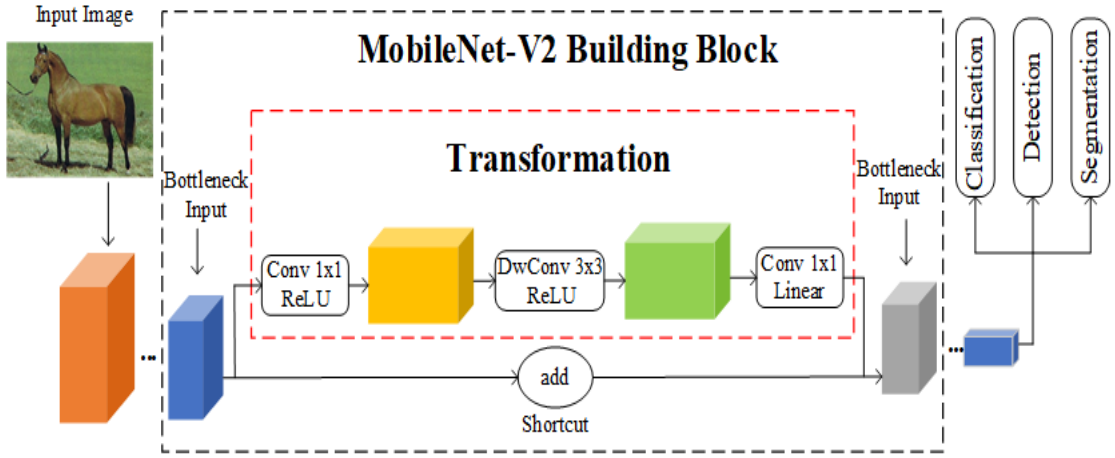

Figure 2.1 - MobileNetV2 network block structure

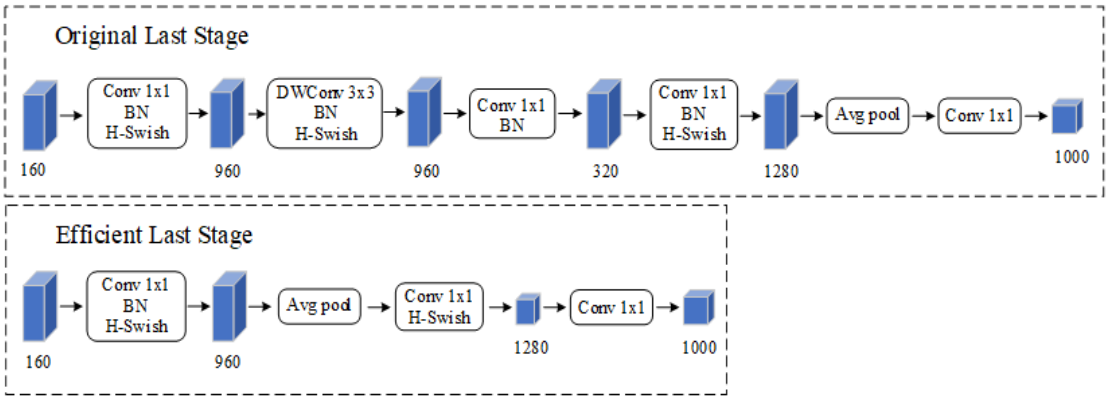

Figure 2.2 - Structural comparison before and after the final stage of the Mobile Net network optimization

In addition, in order to reduce the computing power requirements of the network, Mobile Net V3 not only reduces the number of convolution kernel channels at the front end of the network from the original 3233 to 1633, but also transforms the original swish activation function to the h-swish function. Wish, the function is composed of the relationship formula containing sigmoid activation function. Because the calculation complexity of sigmoid activation function is high, the MobileNetV3 network uses the ReLU6 function with high computational efficiency to improve the swish function, that is, to replace the sigmoid activation function, and

the final processed activation function is defined as h-swish function. The above measures not only improve the operation speed of the model, but also cause little loss of accuracy. In addition, the h-swish activation function is more significantly in the deep network.

MobileNetV3 There are two versions of Large and Small, both the number of bneck and the number of channels in the latter network. In the text, we adopted the Mobile Net V3 large version, whose structure is shown in Table 2.1.

Table 2.1 - MobileNetV3 Large Network Structure

Input	Operator	Exp size	#out	NL	SE
224×224×3	Conv 2d	-	16	HS	-
112×112×16	Bneck, 3×3	16	16	RE	-
112×112×16	Bneck, 3×3	64	24	RE	-
56×56×24	Bneck, 3×3	72	24	RE	-
56×56×24	Bneck, 5×5	72	40	RE	✔
28×28×40	Bneck, 5×5	120	40	RE	✔
28×28×40	Bneck, 5×5	120	40	RE	✔
28×28×40	Bneck, 3×3	240	80	HS	-
14×14×80	Bneck, 3×3	200	80	HS	-
14×14×80	Bneck, 3×3	184	80	HS	-
14×14×80	Bneck, 3×3	184	80	HS	-
14×14×80	Bneck, 3×3	480	112	HS	✔
14×14×112	Bneck, 3×3	672	112	HS	✔
14×14×112	Bneck, 5×5	672	160	HS	✔
14×14×112	Bneck, 5×5	672	160	HS	✔
7×7×160	Bneck, 5×5	960	160	HS	✔
7×7×160	Conv2d, 1×1	-	960	HS	-
7×7×960	Pool,7×7	-	-	HS	-
1×1×960	Conv2d 1×1,NBN	-	1280	HS	-
1×1×1280	Conv2d 1×1,NBN	-	k		-

Where Input represents the shape change of each feature layer in the network, and Operator represents the upcoming block structure of each feature layer, it can be seen from Table 2.1 that the feature extraction network introduces many bneck structures. The third and fourth columns represent the number of channels after the

inverse residue structure in bneck, and the number of channels in the characteristic layer when input to bneck. Finally NL indicates the kind of activation function, HS for h-swish and RE for RELU.

Mobile Net How effective it is is compared with Google Net and VGG 16, as shown in Table2.2. Compared to VGG 16, Mobile Net is slightly less accurate, but better than Google Net. However, Mobile Net has absolute advantages in terms of the computational quantities and the number of parameters.

Table 2.2 - Performance comparison of Mobile Net with Google Net and VGG 16

Model	Image Net Accuracy	Million Mult-AddS	Million Parameters
1.0 Mobile Net-244	70.6%	569	4.2
Google Net	69.8%	1550	6.8
VGG 16	71.5%	15300	138

2.2.2 Shuffle Net series of algorithm

The feature extraction of ShuffleNetV1 [17] is more efficient, which further reduces the model calculation while maintaining the accuracy. Its structure is shown in Figure 2.3.

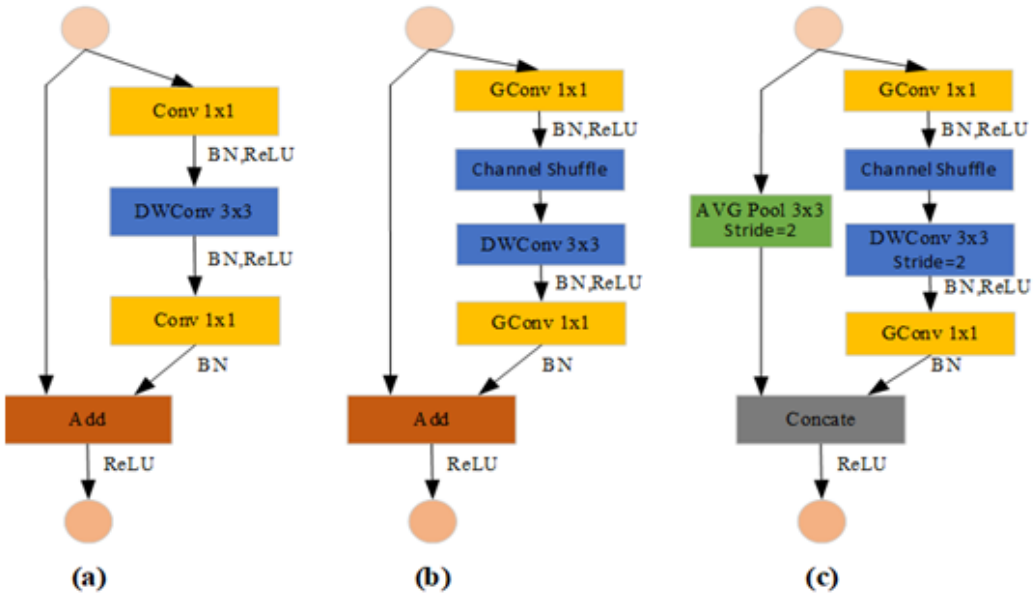

Figure 2.3 - ShuffleNetV1 Basic unit

Figure 2.3 (a) shows the lightweight network based on ResNet structure, using 11 convolution and channel convolution (depthwise convolution) to improve the structure and improve the speed of the model. Figure 2.3 (b) shows an improvement of the ShuffleNetV1 base unit through the introduction of packet convolution and channel stochastic mixing methods. Figure 2.3 (c) shows another improvement of the ShuffleNetV1 base unit. By adding a uniform pooling layer to the left branch of the base unit, and finally connecting the feature map with the output to realize the lightweight of the network. ShuffleNet V1 The network is mainly composed of multiple convolutional layers, pooling layers, stage, and modules (multiple basic units are mixed). Its structure is shown in Table 2.3. In addition, among the three stage modules introduced by the ShuffleNetV1 network, the other two step sizes are 1, except that the first step size is 2.

Table 2.3 - Structure of the ShuffleNetV1 network

Layer	**Output size**	**KSize**	**Stride**	**Repeat**
Image	224×224	-	-	-
Con v	112×112	3×3	2	1
Max Pool	56×56	3×3	2	
Stage 2	28×28 28×28	- -	2 1	1 3
Stage 3	14×14 14×14	-	2 1	1 7
Stage 4	7×7 7×7	-	2 1	1 3
Global Pool	1×1	7×7	-	-
FC	-	-	-	-

According to the feature graph size, the ShuffleNetV2 network optimizes the two basic block units of the ShuffleNetV1 network based on the residual ideas. The structure is shown in Figure 2.4, (a) and (b). Similarly, Figure 2.4 (c) and (d) are the two base block units of ShuffleNetV2.

The difference between Figure 2.4 (a) and (c) lies in that Figure 2.4 (c) has an additional channel segmentation operation at the beginning, which divides the

original packet convolution into the branch network of the two-channel information system, so that the number of channels on both sides of the convolution layer remains unchanged and the speed of the model is improved. Then it can be seen that the grouping convolution operation in the 11 convolution layer is cancelled in Figure 2.4 (c), because excessive packet convolution will slow down the model. Then, the channel random mixing operation is put after the feature cascade fusion (concat, all the features extracted from the convolution) operation. Finally, the element by element increase operation is replaced with the feature cascade fusion operation. The time consumption of element by element convolution is much more than the performance value on FLOPs (float-point operations), so the element by element convolution operation should be reduced as much as possible.

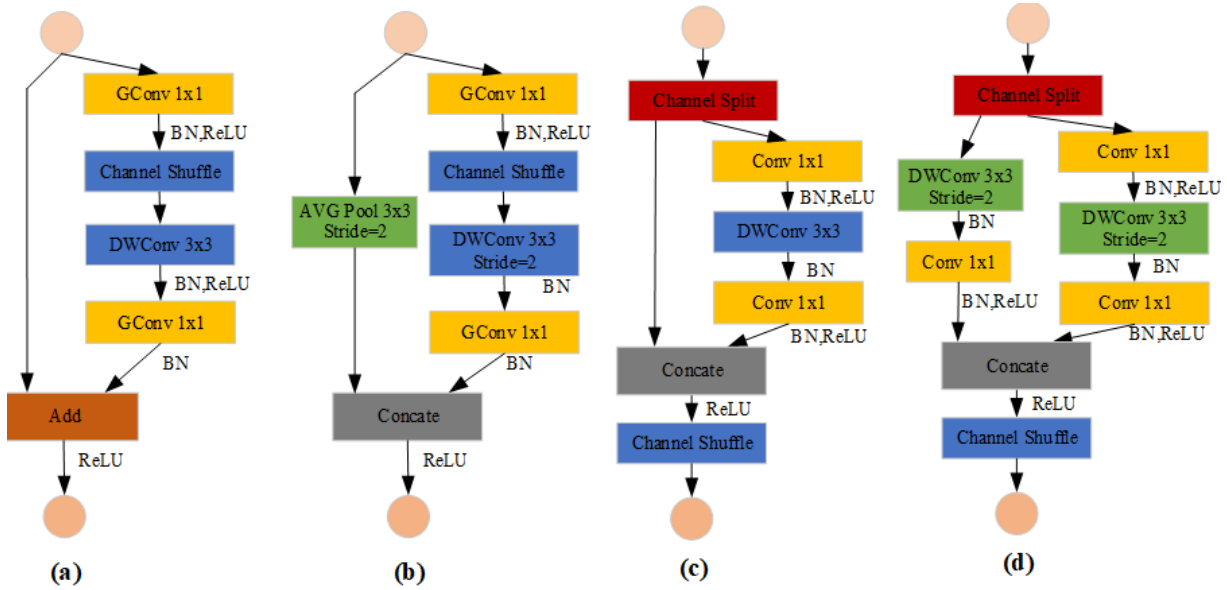

Figure 2.4 - Shuffle Net v1 and v2 block structures compare

Table 2.4 shows the network structure of ShuffleNetV2, and each stage is composed of the blocks shown in (c) (d) in Figure 2.5, and the number of block structures corresponds to the Repeat column in Table 2.3. Moreover, in ShuffleNetV2 [18], a new model speed measure is used- -memory access consumption time. Compared with the general FLOPs index, the former focuses more on evaluating the operation efficiency of the convolution layer, so as to reflect the real model speed.

Table 2.4 - ShuffleNetV2 Network Structure

Layer	Output size	KSize	Stride	Repeat
Image	224×224			
Conv	112×112	3×3	2	1
MaxPool	56×56	3×3	2	
Stage 2	28×28 28×28		2 1	1 3
Stage 3	14×14 14×14		2 1	1 7
Stage 4	7×7 7×7		2 1	1 3
Conv 5	7×7	1×1	1	1
GlobalPool FC	1×1	7×7		

2.2.3 Squeeze Net

Figure 2.5 shows the structure of the Squeeze Net network. It can be seen that the input image is first fed into a convolution layer, and then successively input into multiple fire modules. And connect the maximum pooling layer between these fire modules to achieve the delay of down sampling, so that the front layer has a large feature graph. Inspired by the idea of residual difference in Res Net network, the introduction of different "short circuit" mechanisms in Squeeze Net network can also reduce the number of parameters to a certain extent.

Where the fire module is a specially-designed structure in Squeeze Net, as shown in Figure 2.6. It mainly consists of the squeeze layers and the expand layers. The set of convolutional layers of 11 constitutes the sequeeze layer, which greatly reduces the input channel dimension. The set of 11 and 33 convolutional layers constitutes the expand layer, the structure mainly achieves feature fusion.

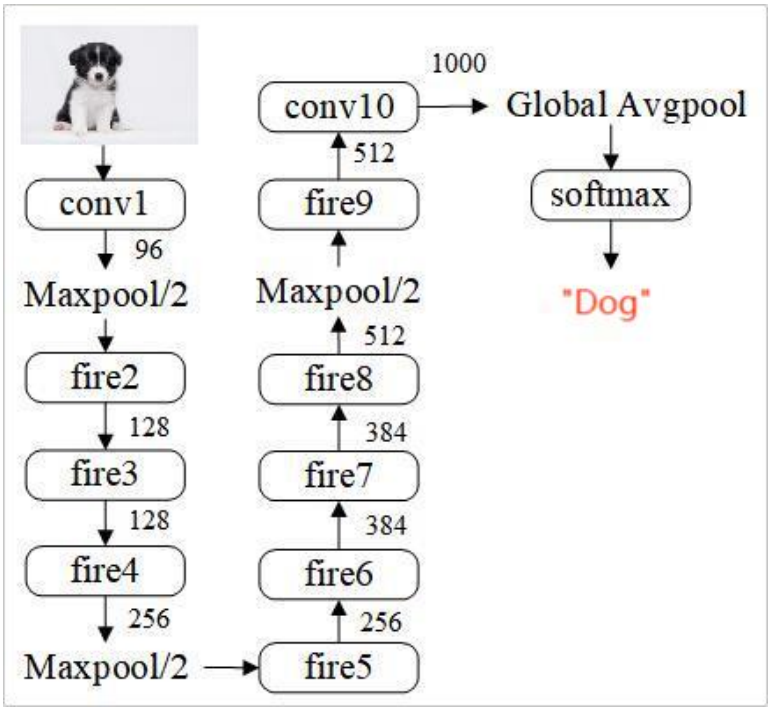

Figure 2.5 - Squeeze Net Network structure

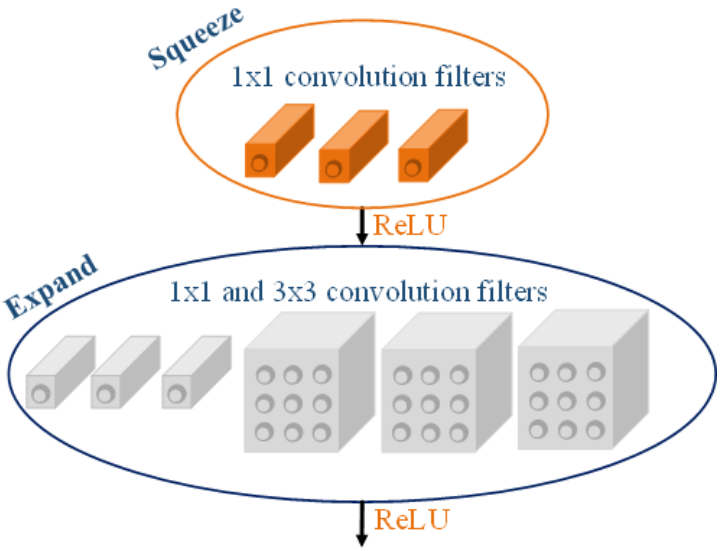

Figure 2.6 - Fire module structure

2.2.4 Xception

Xception, Network is based on deep separable convolution and residual network structure optimization on the basis of Inception network. The Xception network structure is shown in Figure 2.7. Unlike the operation of mixing channels and space during conventional convolution, Xception divides the input data into multiple groups through 11 convolution. Then the convolution operation is used to

realize the accurate control of the channel and the spatial operation, and realize the decoupling of the two correlation. Both the network and the Mobile Net series algorithms add channel-by-channel and point-by-point ideas to the model optimization. However, the original optimization objects are different. The Xception network is improved based on the Inception V3 network to transform its basic units into a combination of channel convolution and point convolution. Moreover, the model size and computational efficiency are also improved compared with the Inception V3. The final model achieved better results than Inception V3 and Resnet-152 on datasets.

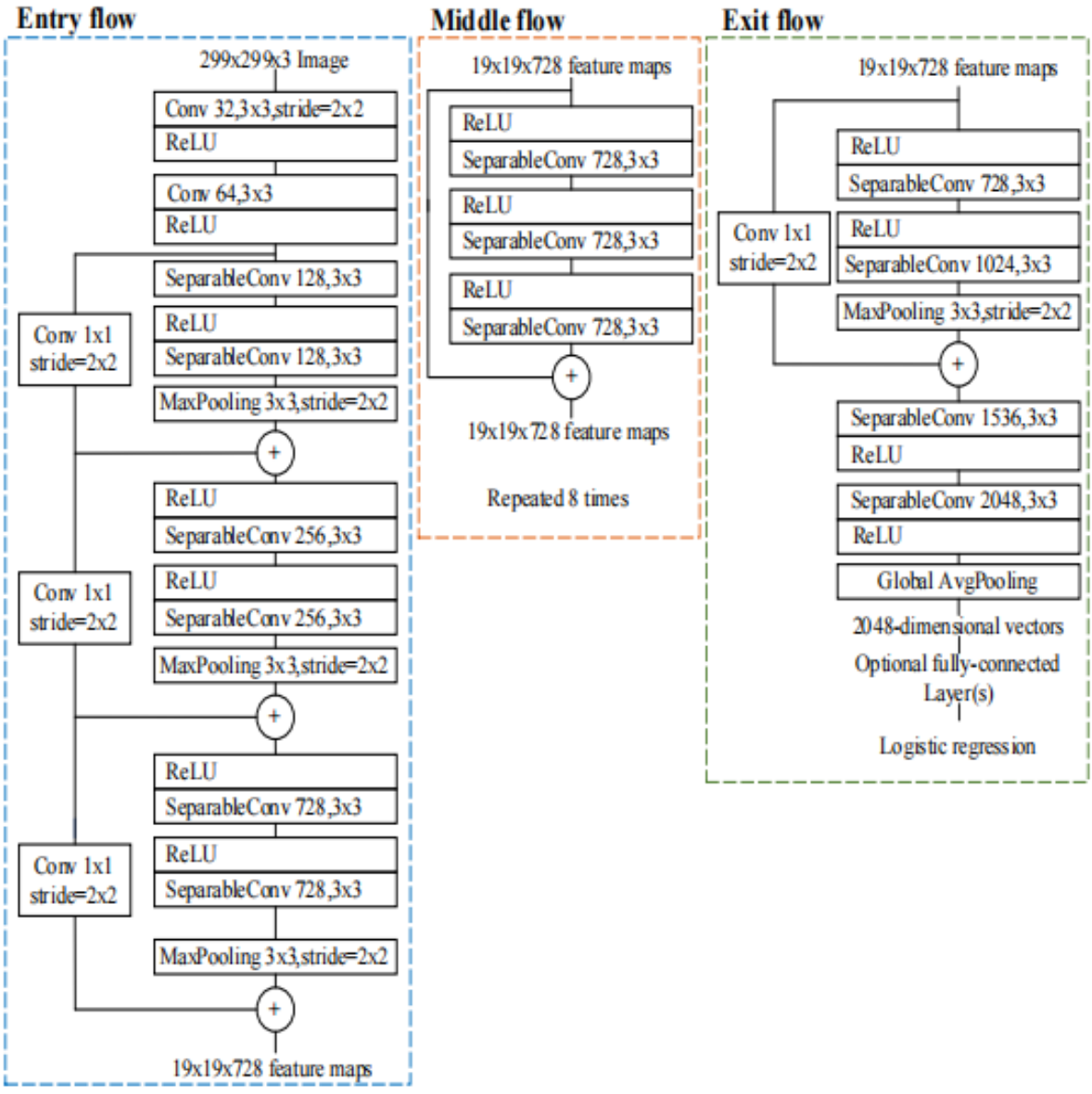

Figure 2.7 - The overall structure of the Xception network

2.2.5 Pelee Net

Dense Net Due to the introduction of the idea of dense connection, the network performance is greatly improved and the consumption of computing power becomes greater, which further leads to it cannot be directly applied on mobile terminals. Therefore, Pelee Net [20] completed the miniaturization of Dense Net to adapt to the deployment and application on the device side. Pelee, the classification network structure is shown in Table 2.5. In order to realize the accuracy of the network feature expression, the Pelee classification network designed the Stem module, and down sampled the input images and increased the number of channels.

Table 2.5 - Pelee Classification network structure

Stage		**Layer**	**Output Shape**
Input			224×224×3
Stage 0	Stem Block	-	56×56×32
Stage 1	Dense Block	DenseLayer ×**3**	28×28×128
	Trabsition Layer	conv 1×1,stride 1	
		Avg pool 2×2, stride 2	
Stage 2	Dense Block	DenseLayer ×**4**	14×14×256
	Trabsition Layer	conv 1×1,stride 1	
		Avg pool 2×2, stride 2	
Stage 3	Dense Block	DenseLayer ×**8**	7×7×512
	Trabsition Layer	conv 1×1,stride 1	
		Avg pool 2×2, stride 2	
Stage 4	Dense Block	DenseLayer ×**6**	7×7×704
	Trabsition Layer	conv 1×1,stride 1	
Classifier Layer		Avg pool 7×7	1×1×704
		1000D fully-connecte,softma×	

In Pelee Net, the number of output channels in the bottleneck layer changes

dynamically with the input shape, rather than 4 times that of growth-rate in Dense Net (growth-rate represents the number of channels), thus reducing the demand of computing power on the network. Moreover, the number of input-output channels remained consistent in the transition layer of the Pelee Net network. Finally, the Post-activation structure replaces the Pre-activation structure in Dense Net, which leads to the fusion of the BN layer and the convolutional layer in the inference stage. Both Post-activation and Pre-activation are residual block structures, where Post and Pre are shown relative to the anterior-posterior position of the first convolutional layer in the ReLu activation function in the residual block structure.

2.2.6 MnasNet

Lightweight neural networks should take into account the balance between accuracy, time and calculation complexity, and realize efficient operation, the accuracy of the model can not have too much loss. At present, the performance of MobileNet series, PeleeNet, EfficientNet and other lightweight networks in low computing devices has been greatly improved, but in practice, it is still difficult to see these algorithms. Mainly because these networks are very fast, but the accuracy is not ideal. So the Google team proposed MnasNet [22], and Figure 2.8 (a) shows MnasNet-A1. As can be seen from Figure 2.8, the entire network consists of multiple layer structures. Moreover, MnasNet not only introduces an efficient special multi-objective search strategy, but also replaces 33 convolutions with combined convolutions of 33 and 55. According to the test results of Pixel series mobile phones, the MnasNet model can achieve efficient real-time identification under multi-task (ImageNet classification and COCO target detection, etc.).

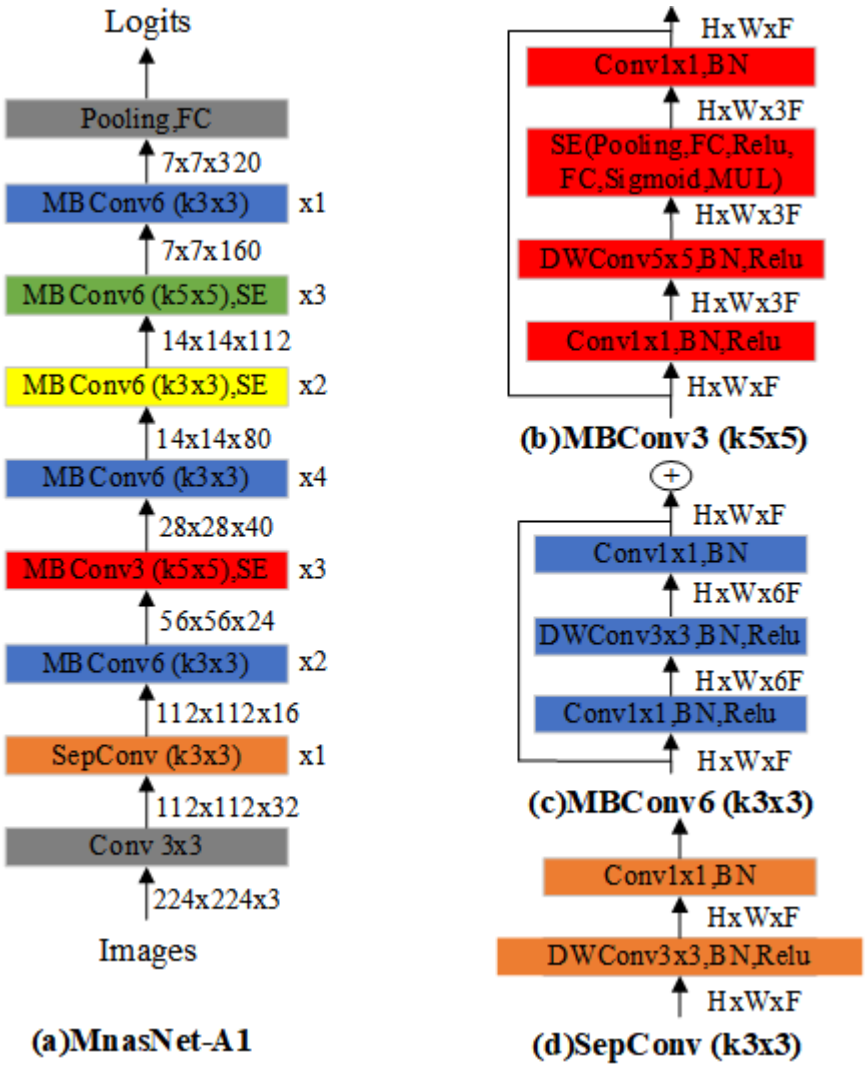

Figure 2.8 - MnasNet network structure

2.3 Mainstream algorithm implementation methods of low-power devices in different application scenarios

2.3.1 Application of the lightweight detection algorithm improved based on the YOLO series algorithm

SlimYOLOV3 [32] Algorithm: In order to realize the use of deep learning algorithm in drones with low computing power and small memory, the researchers increased the SPP module [24] in YOLOV3-SPP 1 to three and obtained the YOLOV3-SPP3network. SlimYOLO V3 [32] was pruned on the basis of YOLOV3-SPP 3 network, and the main process of model pruning is shown in Figure 2.9.

YOLO V3-SPP 3 removes "unimportant" connections during sparse training, that is, the channel information of scale factor effects. The parameters of the sparse treated SlimYOLOV3 model were then adjusted. Then, whether to conduct the sparse processing according to the test results of the model is decided. If the performance after the test fails to meet the requirements, the sparse processing will continue to be iteratively, and otherwise, the model will be exported. The final test results show that the sparse SlimYOLOV3 network accuracy is comparable to YOLOV3 with insufficient computing power, limited memory and insufficient performance, however, but the operating efficiency and hardware performance requirements are greatly optimized.

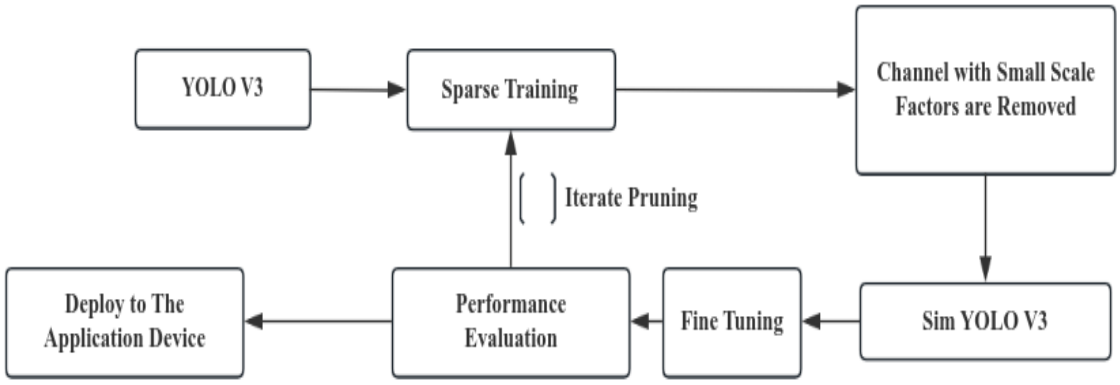

Figure 2.9 - SlimYOLOV3 Model pruning flow

YOLO-Lite Network: The network optimizes the selection of activation function on the basis of the YOLO V2 network. Using the Leaky ReLU activation function, the input image size is compressed to 224*224 pixels, and the smaller input size finally reduces the computational amount. In addition, the YOLO-Lite network balances the pros and cons of the BN (Batch Normalization) layer. Although the BN layer can speed up the convergence of parameters during training, it also produces the computational overhead problem that has the most impact on the application of low power devices.

YOLO Nano Network: The network is mainly composed of the diversified combination of PEP (projection-expansion-projection) modules, EP (expansion-

projection) modules, FCA (full-connected attention) modules and each 33 and 11 convolutional layers. The YOLO Nano network structure is shown in Figure 2.10. Where PEP (x) represents the x channel of the first mapping layer of the residual PEP module, and FCA (x) indicates the dimension reduction rate of x. The residual PEP macro architecture consists of two 11 convolutional mapping layers, an 11 convolution expansion layer, and a channel-by-channel convolution layer.

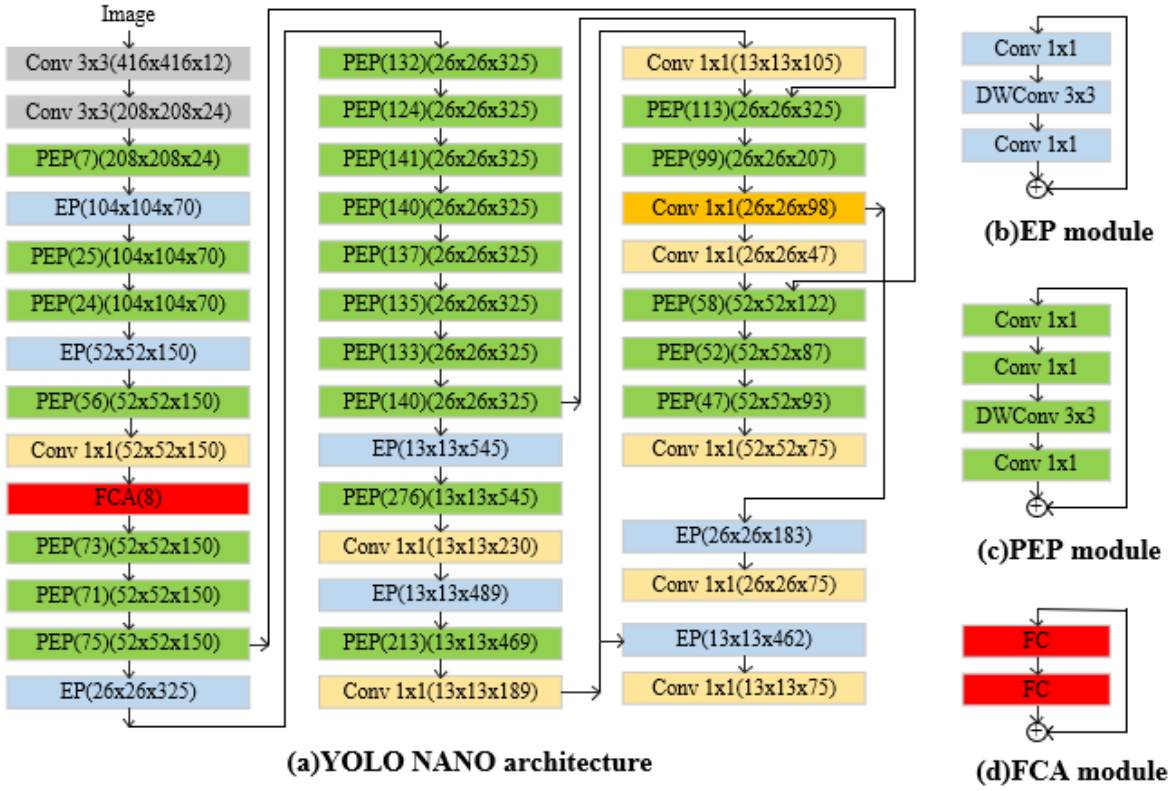

Figure 2.10 - YOLONano Network Architecture

The FCA module consists of two fully connected layers. The fully connected layer in this module learns dynamic, nonlinear internal dependencies between channels and reweights the importance of channels by multiplication at channel-level. The use of FCA not only helps to focus on more informative features based on global information, but also can make the optimal choice between pruning the model architecture, reducing the model complexity, and increasing the power of the model representation. In addition, the researchers tested the inference speed and energy efficiency of YOLONano networks in the Jetson AGX Xavier embedded module.

With the 15W and 30W energy budgets, YOLO Nano can achieve the inference speed of 26.9fps and 48.2fps, respectively.

Tiny-YOLO algorithm: V. Kharchenko et al. proposed to use the Tiny-YOLO algorithm to run on drones to realize the recognition and detection of ground objects. Unlike the YOLOV3 algorithm, Tiny-YOLO uses a more lightweight feature extraction network, and the feature extraction network structure is shown in Table 2.6.

Table 2.6 - Tiny-YOLO feature extraction network structure

Type	Filters	Size	Output
Conv	16	3×3/2	208×208
Conv	32	3×3/2	104×104
Conv	64	3×3/2	52×52
Conv	128	3×3/2	26×26
Conv	256	3×3/2	13×13
Conv	512	3×3/2	13×13

2.3.2 Application of the lightweight detection algorithm improved based on the SSD algorithm

SSD + PeleeNet Network [31]: Real-time small object detection in low-power mobile devices has always been one of the fundamental problems in monitoring applications. Unel, the team uses the SSD + PeleeNet network application to achieve pedestrian and vehicle detection on a miniature aircraft (MAV) with high-resolution images. By feeding the feature map extracted from PeleeNet into the SSD [50] network, the miniaturization of the original SSD network is realized, which is very suitable for the deployment of low-power devices. Compared with the structure algorithm with MobileNet as the backbone network, the PeleeNet-based implementation structure has fewer parameters with computational scale. Mainly because the algorithm connects five scale feature maps to the detection branch, namely (1919,1010,55,33 and 11), and the size design of the anchor frame is shown in Table 2.7.

In addition, lightweight residual blocks were added before the detection layer

of each detection branch to ensure feature expression power. The residual predicted touch block structure is shown in Figure 2.11. The detection layer (in the ResBlock module) replaces the 33 convolution with the 11 convolution, and the calculated quantity reduction rate reaches 21.5%.

Table 2.7 - Comparison of the size design of anchor in each network

Model	Scale of Feature Map : Scale of Default Box					
Original SSD	38*38:30	19*19:60	10*10:110	5*5:162	3*3 : 213	1*1:264
SSD+Mobile Net	19*19:60	10*10:105	5*5:150	3*3 : 195	2*2:240	1*1:285
Pelee	38*38:30.4	19*19:60.8	10*10:112.5	5*5 : 164.2	3*3:215.8	1*1:267.5

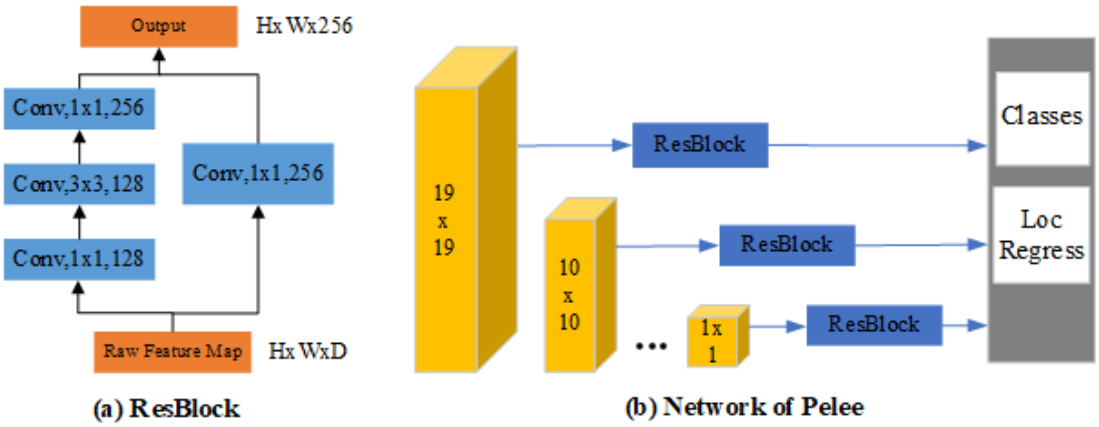

Figure 2.11 - Residual-predicted touch block

2.4 Overview of model compression techniques

2.4.1 Model pruning

Artificial intelligence has begun to emerge in every aspect of our lives, especially in the smart Internet of Things. Therefore, the need to deploy algorithm models in various low-power embedded development devices in the intelligent Internet of Things system arises. The lightweight network research with low

computing power requirements and optimized accuracy and speed has become one of the hot research issues. To meet the above requirements, the algorithm model must be optimized. Model pruning is one of the commonly used techniques in deep learning network model compression technology. It can not only effectively reduce the computational complexity of the model, but also alleviate the overfitting problem caused by overtuning operations.

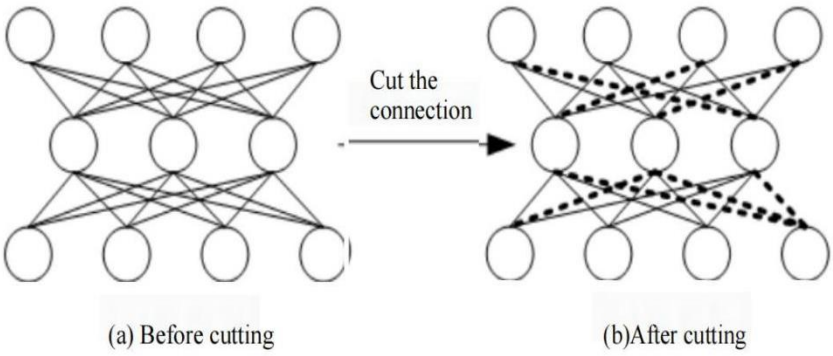

Figure 2.12 - Cropped connections

Model pruning completes redundant connections by setting an evaluation standard for network parameters and performing the standard on the existing model. Common model pruning techniques include unstructured pruning and structured pruning. Dong et al. [39] proposed a hierarchical pruning method based on the error function values of each network layer. In Figure 2.12 (a), the connections at the two ends of the network layer are too dense, and then the screening criteria are designed according to the weight of the connections that do not meet the criteria, that is, the dotted line in Figure 2.12 (b) to realize model compression. Han et al. [40] set the evaluation criteria according to the weight norm between neurons, and deleted the unimportant connections (i. e., the model value is less than a certain threshold) to achieve model compression. Different from other methods on the pre-training model pruning, Lee et al. SNIP (single-shot network pruning) method [41] before the training model set multiple sampling to determine whether the connection between neurons is important to pruning related parameters, then in the model training stage pruning model, and conventional pruning-fine-tuning completely different.

In addition to the above pruning method, the neuronal weights can be directly evaluated as shown in Figure 2.13. Mariet et al [43] will use the point process (DPP) thought to realize neurons importance judgment and make a choice, and the low weight neurons information and remaining neurons complete fusion, this can be very good, restore performance loss caused by pruning, compared with the traditional pruning after fine-tuning training is more convenient, and the effect is better. Carreira-Perpinan et al [44] alternate the "learning" and "compression" method to explore the weight subset method that minimize the loss. Liu et al. convolution can be implemented by DCT domain multiplication, followed by a dynamic pruning of the DCT coefficient of filter.

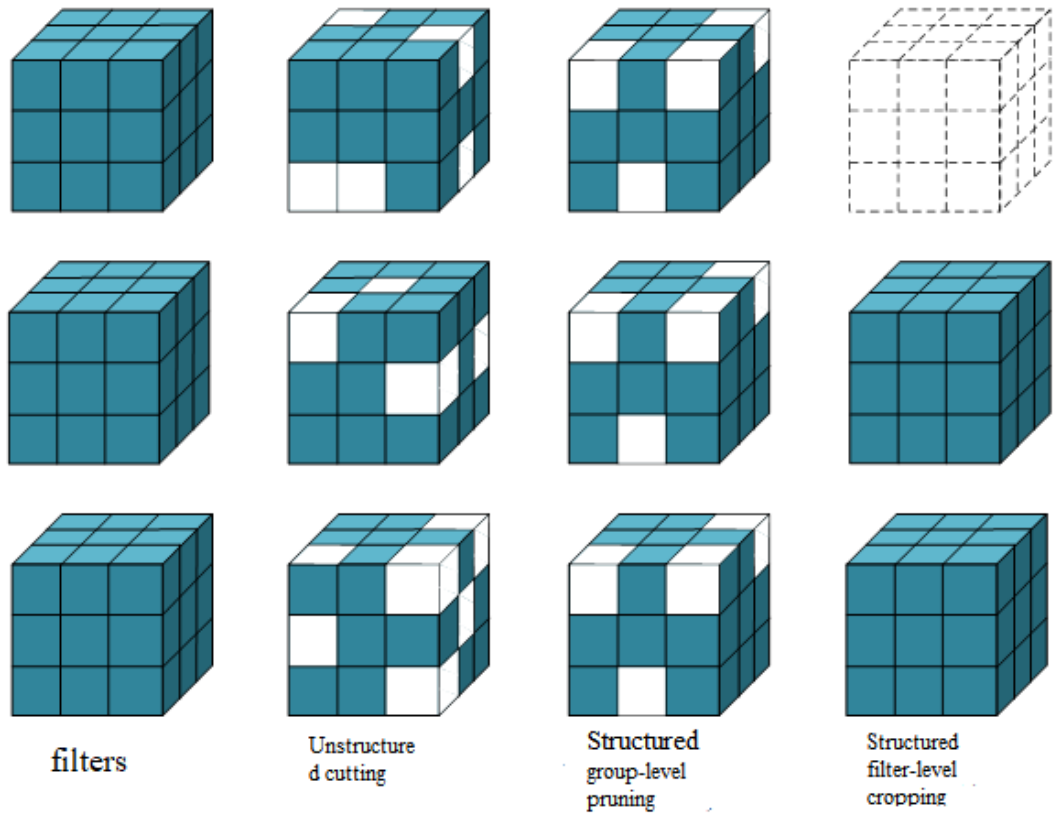

Figure 2.13 - Parameter pruning

Structured pruning mode includes both group level and filter level. As shown in Figure 2.13, group level pruning means that the same sparse pattern is filter for each layer (i. e., each cube removes the small square at the same position) and becomes a sparse matrix with the same structure. Wen et al. [46] introduced Gaussian

regression groups to perform regularization constraints to verify the multi-level structural sparsity.filter Level pruning is the channel level pruning. As shown in Figure 2.13, deleting the entire cube from the deletion graph is equivalent to deleting part of the resulting feature graph and the next part filter that needs to be convoluted with this part of the feature graph. Li et al. [48] calculates the L1 norm of the filter, filters out the feature map corresponding to the filter of the smaller L1 norm, and retrains after pruning. Molchanov et al. [49] problem as an optimization problem, choose an optimal combination from the weight parameters to minimize the loss of the parameter loss, so that the small decay of prediction accuracy after pruning is unimportant.

2.4.2 Model Quantification

Parameter quantification process will selectively use relatively low width to represent high wide floating point parameters, the parameters is mainly in the network forward reasoning, reverse reasoning in the weight value, activation function output value, gradient update value, etc., in the process of quantification generally use unified bit width (such as 16-bit, 8-bit, 2-bit and 1-bit, etc.), sometimes also according to network characteristics or training skills to choose different bit width. Parameter quantification can not only reduce the network computing complexity and hardware resources, but also improve the computing efficiency and reduce the equipment energy consumption. Normally, the model inference process with 32-bit floating-point numbers requires more memory resources, which is about three times slower than the running speed of 8-bit integer data. The more complex the model structure is, the more the corresponding parameters are, and the larger the gap will be. Therefore, no matter in the forward reasoning process of the deep learning model or the reverse reasoning process, the calculation efficiency of the whole data is higher than that of the floating point data, and the accuracy loss will not decline greatly, so as to realize the smooth operation of the devices with low power consumption and low computing power.

Courbariaux et al.[51] applied the binary strategy in forward reasoning and reverse inference to realize the reduction of the number of model parameters, but still

chose the high and wide data calculation method in the gradient update. Liu et al. [53], on the basis of Xnor-net [54], minimized the network structure and constantly adjusted the training strategy, and finally obtained the Bi-real network.

It is often difficult to obtain the optimal solution we want by setting the network parameter bit width manually, but we can set different critical criteria to assist the network to minimize the accuracy loss. Faraone et al. [61] proposed gradient-based learning symmetry quantization, design weight binarization or tripolarization, and define different scaling factors pixel by pixel, line by line or layer by layer to estimate network weights. Zhang et al. [62] proposed Learned quantization (LQ-nets), which enables the quantizer to be trained jointly with the network to adaptively learn the best quantization bit width. In order to reduce the damage of quantization to the model performance, Xiao Guolin [71] not only improves the statistical method of quantization error, but also proposes a new 2-bit weight interaction quantization algorithm.

3 ALGORITHM DESIGN AND TRAINING STRATEGY BASED ON LOW-POWER EMBEDDED DEVICES

3.1 Overall network structure

Compare it to the network accuracy. Figure 3.1 shows the results of the precision and time comparison of the different lightweight network models. The abscissa represents the time that the Google Pixel mobile phone series has completed the test with a single-threaded large kernel in the Tensorflow Lite environment. Figure 3.2 shows a comparison between the MAdds and top-1 accuracy. For different lightweight network models, MobileNetV3 takes picture resolution with input 224224 and uses multipliers of 0.35,0.5,0.75,1 and 1.25 coefficients.

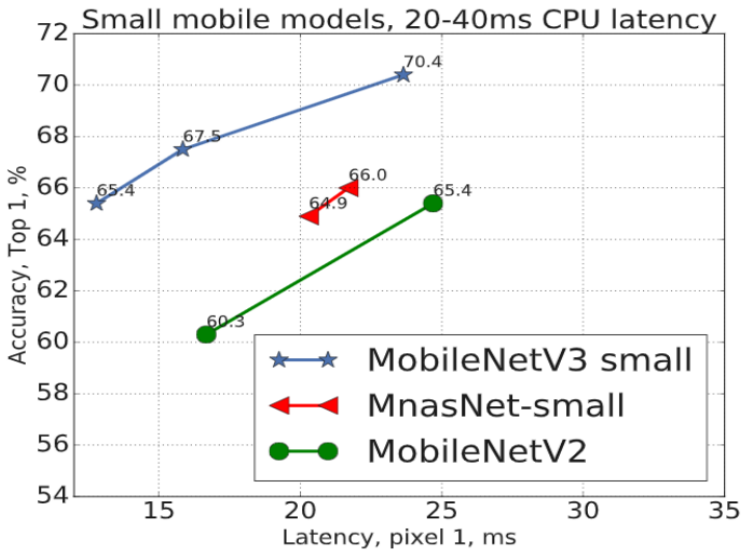

Figure 3.1 - Schematic diagram of the accuracy rate of the Mobile Net V3 network and the other networks

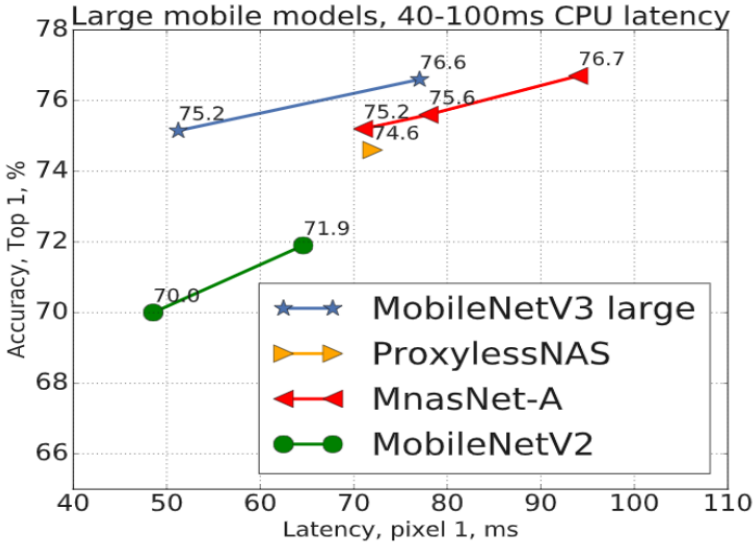

Figure 3.2 - Schematic diagram of the accuracy rate of the Mobile Net V3 network and the other networks

Lightweight networks are optimized based on the bloated network, and they only improve to optimize the performance of the network and reduce the parameters of the network. As can be seen from Figure 3.1 and Figure 3.2, MobileNet V3-Large algorithm is more accurate and faster than lightweight networks such as MobileNet V2, NasNet and ShuffleNet. Thus, the MobileNet V3-Large algorithm becomes the optimal choice to replace the DarkNet-53 network in YOLO V3.

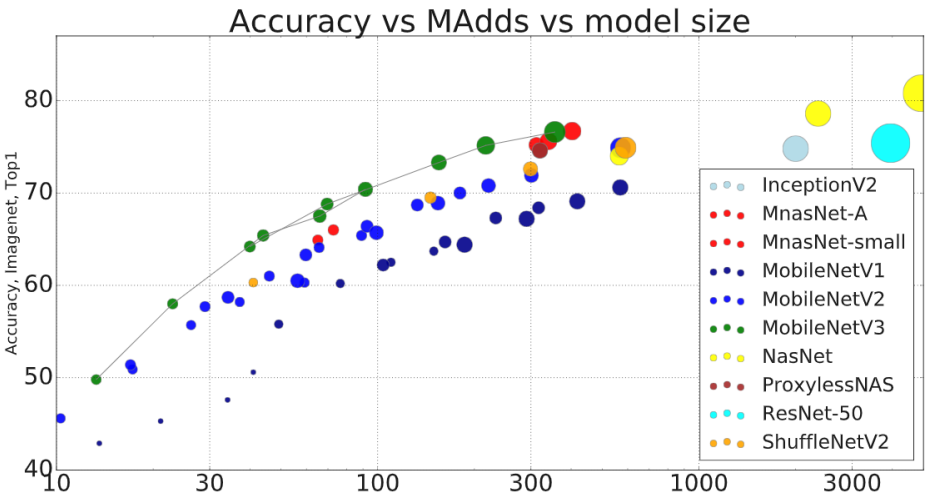

Figure 3.3 - Comparison of Mobile Net V3 network and other networks

Parameter quantification process will selectively use relatively low width to represent high wide floating point parameters, the parameters is mainly in the network forward reasoning, reverse reasoning in the weight value, activation function output value, gradient update value, etc., in the process of quantification generally use unified bit width (such as 16-bit, 8-bit, 2-bit and 1-bit, etc.), sometimes also according to network characteristics or training skills to choose different bit width. Parameter quantification can not only reduce the network computing complexity and hardware resources, but also improve the computing efficiency and reduce the equipment energy consumption. Normally, the model inference process with 32-bit floating-point numbers requires more memory resources, which is about three times slower than the running speed of 8-bit integer data. The more complex the model structure is, the more the corresponding parameters are, and the larger the gap will be. Therefore, no matter in the forward reasoning process of deep learning model or the reverse reasoning process, the calculation efficiency of the whole data is higher than that of floating point data. At the same time, the accuracy loss will not decline greatly, so as to realize the smooth operation of devices with low power consumption and low computing power.

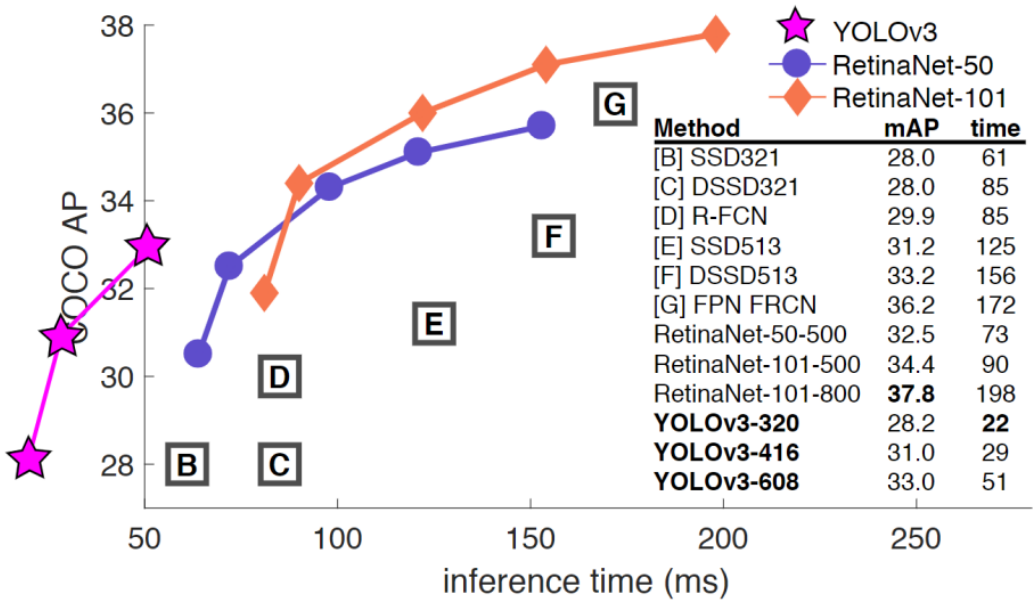

Figure 3.4 - Comparison of YOLO V3 algorithm and other algorithms

In the implementation of the network framework, in order to reduce the difficulty of the target detection model transplantation to the embedded computing platform, this paper integrates the MobileNetV3 Large network into the YOLO V3 algorithm to complete the feature extraction task. Based on this algorithm, the trained network model performs parameter pruning and model quantification. The final model was then deployed in the Raspberry Pi, NVIDIA Jetson Nano, and KD 233 development suite, respectively. The Darknet-53 network using the MobileNet V3 Large network to replace the YOLO V3 algorithm can run smoothly on embedded devices. Figure 3.5 is the improved network structure diagram, where the bottleneck structure is shown in Chapter 2, Figure 2.3. After entering the data into the MobileNetV3 Large network and adjusting the input image size to 416416, the feature map of the three scales will be obtained from the sixth bottleneck layer, the twelfth bottleneck layer, and the last convolution layer of the network. Then, the three feature maps are respectively input into the YOLOV3 algorithm for multiple upsampling, convolution and other operations, and then obtain the final detection results.

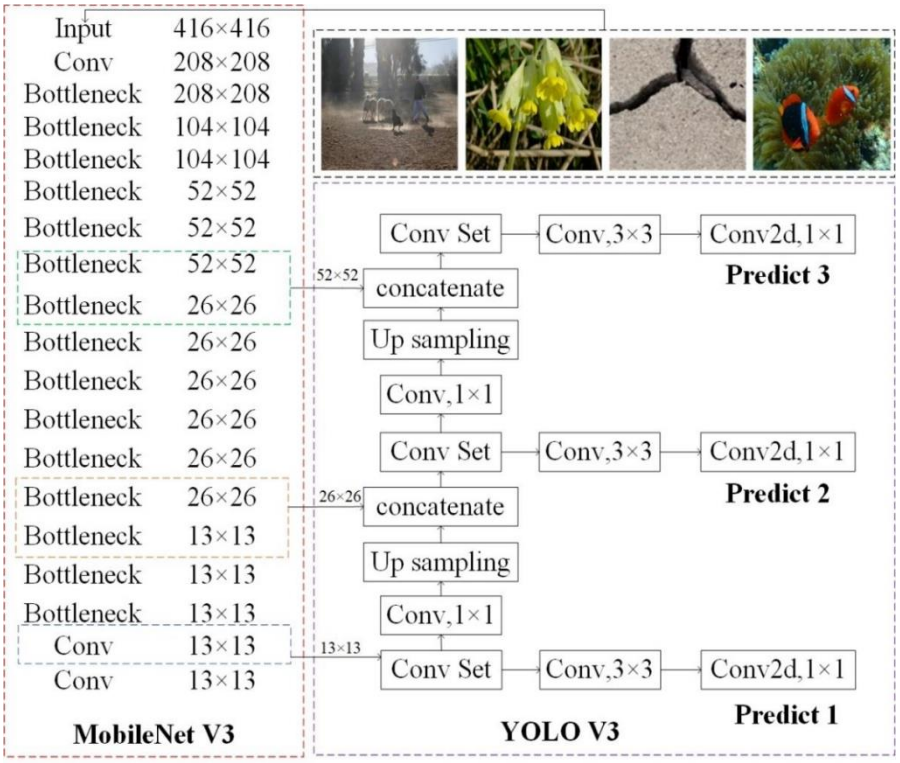

Figure 3.5 - Schematic diagram of the MobileNet V3 + YOLO V3 network

3.2 Improved pruning method based on the pre-training model

Neural networks with more complex width and depth information can not only improve the performance, but also produce a large number of redundant parameters, resulting in the waste of computing power. Therefore, it is necessary to effectively compress the network model, and adopt a method that can keep the size and speed of the model balanced, so that the final compression will not lose the watermelon and pick up the sesame seed. Model pruning is to set an evaluation criterion, and then delete those unimportant connections based on the criteria, thus reducing the parameter scale and computation. However, when the model parameters are excessively pruned, especially the large network model with particularly good performance, the expression ability of the model will be weak, and how to prune the parameters also determines the loss of model accuracy.

Model pruning mainly includes both single pruning and multiple pruning

modes, [68]. Based on the results of the same compression sparse ratio (ratio of trimmed parameters and ratio of total parameters) model, multiple iterative pruning causes less accuracy loss and more robust to the model than single pruning.The iterative pruning process is shown in Figure 3.6.

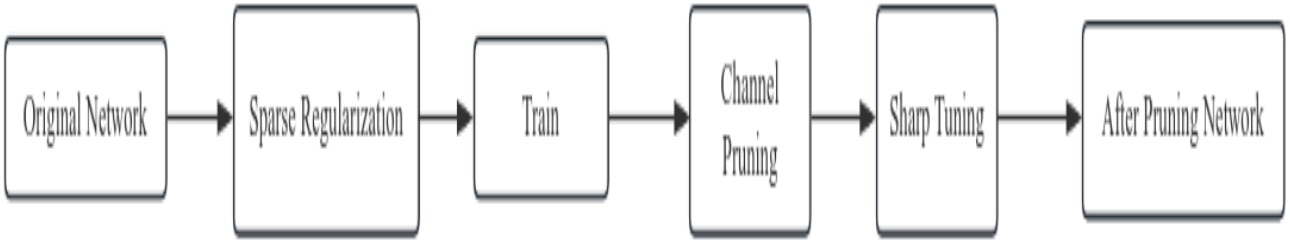

Figure 3.6 - Flow chart of the pruning step

This paper on the YOLO V3 + MobileNet V3 and YOLO V3 + MobileNet V2 model experiment, in order to reduce or avoid interference by other factors, we are all in the same hardware, experimental environment, sparse way, initial learning rate training until network convergence, the model test data set is standard VOC-2012 data set, do not take any optimization training skills under the premise of the sparse process is divided into 4 sections and the final sparse ratio is 60%.

Table 3.1 - Random nonlinear and linear segment sparse ratio changes

	Sparse ratio				
segmentation	0	1	2	3	4
Random nonlinear segmentation	0%	19%	27%	48%	60%
Linear segmentation	0%	15%	30%	45%	60%

For the different segments and sparse ratios of different sparse strategies under multiple iterations, see Table 3.1 Figure 3.7 shows the accuracy loss comparison of the two models applying random nonlinear and linear pruning strategies, where the abscissa coordinate represents the number of segments, and the ordinate represents the top-1 accuracy loss rate, that is, the maximum accuracy loss among all test samples.

From Figure 3.7, when the target sparsity ratio is 60%, the accuracy loss of the two pruning strategies is not large in the first half of the sparsity, but the increase of the sparsity ratio in the second half of the sparsity process leads to a huge decrease in model performance. The final experimental results prove that nonlinear segmentation sparse causes less accuracy loss than linear segmentation.

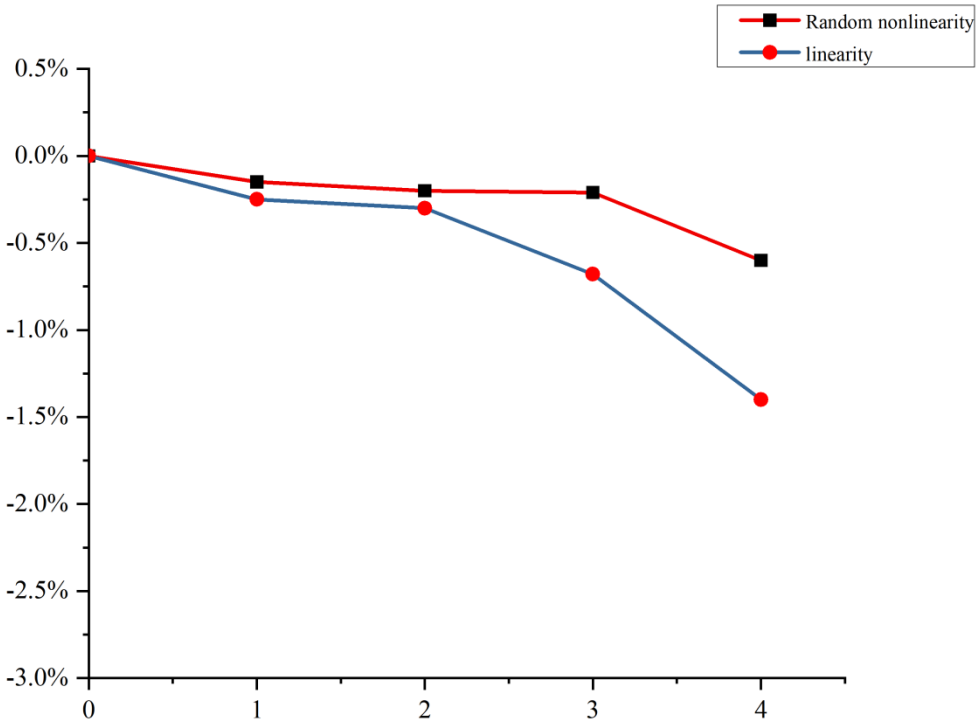

Figure 3.7 - Comparison of precision loss-wise YOLO V3 + MobileNet V3 on VOC-2012 for random nonlinear and linear pruning strategies

3.3 Model quantification method and effect

Model quantification is an effective means to compress the internal weight parameters of the model. The quantification method [26] is to change the original floating point number into the integer number in the internal reasoning process of the convolutional neural network, so as to reduce the computing overhead of the neural network model on low-power devices. Model quantification is to realize the

optimization of accuracy and speed, that is, to maximize the model computational complexity and minimize the accuracy loss.

Pytorch, Tensorflow and other similar mainstream frameworks will use float 32 (Full Precise, FP32) level data to update and pass the parameters (weight, bias, [69], learning rate, etc.) in the deep neural network training. If a network is deep, the number of network parameters will be more and the computation will be larger (for example, the ResNet-152 network reaches 11.3BFLOPS). If there is more computation, if FP32 is used for reasoning, the computing power of low-power embedded devices is difficult to complete such tasks.

The model is composed of weight (w) and bias (b), w, b are stored in float32 format, float 32 format for 32 bit in the computer, int8 format for 8bit in the computer; the model quantification is to int8 or less number of data type instead of the float32 internal data update, the model, so as to achieve the model of computing force reduction and reasoning speed up. Therefore, this paper uses the 8-bit quantization of the floating point model to accelerate the operation of the model on the embedded development board. Figure 3.8 represents the simplified process of the int8 fixed-point quantization method.

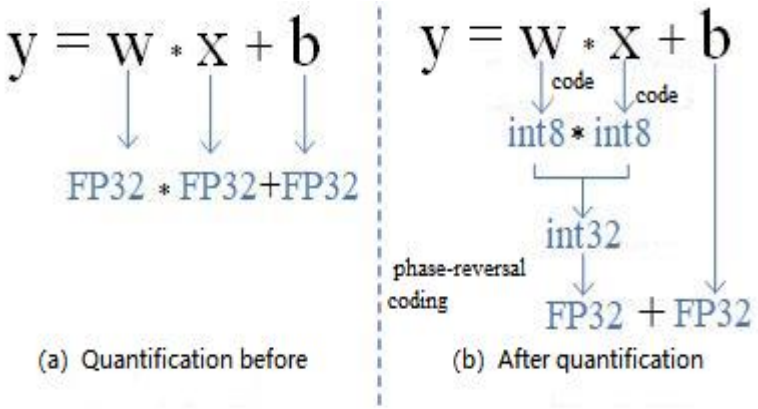

Figure 3.8 - Int8

All computations in the model forward inference process can be reduced to y = = w * x + b, y is the result, x is the input, i. e., the feature graph, w is the weight, and b is the bias. In the actual process, b has little influence on the inference results of the

model and is usually abandoned. Originally w, x is float32 level data, now using int8 as qw, qx; model quantification method can reflect the mapping relationship between fixed point and floating point.

The development of model compression technology leads to the model size is getting smaller and smaller, but the computing power of the equipment still requires 100-200 MFLOPS, which still leads to a gap for a large number of low-end arm-based CPU built in reality. There are two schemes for conventional low-power device acceleration: CPU based and GPU based. At present, NVIDA Jetson series edge equipment development suite based on edge devices. Based on CPU applications, mainly Raspberry Pi series development board and other mobile phone processors based on arm platform, UAV controllers, automotive electronic system management chips and other chips widely used in the Internet of Things. On the deployment of specific quantitative methods, this paper uses NCNN [73] to perform int8 fixed-point quantification method on the Raspberry Pi and Kanzhi KD233 development suite of onnx model, and uses TensorRT to quantify the model on NVIDIA Jetson Nano. Finally, the object detection is realized respectively.

In this study, we were trained on the server side of multiple GPU operation, where the initial value of learning rate is 0.001, decay rate =0.85, momentum coefficient a=0.9, and batch size 64. Four models were obtained after 8000 rounds (myVOC, myFLOWER, myLIFE, myCRACK). The four models were processed separately using int8 fixed-point quantization method, and the accuracy results of the network test after deployment are shown in Table 3.2 below.

Table 3.2 - Comparison of the quantification results

	myVOC	**myFLOWER**	**myLIFE**	**myCRACK**
(Floor floating point calculation) accuracy (%)	91.97%	92.76%	87.48%	90.48%
(The int 8 fixed-point quantification) accuracy (%)	85.06%	86.22%	82.1%	83.26%

3.4 Synchronous mode and multiple GPU parallel training

The training of the deep learning model generally uses the GPU for the acceleration. When the training sample size is too large or the batch size is too large, the training efficiency of a single graphics card will become very low, which further leads to the longer training time and ultimately affects the deployment progress of the engineering application. Therefore, in the model training, this paper adopts the multi-card parallel mode in the synchronous mode to realize the high efficiency of the model acquisition.

Training in synchronous mode, different devices will first after iterative training update the current parameter value, and then random part from the training set, feed the data into the transmission algorithm will get the output parameter prediction value, then the device on the parameter prediction value input to back propagation algorithm will synchronize the parameter gradient value of the output. Although the initial parameters of each graphics card on the server are the same, the final calculated parameter gradient value are different because the command response time may be different. When the backpropagation algorithm is executed and updated with the parameter gradient on all devices, the main program will average the update parameter values output by the backpropagation algorithm on each device, and finally complete the gradient update according to the average and complete one link of the iterative training. Moreover, in the synchronization mode, all devices will update the parameter gradient according to the corresponding output only after performing the backpropagation algorithm.

4 BASED ON THE IMPLEMENTATION OF LOW-POWER EMBEDDED DEVICES

4.1 Setting up of the experimental environment

4.1.1 Building of model training environment

The host configuration for the experiment based on the TensorFlow training platform is: Ubuntu16.04, Intel I7 processor, 32GB DDR4 memory * 4, and Nvidia Geforce Tesla1000 (32GB) graphics card * 4.

In this study, we used the current popular TensorFlow framework, where deep learning is used to complete the training and testing of the model, and used the cuDNN library to accelerate the training process of GPU. The construction and execution of the computational diagram are managed by the TensorFlow front end and back end respectively, as shown in Figure 4.1. TensorFlow The implemented deep learning network model can be transplanted on many low-power platforms, such as Raspberry PI 3B +, KD 233 Development suite, and Nvidia Jetson Nano Development Board.

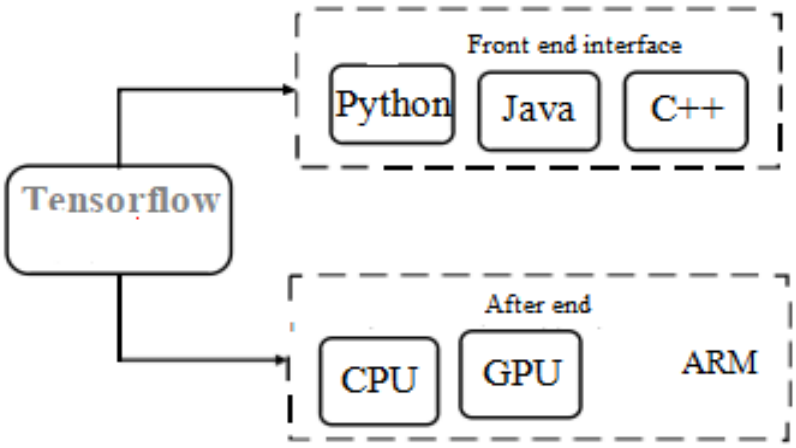

Figure 4.1 - Tensorflow Infrastructure

4.1.2 Model deployment and environment building

The three low-power embedded development devices of NVIDIA Jetson Nano, Raspberry Pi 3B + and KD 233 development suite used in this paper all need to go through the following four steps to carry out environment construction before executing the deep learning detection model:

– write the operating system of each low-power consumption equipment;

– install the sklearn, opencv, pandas, and other base dependency libraries;

– configure the deep learning running environment, which is the installation of the Tensorflow deep learning framework;

– install the forward inference acceleration library, install the NCNN library in the Raspberry PI 3B +, KD 233 development suite, and install the TensoRT library in NVIDIA Jetson Nano.

The three low-power embedded development devices of NVIDIA Jetson Nano, Raspberry Pi 3B + and Kanzhi KD 233 development suite used in this study generally need to go through the following four steps to build the environment before executing the deep learning detection model.

Figure 4.2 - The Jetson Nano Development Board

Through the realization of image recognition, object detection and positioning, semantic segmentation, video enhancement and intelligent analysis of the strong Big functions, these networks can be used to build complex artificial intelligence IoT systems.

The following is a brief introduction to the Jetson Nano development board environment installation, mainly including system burning, tensorflow environment installation and TensorRT library installation.

The first is the installation of the Jetson Nano system. The official website provides three types of system mirror files based on a variety of systems. All of our programs are written under the windows system. After installing the SD Card Formatter software, we can then download the corresponding mirror files from the official website and decompress them. Then insert the SD card into the card reader and connect to the computer, and use the SD Card Formatter software to complete the formatting of the SD card. (select Quick format and click Format to realize the formatting operation).

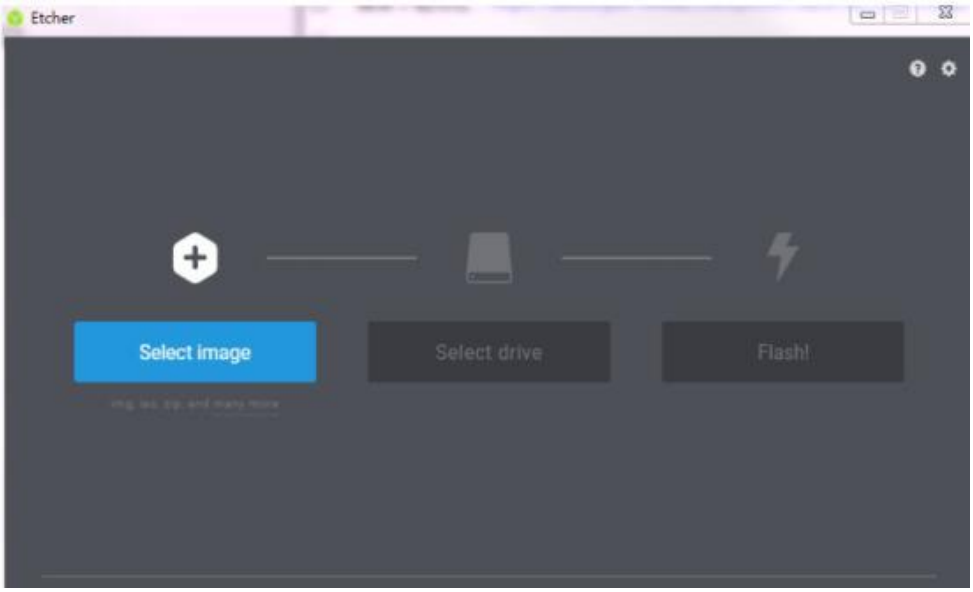

Figure 4.3 - Start interface of Ether Software

After forming the SD card, then use Etcher software to write the system mirror, about 30 minutes. Etcher The beginning of software firing is shown in Figure 4.3, and the end of firing is shown in Figure 4.4.

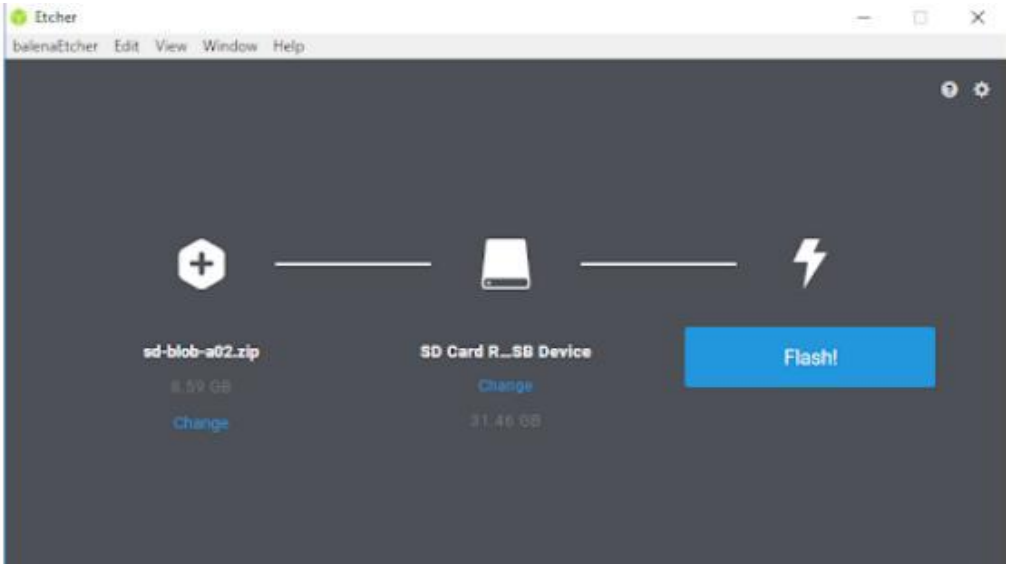

Figure 4.4 - Complete interface of Ether software

After burning the system mirror file to the SD card, insert the SD card into the Jetson Nano development board and connect the required external equipment (development board switch button, shell, camera, antenna, display, etc.) and press the power button to enter the system interface as shown in the following figure below.

Figure 4.5 - Power display screen of Jetson Nano development board

Then is the installation of Tensorflow deep learning framework, open the terminal interface and execute the following commands: sudo apt-get install python3-

pip libhdf5-serial-dev hdf5-tools pip3 install --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v42 tensorflow-gpu==1.12.1

After executing the above command and install the TensorFlow GPU version, then test the TensorFlow GPU version using python3.6 at the terminal and enter the following instructions:

```
import tensorflow astf
a = tf.constant(1.)
b = tf.constant(2.)
print(a+b)
print('GPU:',tf.test.is_gpu_available())
```

The final test results indicate that the tensorflow installation is successful below.

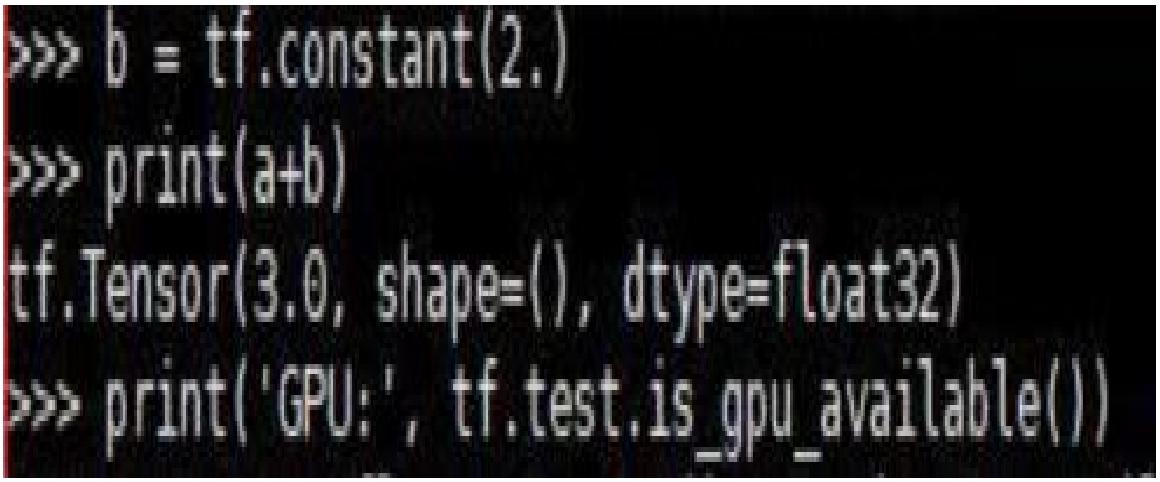

Figure 4.6 - Tensorflow Test Results

TensorRT Installation and testing. Before installation, give a brief introduction to the TensorRT library. NVIDIA The TensorRT framework launched by the company is essentially a C + + library, which can accelerate the reasoning and analysis of input images by graphics processing units. Different from conventional deep learning frameworks (such as TensorFlow and PyTorch), TensorRT is often deployed on low-power devices to accelerate trained models for classification,

detection, semantic segmentation and other visual tasks.

TensorRT has been integrated into various frameworks like TensorFlow and MATLAB to improve the operation efficiency of the framework. Combined layer and kernel optimization decisions are important means for TensorRT framework to improve network, improve efficiency and reduce energy consumption. TensorRT The introduction of the framework will inevitably affect the accuracy, and will also bring less resource consumption. Based on the model trained by the existing deep learning framework, the TensorRT framework can be compressed and optimized to improve the operating efficiency. Finally, the generated lightweight model can be deployed on the low power consumption equipment to improve our work efficiency and quality of life.

Due to the use of the official JetPack 4.4 version mirror in the previous operating system installation process, the system has built-in TensorRT 7.1.3 environment, and test the dpkg-l | grep TensorRT instruction in the termina interface. Figure 4.7 is the return interface after the final execution.

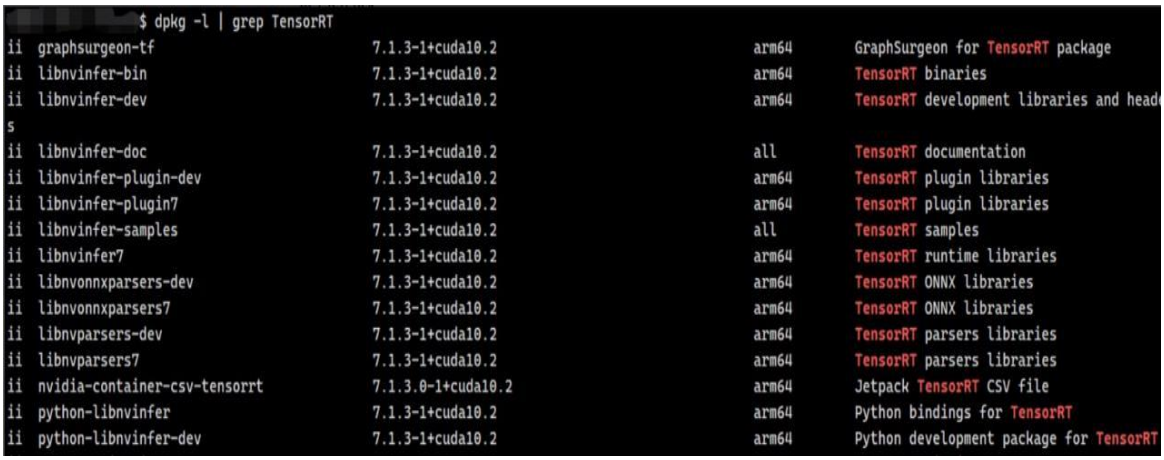

Figure 4.7 - TensorRT Test Results

4.2 Data set preparation

Target detection is one of the directions that researchers pay more attention to in computer vision recognition, and many high-precision target detection algorithms are effectively proved based on public data sets (such as PASCAL VOC[42] and ImageNet[1]). Therefore, this paper using the most authoritative data set in the field of computer vision to obtain training model and verify the algorithm proposed in this paper, then use Oxford-17 class flower data set, Marine biological data set, fracture data set for the former transfer learning of the model and model deployed in me, the previous three low power equipment, through comparison experiment further verify our improved lightweight target detection algorithm in a variety of application scenarios of accuracy, speed and power consumption. The information of all the datasets used in this article is summarized in Table 4.1.

PASCAL VOC Data set includes 20 categories, including people, tables and chairs, as shown in Figure 4.8. Pictures are mainly from various life scenes, including pictures with different resolution, perspective and environment. There is also the phenomenon of multiple target objects overlapping in the same picture, which is challenging to identify.

Figure 4.8 - Legend to the PASCAL VOC dataset

Table 4.1 - Summary table of data set information

Data set / Quality	VOC 2012 Data collection	Oxford flower	Marine biological concrete crack	Distress in concrete crack
The original quantity	11540	1360	1722	1022
Type	20	17	20	1
Data augmentation	No		Left and right mirror images, randomly cropping, adding Gaussian noise, rotation (Mixup)	
Extended quantity	11540			
Original annotation situation	Marked	8517	10610	5610
Data sources	PASCAL *VOC* Challenge	Not marked Oxford University is publicly available Dataset, self Collection supplement	Not marked Future Cup University AICompetition data set, self-directed Collection supplement	Not marked Public data set of Beijing Institute of Technology Collect and supplement by self

The second dataset used in this study is the dataset containing 17 categories of flowers collected by the Oxford University team, with 1360 images. Although the data set is diverse, the number of classes is extremely unbalanced and challenging. Moreover, the small number of samples can easily cause overfitting of the network. Therefore, on the basis of the original public data set, mobile phones are used to shoot related flowers, and in addition, Google, Baidu, and Bing search engines are used to collect supplementary data sets. In order to avoid the problem of overfitting caused by using a small amount of data to train the target detection algorithm, we used the above data augmentation method to expand the data set to 8517 pieces, and the flower data set is shown in Figure 4.9.

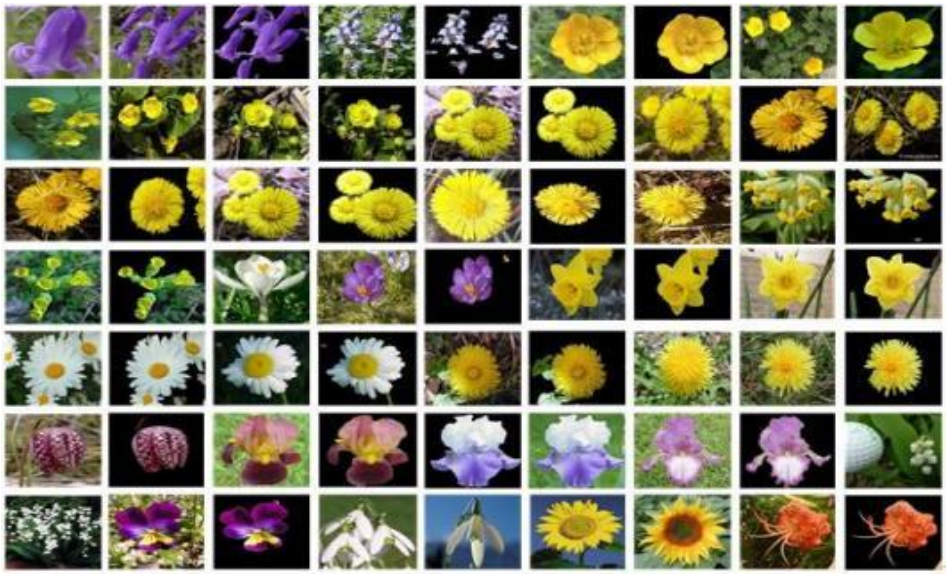

Figure 4.9 - Legend of the Oxford-17 Flower dataset

The third dataset used in this study is the publicly available Marine biology dataset in the 2021 Future Cup University AI Competition, and Figure 4.10 is the legend sample of this Marine biology dataset. This dataset contains 20 classes of marine organisms, with a total of 1,722 images. Due to the small amount of data in the data set, and the difficulty of image acquisition in the actual scene, and the background information in the image is complex, it is difficult to distinguish from the target object, so it is also difficult to detect. In addition, due to the insufficient number of data sets, in addition to collecting part of the data sets, it also used data enhancement means, and the final data set was expanded to 10610 pieces.

Figure 4.10 - Legend for the Marine life dataset

The fourth dataset used in this study is the publicly available fissure dataset [29] of Beijing Institute of Technology on github. Figure 4.11 is a legend sample of the concrete crack dataset with a total of 1022 images. Although this data set has only one category, namely concrete crack, but the concrete crack images are affected by weather, shooting Angle, sunlight and other environmental factors, so the background information of cracks in the actual picture is complex and diverse. In order to get a better detection results, we also took the same method to expand the data set, and finally reached 5610 pieces.

Figure 4.11- Legend of the concrete crack data set

Except for VOC, the other data sets we use are all unannotated images, and these unannotated data need to be used by pre-trained models for transfer learning. Therefore, after data enhancement of these images and annotated by LabelImg software, the number of processed images can meet the requirements of transfer learning. In addition, our data enhancement includes (mixed sample) mixup, left and right mirror image, random cropping, adding Gaussian noise, rotation and other operations, and the number of processed images increases exponentially.

4.3 Experimental test process

The data volume of ImageNet[1] has reached one million levels, and training the model from zero inevitably consumes a lot of hardware computing power and time. Therefore, this paper optimizes the performance improvement based on the pre-training model, and uses the PASCAL VOC[42] data set to complete transfer learning to obtain a more robust model. A total of 11540 VOC 2012 images were used as the training set, and then 4952 images were randomly selected and a part of each image was randomly intercepted as the test set. Feed training set into the third chapter shows the overall network structure, after 7000 rounds of iterative training for not model compression bloated model, and then to the model parameter pruning and int8 fixed quantification, finally after the compression processing model respectively deployed on three low power embedded devices and frame rate, accuracy, detection time analysis and comparison.

In addition, because the data amount of flower data set, Marine biology data set and fracture data set is relatively small, the model performance obtained from restart training will be affected. Therefore, based on the model obtained in the previous step, this study uses three datasets to learn the model and test it on the low power development suite. The main testing process for the low-power equipment is as follows:

- download and burn the system files for the development board;
- the dependency environment required to run the model, such as tensorflow, opencv, etc;
- the trained solidified weight models are deployed on the three embedded development suites and converted into executable file forms on the corresponding low-power devices. For example, the onnx model is loaded with TensorRT on Jetson nano to achieve inference acceleration;
- finally, according to the deployed model, the actual low-power equipment is tested, mainly using real-time cameras and static pictures to test and record the

corresponding detection time-consuming statistics.

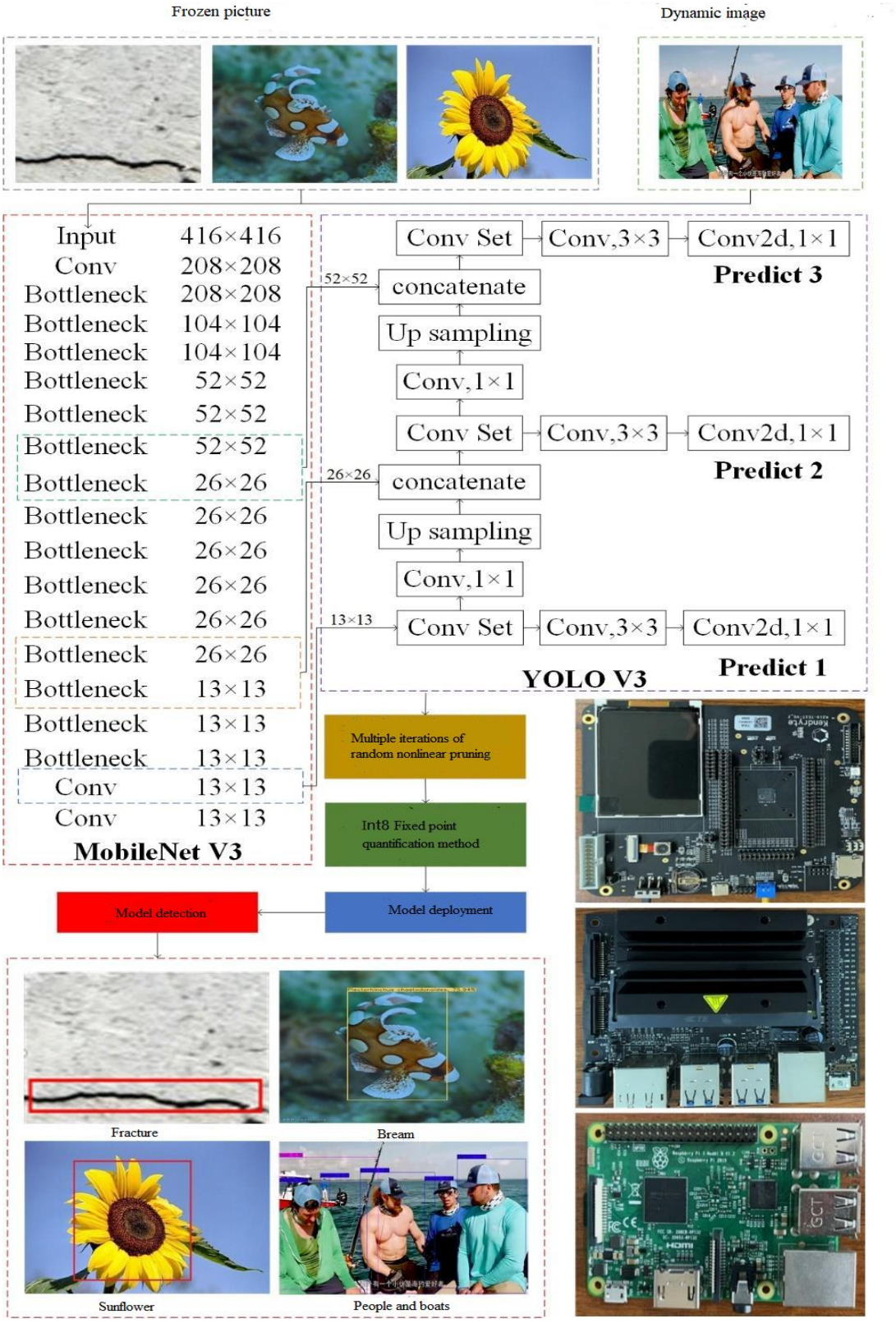

Figure 4.12 - Overall test process

Figure 4.12 shows the overall test process of the experiment in this paper. First

our test data input mainly includes static images and video images (local video, real-time video), through the MobileNetV3 + YOLOV3 algorithm training model, and use the parameter pruning and int8 fixed-point quantification method compression model, and then based on the four data set training model respectively deployed in NVIDIA Jetson Nano, raspberry pie 3B + and can chi KD 233 development suite, the final statistics of the equipment under a single thread actual running time, frame rate and accuracy. Before the model is deployed, the environment needs to build the operation of the target detection model, and the relevant details of the environment deployment can be answered in section 1 of this chapter.

4.4 Analysis of test results of low-power consumption equipment

The average accuracy mAP (mean Average Precision) is the evaluation standard for the detection accuracy of all the experiments conducted in the multi-device and multi-datasets. Since the emergence of PASCAL VOC data set in the International Vision Computer Competition, the average accuracy mAP has become one of the international common evaluation standards. Usually, in this study, multiple images can be input into the target detection algorithm to extract the object-related information in the image (i.e., category, category confidence, coordinate) [69]. The coordinate information can help the program to draw the box where the object is located. If an object is detected from the input image and the confidence of the object is more than 50%, the detection algorithm is detected correctly, otherwise it is false detection. In the actual test, we set a threshold to judge the confidence, and those with confidence greater than g are positive, and those with confidence less than g are negative. Positive samples are detected accurately as true positive, otherwise as false positive. Negative samples were detected as false true negatives, otherwise pseudcases. Accuracy and recall were confirmed from the above variables, defined as follows:

$$\text{Accuracy} = \frac{\text{Number of true positives}}{(\text{Number of true positives} + \text{Number of false positives})} \quad (4.1)$$

$$\text{Recall} = \frac{\text{Number of true positives}}{(\text{Number of true positives} + \text{Number of false negatives})} \quad (4.2)$$

Training on the server side of multiple GPU runs in synchronous mode, with an initial learning rate of 0.001, batch_size of 64, decay rate =0.85, and momentum coefficient a=0.9. After 12,000 rounds, then pruning and quantification, we finally obtain four models (myVOC-all, myFLOWER-all, myLIFE-all, myCRACK-all). The performance test results of the four models under the respective datasets are shown in Table 4.2. The running frame rate pairs of the four models deployed on the Raspberry PI 3B +, NVIDIA Jetson Nano, and KD 233 development suite are shown in Figure 4.13.

Table 4.2 - The Comparison Table of the Performance Evaluation of the Object Detection Model

Detection model	mAP	BFLOPS	Model size(MB)
myVOC-all	82.88%	5.58	8.9
myFLOWER-all	81.23%	5.43	9.2
myLIFE-all	85.19%	5.66	6.8
myCRACK-all	86.79%	5.72	7.6

In this study, the model detection effect of MobileNet V3 + YOLO V3, CPU feedforward time, and running frame number under a single thread were tested on three low-power embedded ARM development platforms. Early after the data set collection and annotation, network selection, and then trained on the server network weight parameters, finally using the model pressure, shrink technology run faster, little effect on the accuracy of the small model and deployed on three embedded development suite, get the final visual test rendering.

Figure 4.13 - Example of VOC 2012 dataset detection based on low-power embedded devices

Except for three models based on the VOC 2012 data set (based on Marine organisms, flowers, concrete cracks), because it is difficult to find test object categories in real life, the three models take static images for accuracy and time-consuming tests. The model based on the VOC 2012 data set adopts the local video and the real-time video test, and the results of a certain frame of the video detection test are shown in Figure 4.14. The experimental results show that the actual test frame rates on NVIDIA Jetson Nano, Raspberry Pi, 6 B + and KZ reach 25 fps on the KD 233 development suite, respectively.

As shown in Figure 4.15, the feature extraction network of the YOLO V3 detection algorithm is improved to reduce the model size by replacing the MobileNetV3 Large network. Compared with the YOLO-Nano algorithm, the NVIDIA Jetson Nano speed is nearly doubled, and the other two devices are also somewhat faster. The average of the single-frame test results of the target detection model based on the VOC dataset, run multiple times on the NVIDIA Jetson Nano embedded development board, resulted in approximately 40 ms. Jetson Nano The embedded development board is more than twice faster than the Raspberry Pi, while the KD 233 development suite runs faster to 35 milliseconds. This shows the advantages of the method proposed here in this paper.

The results of model based on Marine life dataset are as follows (Figure 4.16).

Figure 4.16 - Example of Marine biological detection based on low-power embedded devices

The results of model based on fracture data set are as follows (Figure 4.17).

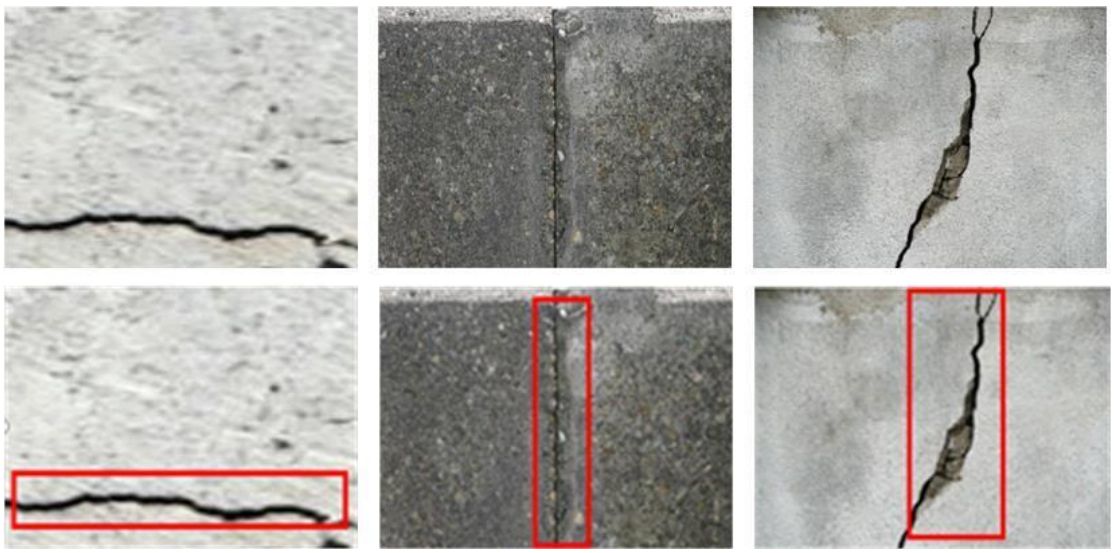

Figure 4.17 - Examples of concrete crack detection based on low-power embedded equipment

The results of the model static picture test based on the Oxford 17th class flower data set are as follows (Figure 4.18).

Figure 4.18 - Example of flower detection based on low-power embedded equipment

Finally, in this study, the detection images in Figure 4.13, 4.16, 4.17 and 4.18 show that the optimized YOLO V3 network model can adapt to multi-morphic and multi-scale images. The MobileNet V3 + YOLOV3 can also identify multiple objects in the image. In addition, as can be seen from Figure 4.13, our algorithm can run smoothly on NVIDIA Jetson Nano, Raspberry PI 3B + and KD 233 development suite.

CONCLUSION

The continuous improvement of the performance of target detection algorithms also leads to the synchronous rise of model complexity, which ultimately leads to the failure of these network algorithms to apply [66] in practical engineering. Therefore, other excellent lightweight models such as MobileNet series and ShuffleNet series were put forward to balance the relationship between time and accuracy. These algorithms in high power consumption of computer or server has been able to run in real time, but can not meet the flow on low power consumption equipment, line, to improve the feasibility of low cost, low power consumption of embedded equipment, can compress model from the algorithm level, reduce the required requirements, and can also be optimized from the embedded equipment hardware level. This paper mainly optimizes from the algorithm level to achieve the purpose of model compression. Using the lightweight Mobilenet V3 algorithm to optimize the YOLOV3 algorithm, the lightweight target detection algorithm for different application scenarios is realized on the TensorFlow framework platform. Finally, the target detection algorithm was deployed to three embedded mobile devices on NVIDIA Jetson Nano, Raspberry PI 3B + and KD 233 development suite. The test results showed that it could find a balance between accuracy and speed to meet the real-time application requirements.

At present, almost all target detection algorithms rely on the cloud server, and then the server-side detection results are transmitted to the mobile phone through the network. The biggest disadvantage of this mode is that the cost is high and requires high Internet transmission efficiency. Therefore, deploying target detection algorithms to local low-power devices is the future direction, especially in the smart Internet of Things and mobile phones. The existing high-precision target detection algorithms are too bloated and lead to a huge increase in computational volume. Therefore, the deployment of such algorithms requires devices with higher power consumption and stronger performance, especially computing devices (GPU,

CPU, etc.). However, the existing low-power embedded devices and other devices limit the output of computing power due to the low power consumption. As a result, when the detection algorithm is deployed in the actual low power device, the feasibility of real-time detection is still not high. Therefore, this paper is based on the VOC 2012 data sets, Oxford-17 class flower data sets, Marine biological data sets, fracture data sets, the target detection algorithm deployed to low, power consumption embedded computing platform, and adopt a series of model optimization strategy to reduce the model parameters at the same time reduce the accuracy loss, speed up the forward reasoning time. The research work and the research results of this paper are as follows.

In the target detection algorithm, we add the MobileNet V3 Large network to the YOLO V3 algorithm to obtain the front-end features. The improved target detection algorithm is trained on the TensorFlow deep learning framework platform, handles parameter pruning and model quantification, and transforms the model using the forward inference framework, and finally integrates the model size of BN, Scale and ReLU operators is 8.2M.

In view of the limited computing power of low-power consumption equipment, this paper uses the lightweight MobileNet V3 Large network to obtain the front-end features. The MobileNet-V3 network depth decomposition convolution replaces the traditional convolution operation to achieve the effective decrease of the number of parameters and the model complexity. Meanwhile, the reverse residue structure feature representation in the MobileNet-V3 algorithm is more robust. In addition, MobileNet-V3 networks perform very evenly in scale, speed and accuracy, and are the most suitable network for many lightweight networks to be deployed in low-power embedded devices.

Model training and optimization aspects. A gradient optimization method to control the learning rate is adopted to train the network, and the acquisition model is also trained on a high-performance server with multi-GPU. In addition to the VOC 2012 data set, other data not only adds data enhancement to realize the expansion of small data set, but also use transfer learning to improve the network performance.

In terms of model compression and quantification, multiple iterative random nonlinear pruning strategy is used to complete the parameter pruning of the target detection model, and the int8 fixed-point quantization method is adopted to achieve further compression of the model. Compared with the conventional reasoning method of float32, int8 fixed-point quantization method has higher efficiency in realizing pure integer forward inference, and the accuracy loss is about 10% compared with the reasoning method of float32, which effectively realizes the optimal balance of model inference speed and parameter scale.

Install the Tensorflow Lite open source framework on the low-power embedded device side and deploy the compressed models. After training four types of models with the VOC 2012 dataset, Oxford. 17 flower dataset, Marine biology dataset and using the proposed model compression scheme, the average single-frame prediction time was 165ms, 40ms and 35ms on NVIDIA Jetson Nano, KD 233 development suite, respectively, and mAP was 84.02% on the VOC 2012 dataset, Oxford. 17 flower dataset. The experimental results show that the target detection algorithm designed based on low-power devices can achieve model scale, speed and accuracy when deployed on low-power embedded devices optimization.

This study is mainly to optimize the YOLO V3 algorithm for lightweight level and realize the smooth operation of low-power embedded devices, so there is no high requirement on the results in model training. It is mainly limited by time and experimental equipment resources, and there are still many problems to be followed up. Future work will focus on the following areas:

In this study, only the target detection based on MobileNetV3 lightweight network was realized, without the verification of ShuffleNet series, PeleeNet, Xception and other lightweight networks. If the time is sufficient, the identification based on other algorithms will be realized, and the performance evaluation and comparative test are completed with the lightweight target detection algorithm adopted in this paper. The target detection algorithm only adopts the YOLOV3 algorithm, and has not yet compared the relevant performance parameters with other single-stage algorithms. Therefore, the model design of the latest lightweight feature

extraction network and the target detection algorithm can be used in the future.

In addition, the four training data sets selected in this study are VOC 2012 data set, flower data set, concrete crack data set and Marine biology data set, and the other four images except VOC 2012 are small. Moreover sample data background information is relatively single, and for what we see with the target object image, the background information, is often complex, even appear the same image contains multiple difficult to distinguish the target, so even in the four own data set accuracy again high, for the actual landing of the product still have gap.

Then, our goal detection scheme on low-power devices is only studied from the deep learning algorithm level. Follow-up research can also combine field programmable gate array technology (Field-Programmable Gate Array, FPGA) to improve the efficiency of deployment and execution of deep learning algorithm at the hardware level and realize the combination of software and hardware.

REFERENCE

1) Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems,2012, 25: 1097- 1105.

2) K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arxiv preprintarxiv:1409.1556,2014.

3) Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. PMLR, 2015: 448- 456.

4) He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C].Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

5) Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

6) REN S, HE K, GIRSHICK R. Faster R-CNN: Towards ReaL-Time Object Detection with Reg ion Proposal Networks[C]//Proceedings of International Conference on Neural Infor mation Processing Systems, MIT Press, 2015:91-99.

7) REDMON J, DIVVALA S, GIRSHICK R, et al. You OnLy Look Once: Unified, ReaL-T ime Object Detection [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016:779-788.

8) Cai Z, Vasconcelos N. Cascade r-cnn: Delving into high quality object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6154-6162.

9) Woo S, Park J, Lee JY, et al. Cbam: Convolutional block attention module[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 3- 19.

10) Zhu Y, Newsam S. Densenet for dense flow[C]//2017 IEEE international

conference on image processing (ICIP). IEEE, 2017: 790-794.

11) Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

12) Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.

13) Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 1314- 1324.

14) Chollet F. Xception: Deep learning with depthwise separable convolutions[C].Proceedings of the IEEE conference on computer vision and pattern recog-nition. 2017: 1251- 1258.

15) Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.

16) Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1492- 1500.

17) Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.

18) Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 116- 131.

19) Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arxiv preprint arxiv:1602.07360, 2016.

20) Wang R J, Li X, Ling C X. Pelee: A real-time object detection system on mobile devices[J]. Advances in Neural Information Processing Systems 31, Curran

Associates, 2018,Inc.pages 1963- 1972.

21) HanK, Wang Y, Tian Q, et al. Ghostnet: More features from cheap operations[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 1580- 1589.

22) Tan M, Chen B, Pang R, et al. Mnasnet: Platform-aware neural architecture search for mobile[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 2820-2828.

23) He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9): 1904- 1916.

24) Zhang S, Wu Y, Men C, et al. Tiny YOLO optimization oriented bus passenger object detection[J]. Chinese Journal of Electronics, 2020, 29(1): 132- 138.

25) Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C].CVPR. 2018: 2704-2713.

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

27) Redmon J,FarhadiA. Yolov3: An incremental improvement[J]. IEEE Conference on Computer Vision and Pattern Recognition.2018

28) Bo Wang.AerialCrackDataset: Towards Object Detection with Dataset,Key Labo- ratory of Optoelectronic Imaging Technology and System, Ministry of Education, Sc-hool of Optoelectronics, Beijing Institute of Technology, 2017.

29) Wang Zhefeng. Design and implementation of the mobile terminal target detection system [D]. Zhejiang University, 2018.

30) Xiao Guolin. Research on the quantification algorithm of the weight interaction of the deep learning model [D]. Harbin Institute of Technology, 2020.

31) 80Ozge Unel F, Ozkalayci B O, Cigla C. The power of tiling for small object detection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 10.1109/CVPRW.2019.00084.

32) Zhang P, Zhong Y, Li X. SlimYOLOv3: Narrower, faster and better for real-time UAV applications[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. 2019.

33) Huang R, Pedoeem J, Chen C. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers[C]//2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018: 2503-2510.

34) Wong A, Famuori M, Shafiee M J, et al. Yolo nano: a highly compact you only look once convolutional neural network for object detection[J]. arXiv preprint arXiv:1910.01271, 2019.

35) Li Y, Huang H, Xie Q, et al. Research on a surface defect detection algorithm based on MobileNet-SSD[J]. Applied Sciences, 2018, 8(9): 1678.

36) Kharchenko V, Chyrka I. Detection of airplanes on the ground using YOLO neural network[C]//2018 IEEE 17th international conference on mathematical methods in electromagnetic theory (MMET). IEEE, 2018: 294-297.

37) Zhang S, Wu Y, Men C, et al. Tiny YOLO optimization oriented bus passenger object detection[J].Chinese Journal of Electronics,2020,29(1):132- 138.

38) LeCunY, Denker J S, Solla S A, et al. Optimal brain damage[C]//NIPs. 1989, 2: 598-605.

39) Dong X, Chen S, Pan S. Learning to prune deep neural networks via layer-wise optimal brain surgeon[J]. Advances in Neural Information Processing Systems. 2017. 4857-4867.

40) Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[J]. Advances in Neural Information Processing Systems. 2015. 1135- 1143.

41) Lee N, Ajanthan T, Torr PHS. Snip: Single-shot network pruning based on connection sensitivity[J]. arxiv Preprint arxiv: 1810.02340, 2018.

42) Everingham M, Eslami S MA, Van Gool L, et al. The pascal visual object classes challenge: A retrospective[J]. International journal of computer vision, 2015, 111(1): 98- 136.

43) Mariet Z, Sra S. Diversity networks: Neural network compression using

determinantal point processes[J]. arXiv preprint arXiv:1511.05077, 2015.

44) Carreira-Perpinán M A, Idelbayev Y. “learning-compression” algorithms for neural net pruning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8532-8541.

45) Liu Z, Xu J, Peng X, et al. Frequency-domain dynamic pruning for convolutional neural networks[C],Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018: 1051- 1061.

46) Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks. Advances in Neural Information Processing Systems. 2016. 2074-2082.

47) Gavai N R, Jakhade Y A, Tribhuvan S A, et al. MobileNets for flower classification using TensorFlow[C]//2017 International Conference on Big Data, IoT and Data Science (BID). IEEE, 2017: 154- 158.

48) LiH, KadavA, DurdanovicI, et al. Pruning filters for efficient convnets[J]. arxiv Preprint arxiv: 1608.08710, 2016.

49) Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient transfer learning[J]. arxiv Preprint arxiv: 1611.06440, 2016.

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

51) Courbariaux M,BengioY, David JP. Binaryconnect:Training deep neural networks with binary weights during propagations[J]. Advances in Neural Information Processing Systems. 2015. 3123-3131.

52) Yang Z, Moczulski M, Denil M, et al.Deep fried convnets[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015:1476- 1483.

53) Liu Z, Wu B, Luo W, et al. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 722-737.

54) Rastegari M, Ordonez V, Redmon J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks[C]//European conference on computer vision. Springer, Cham, 2016: 525-542.

55) Li F, Zhang B, Liu B. Ternary weight networks[J]. arXiv preprint arXiv:1605.04711, 2016.

56) Han S,Mao H,Dally WJ.Deep compression:Compressing deep neural networks with pruning,trained quantization and huffman coding[J]. arxiv Preprint arxiv:1510.00149,2015.

57) Gong Y, Liu L, Yang M, et al. Compressing deep convolutional networks using vector quantization[J]. arxiv Preprint arxiv: 1412. 6115, 2014.

58) Lin Z,Courbariaux M,Memisevic R,et al.Neural networks with few multiplications[J].arxiv Preprint arxiv:1510.03009,2015.

59) Wang N, Choi J, Brand D, et al. Training deep neural networks with 8-bit floating point numbers[J]. Advances in Neural Information Processing Systems.2018.7675-7684.

60) Faraone J, Fraser N, Blott M, et al. Syq: Learning symmetric quantization for efficient deep neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4300-4309.

61) Zhang D, Yang J, Ye D, et al. Lq-nets: Learned quantization for highly accurate and compact deep neural networks[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 365-382.

62) DubeyA, Chatterjee M, AhujaN. Coreset-based neural network compression[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 454-470.

63) Han S, Liu X, Mao H, et al. EIE: Efficient inference engine on compressed deep neural network[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243-254.

64) Masana M, van de Weijer J, Herranz L, et al. Domain-adaptive deep network compression[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 4289-4297.

65) WenW, Xu C, Wu C, et al. Coordinating filters for faster deep neural networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 658-666.

66) Qiu Q, Cheng X, Sapiro G. Dcfnet: Deep neural network with decomposed convolutional filters[C]//International Conference on Machine Learning. PMLR, 2018: 4198-4207.

67) Huang Xuan-kun. Deep learning-based mobile-terminal image recognition algorithm [D]. Beijing University of Posts and Telecommunications, 2018. [69]. Study on pedestrian detection in autonomous driving scenarios [D]. Chongqing University of Technology, 2019.

68) Kong Xue. Research and application of the lightweight target detection algorithm [D]. Northeastern Petroleum University, 2019.

69) Chen Siyu. Research and application of real-time object detection based on arm mobile terminal [D]. University of TC, 2020.