UNIVERSITY OF TURKU

Applications of Intelligent Vision in Low-Cost Mobile Robots

Smart Systems
Master's Degree Programme in Information and Communication Technology
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis

Author:
Yaoyu Zhou

Supervisors:
MSc (Tech) Jorge Peña Queralta
Assoc. Prof. Tomi Westerlund

June 2022

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service.

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Smart Systems
Programme: Master's Degree Programme in Information and Communication Technology
Author: Yaoyu Zhou
Title: Applications of Intelligent Vision in Low-Cost Mobile Robots
Number of pages: 86 pages
Date: June 2022

Abstract.

With the development of intelligent information technology, we have entered an era of 5G and AI. Mobile robots embody both of these technologies, and as such play an important role in future developments. However, the development of perception vision in consumer-grade low-cost mobile robots is still in its infancies. With the popularity of edge computing technology in the future, high-performance vision perception algorithms are expected to be deployed on low-power edge computing chips. Within the context of low-cost mobile robotic solutions, a robot intelligent vision system is studied and developed in this thesis.

The thesis proposes and designs the overall framework of the higher-level intelligent vision system. The core system includes automatic robot navigation and obstacle object detection. The core algorithm deployments are implemented through a low-power embedded platform.

The thesis analyzes and investigates deep learning neural network algorithms for obstacle object detection in intelligent vision systems. By comparing a variety of open source object detection neural networks on high performance hardware platforms, combining the constraints of hardware platform, a suitable neural network algorithm is selected.

The thesis combines the characteristics and constraints of the low-power hardware platform to further optimize the selected neural network. It introduces the minimize mean square error (MMSE) and the moving average minmax algorithms in the quantization process to reduce the accuracy loss of the quantized model. The results show that the optimized neural network achieves a 20-fold improvement in inference performance on the RK3399PRO hardware platform compared to the original network.

The thesis concludes with the application of the above modules and systems to a higher-level intelligent vision system for a low-cost disinfection robot, and further optimization is done for the hardware platform. The test results show that while achieving the basic service functions, the robot can accurately identify the obstacles ahead and locate and navigate in real time, which greatly enhances the perception function of the low-cost mobile robot.

Keywords: mobile robot; intelligent vison; object detection; deep learning.

Table of contents

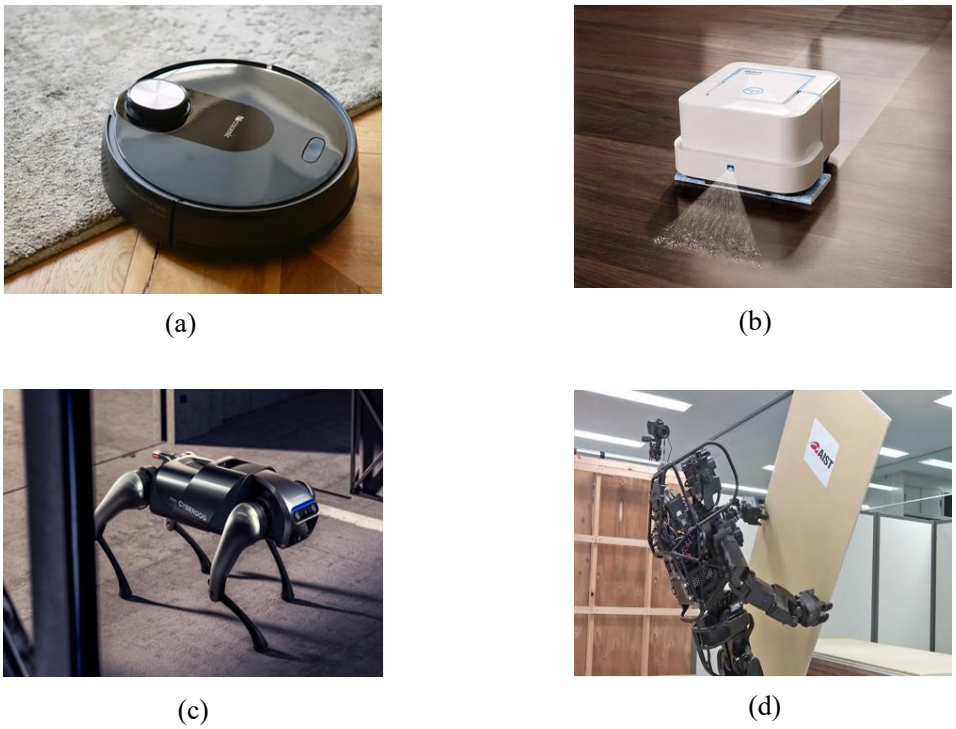
1 Introduction

1.1 Background

With the rapid development of science and technology, after the advent of 5G and the development of Artificial Intelligence (AI) in recent years, we have entered a technological era of 5G and AI, and our lives are full of smart devices equipped with 5G and AI. Mobile robots embody both of these technologies, and as such play an important role in future developments. There are many application scenarios in the development of robots, from wheeled mobile robots on the ground to flying robots in the air and robots on the water. These robots can efficiently complete the established tasks. There are industrial robots in various factory floors, bionic robots in the medical field and anthropomorphic robots or smart dogs in research fields which are developed by companies such as Boston Dynamics. In addition, there are some robots that are targeting the consumer level, which are very popular in recent years. These robots are mainly service-type, and they account for more and more proportions in the market, including sweeping robots, disinfection robots, food delivery robots, etc. A selection of popular mobile robotic platforms are shown in Figure 1-1. These robots can perform tasks including logistics, transportation, sanitation and entertainment[1]. They can autonomously complete service work which is beneficial to humans without too much intervention, and the field of service types is too wide, making them become extremely important in the future. At the same time, with the continuous research and development of Internet of Things (IoT) technology and smart lives, a large number of AI technologies have been deployed on IoT devices. Mobile robots with intelligent interactive features are entering the consumer market, and they are becoming more and more extensive in people's daily lives[2].

The core technology of mobile robots revolves around the three aspects of perception, decision-making and execution. These three architectural pillars include more detailed aspects such as path planning, control, environmental perception, mechanical structure design and other fields. The perception is mainly the robot's understanding of the unknown environment. About how to feel surroundings around the robot, including the depth information and feature information of the target object. This aspect is mainly realized by the sensors and the algorithm which processes the back-end data[3].

Decision-making means that after the robot perceives external information, through internal program settings and algorithms, the robot itself makes a decision on how to act in the next step. Next step is execution. Execution is to control the robot itself according to the results of the decision-making. It mainly includes two types, one is motion control, which is the speed, and the other one is dynamic control, which is the driving torque[4]. In these three aspects, perception is the most core technology. Decision-making and execution are firmly dependent on the robot's perception. Perception mainly relies on the robot's vision system. Therefore, the development of robot vision system has become particularly important[5].



(a) (b)

(c) (d)

Figure 1-1 Application of Robots (a) Proscenic's sweeper robot (b) iRobot's sweeper robot (c) Xiaomi's CyberDog (d) AIST's HRP-5P

At present, robots have a wide range of applications in all walks of life, especially consumer-level robots appear more and more frequently. Therefore, further application research based on robots has become extremely important. In all aspects of robots, perception is particularly important. In view of this, based on a general low-cost consumer-grade mobile robot platform, the thesis has developed an upper-level intelligent visual system, combined with deep learning technology in AI, Simultaneous Localization and Mapping (SLAM) technology and edge computing platforms. With this intelligent vision system, traditional vision system which rely on infrared and

ultrasonic sensors has become more intelligent. Mobile robots can realize real-time obstacle detection, localization and automatic navigation function while finishing basic service functions efficiently.

1.2 Review of Current Research

1.2.1 Mobile Robot Applications

Many researchers and engineers are currently working in the field of mobile robots. The proportion of robots in the market is increasing because there are variety of application scenarios. In industrial-level scenario, mobile robots can be used to perform automatic operations and intelligent transportation in the factory floor. The position and navigation of the robot is a key point in this application[6-7]. While the robot not only position itself, it also needs to coordinate with other robots. This function has higher requirements on the robot's vision perception and decision-making system[8]. When transporting goods autonomously, mobile robots will simulate human hands to carry out the function of grabbing goods. Some mobile robots will adopt a feature of rigid hands. B.Fang et al. designed a mobile robot that uses soft hands to do it, so that the robot’s execution can be better integrated with vision perception[9]. For the application scenarios of flying robots in the air, A. Sánchez-Orta used a monitoring flying robot to obtain better visual perception to assist the decision-making of the autonomous vehicle[10]. W. Tabib used autonomous flying robots to perform cave detection. These robots are equipped with depth cameras so they can obtain both image and distance information. This robot vision perception performs better detection function and reduce the risk of people working in such dangerous situations[11].

In the medical field, Z.LI et al. used brain-computer interfaces to assist some patients suffering from certain diseases or nerve loss. They developed an exoskeleton robot that can be controlled through the patient's own thoughts[12]. The bionic robot designed by G.FICHT can coexist with humans in this society. This kind of humanoid robot can replace humans to complete tasks in many dangerous scenarios[13]. There are also application scenarios different with mechanical work. M. Langen et al. designed a bionic robots which can be used as a companion for many lonely people for they have AI-based emotional processing capabilities[14].

In daily life scenarios, people may be more concerned about the interaction between

robots and humans[15]. L. Pang et al. used mobile robots to perform automatic following job for human beings. There are many potential applications for following robots, including auto-following suitcases, shopping carts and assistance for the elderly and disabled people[16]. J. Chen et al. developed one type of following robots which can recognize different human gestures and have different interactions based on these gestures. This type of following robot mainly considers three aspects. Perceptual detection of human interaction, positioning of unfamiliar environment and own decision-making[17-18].

At present, because of the global epidemic of covid-19 virus which began in 2020, almost all countries on all continents in the world have been severely affected, and people's daily lives have undergone great changes[19]. The application of disinfection robots has emerged on the market. Since the process of using a large amount of chemicals to disinfect the environment will seriously threaten human health and safety if it is done manually, the solution of using mobile disinfection robots is a very effective tool against viruses[20-21]. Ackerman et al. developed large-scale disinfection robots deployed in hospitals[22], and Guettari et al. designed small household disinfection robots which use UVC light as disinfection tools[23]. With the emergence of these disinfection robots, people's health has been further guaranteed in today's epidemic situation.

(a) (b)

Figure 1-2 Disinfection Robots (a) UBTech's Adibot (b) CloudMinds's CCLE-GA-2-50

Although there is a lot of research on the application of robots at home and abroad, most of the researchers and engineers mainly focus on large-scale industrial scenes. The research on consumer-grade mobile robots in daily lives is relatively small and the development of robots mainly revolve around the decision-making and execution aspects. Further work is needed for the development of robot visual perception.

1.2.2 AI on Robotics

AI has experienced rapid development in this era, which has penetrated into all walks of life in modern society, from industry to medical care, to humanities education, and even to government departments. AI-based systems currently have excellent perception, learning, inference and generalization capabilities. Our daily lives are full of AI[24]. At the same time, the development of deep learning has witnessed a climax in the field of machine learning, brought about a vigorous development of AI applications. Deep learning has had a great impact on many fields, including robotics and smart phones. Among the deep learning in the field of robotics, computer vision and natural language processing are the most popular research areas[25-26]. The application of computer vision and natural language processing based on AI are shown in the Figure 1-3 and 1-4.

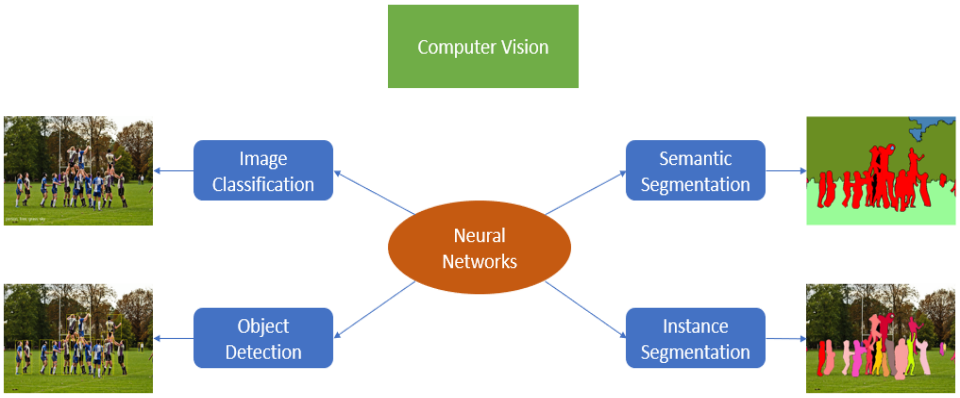


Figure 1-3 Computer Vision in AI

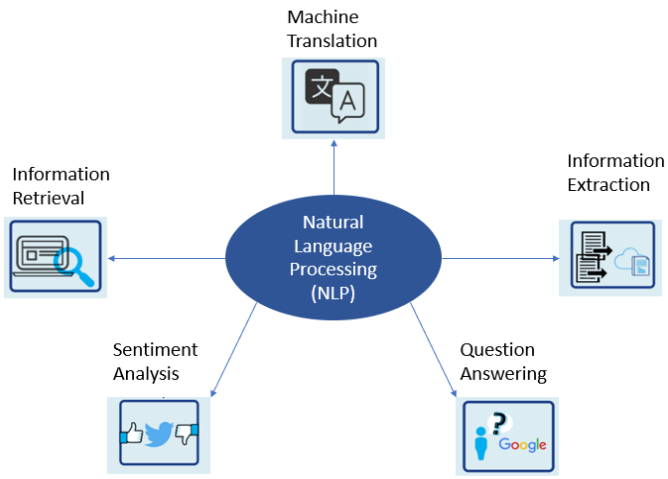


Figure 1-4 Natural Language Processing in AI

The application of deep learning on mobile robots is becoming an increasingly hot topic. Some researchers have combined the technology of deep learning and reinforcement learning, applied Deep Reinforcement Learning (DRL) to the robot to improve the accuracy of the robot's visual perception and decision-making[27-28]. H. Jiang et al. used DRL to solve the problems encountered by robots in dynamic environment navigation, because generally mobile robots perform positioning and navigation tasks in a relatively static environment, while the dynamic environment is full of uncertainties, utilization of DRL is better to solve this problem[29]. T. Hiejima et al. also applied reinforcement learning to cluster robots, letting clusters to make better decisions when they collaborate efficiently[30]. P. Kirsanov et al. combined deep learning and SLAM technology, using neural network based semantic segmentation and panoramic segmentation to improve the accuracy of visual SLAM mapping[31].

Although the development of AI in mobile robots is becoming more and more popular, most of the researchers focus on the decision-making aspect of robots. There are not many studies on the perception system of robot. While perception has a very important influence on the entire operation of robots. Therefore, the development of intelligent vision system based on machine learning on mobile robots is one of the current research directions.

1.2.3 Era of Edge Computing

With the rapid growth of the consumer electronics industry and continuous development of Information Communication Technology (ICT), from cloud servers to personal computers and smart mobile phones, we are in an era full of information computing. More and more smart devices are developing in the direction of wearable devices and IoT. It is estimated that by 2030, there will be about 500 billion devices connected to IoT. With the emergence of these devices, the demand for computing and storage is also increasing. The traditional server-dependent computing solutions need to undergo a process of transformation and new computing methods need to be proposed[32-34]. Therefore, edge computing has emerged. Edge computing is a new form of computing and different from traditional data processing. In the past, developer always considered uploading the data to a cloud server, processing and storing the data in the cloud, and then sending some required results back to the edge terminal[35-36]. Edge computing refers to data processing and computing directly on the appropriate

device on the edge terminal, without uploading to the cloud. All operations are run locally[37-38].

At the same time, with the application of AI technologies on IoT scenarios, machine learning brings more data processing needs. How to introduce AI into edge computing field has become a hot topic. Edge AI computing is different from traditional machine learning computing. Traditional technology relies on large servers with strong computing power[39]. While neural network inference on IoT devices has many constraints, including the computing power, memory, and input and output throughput limitations of the hardware platform. These all make it difficult to deploy AI neural networks on the edge devices. However, the IoT devices have the advantages of real-time, strong distribution, positioning information, high mobility, etc. Also due to too many IoT devices, it is impossible to upload all edge data to the cloud. Therefore, it is necessary to develop edge AI computing with real-time and high mobility[40-41].

In some edge computing scenarios with small data volumes, it is generally possible to rely on tiny single-chip processors and Real-Time Operating System (RTOS). While this kind of hardware is not enough when AI algorithms need to be deployed on the edge devices, neural networks have computing power requirements for the hardware platform. So, the hardware platform requires a specific architecture design. At present, the hardware for edge AI acceleration is mainly divided into three categories, utilizing CPU, GPU or ASIC for neural network inference. Based on general-purpose CPUs and GPUs, many companies and researchers have developed software acceleration optimizations, including intel's OPENVINO, NVIDIA's cuDNN and TensorRT. For ASICs, there are a variety of hardware structures, including DSP, Neural Processing Unit (NPU), Tensor Processing Unit (TPU), and other hardware acceleration units specifically for neural network algorithms. With the support of these hardware structures, the deployment of AI algorithms on IoT and mobile terminals will become simpler, more feasible and efficient.

Figure 1-5 shows several edge computing platforms which have optimization and acceleration for neural networks inference. The mobile robot intelligent vision system in this thesis also selects the edge AI computing solution, and the selected RK3399PRO platform has an NPU for neural network acceleration.

(a) (b) (c)

Figure 1-5 Different Edge Computing Platform (a) Intel's Neural Compute Stick (b) Google's Edge TPU Platform (c) Nvidia's Jetson Nano

1.3 Thesis Work and Contributions

In this era, as the application of robots in all walks of life becomes more and more widespread, especially consumer-level mobile robots occupy more and more shares in the market, including sweeping robots, guidance robots. And because of the global popularity of Covid-19 virus, disinfection robots are also popular in the market now. The main technology of this kind of robots are three parts: perception, decision-making, and execution. Among these three parts, perception is a particularly important aspect. The latter two depend on the robot's perception of the environment, obstacles and unknown areas. However, most of the existing consumer-grade mobile robots on the market rely on simple infrared and ultrasonic sensors for their perception. They are not as powerful as industrial-grade and bionic robots in terms of perception. This part is still need for development. For this reason, this thesis mainly focuses on the research and design of perceptive function on consumer-level mobile robots, which is the development of the mobile robot's intelligent vision system.

The main work and contributions of this thesis include:

Intelligent vision system frame design: The visual development of mobile robots is the most critical part of the entire system. Only mobile robots with a perfect vision system can successfully complete the tasks set out at the beginning. Based on this, an intelligent vision system is designed and developed on the existing disinfection robot platform. From the construction of sensors to the selection of perception technology, the development of algorithms, the allocation of hardware resources and the communication between different modules. Finally, the architecture design of an upper-

level vision system is completed, and this vision system innovatively introduces SLAM and deep learning neural network technologies.

Development of deep learning neural network algorithm: This is an era of 5G and AI, so the application of AI has penetrated into all walks of life, and the development of deep learning is particularly prominent in AI. Therefore, the obstacle object detection vision module based on deep learning is applied to mobile robots in this thesis. The thesis compares the structure of different object detection neural network algorithms and analyzes the advantages and disadvantages of various tricks. Through construction of own dataset, different algorithms are trained and tested on a high-performance hardware platform. Also, they are deployed to low-power hardware platform for inference verification. The degree of adaptation of these neural networks in the mobile robots and the possibility of further optimization are analyzed according to the training and inference results. Finally, the state-of-the-art object detection neural network YOLOV4, which is not adopted by most developers, is innovatively selected as the main idea of the obstacle detection module in the intelligent vision system.

Neural network optimization for edge deployment: As a smart device of the IoT, a mobile robot will have many restrictions if data processing and neural network inference are uploaded to the cloud server. Meanwhile the application scenario of mobile robot determines its high requirements for real-time performance. In view of this, this thesis uses the edge computing solution to deploy the neural network algorithm and chooses RK3399PRO edge AI platform. Based on the features of the hardware, a series of optimization processes have been done on the neural network structure, including splitting, replacement, and fusion of different operators, conversion of data layout and optimization of activation function and pooling layer. At the same time, the model is quantized and compressed. The main innovations are the utilization of asymmetric affine and dynamic fixed point algorithms for quantization and the minimize mean square error (MMSE) and Moving Average Min Max algorithms are introduced in this process to update the quantization parameters to reduce the accuracy loss of the quantized model. All of these optimizations are completed on ARM platform. At last, the inference performance of the neural network on the RK3399PRO hardware can approach that on the high-performance server.

Finally, an upper-level intelligent vision system with deep learning object detection function and SLAM navigation planning is developed on the mobile robot platform. The optimized object detection neural network can detect and visualize real-time

obstacles in the 1080p60fps video stream captured by the monocular camera, whose inference performance has a 20 times improvement compared to the original YOLOV4 network. The robot's vision system can process data up to 15 frames each second and the detection accuracy for obstacles can reach more than 90%. With this intelligent vision system, the mobile robot can realize real-time obstacle detection, localization and navigation while finishing basic service functions in unfamiliar environments, which significantly enhances the robot's perception.

1.4 Thesis Structure

Chapter 1 contains introduction to the whole thesis, including current research and development of robots, daily life applications, core technologies of mobile robots and combination AI and edge computing technologies. It summarizes the main work and contributions to the development of mobile robot vision system.

Chapter 2 focus on the framework design of the robot intelligent vision system. It analyzes and introduces the two key technologies, describes the construction of sensors of the vision system, the allocation of hardware resources, the selection of perception algorithms and the communication between different modules. Finally, it introduces the architecture diagram of the intelligent vision system.

Chapter 3 covers the research and development of deep learning algorithms on robots. It introduces the principle of object detection algorithms based on deep learning, analyzes its application on mobile robots and compares different object detection neural networks. These networks are trained on high-performance platform and deployed on low-power platforms to compare their pros and cons. Finally, the state-of-the-art YOLOV4 neural network is selected as the main idea of the obstacle object detection module in the intelligent vision system.

Chapter 4 focus on the optimization of neural network on the hardware. It introduces the structure of the RK3399PRO hardware and the basic idea of the deep learning neural network acceleration. Then it introduces operator splitting, replacement and fusion, conversion of model data layout and the development of activation function and pooling layer. This chapter also introduces the quantization process and how to improve quantization accuracy. Finally, the optimized neural network has a much lower resource usage and better inference performance on hardware platform.

Chapter 5 includes the results and data analysis. It shows the construction of dataset,

training and testing results of different object detection neural networks on high-performance platform and the inference results on low-power platform. According to the accuracy and performance, it is proved that the optimized neural network has the best inference performance with acceptable accuracy.

Chapter 6 concludes the whole thesis and discusses future research directions. It summarizes the work and contributions to the intelligent vision system, analyzes some deficiencies in the software algorithm level and the hardware structure level and provides some reference ideas for future vision system development.

2 Robot Vision System Design

This chapter introduces architecture design of intelligent vision system on mobile robot. It describes the principles of mobile robot's vision and analyzes two key perception technologies, including their principles and research status. The chapter also introduces sensor construction, communication of different modules and allocation of hardware resources of the whole system. Finally, this chapter shows the whole architecture diagram of the vision system.

2.1 Mobile Robot Vision Principle

The visual development of mobile robots is to develop how the robot perceives the surroundings. Traditional robot vision system is mainly divided into monocular vision, binocular stereo vision, multi-eye vision, panoramic vision and mixed fusion solutions. The classification is based on different sensors. Monocular vision uses only one vision sensor. During the imaging process, it is projected from the three-dimensional objective world onto the two-dimensional image. However, monocular vision system cannot obtain depth information. It can only be roughly estimated by some algorithms. This is the main disadvantage of this type of vision system[42]. But monocular vision is the basis of other types of vision systems, which are realized by adding other means and measures. Binocular stereo vision is composed of two cameras. It uses the principle of triangulation to calculate the depth information of the object. And it can reconstruct the three-dimensional shape and position of the surrounding scenery, drawing on the principle of the human eye. While the difficulty of the binocular vision is the matching of left and right vision at a same point. The multi-eye vision and others solve the ambiguity of binocular vision system matching. They use multiple sensors to carry out a vision fusion solution, which highly improves the robot's visual positioning accuracy[43]. The higher level vision system requires more complicated processing algorithm, which needs more time and hardware resource.

At present, there are many research developments in the field of robot vision. But most of them are aimed at industrial-level robots and some advanced bionic robots. There is not much development of vision applications on consumer-level mobile robots. On this kind of robots, such as sweeping robots and disinfection robots, vision systems only

rely on conventional infrared or ultrasonic sensors to perform some obstacle avoidance functions. This vision system is greatly affected by environmental interference and data errors occur from time to time. And it does not include automatic localization, automatic navigation, object detection, target tracking and other intelligent functions. While the perception of environment and the path planning are core technology of whether the mobile robot can successfully perform tasks[44]. Therefore, research on vision development on such mobile robots, combined with AI and SLAM algorithms to achieve high-precision vision systems, can enable these robots to work more efficiently. And in this intelligent vision systems, high-precision positioning and as much surrounding information as possible are required. So, this thesis designs a multi-sensor fusion system.

2.2 Sensor Construction

The visual development of the robot mainly includes two parts. One is the positioning in an unfamiliar environment, which is related to the functions of automatic positioning, mapping and navigation. The other one is the cognition of unknown things, which involves an object detection function for obstacles, and face detection on robots is also belongs to this part.

In the mobile robot platform, in order to realize these two visual development parts, the construction of sensors is required to collect data. This thesis uses a multi-sensor fusion vision frame, including a monocular camera and a 2D LiDAR for intelligent vision system, which is shown in Figure 2-1.

(a) (b)

Figure 2-1 Sensors of Vision System (a)Monocular camera (b)2D LiDAR

The monocular camera is mainly used for object detection function based on deep learning. It uses the USB interface to transmit video stream information to the RK3399PRO board. The 2D LiDAR is mainly used for robot's SLAM algorithm and

automatic navigation function. It uses network interface is used to transmit the point cloud information which is reflected by laser. With the construction of these two sensors, mobile robots can complete tasks more efficiently and intelligently in unknown environments. The control panel of the mobile robot still uses infrared and ultrasonic sensors as an auxiliary obstacle avoidance.

2.3 SLAM on Mobile Robot

SLAM is mainly about solving the problem of positioning, navigation and mapping when the robot is running in an unknown environment. Construct and update the map of the unknown environment through the LiDAR sensor carried by the robot[45]. When the robot is working in an unfamiliar environment, it first needs to locate its own position and it needs to construct a map based on the surrounding environment information, so the robot can perform synchronize positioning in the map. Then it can lead to the next step of the navigation function, the robot decides which direction to go through its own decision-making. This process can be divided into four parts: perception, positioning, mapping and navigation. Among these four parts, positioning is the most critical aspect. High-precision positioning is a prerequisite for high-precision mapping and accurate navigation[46]. At present, there are visual SLAM based on traditional images and laser SLAM based on point cloud information. The development of laser SLAM started earlier than visual SLAM, and it is more mature than visual SLAM in all aspects. However, laser SLAM strongly relies on the accuracy of the sensor, and it will be time-consuming to construct and update maps. Visual SLAM cost less and obtain more information, but the reliability and accuracy are not as good as laser SLAM[47]. On mobile robots, these two SLAM technologies have been researched and developed. Laser SLAM is mainly used in indoor areas. Visual SLAM can be used both indoors and outdoors, but the dependence on light is relatively high and it cannot work in dark places or areas with few texture features. The process of SLAM is shown in Figure 2-2.

The mobile robot intelligent vision system in this thesis utilizes a 2D LiDAR to perform a SLAM mapping and navigation. The research and design of the SLAM algorithm part is in charge of my partner Zihan Liu. I am responsible for the data post-processing and fusion of the SLAM algorithm results.

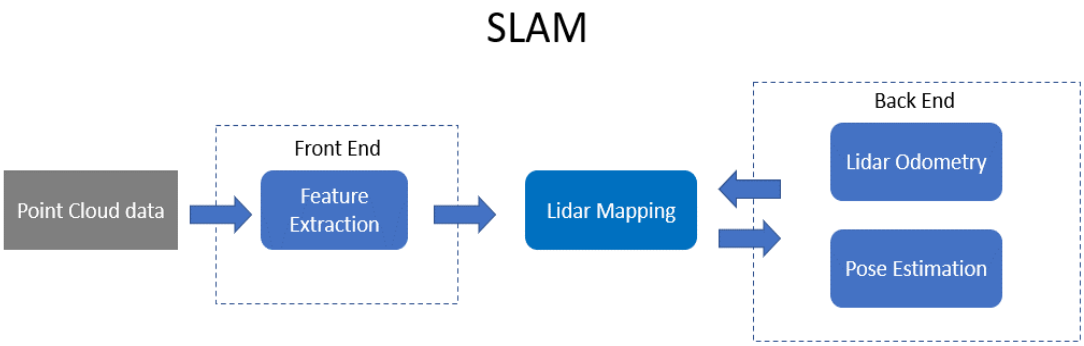


Figure 2-2 Main Idea of SLAM

2.4 Object Detection on Mobile Robot

Object detection is the core technology of computer vision and image processing. It plays a key role in applications such as robot navigation, intelligent monitoring and face recognition. It is a hot topic in the field of robotics research and application. Object detection is to enable the robot to detect and recognize each object that can be perceived by the sensor. On the mobile robot, the object of interest in the image obtained by sensors can be found through object detection. The category and location information of the object can be marked, so that the robot has a better cognitive function for unknown surrounding obstacles. Based on this function, a more intelligent robot can be further developed to achieve better service functions. Many researchers and engineers have studied object detection algorithms and applied them to robot's vision system. Object detection algorithm is mainly divided into two categories, one is a traditional image recognition algorithm based on conventional image segmentation and feature extraction. The classification and recognition are based on edge texture features of image. The other one utilizes deep learning technology. Neural network is used to extract the features of the image, and then perform the detection and classification at the backend. Recent years, the development of AI is extremely hot, and deep learning is the most popular part of the research, especially in the field of computer vision. There are a large number of neural network algorithms for image segmentation, classification, object detection and other functions. These algorithms have excellent achievements. At present, there are many developers researching in academia and industry.

The process of object detection based on deep learning is shown in Figure 2-3.

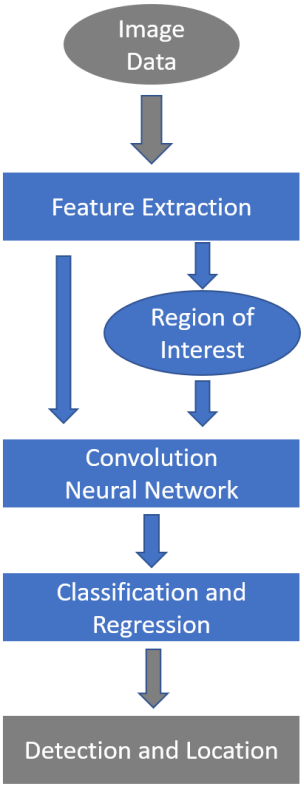


Figure 2-3 Main Idea of Object Detection based on Deep Learning

However, a lot of research is still in the theoretical algorithm stage. Most of the neural network algorithms for object detection are implemented on powerful processors. In the IoT application scenarios, there is still a lack of research when mobile robots are equipped with low-power chips. Few developers have done a deep learning algorithm performance optimization for the hardware platform, which is lacking in the current academic research field and the market. How to implement a very computationally expensive and complex neural network algorithm on a chip with lower computing power and memory requires further research and development. This is a difficult point in this thesis, and it is the main research direction.

2.5 Whole System Architecture

The mobile robot platform used in this thesis is a disinfection robot, and the intelligent vision system is a multi-sensor fusion vision solution. The object detection function is based on the deep learning neural network algorithm. The neural network processes the real-time video stream data obtained by monocular camera. The synchronous

positioning and automatic navigation function is based on laser SLAM algorithm. This algorithm processes the point cloud data obtained by 2D LiDAR. In addition, this mobile robot includes obstacle avoidance functions based on the infrared and ultrasonic sensors. The whole system design involves hardware construction, allocation of chip resource, message communication between different modules and algorithm implementation. The entire intelligent vision system is deployed to the low-power edge computing IoT platform. The RK3399PRO is the core processing chip. The specific system architecture is shown in the Figure 2-4.

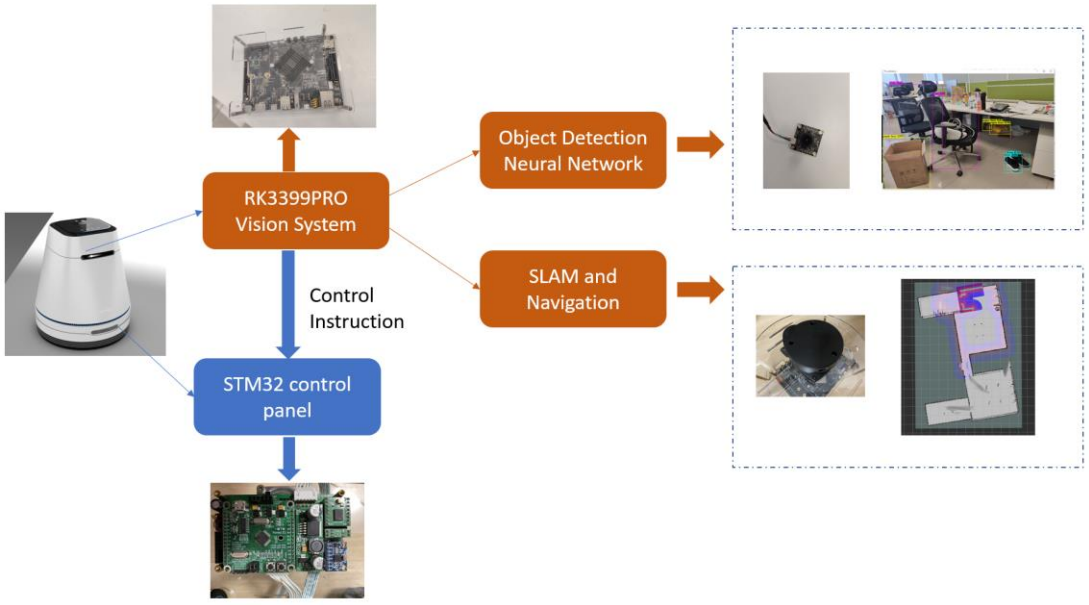


Figure 2-4 Whole Vision System Architecture

Inside the RK3399PRO platform, GPU and VPU are responsible for video stream encoding and decoding, image enhancement, noise reduction and other preprocessing, NPU is responsible for the inference of the object detection neural network algorithm, and the dual-core Cortex-A72 of CPU is responsible for the 2D LiDAR data processing and SLAM algorithm implementation , the quad-core Cortex-A53 is responsible for the post-processing and data fusion of neural network and SLAM algorithm. The idle core of Cortex-A53 implements the communication function with STM32F series control panel. The mobile robot's motor drive, data processing of obstacle avoidance sensors and the realization of service functions like disinfection or sweeping are all controlled by the STM32F series chip. The STM32F control panel is connected to the RK3399PRO chip of the upper intelligent system for communication. The RK3399PRO chip is responsible for the algorithm implementation of the intelligent vision part,

navigation control command issuance and data visualization.

The whole mobile robot platform uses a hardware architecture of RK3399PRO processing chip and the STM32F series control chip. The RK3399PRO is the core of the intelligent vision system.

2.6 Summary

This chapter describes the architecture design of the intelligent vision system on mobile robots, including the SLAM algorithm for positioning and navigation in unknown environments and the deep learning neural network for obstacle object detection. It also introduces the construction of sensors, the allocation of chip resource and communication between different modules. Finally, this chapter shows an overall intelligent vision system architecture.

3 Deep Learning Object Detection on Robot

With the development of the IoT technology and AI algorithm, Artificial Intelligence and Internet of Things (AIoT) has become a popular research area in recent years and there are more and more AIoT applications on mobile robots. Among the development of AI and machine learning, deep learning neural network has achieved a breakthrough. It has been a hot topic of research and development in recent years. Convolutional Neural Network (CNN) based on deep learning has a significant achievement in both computer vision and natural language processing[48]. On mobile robots, both of them have applications and research, but they are mainly in early stages. In the intelligent vision system on mobile robot, this thesis mainly focuses on the development of the object detection function based on the deep learning. This chapter researches the deep learning neural network algorithms on the mobile robot, compares several open source object detection neural networks and analyzes their advantages and disadvantages. These neural network algorithms are trained and tested on a high-performance server and a low-power IoT platform. According to the accuracy and performance of different neural networks, the YOLOV4 network is selected as the main idea of the algorithm for the obstacle object detection module in the intelligent vision system.

3.1 Traditional Object Detection Neural Network

Object detection refers to identifying the target in the image and locating it. This function extracts the location feature information on the basis of the conventional image classification algorithm and outputs the coordinate information in pixels of each category's object. However, the size of the object varies widely, the angle and posture of the object are not fixed, and the object can appear anywhere in the image. Generally, one image includes a variety of categories. So, it is difficult to achieve extremely accurate object detection and there are a variety of different algorithms. With the success of deep learning in these years, most of the research directions have turned to deep learning based object detection. There are many successful classic algorithms which have proved that applying deep learning to object detection can greatly improve the accuracy and performance[49].

General structure of object detection neural network is shown in Figure 3-1.

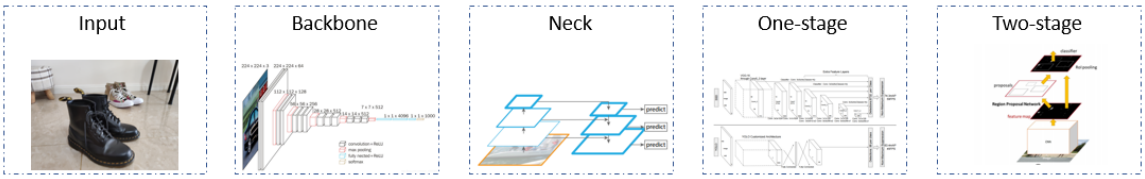


Figure 3-1 General Structure of Object Detection Neural Network

This kind of neural network mainly includes four parts: the input preprocessing part, the backbone network, the neck and the output head. The input part is generally preprocessing such as splicing and enhancement of a batch of pictures. The backbone network is generally divided into ResNet, Cross Stage Partial Network (CSPNet) or MobileNet according to the different hardware of neural network deployment, using GPU or CPU. The neck network is generally composed of several top-down paths and bottom-up paths to extract and fuse features. The output head part is usually divided into two categories, one-stage and two-stage. The difference is that the one-stage algorithm directly outputs the probability and coordinate of each object category. The two-stage algorithm has two steps: the first step is to generate candidate regions through Region Proposal Network (RPN), and the second step is to classify and regress each proposal region to obtain the final result. The one-stage algorithms usually have faster inference speed while two-stage algorithms have higher accuracy. The rest of this chapter overviews the basic concepts behind popular one-stage (SSD, YOLO) and two-stage detectors (R-CNN).

3.1.1 R-CNN

R-CNN is a classic two-stage computer vision neural network algorithm. Two-stage must first filter out proposal regions and then perform feature extraction and segmentation. R-CNN has developed from the initial neural network structure to current neural network of R-CNN, Fast -RCNN, Faster-RCNN and mask-RCNN. The main application scenarios of R-CNN include image classification, object detection, face detection and image segmentation, many subsequent computer vision neural networks have borrowed ideas from R-CNN[50].

The main idea of the R-CNN neural network algorithm is to first extract about 2000 region proposal from the input image through the selective search algorithm, and then input each proposal region into CNN for feature extraction. The original R-CNN

network scales each proposed area to 227x227, while Fast-RCNN and Faster-RCNN directly input the entire image into CNN for feature extraction so that the features of the entire image are extracted at one time. Compared with the original R-CNN network, there is no need to put each proposed region into CNN to extract feature, which greatly reduces the amount of calculation. The final step is to classify the extracted features. Original R-CNN network transfer these features into Support Vector Machine (SVM) for classification and positioning, but the speed of final window coordinate prediction through SVM regression is too slow, and the features are not updated. In the following Fast-RCNN and Faster-RCNN, this structure is cancelled, and the Pooling layer of Region of Interest (ROI) is used to generate fixed-size feature maps. Then SoftMax and Smooth L1 Loss function are connected to calculate probability of classification and bounding box regression is used to generate prediction window.

The latest Faster-RCNN network structure in the development of R-CNN is shown in Figure 3-2.

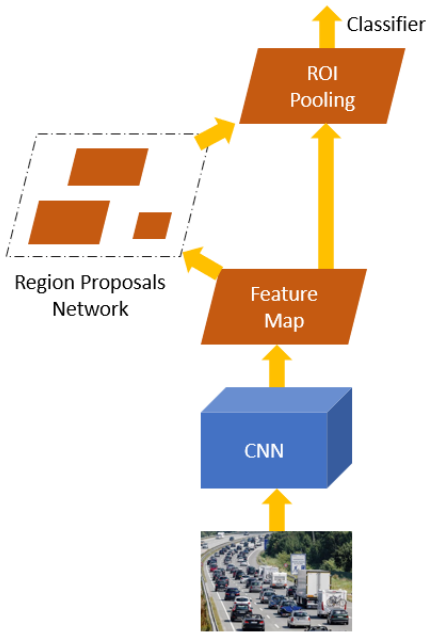


Figure 3-2 Neural Network Structure of Faster-RCNN

In Faster-RCNN, when generating the suggestion window, the RPN is used instead of the original selective search algorithm to generate the proposal region. At the same time, the concept of anchor box is utilized to deal with the diversification of object shape and size. The area extracted by these anchor boxes is used to determine whether it is in the foreground or the background, and then suggestion window is generated. This structure will greatly improve the efficiency.

From R-CNN, Fast R-CNN, Faster R-CNN along the way, the detection process is becoming more streamlined, the accuracy is getting higher and higher, and the inference speed is getting faster and faster. R-CNN is the most important branch in the development of the field of deep learning object detection. However, when applied to a mobile robot platform, the computing power and memory resources of the hardware are limited, the two-stage object detection neural network is not very suitable, which will first extract the anchor box from the feature map based on the proposal region, then map the anchor box to the feature map, and finally perform a classification and regression on this whole feature map area. There is a neural network layer for proposal region generation in the middle, which greatly increases the requirements of neural network for hardware platform resources. Even on faster-R-CNN, which requires the minimal resources, the memory used for training on the server will be 50% higher than that of other neural networks. When the model is deployed on the RK3399PRO platform, the inference time is several times that of the one-stage neural networks. Of course, the performance also related to the hardware platform's support for different network structures and different operators. Therefore, when deploying edge AI computing solution on mobile robots, the R-CNN is not applicable. In view of this, the object detection neural network selected for the intelligent vision system on mobile robot platform in this thesis is one-stage type. However, the R-CNN network is still an important beginning of deep learning object detection neural networks.

3.1.2 SSD

The Single Shot MultiBox Detector (SSD) algorithm is one of the most widely used algorithms in the field of target detection so far, and many researchers and engineers have developed a lot of improved algorithms on the original SSD neural network[51]. As mentioned above, the RPN layer used by Faster-RCNN is used to extract the features of the entire image. However, one of the drawbacks of RPN is that the detection effect for small target objects is very poor, because its anchor is fixed. Each point on the feature map is mapped back to the original image. If the size of the input image is large and the size of the output feature map is small, each point on the feature map is responsible for a large area, and generally the feature map of the last layer is relatively abstract. In this way, the features of small target objects on the original image cannot be extracted well. In view of this, SSD was born. Unlike RPN processing the results of

prediction and detection on the final feature map, SSD performs information fusion from multiple scales feature maps. It predicts and detects results on feature maps of various levels and finally integrate the different prediction results.

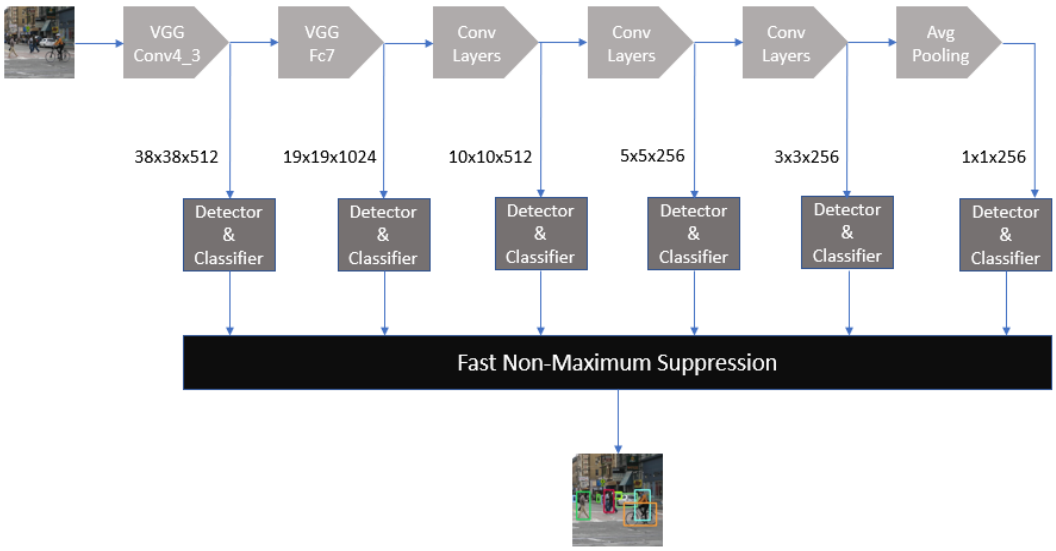


Figure 3-3 Neural Network Structure of SSD

The structure of SSD is shown in Figure 3-3. The backbone network part of the SSD adopts the idea of VGG16, but SSD has some modifications: the two Fully Connected (FC) layers are replaced with convolutional layers, the dropout layer is removed, the atrous convolution algorithm is introduced and parameters of pooling layer are improved. In order not to reduce the size of the feature map, several feature maps of multiple scales can perform accurately detection at the same time. The receptive field of the feature map at the bottom is relatively small, and the receptive field at the high-level is relatively large. The shallow layers are more interested in the edges, while the deep layers are more interested in the complex features composed of the shallow features.

SSD also draws on the anchor structure in Faster-RCNN, which will generate a series of default boxes from the feature map and select out one prior box as the initial coordinates. The final predicted positioning information is actually the relative coordinates of the bounding box after regression and the prior box before regression. That is, some default boxes are preset, and then the real object detection window is obtained by SoftMax classification and bounding box regression. Also, different default boxes are selected for feature maps of different scales. Finally, the predicted detection

boxes are sent to the Non-Maximum Suppression (NMS) module to calculate the final detection result.

The inference results of SSD neural network deployed on the RK3399PRO platform is shown in Figure 3-4.

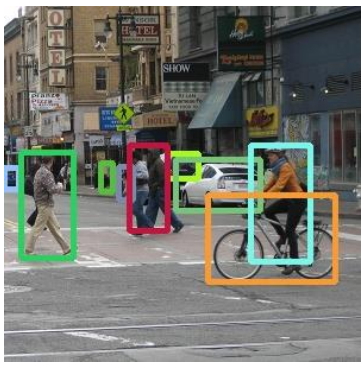


Figure 3-4 Inference Result of SSD on RK3399PRO

In general, SSD not only borrows part of the mechanism in Faster-RCNN, but also speeds up the detection process based on it. Because SSD is a one-stage network, it outputs detection results only in one step, Faster-RCNN is a two-stage network which requires two networks to perform operations. The data enhancement, multi-scale fusion, and FC layer adopted by the SSD have all verified this. The accuracy of the SSD does not lose much compared to Faster-RCNN. In the AIoT scenario, SSD is also one of the most widely used algorithms. However, the SSD neural network algorithm also has certain shortcomings. Although there is a multi-scale pyramidal feature hierarchy idea, the detection of small targets is still not good. In the application scenario of the mobile robot in this thesis, small target objects are still in a more important part. And some parameters of the SSD's prior box need to be set manually and the basic size cannot be obtained through self-learning. The size and shape of the prior box used by each layer of the feature map in the neural network are different, which causes the setting up process rely heavily on experience. At this point, compared to the You Only Look Once (YOLO) algorithm proposed below, SSD has no advantage. The anchor box shape calculated by YOLO through adaptive clustering is much better than the manual setting in SSD.

MobileNet-SSD is one of the successfully improved algorithms of original SSD networks. The inference data of MobileNet-SSD neural network deployed on the RK3399PRO is shown in Table 3-1. The inference performance depends not only on

the computing power, but also on the memory usage and the hardware platform operator support. Although the SSD is light enough, it also brings a decrease in accuracy. And due to the particularity of the depthwise convolution compared to the traditional 2D convolution structure, the accuracy loss of the depthwise convolution kernel after int8 quantization is large, while the core of MobileNet-SSD is the depthwise convolution. Therefore, SSD is not suitable for the object detection field on the mobile robot platform in this thesis.

Table 3-1 Inference Data of MobileNet-SSD on RK3399PRO

	Initial Model	Quantized Model
Model Size (MB)	13.1	6.62
System Memory (MB)	186.48	90.56
NPU Memory (MB)	106.70	52.41
Inference Cost (us)	181017	25255
FPS	5.52	39.6

3.1.3 YOLO

The YOLO algorithm is comparable to the SSD algorithm in the field of computer vision, and it is also one of the most used object detection neural network algorithms so far. It has developed from YOLOV1 to YOLOV4, and each version has different innovations. It is still being developed and updated, and there are many developers' contributions in the current object detection field[52].

The structure of YOLO network is shown in Figure 3-5.

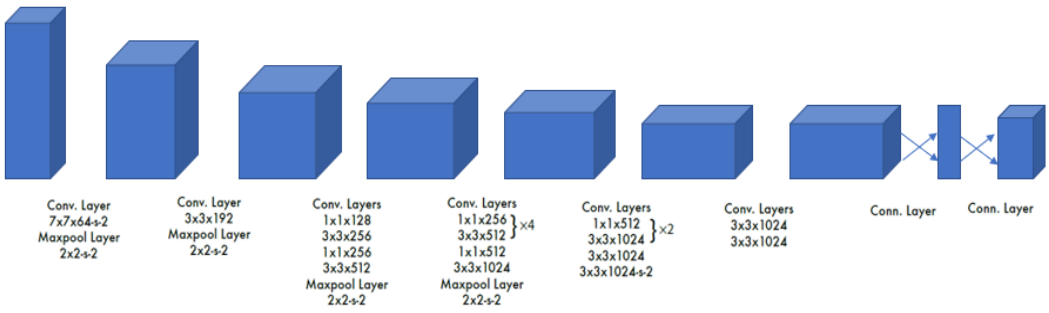


Figure 3-5 Neural Network Structure of YOLO

When the first generation of YOLO was proposed, the core innovation was to convert the object detection into a regression problem. This is completely different from the R-CNN at the time, which is the characteristic of one-stage, using only one step to output the bounding box and category probability. YOLO's network structure is very clear, that is, through several fully convolutional layers and maximum pooling layers to extract the abstract features of the input image, then through the FC layers to output the predicted object's bounding box and category probability. YOLO has achieved a major breakthrough in real-time performance, but the accuracy is not good, the result is very poor in the detection of small targets, especially when two targets are very close to each other, the detection result is terrible. And YOLO's positioning is very poor.

Therefore, on YOLOV2, the author has modified many places to solve these problems. Compared with YOLO, it is faster and more accurate. The main idea of YOLOV2 is still similar to the YOLO network, but there are some adjustments. First, on the backbone network, YOLOV2 uses Darknet-19, which is similar to the VGG used in SSD, but the difference is that after convolution on each layer, Batch Normalization (BN) is used to preprocess the input data, so that the output of each layer can be normalized distribution. In this way, the internal covariate shift caused by the data distribution changes will not occur and the network does not need to learn the distribution of the data at each layer, which reduces the learning rate during training. And YOLOV2 uses a dimensionality reduction idea by putting the 1x1 convolution between 3x3 and using it to compress the features. Overall, this structure is almost 6 times faster compared to VGG, and there is not much loss in accuracy. The structure of YOLOV2 is shown in Figure 3-6.

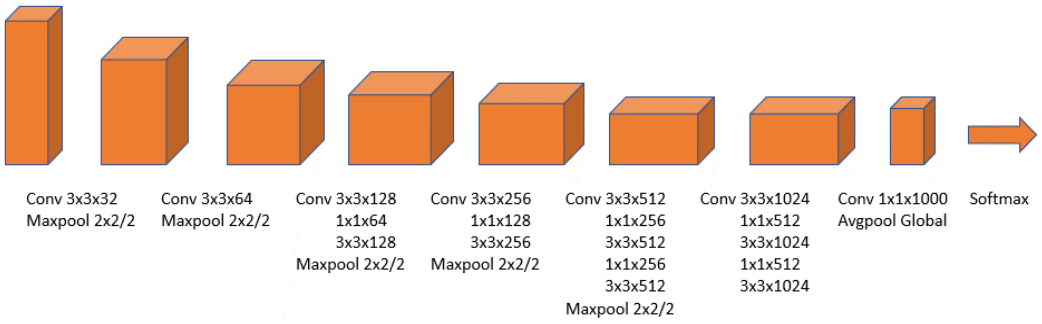


Figure 3-6 Neural Network Structure of YOLOV2

YOLOV2 also introduces the anchor mechanism in Faster-RCNN, which is also applied in SSD. It uses the predicted offset and confidence of the anchor box to make the final prediction, instead of the direct coordinate prediction in YOLOV1. The difference with

Faster-RCNN is that YOLOV2 uses the sigmoid function to normalize the offsets to the 0-1 interval in order to constrain the centroid of the bounding box to the current grid, which makes the training of the model more stable. The size and proportion of the anchor box selected by Faster-RCNN are empirically set and not very representative, so the K-means clustering method is used to obtain the size of the anchor box in YOLOV2. In order to fuse the features of different feature maps, YOLOV2 uses a passthrough layer to connect the high resolution shallow features to the low resolution deep features and stacking features in different channels. And then the detection and regression are performed. This structure can improve the detection ability of small targets. The inference results of YOLOV2 network deployed on the RK3399PRO platform is shown in Figure 3-7.

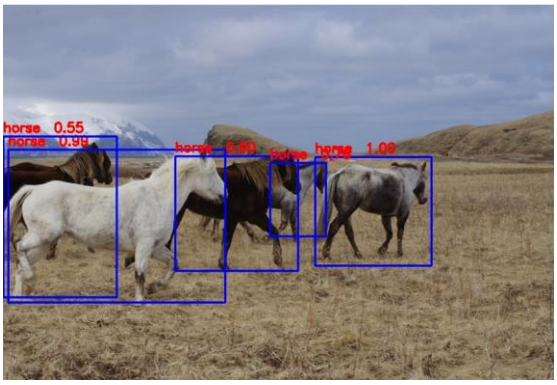


Figure 3-7 Inference Result of YOLOV2 on RK3399PRO

Overall, compared to YOLO, YOLOV2 has a huge improvement in accuracy and maintains the speed as fast as YOLO. In edge AI computing scenario, the real-time performance of YOLO series neural network is much higher than that of Faster-RCNN. And YOLOV3 and YOLOV4 were further optimized and developed with greater improvements. So, in this thesis, YOLO's algorithm was selected on the mobile robot platform, and the advantages and disadvantages of different algorithms were compared.

3.2 YOLOV3 Deployment

The YOLOV3 algorithm is the pinnacle of the YOLO algorithm, and many developers still choose the YOLOV3 framework to do their own development work for deep learning object detection neural network[53]. The intelligent vision system on mobile robot platform in this thesis also deploys YOLOV3 algorithm network to do obstacle detection.

The backbone part of YOLOV3 has a huge improvement compared to YOLOV2 by improving the original Darknet-19 structure into Darknet-53 structure.

The idea of ResNet network is also used. The backbone Darknet-53 consists of 1x1 and 3x3 convolutional layers, each convolution operator followed by a BN layer and Leaky ReLU activation function. The BN layer for data pre-processing is used to improve the learning rate. These components form the basic convolutional unit: Conv+Bn+Leaky ReLU (CBLR) of YOLOV3, and there is no pooling layer or FC layer in the whole network. Then the idea of Resnet is adopted, each input is passed through two CBLRs and then add with the original input, so the features are fused. This ResNet unit allows the network to extract deeper features and at the same time avoid the gradient vanishing or exploding problem that occur during training.

Then some layers in the middle of the backbone network are concated with the upsampling of some layers behind, which can achieve a concept of multi-scale features fusion as used in YOLOV2 and SSD network. Concat is not the same as the add operation of ResNet, concat is to expand the dimension of the tensor, while add is only to superimpose features without expanding the dimension. These are the improvements of this backbone network, and one of the most innovative points. Another feature of YOLOV3 is multi-scale detection.

The output of YOLOV3 is not only one output box, but three different scales of network output, each scale is responsible for three bounding boxes, adding up to a total of 9 anchor boxes, which is better for detecting small target objects. The output tensors are not completely independent, unlike the SSD which uses the processing result of the middle layer in the backbone network as the output of the feature map, the three output tensors of YOLOV3 here are the output of the feature map after sampling and stitching, which drawing on the idea of Feature Pyramid Network (FPN).

The structure of YOLOV3 is shown in Figure 3-8.

In the intelligent vision system on mobile robot platform in this thesis, the neural network of YOLOV3 was chosen for deployment test, by neural network training on MS coco dataset with 80 target species and also on constructed dataset (see Chapter 5 for details).

The training process of own dataset are shown in Figure 3-9.

Since further training of more batches will lead to a decrease in accuracy and over-fitting, the training weights with the highest accuracy are extracted.

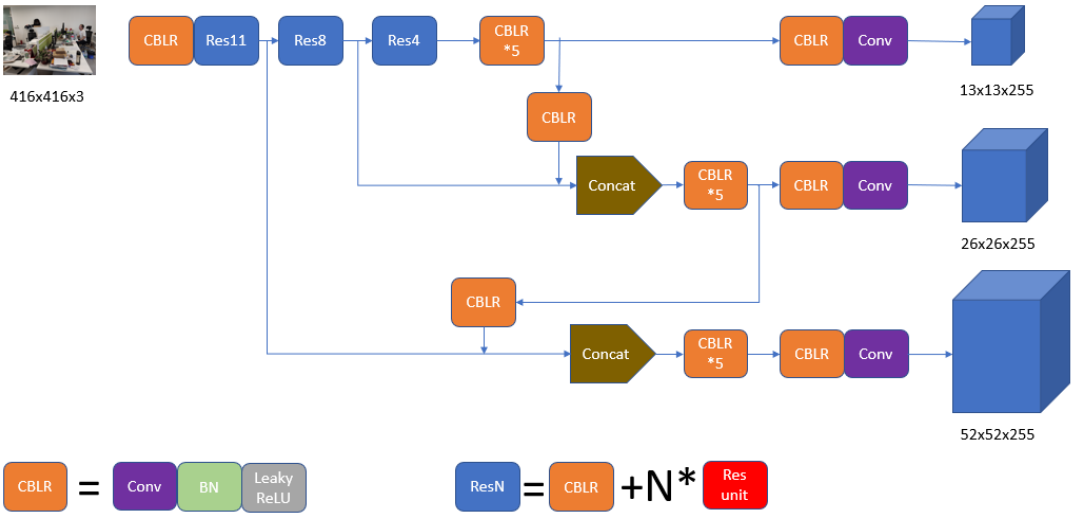


Figure 3-8 Neural Network Structure of YOLOV3

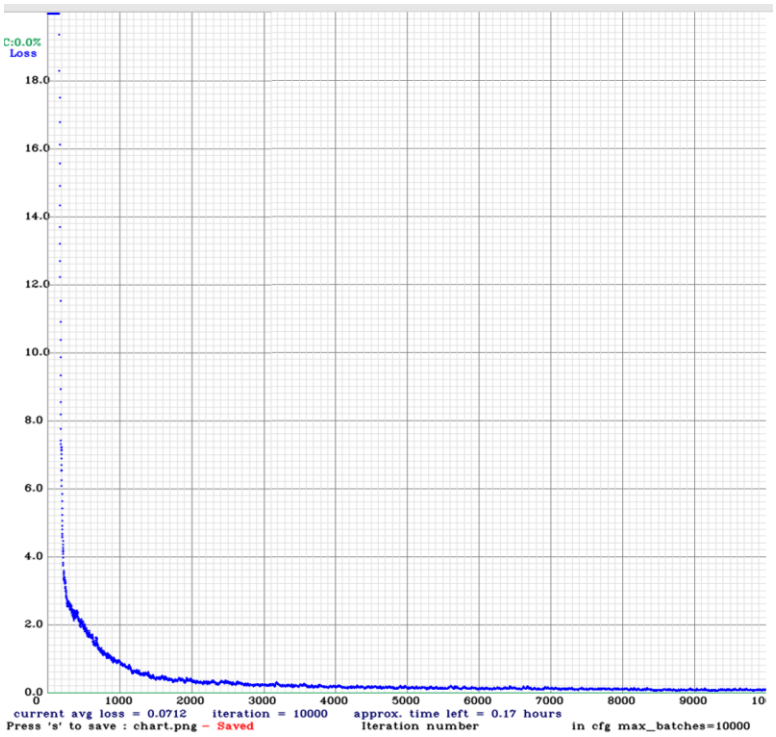


Figure 3-9 Training Process of YOLOV3

The testing results of YOLOV3 neural network on high-performance server are shown in Figure 3-10, each window shows one detected object, and each followed number is

the confidence of the object, which represents the probability that the object is included in the detection window and that the detection window fully includes the object profile. The neural network model is then transferred and deployed to the RK3399PRO platform for inference. The thesis processes the output data of the three output tensors of the neural network for visualization. When the input image's size is 608x608, the sizes of the three output tensors are 19x19x255,38x38x255, 76x76x255. The number 255 corresponds to 3x(80+5), in which 3 corresponds to the 3 anchor boxes each tensor is responsible for, 80 corresponds to the probability of each of the 80 object categories in the MS coco dataset and 5 corresponds to (x, y, w, h) and the confidence of each detection box. The inference results of YOLOV3 network on RK3399PRO platform are shown in Figure 3-11. Labels and confidences of each object are shown in captions.

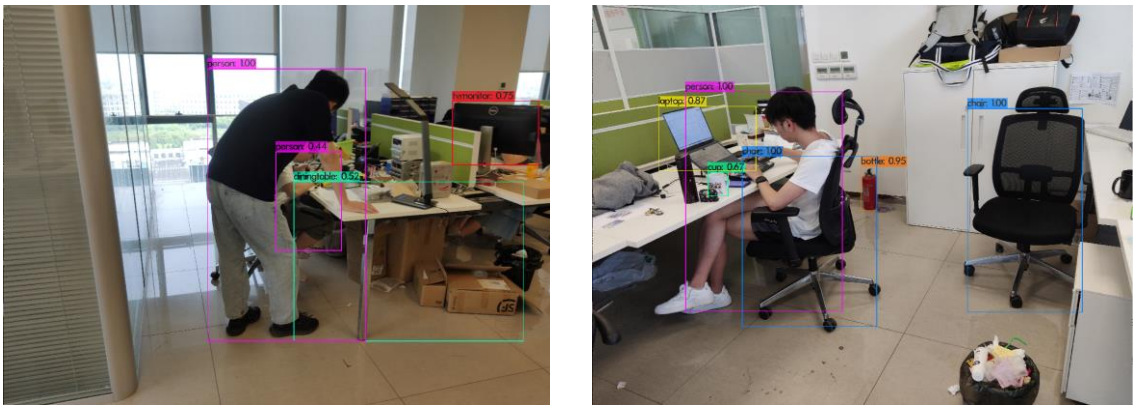


Figure 3-10 Inference Result of YOLOV3 on Darknet

(Left) Person 1.00 TV Monitor 0.75 Dining Table 0.52 Person 0.44

(Right) Person 1.00 Chair 1.00 Chair 1.00 Bottle 0.95 Laptop 0.87 Cup 0.67

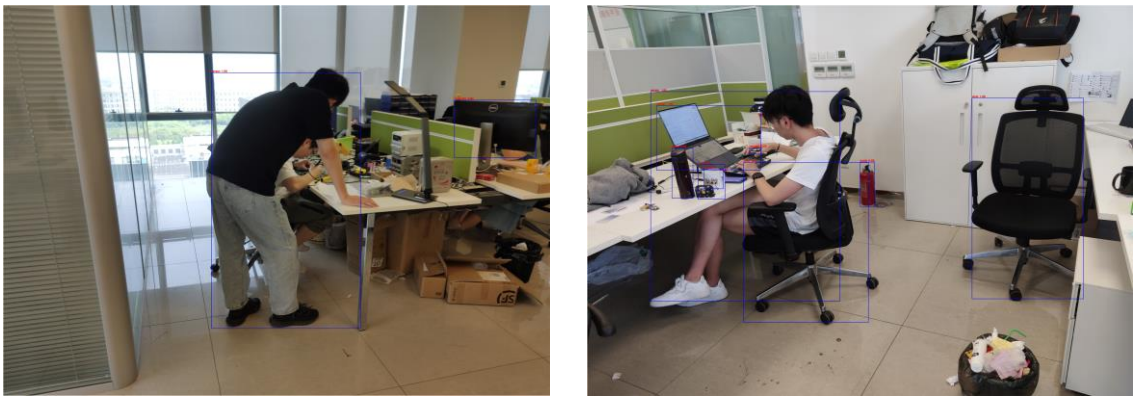


Figure 3-11 Inference Result of YOLOV3 on RK3399PRO

(Left) Person 1.00 TV Monitor 0.82

(Right) Person 1.00 Laptop 1.00 Chair 1.00 Chair 0.99 Cup 0.94 Mouse 0.72 Bottle 0.68 Bottle 0.55

Table 3-2 Inference Data of YOLOV3 on RK3399PRO

	Initial Model		Quantized Model	
Size (MB)	118.29		59.28	
Input Size	416x416	608x608	416x416	608x608
System Memory (MB)	407.35	447.43	199.88	241.31
NPU Memory (MB)	293.77	315.73	142.07	153.20
Time Cost (us)	2165990	4712463	75756	171060
FPS	0.46	0.21	13.2	5.85

The inference data of YOLOV3 network on RK3399PRO platform are shown in Table 3-2. In general, YOLOV3 is a very good neural network in the field of engineering, which is widely used in edge computing scenarios and mobile robotics platform. YOLOV3 has been introduced for a long time and is a very mature target detection neural network framework, while compared to the following state-of-the-art object detection neural network algorithm YOLOV4, it still has a lot of shortcomings.

3.3 YOLOV4 Deployment

YOLOV4 is a state-of-the-art object detection neural network proposed in 2020. YOLOV4 was not introduced by Redmon, the original author of YOLOV1-V3, but by Alexey. Compared with the network framework of YOLOV3, YOLOV4 does not make particularly large changes on the backbone network Darknet-53, but YOLOV4 has been improved by a series of small tricks, and the final experiment data prove that YOLOV4 can achieve higher inference speed with the same detection accuracy[54]. YOLOV4 has some optimizations and is more suitable for single-GPU training and single-GPU inference. In general edge computing scenarios, the hardware architecture is usually a single GPU, NPU or AI acceleration unit, so YOLOV4 has an advantage in this scenario. The structure of YOLOV4 network is shown in Figure 3-12.

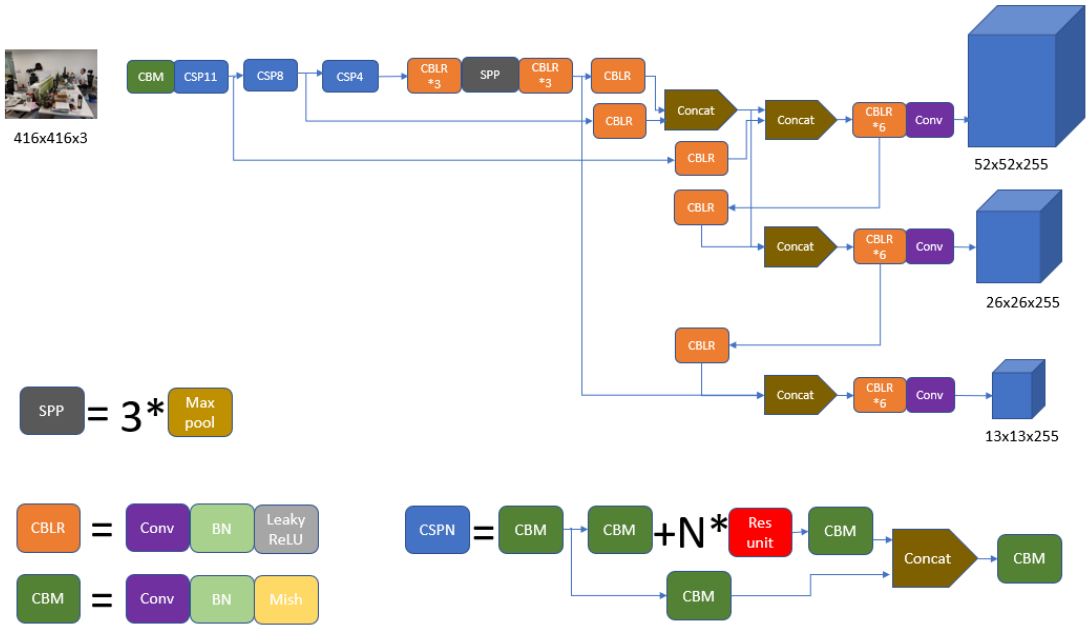


Figure 3-12 Neural Network Structure of YOLOV4

The backbone network of YOLOV4 is the CSPDarknet53, which is a combination of YOLOV2's Darknet53 with CSPNet. CSPNet solves the problem of that previous neural network structures require a lot of inference calculations, mainly by dividing the basic feature graph into two parts and then merging and implementing them through a cross stage hierarchy[55-56]. In YOLOV4, the original ResNet block of YOLOV3 is split. The main part continues the original ResNet block operation, while the secondary part is directly connected to the final network layer after a small amount of processing. This structure design can achieve richer gradient combinations while reducing the calculation. Another feature of YOLOV4 network is that the original Leaky ReLU activation function is replaced by the Mish activation function.

In YOLOV3, there was no pooling layer, the tensor size was changed by increasing the stride of the convolution kernel, but in YOLOV4, the pooling layer is re-introduced, the role of pooling is to reduce the dimensionality, reduce the amount of information, and prevent the neural network model from overfitting. Different pooling methods have different functions. YOLOV4 instead chooses the max-pooling, which takes the largest value of a local area, so that it can better learn the edge and texture structure of the image and better retain the whole feature of the image. YOLOV4 uses Spatial Pyramid Pooling (SPP) structure, which was originally designed to allow CNN to calculate without the limitation of fixed input size by max-pooling and fusing features at multi-

scales[57]. YOLOV4 utilizes this structure here because SPP significantly increases the receptive field and separates out the most important contextual features. At the same time this SPP structure does not affect YOLOV4's inference performance, by max-pooling at 4 different scales, 1x1,5x5,9x9,13x13, and where 1x1 pooling is equivalent to doing nothing. The structure of SPP is shown in Figure 3-13.

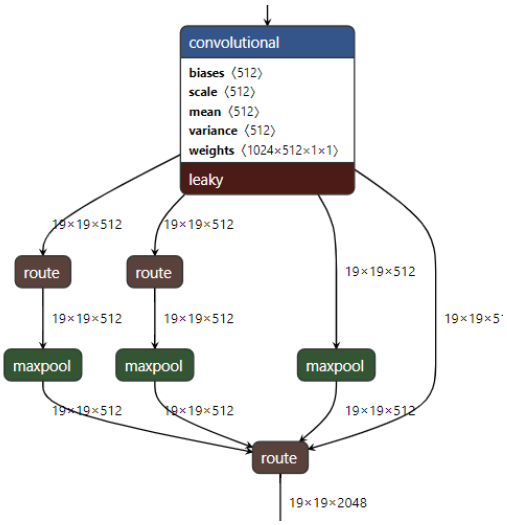


Figure 3-13 Structure of SPP

After the SPP module, at the neck network of object detection, YOLOV4 uses Path Aggregation Network (PANET), which mainly contains FPN, bottom-up path augmentation, adaptive feature pooling and fully- connected fusion. The FPN feature pyramid is also used in YOLOV3, which is top-down and fuses the high-level information by upsampling. Bottom-up path augmentation is a bottom-up one, mainly considering the shallow feature information, feature sampling of this piece, so as to learn some edge feature of the target. Adaptive feature pooling is about feature fusion, using each ROI to fuse the feature map of each layer. FC fusion is to introduce a FC branch for the original segmentation branch and get more accurate segmentation results by fusing the output of these two branches. This structure is designed for semantic segmentation, so it is not used in YOLOV4 for object detection. And the shortcut connection in the original PANET is turned into concat in YOLOV4, so that the dimensionality of the feature tensor can be expanded.

The head of YOLOV4 is similar to the head of YOLOV3, which outputs three output tensor and each tensor is responsible for 3 anchor boxes. The difference is that YOLOV4 contains a bottom-up path enhancement, so the order of output tensor is reversed. When the size of input image is 608x608. The sizes of the three output tensors are 76x76x255, 38x38x255, 19x19x255, and each number represents the same meaning

as that in YOLOV3.

There are also some other optimization contributions in YOLOV4. It introduces the CIOU_LOSS to improve the accuracy of training.

$$CIOU_{LOSS} = 1 - CIOU = 1 - \left(IOU - \frac{{Distance_2}^2}{{Distance_C}^2} - \frac{v^2}{(1 - IOU) + v} \right) \quad (3-1)$$

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w^p}{h^p})^2 \quad (3-2)$$

Compared with the traditional IOU LOSS function which only considers the overlap area of the detection window and the target window, CIOU LOSS not only considers this overlap area, but also the distance between the centroids of the two windows and the aspect ratio of the two frames. This approach makes the regression of the prediction faster and more accurate, and these small optimizations are considered as free bags and do not add any performance decrease and accuracy loss, unlike the SPP and other special bags which adjust network structure. Because the free packages do not affect the forward inference process, but only change the reverse propagation and have some effects on the input and output sides.

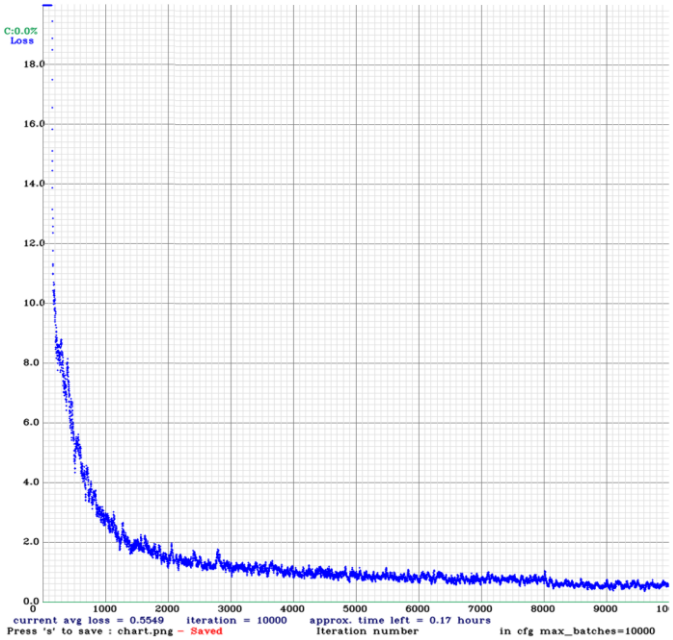


Figure 3-14 Training Process of YOLOV4

(Loss function has more oscillation than YOLOV3)

Similarly, on the mobile robot platform in this thesis, the YOLOV4 neural network is selected to train on the MS coco dataset with 80 target classes and own dataset. The training process of YOLOV4 network is shown in Figure 3-14 and the inference results of YOLOV4 network on high-performance server is shown in Figure 3-15.

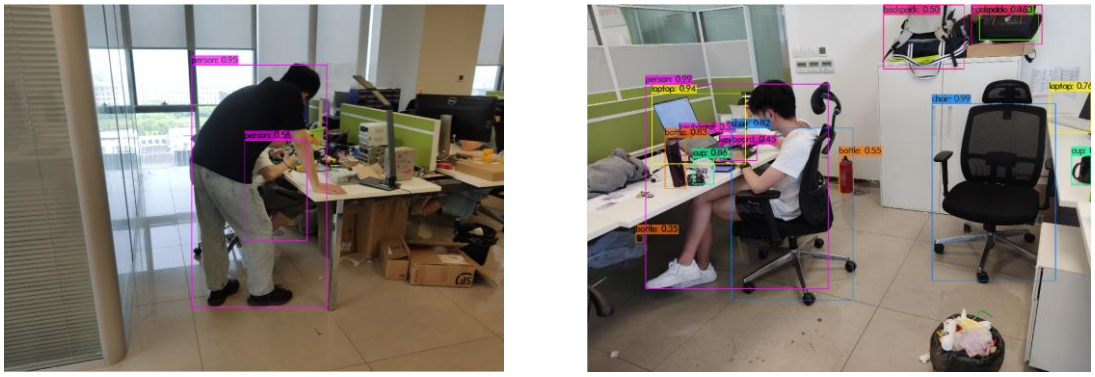


Figure 3-15 Inference Result of YOLOV4 on Darknet

(Left) Person 0.95 TV Person 0.56

(Right) Person 0.99 Chair 0.99 Laptop 0.94 Cup 0.86 Bottle 0.83 Chair 0.82 Backpack 0.50 Keyboard 0.45

Then the neural network model is deployed to RK3399PRO platform for inference through model conversion. The post-processing of the neural network here is similar to the post-processing of YOLOV3, because the output heads are similar, only the order of output tensors are different. And here I refer to the ideas of DIOU LOSS and CIOU LOSS[58], using DIOU NMS in the filtering of the output object detection windows. The traditional NMS only considers the overlap of the two detection windows and the confidence, while DIOU NMS also considers the distance between the centroids of the two detection windows and the aspect ratio of detection windows. It does not need to calculate the ground truth when performing forward inference. In the final object detection results with overlap, DIOU NMS outperforms the normal NMS, and it does not increase the computation much, without adding too much memory and CPU burden.

$$DIOU = IOU - \frac{Distance_2{}^2}{Distance_C{}^2} \tag{3-3}$$

$$s_i = \begin{cases} s_i, DIOU(M, B_i) \geq thresh \\ 0\ , DIOU(M, B_i) < thresh \end{cases} \tag{3-4}$$

The inference results of YOLOV4 model on RK3399PRO platform is shown in Figure 3-16 and the data is shown in Table 3-3.

According to Table 3-2 and 3-3, before optimizing the neural network structure of YOLOV4, the performance of YOLOV4 on RK3399PRO platform is much worse than YOLOV3. However, on the high-performance server it is quite different, there is not much difference between the inference performance of YOLOV4 and YOLOV3. Some comparative data between YOLOV4 and YOLOV3 on the high-performance server are shown in Table 3-4 (without any network structure adjustments to YOLOV4 and YOLOV3).

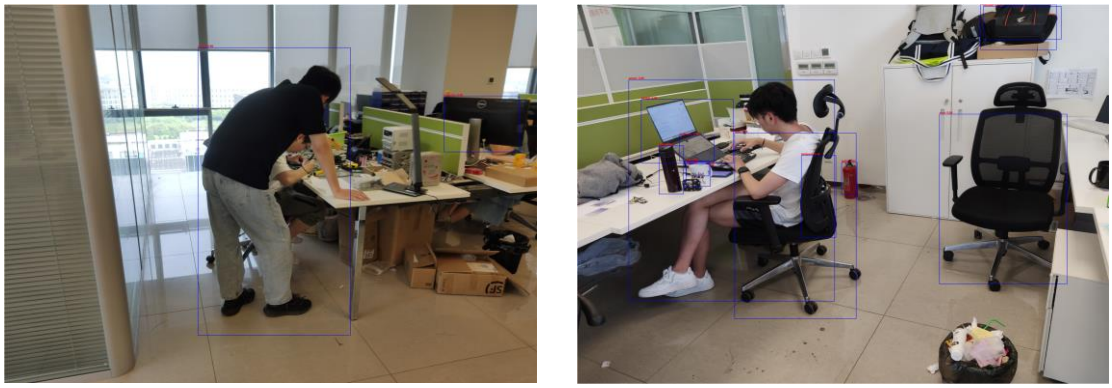


Figure 3-16 Inference Result of YOLOV4 on RK3399PRO

(Left) Person 1.00 TV Monitor 0.82

(Right) Person 1.00 Chair 1.00 Laptop 1.00 Cup 0.94 Chair 0.88 Mouse 0.72 Bottle 0.68

Table 3-3 Inference Data of YOLOV4 on RK3399PRO

	Initial Model		Quantized Model	
Model Size (MB)	122.95		61.66	
Input Size	416x416	608x608	416x416	608x608
System Memory (MB)	530.40	609.49	307.08	344.99
NPU Memory (MB)	327.54	362.55	158.67	176.84
Time Cost (us)	2622625	5751369	542642	1166211
FPS	0.38	0.17	1.84	0.86

Table 3-4 Comparison of YOLOV4 and YOLOV3 on Server

	YOLOV3		YOLOV4	
Model Size (MB)	236		245	
AP50:95	30.9		40.3	
Input Size	416x416	608x608	416x416	608x608
Memory Usage (MB)	18.88	36.22	18.88	36.22
BFLOPS	65.879	140.723	60.137	128.459
Time Cost (ms)	31.07	60.137	34.012	63.16
FPS	32.4	17.1	31.8	16.2

Although the inference performance difference between YOLOV3 and YOLOV4 on RK3399PRO hardware platform is large, the performance difference on the server is minimal, and the accuracy of YOLOV4 on server can be much higher than YOLOV3, accordingly, the accuracy of YOLOV4 is also higher than YOLOV3 by comparing the inference image results in RK3399PRO platform. YOLOV3 has a lot of mature development in the industry, while YOLOV4, as the latest state-of-the-art neural network framework for object detection. Considering the better accuracy, many optimizations can be made to make YOLOV4's performance on the hardware side exceed that of YOLOV3, approaching the data of high-performance servers. In view of this, the intelligent vision system on mobile robot platform in this thesis also adopts the algorithm idea of YOLOV4, and the subsequent chapter 4 describes in detail about specific optimization process done on the hardware side, engineering implementation adaptations and neural network structure adjustment.

3.4 YOLOV4-Tiny Deployment

YOLOV4-Tiny is an official light model provided by the authors of YOLOV4, a network specifically for scenarios with high real-time requirements, as well as

YOLOV3-Tiny, which was also provided by YOLOV3 official. YOLOV4-Tiny has about a quarter of YOLOV4's layers, which is equivalent to deleting a large chunk of the original network structure. When testing on RTX2070, it can achieve more than 300 FPS and 20 AP on MS coco dataset. It is perfectly acceptable to compromise this accuracy for high speed. Tiny is essentially a lightweight target detection network that can be deployed directly to low-end AI acceleration platforms, and it takes into account not only the limitations of computing power on these platforms, but also the impact of memory limitations, context, DRAM speed, and cache hit rate, all of which are particularly influential factors on edge computing platforms. The backbone network of YOLOV4-Tiny is OSANet, and the same with YOLOV4, it refers to the idea of CSPNet, where the main layers are connected by the ResNet part and perform the convolutional operation, and the secondary part is directly connected to the final network layers with the output of the main ResNet layers for concat operation. Some tricks are still similar to YOLOV4, but the Mish activation function is not used, and the structure of SPP is not used, only the maximum pooling layer is added in the middle two layers. The output of YOLOV4-Tiny has only two tensors, which adds up to only 6 anchor boxes, so YOLOV4-Tiny is much faster than YOLOV4.

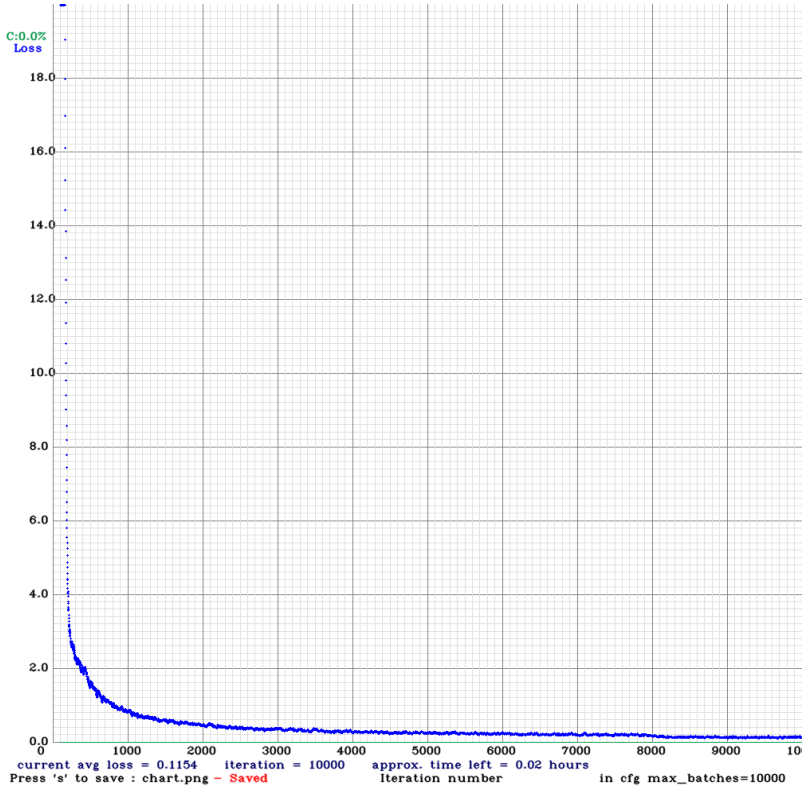


Figure 3-17 Training Process of YOLOV4-Tiny

(The Loss function is more stable than YOLOV3 and YOLOV4 and has less average loss)

In this thesis, YOLOV4-Tiny is also select to train on the MS coco dataset with 80 target species, and own dataset, the training process is shown in Figure 3-17. The inference result of YOLOV4-Tiny on server is shown in Figure 3-18.

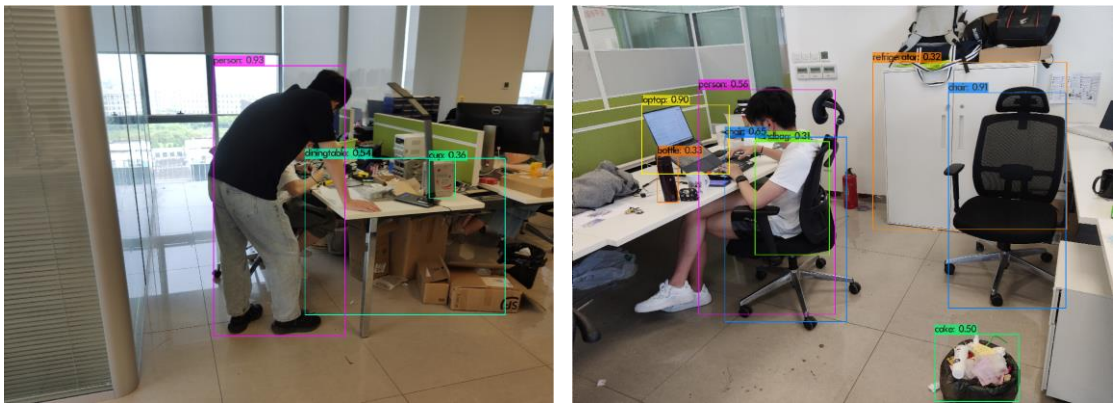


Figure 3-18 Inference Result of YOLOV4-Tiny on Darknet

(Left) Person 0.93 Dining Table 0.54 Cup 0.36

(Right) Chair 0.91 Laptop 0.90 Chair 0.65 Person 0.56 Cake 0.50 Bottle 0.33 Refrigerator 0.32

Then the YOLOV4-Tiny network model is converted to deploy on the RK3399PRO platform for inference. The post-processing here is also referred to the post-processing of YOLOV4, but the output tensor here is only two, so some adjustments are made. In the filtering process of multiple detection windows, the DIOU NMS selected by the previous YOLOV4 post-processing was still used here. The inference result of YOLOV4-Tiny neural network is shown in Figure 3-19. The comparison of YOLOV4 and YOLOV4-Tiny is shown in Table 3-5.

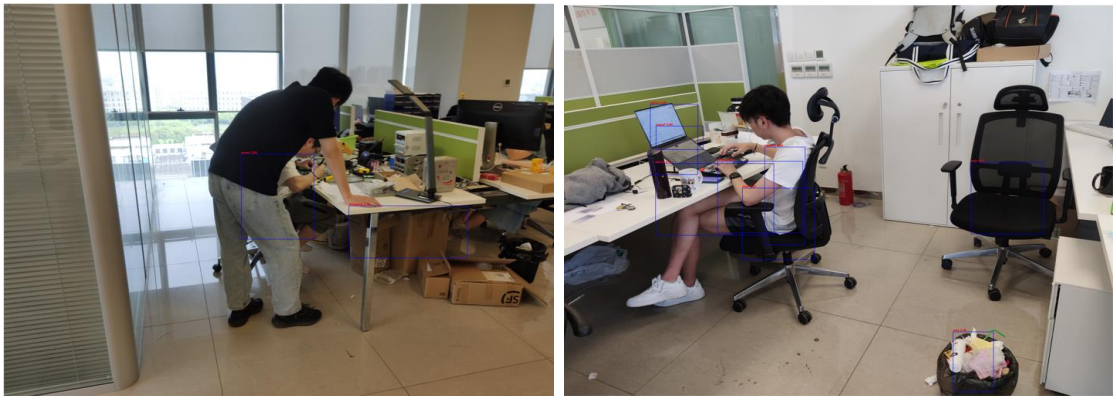


Figure 3-19 Inference Result of YOLOV4-Tiny on RK3399PRO

(Left) Person 0.82 Dining Table 0.42

(Right) Laptop 0.90 Person 0.68 Chair 0.63 Chair 0.48 Cake 0.46

(Almost all the detected boxes are wrong)

Table 3-5 Comparison of YOLOV4 and YOLOV4-Tiny on RK3399PRO

	YOLOV4		YOLOV4-Tiny	
Model Size (MB)	245		22.4	
AP50:95	40.3		20.6	
Quantization	Yes	No	Yes	No
System Memory (MB)	307.08	530.40	31.11	48.02
NPU Memory (MB)	160.03	327.54	19.01	38.89
Time Cost (us)	542642	2622625	20792	453759
FPS	1.84	0.38	2.2	48.1

According to the Figure, YOLOV4-Tiny network has a serious loss of accuracy on the RK3399PRO hardware platform. Although it is lighter, occupies less memory and faster than YOLOV4, the loss of accuracy due to the reduced network structure of YOLOV4-Tiny is unusually high. Whether quantized or not, the minimum accuracy requirements cannot be achieved. Therefore, this thesis finally gives up the choice of YOLOV4-Tiny and still select YOLOV4 as the object detection neural network of the intelligent vision system on the mobile robot platform.

3.5 Network Framework

All deep learning neural networks need a framework for training and inference, only has a network is not enough, you need to build the structure of the network on a framework, and through the input and output to process the data needed. Deep learning has developed to a very mature ecology, including some open source deep learning framework as well as in each hardware manufacturer's closed-source neural network inference framework, which can only deploy on their own chips. Through these

frameworks, the input and output interfaces, network layer construction, model training, model visualization and model testing deployment can be all customized. The same neural networks are implemented differently in different frameworks. Currently the most contributed by developers are TensorFlow, PyTorch, Caffe, etc. The following of this chapter introduces the deep learning framework platform used for the mobile robot platform in this thesis.

3.5.1 Darknet

Darknet is a framework developed by the original authors of the YOLO series for the YOLO algorithm. Darknet was designed with the same idea as YOLO, to be efficient and light, and to greatly improve performance and inference speed while maintain acceptable accuracy. So, Darknet basically has no other package dependencies, and is written from the bottom of C language. It later added NVIDIA's CUDA packages and cuDNN acceleration units for GPU acceleration and support for OPENCV compilation. However, Darknet does not have a large number of developers like TensorFlow and other frameworks, only a small team is developing. The advantage of darknet is that it is easy to compile and no need to install other dependencies. It has clear structure, source code and model network framework and its weight files are very clear, easy to change and transplant. It is easy to deploy darknet on the actual engineering project, which is one of the ideas when YOLO was founded. The object detection neural network on high-performance server in this thesis is built on Darknet framework.

3.5.2 RKNN

RKNN is a framework for neural network inference on Rockchip's own NPU hardware platform, which consists of converting trained neural network models from general deep learning frameworks such as TensorFlow, PyTorch and darknet into RKNN framework format and deploying them to Rockchip’s deep learning acceleration NPU platform to perform forward inference. If the neural network models in other deep learning frameworks are put on ARM platform directly without converting to RKNN framework format, they can only utilize the CPU of ARM, which has very low computing power. While the inference speed can be greatly improved by converting the models into RKNN and accelerating on NPU. The RKNN framework also includes

some operator substitution, merging and optimization, as well as built-in functions such as the accuracy analysis, performance evaluation and memory consumption analysis. Model segmentation can also be performed to improve the accuracy and performance of the model inference. In this thesis, several neural networks are used in the intelligent vision system on mobile robot platform: YOLOV3, YOLOV4, YOLOV4-TINY and other optimized neural networks, they are also converted to the RKNN framework for neural network inference on RK3399PRO platform, the operating system used in the RK3399PRO is the Debian10 Linux distribution. Based on the features of the RK3399PRO hardware, this thesis has done some hardware allocation of computing resources, neural network optimization, operator development, etc.

3.6 Summary

This chapter focuses on different deep learning object detection neural networks and their developments. Their respective characteristic advantages in different scenarios are compared. This chapter then applies ideas of these neural network algorithm and deploys them to high-performance server for training and low-power hardware platforms for inference. The performance and accuracy of different network structures are compared on both RTX2070 hardware and RK3399PRO hardware, and the potential optimization effects of each neural network structure is analyzed. At next this chapter describes the difference of neural network deployment between edge side and cloud server side and introduces deep learning frameworks. Finally, YOLOV4 is selected as the most suitable neural network algorithm for obstacle detection function of intelligent vision system on mobile robot platform in this thesis.

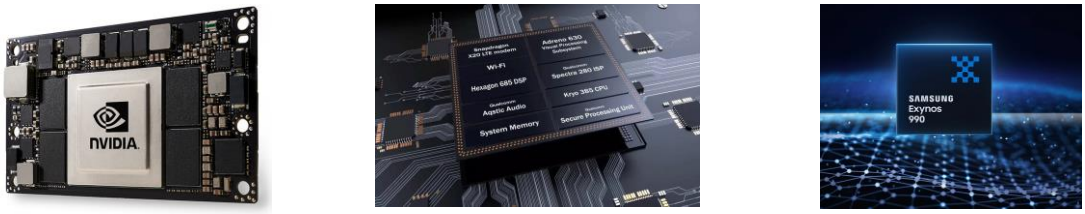
4 Neural Network Optimization on Hardware

Although deep learning neural network is a popular topic in research and engineering, the current neural network inference is basically based on the powerful GPU acceleration in the cloud server, which has very high requirements for the server's computing power, memory speed, multi-threading support, distribution, etc. However, when the neural network inference is performed on smart phones and embedded terminals, the hardware support is not as rich as that on the cloud, so we can only rely on low-end chips for inference. Some developers use cloud-based upstream and downstream data, sending sensor’s data to the cloud for inference and then sending the results back to the low-power terminal, but this approach is very dependent on the performance of the network and consumes a lot of energy to transfer the data to the cloud. Therefore, in most cases, when doing neural network inference in edge computing scenarios, we still have to rely on the local hardware for computing, so we need to make a comprehensive optimization improvement of the neural network structure. This chapter focus on the optimization improvement of YOLOV4 deep learning neural network on the RK3399PRO hardware platform, including operator splitting, replacement, and fusion, conversion of model data layout and improvement of activation function and pooling layer. Also, the model is quantized to 8-bit, and MMSE and Moving Average Min Max algorithms are utilized in the quantization process to update the parameters to minimize the accuracy loss. Finally, the optimized neural network has a good optimization effect on the RK3399PRO hardware platform compared to the original YOLOV4 network.

4.1 General AI Chip Platform

In a board sense, the chips which can run AI algorithms can be called AI chips, but the general meaning of the AI chip platform refers to the chip specifically to AI algorithms to do a special acceleration design. AI chips are mainly optimized for deep learning neural network algorithms and other machine learning algorithms are also involved. At present, the main solutions of major semiconductor manufacturers are based on GPU, FPGA and ASIC, which are the main directions. The ASICs include a variety of TPU, NPU, DSP, etc. GPUs are versatile, fast and efficient, but along with them comes high

power consumption. FPGAs have the advantages of low power consumption and programmability, but the requirements for developers are much higher. The ASIC side includes hardware-level optimizations for higher performance, power consumption ratio, and is tailored for AI, but with the rapid development of deep learning algorithms, the hardware IC level design side also needs to be updated quickly to accommodate the latest algorithms. The hardware platform AI chip used in this thesis is the RK3399PRO, and the solution chosen is a customized hardware with a built-in NPU for neural network inference acceleration, and the optimization of the YOLOV4 neural network is also based on this NPU to do the optimization. Several chips with AI acceleration units are shown in Figure 4-1.



(a) (b) (c)

Figure 4-1 Different AI Acceleration Chips (a) Nvidia's GPU (b) Qualcomm's DSP (c) SAMSUNG'S NPU

4.2 Hardware Architecture

The mobile robot vision platform in this thesis uses a development board based on the RK3399PRO chip. In the architecture of RK3399PRO, the CPU of the chip adopts a 6-core 64-bit ARM processor, including a dual-core Cortex-A72 and a quad-core Cortex-A53, and the access consistency between these two clusters is ensured by CCI500, each A72 core has 48KB of L1 instruction cache and 32KB of data Each A72 core has 48KB of L1 instruction cache and 32KB of data cache (4 groups are linked); each A53 core has 32KB of L1 cache and 32KB of L1 data cache (4 groups are linked), and each cluster shares one L2 cache, 1MB for the large cluster (A72) and 512KB for the small cluster (A53). Then in the multimedia image processing part, the GPU uses quad-core ARM Mail-T860 MP4 GPU, the VPU supports 4K VP9 and 4K 10bits H265/H264 video decoding up to 60 fps and the Image Enhancement Processor (IEP) has a number of video post-processing functions, including image enhancement, image noise

reduction, anti-interlacing, edge sharpening, detail and color optimization, etc.

The core part of RK3399PRO is an AI acceleration unit NPU, which can support 8bit and 16bit neural network inference operations with computing power up to 3.0 TOPs, and the main AI acceleration core of this NPU is the use of Multiplier and Accumulation (MAC) acceleration array to achieve the acceleration of the most important convolutional operations in CNN. The built-in core can reach a maximum of 1920 MACs per cycle at int8 data type and 192 MACs per cycle at int16. While the amount of data contained in the neural network is particularly large and requires a lot of memory bandwidth. The NPU has a built-in buffer of 512 kilobytes, which can also be used to store the data that needs to be read many times to improve the inference performance of the whole network. The architecture of the System on Chip (SoC) and the resource allocation is shown in Figure 4-2.

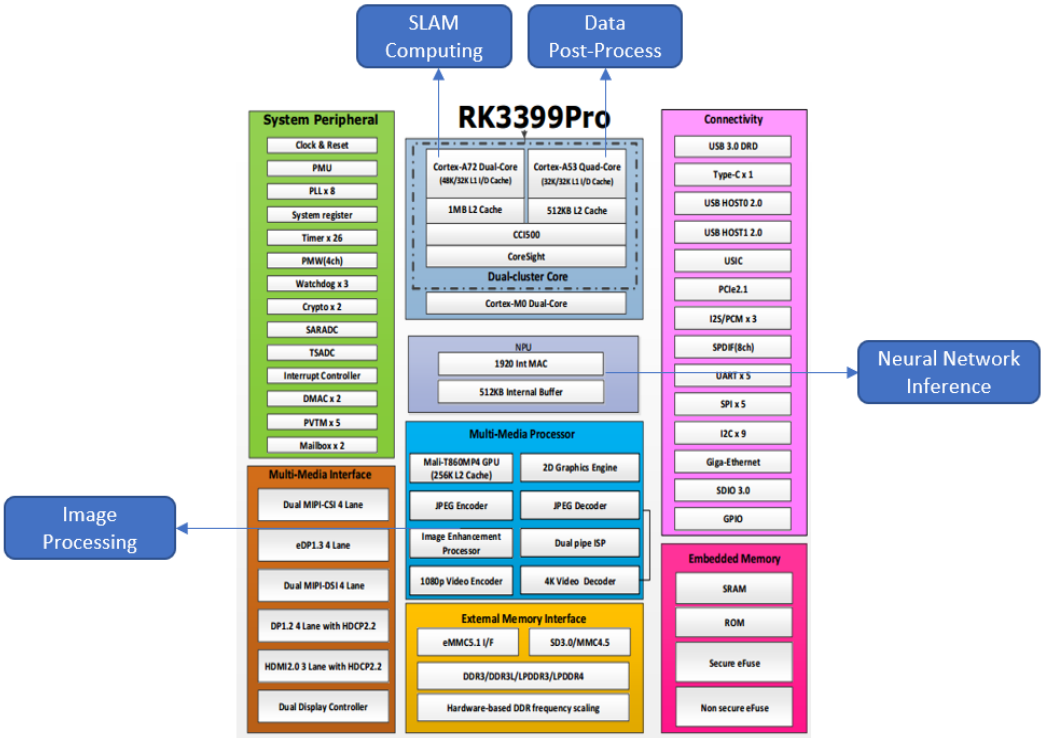


Figure 4-2 RK3399PRO Soc Architecture

The peripheral circuitry of the development board is arranged as follows: the memory includes 32bit 2GB LPDDR3 memory provided to the CPU, while the NPU section provides a separate 32bit 1GB LPDDR3 memory. This 1GB of dedicated memory is sufficient to store the weights and activations included in the neural network operation,

plus the built-in buffer can greatly improve the time consumption required for data accessing during neural network inference. Meanwhile, the peripheral interfaces include USB, PCIE, Ethernet port and MIPI-CSI camera interface. The specific use of these interfaces includes the connection of subsequent monocular cameras, 2D laser LiDAR and other sensors. The development board structure and the connection of some interfaces are shown in figure 4-3.

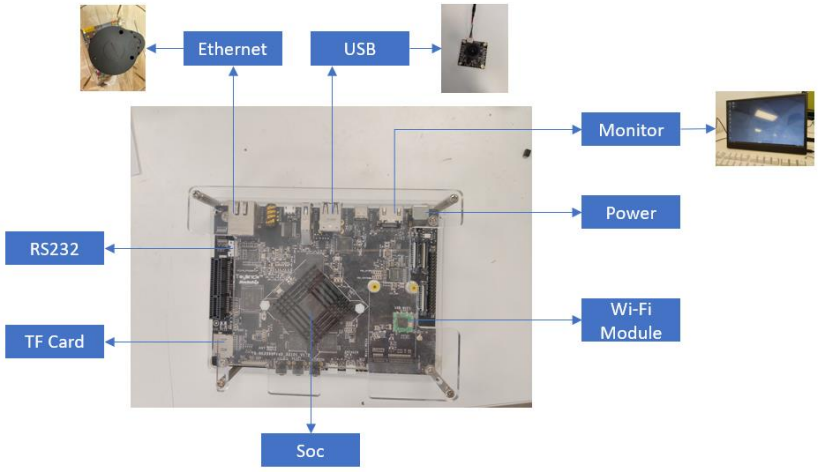


Figure 4-3 Development Board Connection Diagram

4.3 TVM

AI has been in a limelight today, but most developers and researchers only focus on algorithm level research. When AI needs to be practically applied to daily lives, engineering related issues is crucial. Usually, the deployment of a deep learning neural network is divided into two parts, training and inference. Currently for training, the most popular way is to use TensorFlow, PyTorch and other open source frameworks, the mainstream frameworks have basically determined the process. However, in neural network inference, because of the different hardware platforms are deployed, inference frameworks are also different. In order to achieve a model transformation that enables neural networks to run efficiently on inference frameworks of various hardware platforms, TVM emerges. The front-end of TVM is a deep learning compiler that compiles models under various neural network frameworks, which outputs an Intermediate Representation (IR) layer called computational graph IR, and the back-end of TVM optimizes these computational graph IRs to spit out an IR for a back-end

specified hardware platform, and finally puts them into the hardware platform for compilation and inference[59]. The compilation and optimization process of different neural network models on TVM is shown in Figure 4-4.

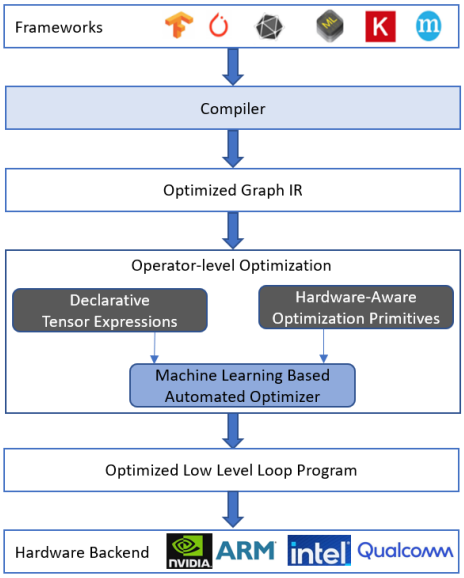


Figure 4-4 Compilation and Optimization Process of TVM

There are many engineers and researchers who are developing projects based on TVM, and there are also many developers who have borrowed the optimization ideas of TVM. In this thesis, the model conversion from DARKNET to RKNN is performed, and the optimization of the operator development is also borrowed from the idea of graph IR optimization. Some optimized algorithms are used to do the optimal allocation of the underlying operators of the neural network, which can significantly improve the performance of neural network inference on the hardware platform.

4.4 Model Transform

Before neural network inference, we need to convert the original model under the training framework to the model under the inference framework. So, we need a model conversion process. During the conversion, the operator replacement, memory layout and scheduling all greatly affect the performance of the final inference process, without considering the neural network weights change, these operations do not affect the accuracy.

For the memory layout part, the storage format of each tensor in neural network model

under the Darknet framework is HWC format, because only one batch data will be input at one time, that is, one image, and the output is a format of HxWx255 as mentioned in Chapter 3. While the hardware platform is not only for CV-related neural network algorithms, but also for other neural network algorithms. Multiple batch inputs like timing sequence may be involved in other neural network algorithms. Therefore, it is necessary to change the storage format corresponding to each layer of individual tensor to NHWC format during the model conversion. The advantage of NHWC format is that the values of the three RGB channels of a single point can be read at one time, refer to Figure 4-5, because the data is stored sequentially inside the hardware memory structure, which reduces the time consumed to read the memory.

Meanwhile, in the dedicated memory of NPU and the built-in buffer, the hardware memory has been optimized for accessing the weights data of each layer of convolution kernel in the neural network. Under the Darknet framework, the weights' storage format is IOHW, while under the RKNN framework, the weights need to be converted into HWIO format. Regardless of how many dimensions the data is logically expressed, it is stored in the computer according to 1D format. Therefore, when reading the data serially, the RK3399PRO hardware platform can read the values in the height and width direction of each convolutional kernel sequentially, and then process each feature map sequentially. So that after the same convolutional kernel is processed, it can be quickly passed to the next layer for operation or temporarily stored in the internal buffer. If IOHW is used, it is not possible to split the parameters of individual convolutional kernels. It is necessary to read the HW of all convolutional kernels from DDR at once and then perform the operation. The memory format difference of the convolutional kernels can be referred to the memory format difference of the tensor's format, which is similar and shown in Figure 4-5.

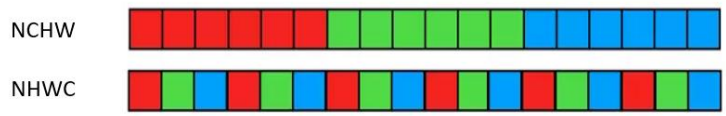


Figure 4-5 Memory Layout Difference

YOLOV4 fuses three parts of operations, Conv, BN and Mish, into one single unit CBM unit in the Darknet framework. While in the RKNN framework, it is necessary to first separate this single fused unit. Since the inference does not require the neural network

to continuously learn the distribution of the data as in training, only the input layer needs to add a normalization of the input image with setting the standard deviation value, mean value and the channel. Therefore, the BN layer can also be optimized separately. The parameters of Conv and the activation layers are quantized separately, and the optimization can be performed at different operator levels. On the mobile robot platform in this thesis, the input data format needs to be set to RGB, and the quantization, pruning and other operator developments need to be performed during the model conversion.

4.5 Quantization

With the development of deep learning, more and more state-of-the-art neural network algorithms are getting deeper and deeper, the operators are more and more complex. These models introduce a huge number of parameters and computation. When need to deploy these models to the mobile terminals, it will encounter some bottlenecks in hardware resources. In order to improve the performance of neural networks in edge computing scenarios, the model quantization came out, which means that the original neural network's floating point operations to be converted into low-bit fixed-point operations in order to achieve lower memory occupation and higher inference efficiency. Many embedded platforms do not support floating point operations like MCUs and DSPs. Because the CNN is insensitive to noise, the quantization does not seriously affect the noise if trained properly. If the AI chip is optimized specifically for fixed-point operations, it can bring higher inference performance, but it also brings a problem, that is, the loss of neural network accuracy. Thus, before quantization, training has to be done with accurate float point. Then in order to achieve optimizations, parameters of the algorithm used in quantization are as challenging as the parameters of the neural network itself.

There are also a variety of different quantization algorithms. The major categories are divided into Quantization Aware Training (QAT) and Post-Training Quantization (PTQ). PTQ is the quantization of the completed training model. In QAT, the quantization is achieved during training, while the gradient will be 0 due to quantization, and it cannot be back-propagated to the front layers of neural network, which will lead to a serious decrease in training accuracy, and there is no good solution for this problem in engineering now. In most of the cases developers still using PTQ to quantize the

trained model weights.

The PTQ process steps are shown in Figure 4-6.

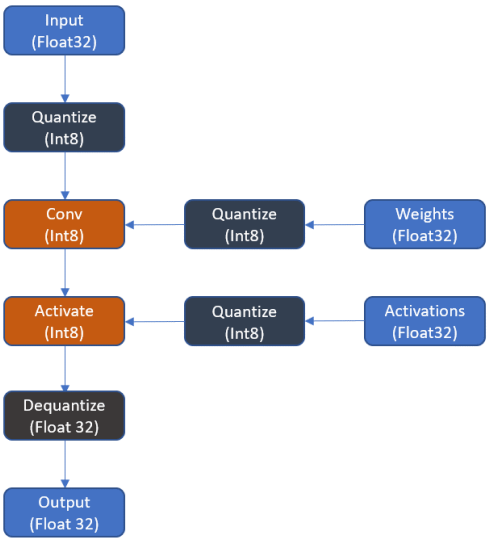


Figure 4-6 PTQ Process

4.5.1 Asymmetric Affine Quantization

The asymmetric affine quantization algorithm refers to linear mapping the neural network data of fp32 to the representation range of int8. The asymmetric means that the distribution range of the data after quantization is not symmetric with the distribution range before quantization, that is, the zero point is different, and the zero point after quantization is not the same as the zero point of initial weights. The advantage of this quantization is that the data distribution of weights and activation of the original model is not evenly distributed on both sides of the zero point in most cases. The formula is as follows.

$$x_{quantized} = round\left(scale \cdot x_{float} - Zero_{point}\right) \qquad (4-1)$$

Since the bias of the original convolutional layer does not have a significant impact on the accuracy when performing network inference on hardware platform, it is simply removed here. In order to calculate the scale and zero point in the quantization process we need to know the actual dynamic range of fp32 weight and activation. For the inference process, weight is a constant tensor that does not require additional data sets to be sampled to determine the actual dynamic range. The actual dynamic range of activation, however, must be sampled (a process generally referred to as data

calibration), which means that a certain amount of input data is needed for quantization, usually several images covering all the application scenarios. Through these actual input data, the parameters for quantization are calibrated.

One of the simplest and most used sampling methods is the maximum-minimum (MinMax) algorithm. The basic idea is to determine the actual dynamic range by selecting the maximum and minimum values directly from the fp32 tensor, which is shown in the following formula.

$$x_{min} = \begin{cases} \min(X), & if\ x_{min} = None \\ \min(x_{min}, \min(X)), & otherwise \end{cases} \quad (4-2)$$

$$x_{max} = \begin{cases} \max(X), & if\ x_{max} = None \\ \max(x_{max}, \max(X)), & otherwise \end{cases} \quad (4-3)$$

For weights, this sampling method is not saturated. For activation, it is not saturated in the sampled data. But it can be saturated if there are data outside the actual dynamic range in the validation dataset. The advantage of this algorithm is that it is simple and straightforward. However, for activation data, if there are outliers in the sampled data, the actual dynamic range can be significantly extended. For instance, 99% of the data are evenly distributed between [-300, 300] in the actual calculation, but there is an outlier with a value of 50,000 when sampling, so the dynamic range obtained by this sampling algorithm becomes [-300, 50,000], which will seriously affect the accuracy after quantization.

Therefore, the thesis using Moving Average MinMax sampling algorithm. Unlike the MinMax algorithm, which is a direct sampling, Moving Average MinMax uses a hyperparameter c (default value is 0.01) to update the dynamic range step by step.

$$x_{min} = \begin{cases} \min(X), & if\ x_{min} = None \\ (1-\mathrm{c})x_{min} + \mathrm{cmin}(X), & otherwise \end{cases} \quad (4-4)$$

$$x_{max} = \begin{cases} \max(X), & if\ x_{max} = None \\ (1-\mathrm{c})x_{max} + \mathrm{cmax}(X), & otherwise \end{cases} \quad (4-5)$$

The dynamic range of activation data obtained by this algorithm is generally smaller than the actual dynamic range. While for weights, the effect of Moving Average MinMax is the same as MinMax because there is no iteration of sampling.

The different parameters selected during quantization can directly affect the final accuracy loss. The error analysis data plot of the quantization using asymmetric affine algorithm is shown in Figure 4-7. The error analysis represents the calculation of the Euclidean distance and Cosine distance between the original layer of the model and the quantized layer, both of them are normalized to the range of 0-1. When the Euclidean distance is smaller and the Cosine distance is larger, it represents the smaller data difference between the quantized layers and original layers, which means the less accuracy loss after quantization.

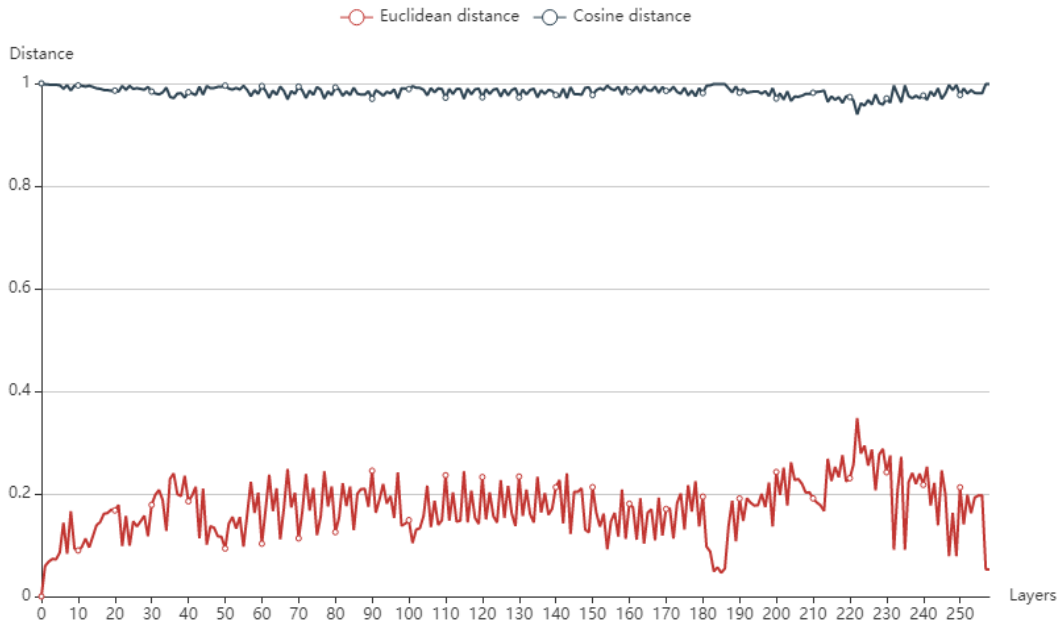


Figure 4-7 Accuracy Analysis of Asymmetric Affine Quantization

4.5.2 Dynamic Fixed Point Quantization

Another point we need to consider during quantization is that the range of data value in each layer of the neural network is not necessarily the same, which gives rise to a problem of dynamic value range. In large layers, the output is the result of the accumulation of thousands, so the network weights' parameters are much smaller than the output. The fixed point algorithm has only a limited ability to cover a wide dynamic range, where fixed point refers to the number of decimal places. Therefore, a dynamic fixed-point quantization algorithm is used to quantize the parameters of different layers separately, which means different layers and groups have different number of decimal

places. The formula of this algorithm is as follows.

$$n = (-1)^s \cdot 2^{-fl} \cdot \sum_{i=0}^{B-2} 2^i \cdot x_i \qquad (4-6)$$

Here the number of decimal places is constant within the same group, the front part of the formula represents the sign digit, the middle is the length of the fraction taken, and the back represents the trailing digit, so the number of decimal places will be different for different network layers. And the data in each network layer of the neural network can be divided into three groups, one for the input layer, one for the weights, and one for the output layer. The number of decimal places is different for different groups. The quantization of activation layer is the same. According to the number of bits needed to quantize the integer part, subtracting this bit width from 8 bits is the bit width of the fractional part. The scale parameter is also calculated by a dynamic range selection algorithm which is similar to the asymmetric affine quantization. The dynamic fixed-point algorithm allows better coverage of the dynamic range of the data in each part, since the weights are generally smaller than the input and output of the layer. And the normal round operation brings some errors, so here use the stochastic method to perform round operation. The formula is as follows.

$$Round(x) = \begin{cases} \lfloor x \rfloor, & with\ probability\ 1 - (x - \lfloor x \rfloor) \\ \lfloor x \rfloor + 1, & with\ probability\ x - \lfloor x \rfloor \end{cases} \qquad (4-7)$$

The accuracy analysis data of dynamic fixed point quantization algorithm is shown in Figure 4-8. And the comparison of quantization accuracy between asymmetric affine and dynamic fixed point algorithms is shown in Figure 4-9.

According to the results, the asymmetric affine quantization algorithm has less accuracy loss compared to the dynamic fixed point quantization algorithm, so the quantization process of the object detection neural network in this thesis adopts the asymmetric affine quantization algorithm.

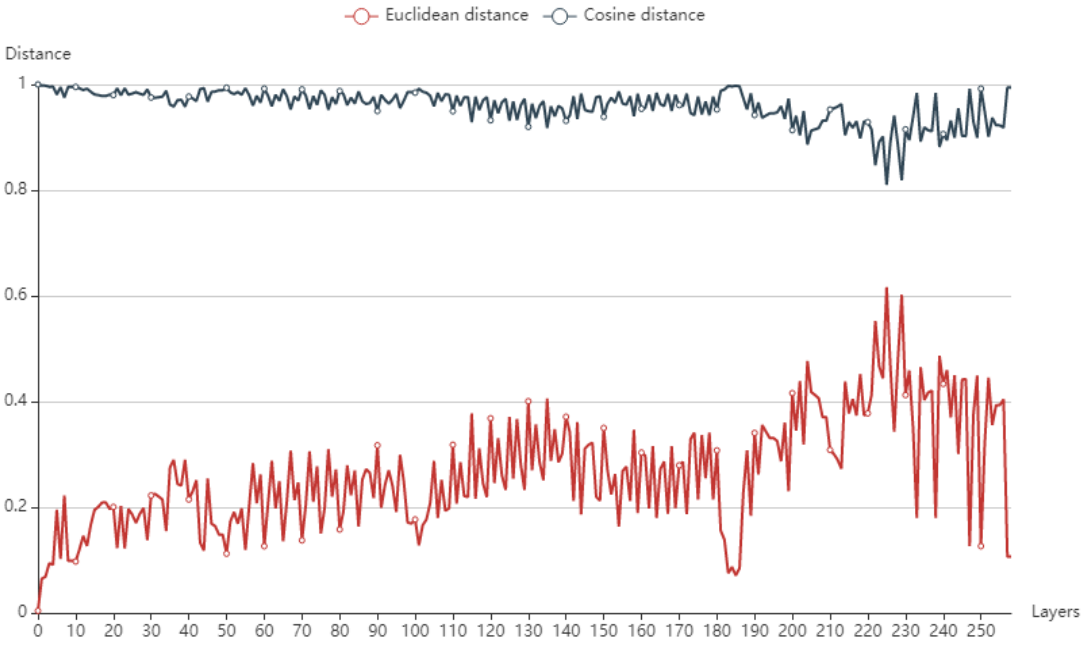


Figure 4-8 Accuracy Analysis of Dynamic Fixed Point

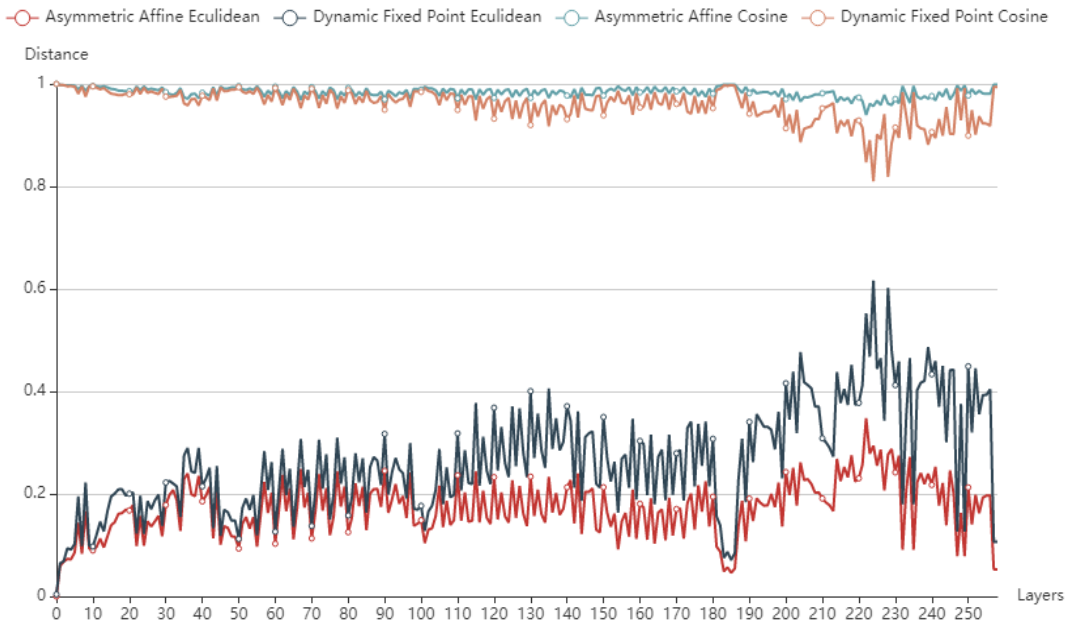


Figure 4-9 Comparison between Two Quantization Algorithms

4.5.3 MMSE Optimization

MMSE algorithm refers to minimize the mean squared error, it focuses on a set of random data to calculate their distribution characteristics, representing the average of the sum of the squares of the differences of each data deviation from the true value, which is initially used in communication engineering. As an equalizer, MMSE is a post-processing algorithm that helps us to find out the received data and make it is as close as possible to the original transmission data. In short, the most important step in MMSE is to find a matrix G, which can simply be the inverse of the channel transfer matrix H^{-1} if we assume that there is no noise. When there is some noise, MMSE is used to reflect the noise. While in deep learning, MMSE is usually used as a loss function during neural network training, and the mean square error is calculated by the difference between the predicted value and the ground truth, and then the weight parameters of each convolutional kernel of the previous neural network are continuously updated to achieve an optimal model training.

However, the MMSE algorithm used in this thesis is mainly for the optimization of the quantization algorithm. MSE represents the expected value of the square of the distance between the quantized layer and the original fp32 layer. The smaller the MSE value is, the smaller the loss of accuracy after quantization. So, the MMSE quantization algorithm here is to calculate the MSE value by setting different scale, zero point and other quantization parameters each time, and then iterate this operation by changing the parameters until the minimum MSE is reached. That is, the accuracy of the quantized model is the closest to the original model. In the intelligent vision system on mobile robot platform in this thesis, MMSE optimization is applied to the asymmetric affine quantization algorithm. The formula is as follows.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(quantized - float)^2 \qquad (4-8)$$

The accuracy analysis comparison between normal asymmetric affine quantization algorithm and MMSE optimized quantization is shown in Figure 4-10. According to the results, after using MMSE to optimize the quantization, it is obvious that the error of the model is much smaller and the decrease in accuracy is also much lower than the normal quantization. The results shown are only the neural network model quantized

after 5 iterations of MSE calculation. Theoretically, the more iterations, the longer it takes to run in quantification, but the accuracy will be better after quantization.

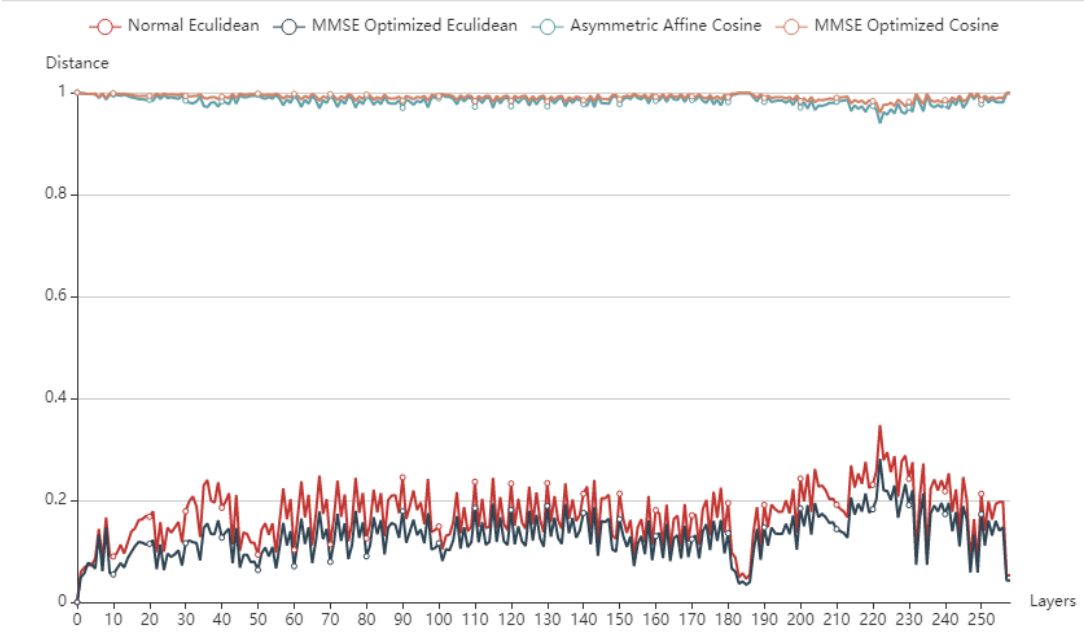


Figure 4-10 MMSE Optimization Result

4.5.4 Mixed Precision

Mixed precision refers to the different precision of data between different layers of the neural network, some layers use fp32, some layers use int16 and some layers use int8. The advantage of mixed precision is that some layers of the network will lose accuracy if they are quantized by int8. If we just keep the original float32 data type for these layers and use quantized int8 for the other layers, the accuracy loss is not too much and the performance of the forward inference of the neural network will not be degraded too much. There are two ways to implement mixed precision, one is mixed precision training, and the other is mixed precision quantization. The mixed precision training has a certain effect on the gradient and is not conducive for backward propagation and requires a better loss function and hyperparameters to train a model with better accuracy. In practice this implementation is more complicated and troublesome.

Therefore, in this thesis, mixed precision quantization is deployed. For different neural network layers, different quantization methods and data formats are chosen. The higher precision data format is used for some "sensitive" layers of the network, while the lower

precision data format is used for the "non-sensitive" layers. However, it is not easy to achieve the optimal solution for mixed precision, which requires a lot of repeated quantization, repeated inference and even sometimes repeated training to evaluate the accuracy of the current mixed precision model. Also, the inference performance needs to be considered, as the inference speed drops rapidly when the number of fp32 becomes larger. This thesis also has tried making part of the layer using fp32 format and other layers using int8 quantization. While because the inference optimization of fp32 data format is really so poor on the NPU of RK3399PRO. Finally, the thesis still uses a whole int8 quantization idea. But the idea of mixed precision is fine, and if in other hardware platforms with better support and expandability, this structure should be able to achieve an optimal result. The idea of mixed precision is shown in Figure 4-11.

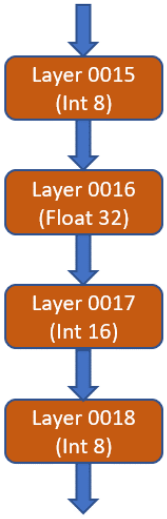


Figure 4-11 Mixed Precision Process

4.6 Pruning

Pruning is also a commonly used method for neural network model compression. As the name implies, pruning means to prune a neural network to remove some neurons that do not contribute much. And this operation, like the quantization described in the previous chapter, is performed on a trained neural network model. The redundancy of the model is reduced by cutting out some unimportant connections or filters by certain criteria. The criteria for pruning algorithms are easy to obtain, including the accuracy

and the inference performance of the neural network model before and after pruning. While how to prune and which criteria to set to judge the degree of contribution of a single neuron on the whole network, these are the focus of attention and the core part of pruning.

Pruning is also an iterative process like quantization, through continuous pruning, continuous training and testing to evaluate the performance and accuracy. Currently, there is not any pruning criterion which has a high degree of conviction in engineering. Only a few developers have tried large pruning algorithms in practical engineering applications. The reason is that these algorithms are often designed to optimize a single neural network model, but when moving in the direction of universality, they often lead to a significant loss of accuracy. And the weights and activation values are different, which represents different convolutional kernels have different levels of contribution. This is also an important factor, which leads to the fact that pruning is not very practical at the moment.

The current pruning algorithms are divided into two parts, one is weight-based pruning, another one is activation-based pruning. Weight-based pruning using the magnitude of the weights value of each layer to judge the importance of each convolutional kernel. For instance, using the sum of the absolute values of all the weights in one filter as a criterion, then the filters with low value will be pruned. This approach can effectively reduce the model complexity and does not bring great loss to the model[60]. In large neural networks, most of the neuron activations tend to be 0. For the neurons with 0 activations, they can be pruned to greatly reduce the size of the model and the number of operations. This approach will not affect the performance of the model. The number of values with zero activation in each filter can be measured by Average Percentage of Zeros (APoZ), and it can be a criterion to evaluate whether a filter is important or not[61]. This thesis also chooses the weight parameter as the criterion to implement pruning operation on the YOLOV4 neural network framework. First, consider the minimum block: CBM in YOLOV4, and the minimum number of channels to be retained for each CBM is 2^n. Then, set the threshold value for the γ parameter of the BN layer according to the evaluation criteria, and the channels smaller than the threshold value are pruned. But we need to retain the minimum number of channels required for each convolutional kernel, so we cannot prune them all and the channels with the largest weights are kept according to the weight data sorting. The neural network pruned in this way has an inference speed increase in high-performance platform. While because the difference

of memory resource between server and RK3399PRO platform, according to the actual inference results, there is almost no big difference between original model and pruned model. And it brings the problem of accuracy loss, so the thesis finally did not implement pruning for the network framework. However, the pruning algorithm is still evolving, and the pruning effect may be better for large neural network models. On the YOLO, MobileNet and other network frameworks which are hardware-side friendly, efficient and light, the pruning results are not good, each convolutional kernel of the neural network occupies a certain impact. So, in the low-power embedded platform, there is a need for further development of pruning algorithms.

4.7 Operator Development

The development of operators is an extremely important part of the model conversion and deployment of neural network on hardware platform. The minimal granularity of neural network is the operator, so while doing data layout conversion, quantization and pruning, the development of operators is also necessary during the deployment of neural network on the hardware platform. The development of operators includes the splitting, replacement, fusion and discarding. These operations are based on different hardware structures, characteristics to specific analysis and optimization, because some hardware manufacturers will do certain optimization for different types of operators on the AI acceleration chip.

4.7.1 Activation Function

The activation function is an integral part of a neural network, as the name implies, the activation function is used to activate each neuron. Each convolutional kernel in neural network has a corresponding activation function, which receives the output of each convolutional kernel, activates it, and then transfers it to the next convolutional kernel. The main purpose of the activation function is to add nonlinear representation to the neural network. If all the neurons of one neural network are transferred linearly, then even if the network is very deep and complex, the output of the whole network will always be a linear combination of the inputs, and the functions which can be represented by neural networks is very limited. With the nonlinearity brought by the activation function, the expressiveness of the neural network is greatly improved, and it has the

ability to approximate any function. At the same time, the neural network needs to perform backward propagation to update the weights during training, so the activation function needs to be differentiable to apply gradient descent.

The idea of activation function in neural network is shown in Figure 4-12.

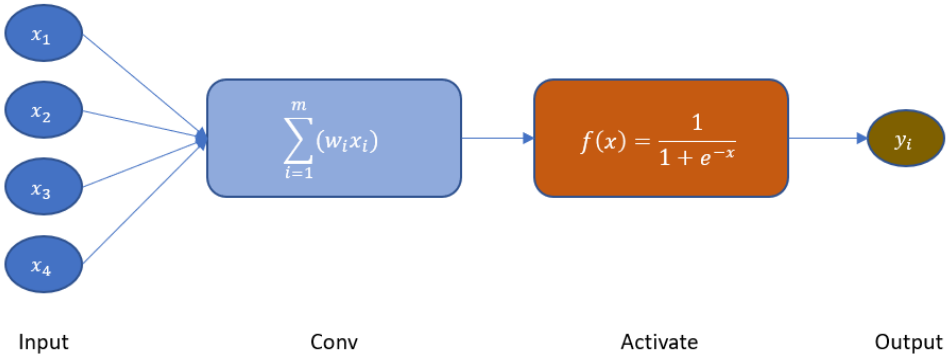


Figure 4-12 Activation Function Process

There are many different activation functions that can be applied, and each of them has its own advantages and disadvantages.

Sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (4-9)$$

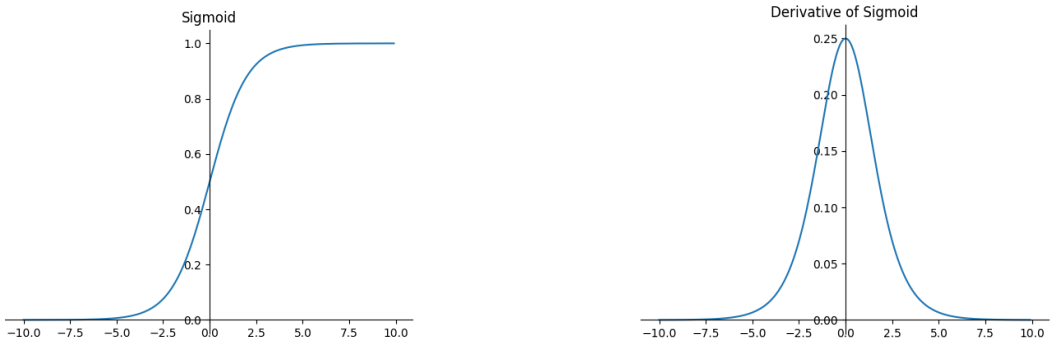


Figure 4-13 Sigmoid Function Curve

Sigmoid transforms the continuous real value of input into an output between 0 and 1, positive infinity of input tends to 1 and negative infinity of inputs tends to 0. Previously, sigmoid is a function used by many developers, in recent years less people use it, mainly

because that when we initialize the neural network weights to random values between [0, 1], during the backward propagation derivation in neural network, the gradient is reduced to 0.25 times of the original value at each layer, and if the neural network has many layers, the gradient will become especially small and close to 0 after passing through these layers, which will leads to the phenomenon of gradient vanishing. If we initialize the weights of the neural network at random values between $[1, +\infty]$, then there is a possibility of gradient explosion. And the mean value of output is not 0, which will reduce the efficiency of weights update. Another disadvantage is that sigmoid with power of exponential operations, the calculation of this function is more time-consuming. For neural network forward inference, it is okay, but for backward propagation training, this function will greatly increase the time.

Tanh:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4-10)$$

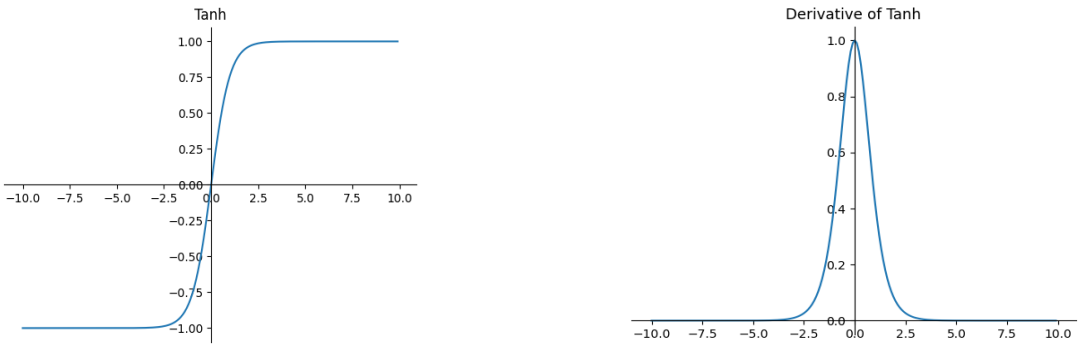


Figure 4-14 Tanh Function Curve

The curve of Tanh is similar to the Sigmoid, but Tanh has some advantages. The output is almost smooth and has a small gradient when the input is extremely large or small. And Tanh solves the problem that the mean value of layer's output is not 0, the output of tanh has an interval of 1 and centered on 0. However, the same problem of gradient disappearance and power operation of Sigmoid also exist in Tanh.

ReLU:

$$f(x) = \max(0, x) \qquad (4-11)$$

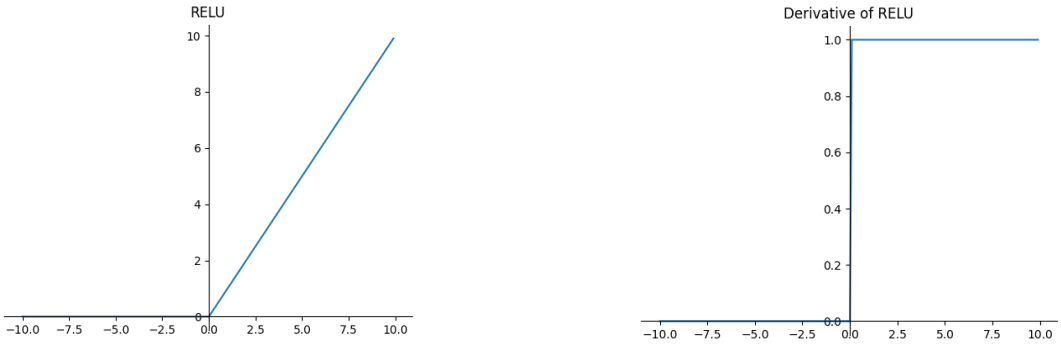


Figure 4-15 ReLU Function Curve

ReLU is currently one of the most used activation functions by developers. The calculation of ReLU is very simple, it is a function that takes the maximum value. ReLU is not fully interval differentiable, it is not differentiable at 0, but it can consider calculating the sub-gradient. Because the output of ReLU is not interval bounded and it tends to infinity. When the input of the activation function is positive, ReLU does not have any problem of gradient vanishing or exploding which exist in sigmoid and Tanh. However, ReLU also contains some problems, its mean value of output is not 0. And when the input is negative, the neuron completely fails and will no longer be activated. It is fine during forward inference, but during the backward propagation training, the gradient will become 0, which may seriously affect the training accuracy. Despite these two minor problems, ReLU is still the most widely used activation function.

Leaky ReLU:

$$f(x) = \max(0.01x, x) \qquad (4-12)$$

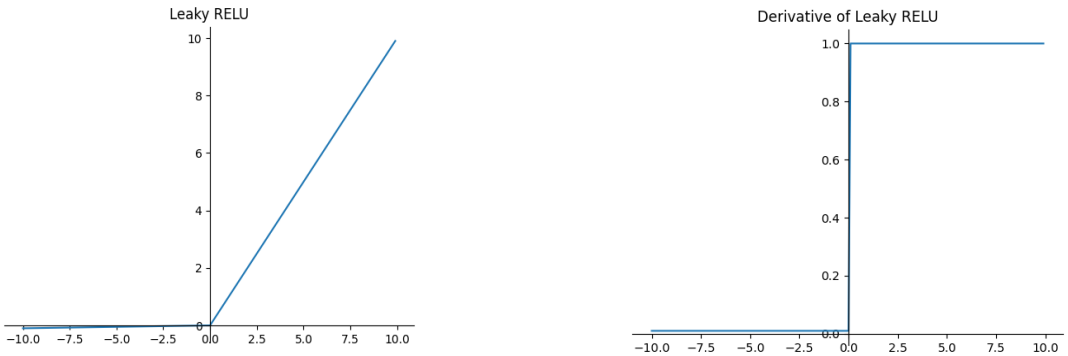


Figure 4-16 Leaky ReLU Function Curve

In order to solve the dead ReLU problem in ReLU, Leaky ReLU emerged, by giving very small linear components to the negative output value, so that these neurons can be activated when the input value is negative, and the gradient vanishing problem will not disappear during backward propagation training. Leaky ReLU extends the range of functions of ReLU, where both input and output are positive infinity to negative infinity. Theoretically, Leaky ReLU has all the advantages of ReLU without consuming more computing resources.

Mish:

$$f(x) = x \cdot \tanh\left(\ln(1 + e^{x})\right) \qquad (4-13)$$

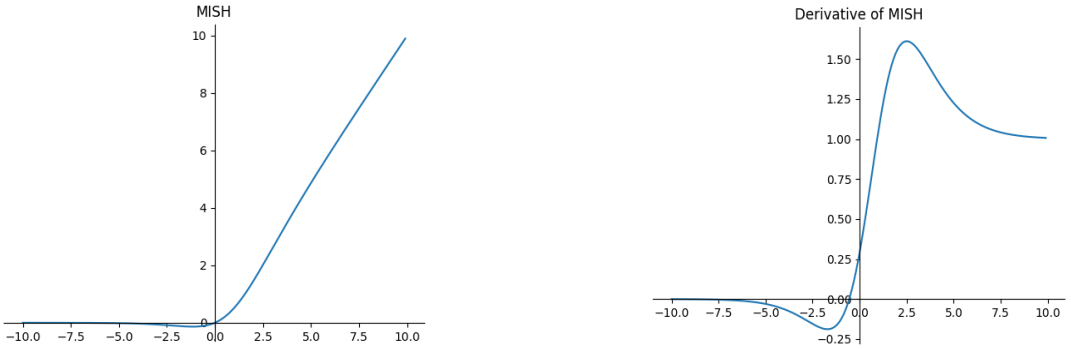


Figure 4-17 Mish Function Curve

Tanh has a strong nonlinear expression and good derivative properties but has some problems such as gradient exploding and vanishing. ReLU avoids the problem of gradient saturation, but has a weak nonlinear expression, requires a very deep network, needs to stack multiple layers, and has bad derivative properties. In order to integrate the advantages of both, Mish emerged, and Mish is a combination of Tanh and ReLU. YOLOV4 object detection neural network has utilized Mish activation function. Mish, compared to ReLU, allows a small gradient propagation when the input is negative, rather than directly to 0, which leads to most neurons failing to activate. This is consistent with the idea of Leaky ReLU. Mish has better smoothness than Leaky ReLU, it is fully differentiable. Smooth activation functions allow for better data propagation in the neural network, which can improve the accuracy and the generalization of the model. According to the experimental data, Mish is much more stable and accurate than ReLU during training, while the computational complexity of Mish is a bit higher.

However, the above results are only based on the powerful GPU on cloud server, when

the neural network is deployed on low-power hardware platform, the consumption of computing resources of different activation functions will be magnified. The comparison of neural network models with Leaky ReLU and Mish respectively on RTX2070 GPU platform are shown in Table 4-1 and the comparison results on RK3399PRO platform are shown in Table 4-2.

It can be seen that there is little difference in the performance of the models with Mish and Leaky ReLU on the high-performance hardware terminal. However, Mish will occupy much more memory than Leaky ReLU on NPU of RK3399PRO platform.

Table 4-1 Comparison of Models with Leaky ReLU and Mish on Server

	Leaky ReLU	Mish
Model Size (MB)	244	245
AP50:95	39.6	40.3
Allocate Memory (MB)	18.88	18.88
BFLOPS	60.137	60.137
Inference Cost (ms)	34.48	34.012
FPS	32.2	31.8

The inference time spent by each activation function layer in these models are shown in Figure 4-18, the models include the original and quantized models. It can be seen that the inference time consumed by activation function layer with Mish far exceed that consumed by activation layer with Leaky ReLU, which leads to the inference speed of the neural network with Mish activation function becoming very slow and failing to meet the real-time requirements on the embedded terminal. And the inference performance of Leaky ReLU increases significantly after quantization, while there is almost no difference of Mish between original layer and quantized layer. The accuracy improvement brought by Mish does not have any advantage after model conversion and quantization. Mish's advantage over Leaky ReLU on high-performance hardware platform seems negligible on the RK3399PRO platform.

Table 4-2 Comparison of Models with Leaky ReLU and Mish on RK3399PRO

	Leaky ReLU		Mish	
Input Size	416x416	608x608	416x416	608x608
System Memory (MB)	258.24	259.06	307.08	344.99
NPU Memory (MB)	158.94	159.60	160.03	178.20
Inference Cost (us)	89856	90145	542642	1166208
FPS	11.13	11.09	1.84	0.86

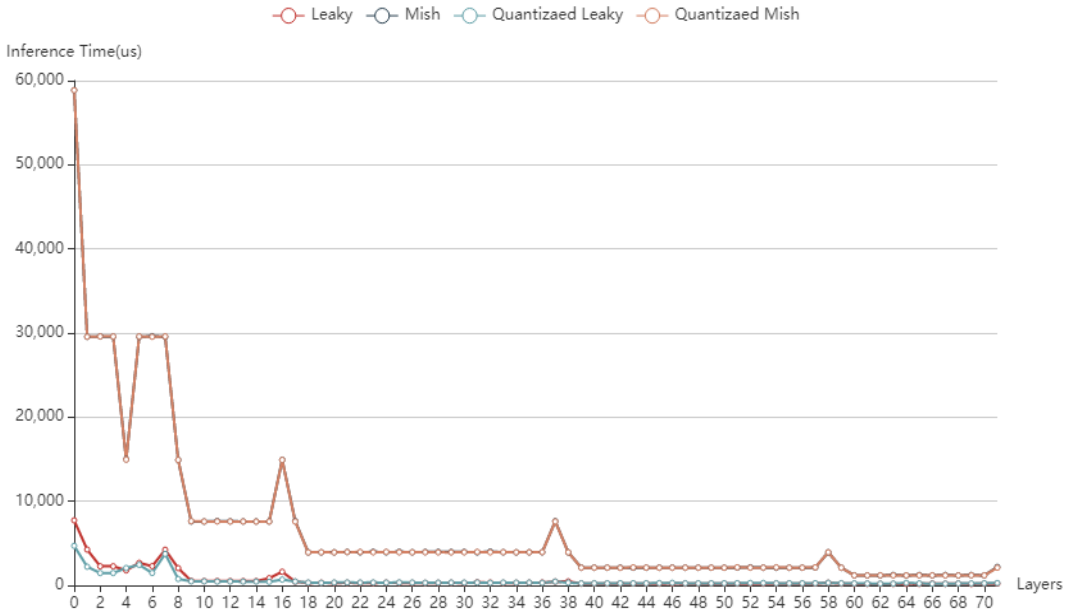


Figure 4-18 Inference Time of Leaky ReLU and Mish Layers on RK3399PRO

The inference time spent by each activation function layer in these models are shown in Figure 4-18, the models include the original and quantized models. It can be seen that the inference time consumed by activation function layer with Mish far exceed that consumed by activation layer with Leaky ReLU, which leads to the inference speed of the neural network with Mish activation function becoming very slow and failing to

meet the real-time requirements on the embedded terminal. And the inference performance of Leaky ReLU increases significantly after quantization, while there is almost no difference of Mish between original layer and quantized layer. The accuracy improvement brought by Mish does not have any advantage after model conversion and quantization. Mish's advantage over Leaky ReLU on high-performance hardware platform seems negligible on the RK3399PRO platform.

And the RKNN framework of RK3399PRO has a special optimization for the ReLU activation function. After the single unit of YOLOV4 neural network is split in the original Darknet framework, some of the methods used during the model training will be removed first, and then if the activation function is ReLU, RKNN will perform a fusion operation on the Conv and ReLU operators, which is essentially the mathematical expression of several operators substituted and simplified, and the weights and other data of these operators will be accessed at the same time. The formula of initial Conv, BN and ReLU layers are as follows:

$$y_{conv} = \omega \cdot x + b \tag{4-14}$$

$$y_{bn} = \frac{\gamma \cdot (y_{conv} - mean)}{\sqrt{var}} + \beta \tag{4-15}$$

$$y_{ReLU} = \max(y_{bn}, 0) \tag{4-16}$$

While after the operator fusion, the formula are as follows:

$$y_{ReLU} = max(\frac{\gamma \cdot \omega_{new} \cdot x}{\sqrt{var}} + \beta + \frac{\gamma \cdot b_{new} - E[x]}{\sqrt{var}}, 0) \tag{4-17}$$

$$\omega_{new} = \frac{\gamma \cdot \omega \cdot x}{\sqrt{var}},\ b_{new} = \frac{\gamma \cdot (b - mean)}{\sqrt{var}} + \beta \tag{4-18}$$

After performing the fusion of operators, the powerful NPU can be utilized at one time, and the original inference of three operators can be processed in just one pass now. This fusion of Conv, BN and ReLU can greatly reduce the memory accessing consumption and thus improve the inference speed of the whole neural network model. Because the object detection neural network in this thesis has a large number of (Conv+BN+Activation) structures, so the activation functions used in the neural network in this thesis are all converted from mish to leaky ReLU, which sacrifices a very small amount of accuracy to significantly improve the inference performance of the neural network model.

The operator fusion of ReLU is shown in Figure 4-19.

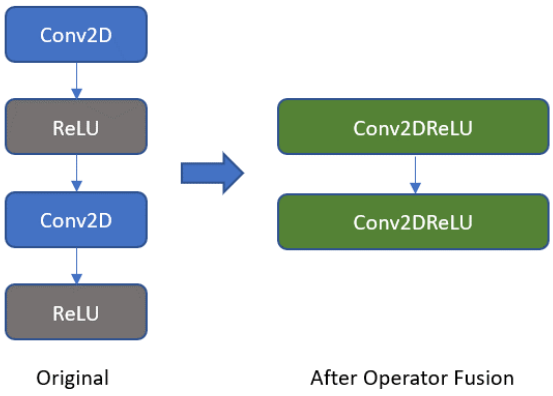


Figure 4-19 Operator Fusion of ReLU

4.7.2 Pooling

Pooling layer is a function of downsampling in neural networks, which has two main purposes. On the one hand, in order to reduce the size of feature maps, compress the amount of data and parameters, reduce information redundancy, and prevent overfitting of neural networks, and on the other hand, to extract local features and extract the most significant part of a pair of feature maps, while ensuring feature invariance, including translation, rotation and scale invariance.

Common pooling operations include max pooling and average pooling. The average pooling is to calculate the average value of the selected region of the feature map, and the average value is set to be the pooled value of the region, while the max pooling is to select the largest value of the selected region to be the pooled value of the region. When doing these pooling operations, there may be overlapping pooling process, that is, different pooling selected areas have overlapping parts. In general, average pooling is used to reduce the increase in variance of the predicted values due to the restricted size of the domain, so an average value is calculated, while max pooling is used to learn some details of the edge texture of the image to better extract the features of the image. And during backward propagation, average pooling is to divide the gradient of an element into N equal parts and assign it to the previous layer, so that the sum of the gradients (residuals) before and after pooling layer remains the same. The maximum pooling passes the gradient directly to one pixel in the previous layer, while the other

pixels do not accept the gradient, whose gradient is 0, in this way the gradient will be constant during pooling. The process of max pooling is shown in Figure 4-20.

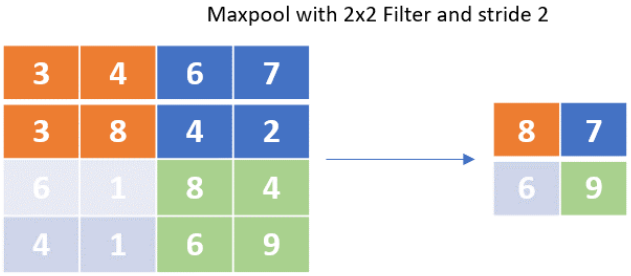


Figure 4-20 Max Pooling Process

The neural network in this thesis uses the maximum pooling method, which can better extract the edge texture features from the feature map and obtain more accurate regression results. The initial YOLOV4 neural network model also contains a SPP, which means several max pooling layers of multiple scales is operated simultaneously, and then the output are connected by a route operation. The advantage of SPP is to convert feature maps at any scale into the same dimension, which allows the neural network to process images at any scale while avoiding the cropping and warping operations, so that it is not limited by the fixed size input. And the receptive field can be significantly increased if deploy SPP in the object detection neural network, as the SPP separates the most important contextual features and hardly affects the inference performance of the neural network.

However, in the framework of RKNN, the four 1x1,5x5,9x9,13x13 max pooling layers in SPP are executed parallel, after storing the previous feature map into the dedicated memory of NPU on RK3399PRO, the operations at the four scales of this feature map are done at the same time, which greatly occupies the memory and computing power of NPU. Since 1x1 max pooling does not do any operation and the other three scales of max pooling affect the performance of inference greatly, the thesis considers only using two scales of max pooling in the SPP. Also, in order to ensure that the output is not affected by the fixed input size, the SPP structure need to add another 1x1 max pooling, that is, just transfer the original feature map of last layer. The SPP structure adjustment is shown in Figure 4-21. The inference result after this adjustment is shown in Table 4-3.

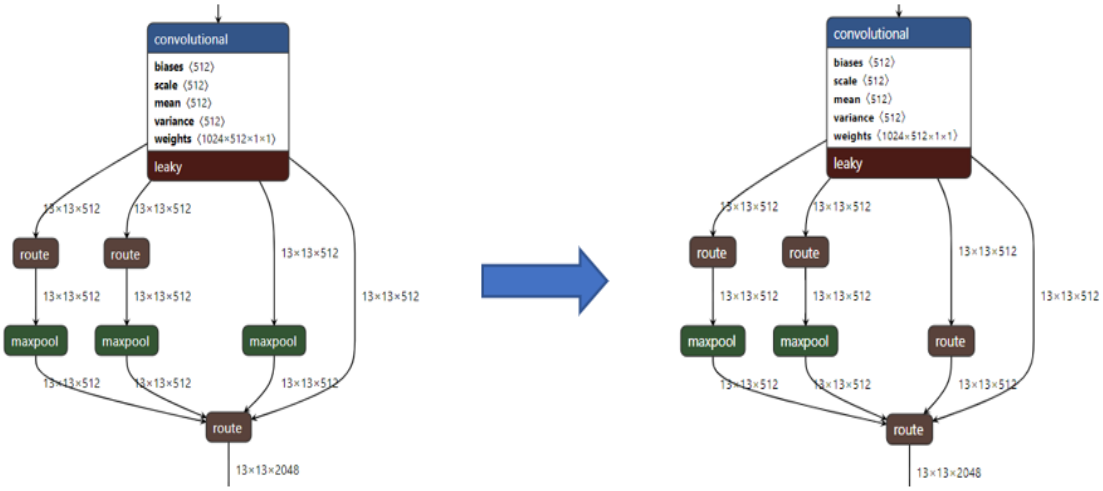


Figure 4-21 SPP Structure Adjustment

Also, in the pooling layer of neural networks on RKNN framework, RKNN has the same operator fusion optimization, which is similar to the operator fusion of ReLU and Conv introduced in the previous chapter. If the pooling layer is connected behind ReLU activation function, then the RKNN framework can also automatically fuse these operators, similarly, the four operators (Conv+BN+ReLU+Pool) are directly fused. But here in the SPP it is necessary to split these several max pooling layers and perform the operations separately, so that the three pooling operators can be fused with other operators, and then calculate these fused operators and connect them through the concat layer. Similar to the operator fusion in the previous chapter, this fusion can greatly improve the memory accessing efficiency and the inference performance of the neural network model.

By iteratively changing the model parameters and training the model for inference comparison, the inference performance is improved, and the loss of accuracy is not significant after the structure adjustment of pooling layer.

The operator fusion process of pooling layer is shown in Figure 4-22.

Table 4-3 Inference Results after SPP Adjustment on RK3399PRO

	Initial Model	After SPP Adjustment
Model Size (MB)	61.66	61.27
System Memory (MB)	293.71	258.04
NPU Memory (MB)	174.03	158.92
Inference Cost (us)	147909	135781
FPS	6.76	7.36

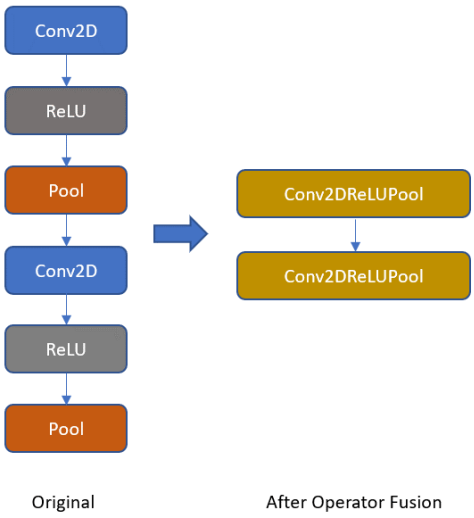


Figure 4-22 Operator Fusion of Pooling

4.7.3 Other Optimizations

The development of operators in this chapter also includes some other optimization, such as discarding some operators that only affect backward propagation in neural network training, including dropout, mosaic and others. In addition to operator replacements, the thesis also deals with operator form conversion. It is important to note

that the same operator in different frameworks needs to be converted to RKNN framework to inference. This means change the form of the operator in different frameworks but maintain the original calculation implementation. Because the RKNN framework need to maintain generality and support various neural networks of different framework. In neural network in this thesis, the operator replacements include the conversion of the shortcut layer into add layer, route layer into concat layer, unsample layer into resize layer, etc. Also, the logistic function used during network training is converted to sigmoid function at the final output layer. During the forward inference process on RKNN framework, there is no need to do a backward propagation. And the linear function is directly integrated into the Conv operator in RKNN framework, no need to add a new operator for calculation.

4.8 Summary

In this chapter, we have introduced the ideas about the neural network adjustment and forward inference optimization of models on RK3399PRO hardware platform, which includes model conversion, data layout transformation, operator development and quantization. Also, this chapter introduces the architecture of RK3399PRO, utilizes from the optimization idea of TVM and tries network pruning. During the quantization, the chapter applies two algorithms and utilizes MMSE algorithm and Moving Average Min Max algorithm to update the parameters to minimize the accuracy loss. And this chapter focuses on the effect of activation functions and pooling layers of neural networks on the hardware platform. Finally, on the RK3399PRO platform, the object detection neural network can achieve real-time video stream detection, and the accuracy does not drop to a very low level, meeting the inference performance. And the optimized neural network has a 20 times improvement in inference performance on the RK3399PRO hardware platform compared to the original YOLOV4 neural network.

5 Experiment and Results

This chapter shows the experiment results, including the construction of dataset, training and testing results of different object detection neural networks on high-performance platform and the inference results on low-power platform. These results are analyzed and shown in the figures and tables in this chapter. According to the data, it is proved that the optimized neural network in this thesis has the best inference performance with acceptable accuracy among the tested networks.

5.1 Dataset Construction

The neural network training in this thesis has used two datasets, one is the open source MS coco dataset, another is self-constructed dataset.

MS coco dataset has a total of 80 categories, including some common target types: people, dogs, bicycles, cars and so on. This is a dataset open source by Microsoft team, including five types of data annotation: object detection, key point detection, material segmentation, panoramic segmentation and image description, which are stored in json format. The chapter extracts the position and category information of each object from the annotated json format and generates the txt format for training on the darknet framework.

During the construction of own dataset, among the general obstacle types in the perspective of mobile robot, this thesis identifies five categories of obstacles: shoes, paper box, socket, headset, chair. This thesis labels the images which are downloaded from some open source websites and taken inside the laboratory. The labeling tool is Labelimg. The Labelimg is an open source image labeling software, which can be used to label the ground truth and category of each object in a graphic user interface, and the software will automatically generate labeled files in voc format or yolo format. In each line of the yolo format, the first number represents the category number, and the following numbers are (x, y, w, h) of each bounding box. Every image in the dataset should be fully labeled with each object.

The labeling process is shown in Figure 5-1.

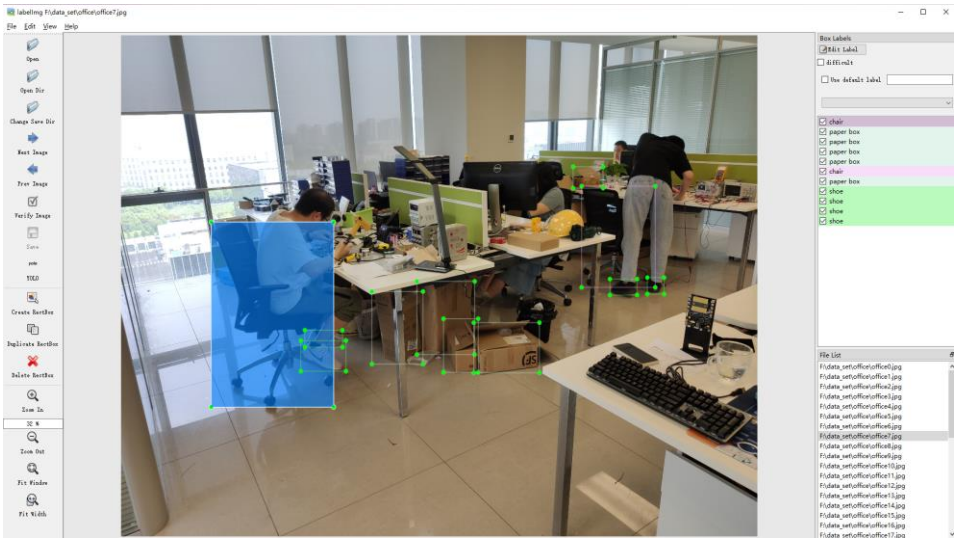


Figure 5-1 Dataset Labeling Process

Because the number of images in own dataset is very small, there are only 1108 images. The thesis uses some dataset enhancement works to improve the richness of the dataset, including some random scaling, cropping, and arrangement of the images to stitch. These enhancements are applied to each image. With these works, the generalization ability of the neural network is improved during training. The process of data enhancement is shown in Figure 5-2.

Figure 5-2 Dataset Enhancement Process

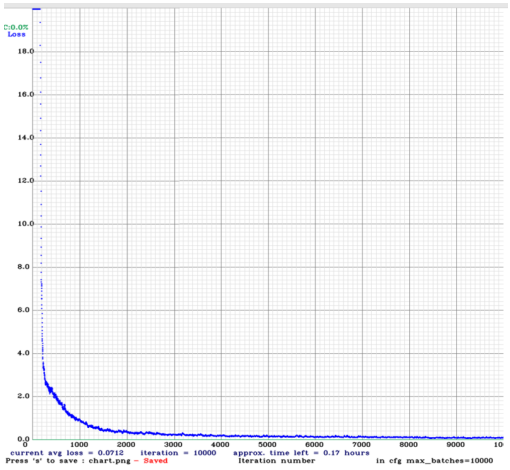
5.2 Model Training and Deployment

All the deep learning object detection neural networks need continuous training and optimization before deploying to the hardware platform, the optimization at each stage has been introduced in the previous chapter, this chapter is mainly about displaying some test results, including the comparison of different networks on high-performance server and RK3399PRO platform. Only a few representative neural network models are shown here, including the original YOLOV3, YOLOV4, and YOLOV4-Tiny neural network models. This chapter also tests the YOLOV4-CSP network model, which changes the PANET of YOLOV4 to CSPNet. Finally, this chapter tests own neural network which is optimized in chapter 4. This neural network is noted as YOLO-Mine.

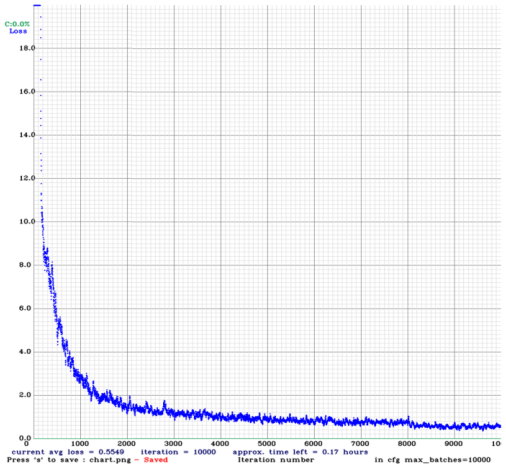
5.2.1 Training

The training of the neural networks in this thesis are implemented on the windows platform, based on the Darknet deep learning framework. The initial anchors are calculated according to the ground truth of the dataset and these anchors are used to train the neural network. The average loss value and the mean average precision value are used to observe the real-time accuracy change of during the model training. The training accuracy of the neural network is modified by setting different learning rates and threshold parameters. Not all models are the trained weights with maximum batches, because there is overfitting and accuracy loss if training more batches. The selected neural network models have the highest accuracy, and some models are trained for less batches. The comparison results of these models in high-performance server are shown in Table 5-1, all the data are tested when the size of input image is 416x416.

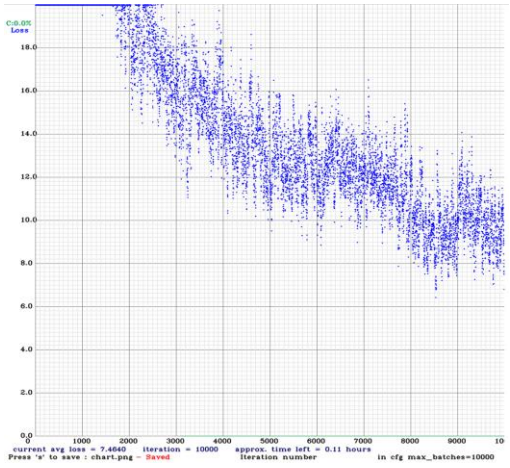
The training process of these neural network models are shown in Figure 5-3. According to the data, the accuracies of all the models are relatively high except the YOLOV4-CSP model. There reason is that the Residual units in CSPNet make the neck network of YOLOV4 more complex, so this neural network need richer dataset. But this low accuracy model can still be used to test the inference performance on RK3399PRO platform, for that accuracy does not affect performance.



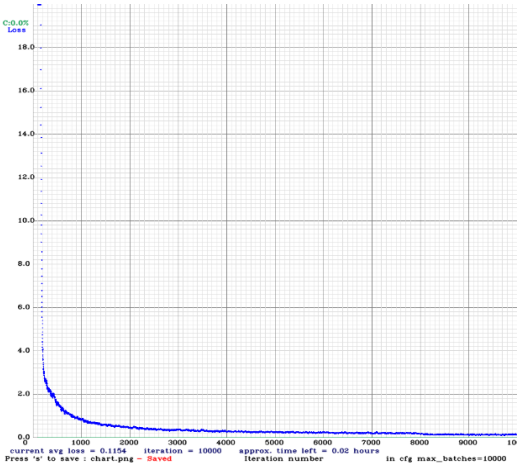
(a)



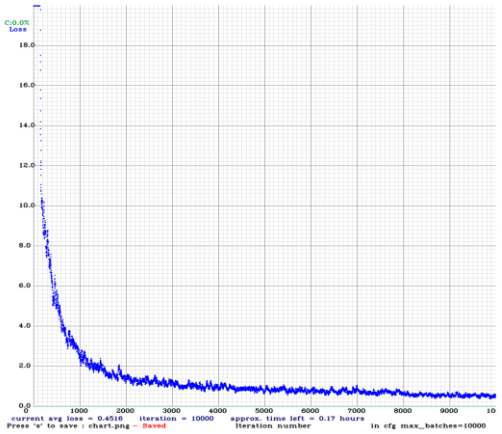
(b)



(c)



(d)



(e)

Figure 5-3 Neural Network Training Process (a) YOLOV3 (b) YOLOV4 (c) YOLOV4-CSP (d) YOLOV4-Tiny (5) YOLO-Mine

Table 5-1 Inference Results of Models on Server

Model	V3	V4	CSP	Tiny	Mine
Size (MB)	234	245	202	22.4	244
AP50:95	30.9	40.3	39.6	20.6	38.5
Memory (MB)	18.88	18.88	18.88	72.42	18.88
BFLOPS	65.879	60.137	50.834	6.910	61.134
Time Cost (ms)	31.07	34.012	33.655	7.663	34.48
FPS	32.4	31.8	20.8	43.3	32.2

According to the data, except YOLOV4-Tiny network, other networks have similar inference performance on high-performance server. While the accuracy of YOLOV3 is much lower than other networks.

5.2.2 Deployment

After the object detection neural networks are trained with a good accuracy on the server, the models are then deployed to the RK3399PRO hardware platform. The operation system of the platform is the Debian 10 system based on ARM. The neural network models on the darknet framework are converted to the RKNN framework. Then the thesis uses images and video stream to test the inference performance and accuracy of these object detection neural networks. The results of initial models are shown in Table 5-2 and the results of quantized models are shown in Table 5-3. Similarly, these data are tested when the input image size is 416x416.

According to the data, the final optimized YOLO-MINE network model has the best inference performance with acceptable accuracy on RK3399PRO hardware platform, which has higher accuracy than YOLOV3 and YOLOV4-Tiny and faster than YOLOV4 and YOLOV4-CSP.

Table 5-2 Inference Results of Initial Models on RK3399PRO

Model	V3	V4	CSP	Tiny	Mine
Model Size (MB)	118.29	122.95	101.13	11.59	122.88
System Memory (MB)	407.35	530.40	478.64	48.02	483.37
NPU Memory (MB)	295.12	327.54	276.48	38.89	159.60
Inference Cost (us)	2165990	2622625	2323044	453759	2167076
FPS	0.46	0.38	0.43	2.3	0.46

Table 5-3 Inference Results of Quantized Models on RK3399PRO

Model	V3	V4	CSP	Tiny	Mine
Model Size (MB)	59.28	61.66	50.7	5.83	61.28
System Memory (MB)	199.88	307.08	275.54	31.11	259.06
NPU Memory (MB)	143.42	160.03	134.08	19.01	147.58
Inference Cost (us)	75803	542784	555320	20793	89832
FPS	13.19	1.84	1.80	48.1	11.13

Finally, the optimized object detection neural network is applied in the intelligent vision system in a low-cost disinfection robot. Figure 5-4 shows the whole framework of this disinfection robot and the functions of different modules. While this robot achieves the basic function, it can accurately detect the obstacles ahead through the monocular camera, and the accuracy can reach more than 90%, at the same time, the robot can automatically map and navigate its surroundings through the 2D LiDAR, with that it can disinfect the surrounding environment without repetition. This intelligent vision system can process real-time video streams at up to 15fps. With this system, the disinfection robot becomes more effective and intelligent.

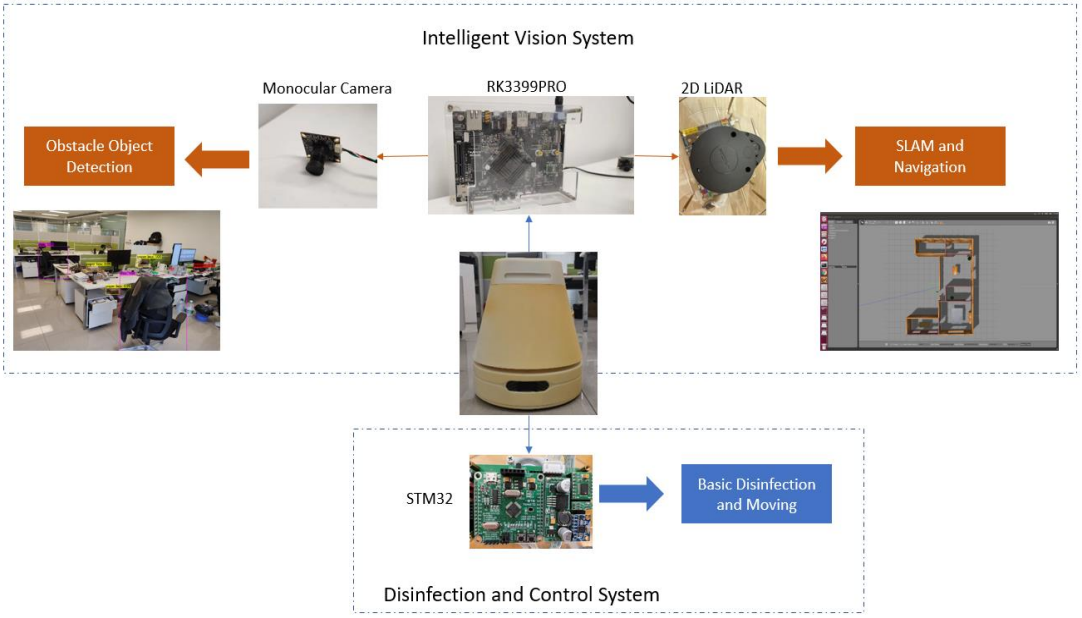


Figure 5-4 Disinfection Robot with Intelligent Vision System

5.3 Summary

This chapter is mainly about data display, including dataset construction, training of different neural networks and the inference results of these networks on server and RK3399PRO platform, including the initial models and quantized models. According to the comparison results, the optimized neural network in this thesis has the best inference performance with acceptable accuracy among all the tested neural networks, based on this neural network, the obstacle detection of real-time video streaming can be implemented on the mobile robot platform. This chapter finally shows the whole architecture of the disinfection robot, with this intelligent vision system, the robot will

become more effective and intelligent. And in the future, this vision system can be applied to other type of low-cost mobile robots.

6 Conclusion and Future Research Directions

6.1 Conclusion

With the development of the information technology, mobile robots embody both of 5G and AI, and as such play an important role in future developments. The core technology of robot includes perception, decision-making and execution. Perception is one of the most important aspects bringing understanding of the environment to intelligent systems. However, most researchers and engineers are not developing vision system for consumer-grade mobile robots. In this kind of edge computing scenario, all data processing needs to be deployed on low-power hardware platform. In view of this, this thesis designs an intelligent vision system based on a mobile robot platform. This intelligent vision system mainly includes two vision modules, one is the SLAM localization and navigation function, and the other one is the deep learning based obstacle object detection function.

The main work and contributions of this thesis is divided into three parts:

(1) In this thesis we design an architecture of the intelligent vision system, including the sensor construction, the allocation of hardware resources, the selection of perception algorithms and the communication between different modules.

(2) Through the thesis, we compare different deep learning based object detection neural networks and analyzes the effect of different tricks on the overall model. Based on the constructed dataset, the different neural network models are trained and tested on a high-performance hardware platform, and they are also deployed to low-cost hardware platform for forward inference test. According to their performance and accuracy, the state-of-the-art object detection neural network YOLOV4, which is not adopted by most developers, is innovatively selected as the main idea of the obstacle detection module in the intelligent vision system.

(3) The thesis implements a series of optimizations for the YOLOV4 object detection neural network based on the low-power RK3399PRO hardware platform, including conversion of the memory layout, optimization of the activation function and pooling layer and splitting, replacement, and fusion of different operators. In this thesis, we also show how quantization can be used to compress a neural network. The MMSE and

Moving Average Min Max algorithms are innovatively utilized in the process to update the parameters to reduce the accuracy loss of the quantized model and stochastic method is applied to do the round operation. Then the thesis tries model pruning and mixed accuracy quantization. Finally, the optimized network has a 20 times improvement in inference performance on the RK3399PRO hardware platform compared to the original YOLOV4 network and the accuracy only has a little loss.
The finally designed intelligent vision system on the mobile robot can perform real-time obstacle detection on 1080p60fps video streaming. The detection accuracy reaches more than 90%, while the processing rate can reach 15fps. At the same time the mobile robot can complete automatic localization and navigation function without cloud connection. With this system, the mobile robot can be lower-cost, more power-efficient and the intelligent is more robust.

6.2 Future Research Directions

This thesis designs an intelligent vision system with two modules, a deep learning-based obstacle detection module and a SLAM localization and navigation module. The object detection is based on an open-source algorithm and the neural network is optimized and deployed on a low-power hardware platform. However, there are still some limitations and shortcomings of the whole vision system, which need further work in the future:
(1) The vision system can apply 3DToF sensors to improve the accuracy of SLAM and depth cameras to obtain distance information for the object detection neural network, which can optimize the data post-processing fusion.
(2) Non-linear quantization algorithm can be used in quantization and the data can be quantized to 4bit when the accuracy loss is within a certain acceptable range. In that way the network can be compressed even further in order to consume less memory and computing power.
(3) A more efficient AI accelerator chip can be used. Memory access is an important factor when performing inference on the edge platform. It is possible to use Single Instruction Multiple Data (SIMD) based hardware to achieve more efficient convolutional data multiplexing. And it will be much faster if the memory is extremely close to the computation block on the hardware. Another idea is to use computation and storage integration technology, using new non-volatile storage devices such as ReRAM,

so the operators of neural networks can be built into the storage array, which can significantly improve the inference and power performance[62].

Reference

[1] K. Xue, J. Li, N. Xiao, J. Liu, X. Ji and H. Qian, "Improving The Robot Localization Accuracy Using Range-only Data: An Optimization Approach," 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM), 2021, pp. 785-790, doi: 10.1109/ICARM52023.2021.9536151.

[2] C. Breazeal, "Social Robots: From Research to Commercialization," 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI, 2017, pp. 1-1.

[3] X. Zhang, Y. Fang, B. Li and J. Wang, "Visual Servoing of Nonholonomic Mobile Robots With Uncalibrated Camera-to-Robot Parameters," in IEEE Transactions on Industrial Electronics, vol. 64, no. 1, pp. 390-400, Jan. 2017, doi: 10.1109/TIE.2016.2598526.

[4] J. Liao, Z. Chen and B. Yao, "Performance-Oriented Coordinated Adaptive Robust Control for Four-Wheel Independently Driven Skid Steer Mobile Robot," in IEEE Access, vol. 5, pp. 19048-19057, 2017, doi: 10.1109/ACCESS.2017.2754647.

[5] A. Munawar et al., "MaestROB: A Robotics Framework for Integrated Orchestration of Low-Level Control and High-Level Reasoning," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 527-534, doi: 10.1109/ICRA.2018.8462870.

[6] X. Liu et al., "Accurate Localization of Tagged Objects Using Mobile RFID-Augmented Robots," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1273-1284, 1 April 2021, doi: 10.1109/TMC.2019.2962129.

[7] M. Sorour, A. Cherubini, P. Fraisse and R. Passama, "Motion Discontinuity-Robust Controller for Steerable Mobile Robots," in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 452-459, April 2017, doi: 10.1109/LRA.2016.2638466.

[8] F. G. Lopez, J. Abbenseth, C. Henkel and S. Dörr, "A predictive online path planning and optimization approach for cooperative mobile service robot navigation in industrial applications," 2017 European Conference on Mobile Robots (ECMR), 2017, pp. 1-6, doi: 10.1109/ECMR.2017.8098677.

[9] B. Fang et al., "A Novel Humanoid Soft Hand with Variable Stiffness and Multi-modal Perception *," 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM), 2021, pp. 99-105, doi: 10.1109/ICARM52023.2021.9536073.

[10] A. Sánchez-Orta et al., "Aerial Following of a Non-holonomic Mobile Robot subject to Velocity Fields: A Case Study for Autonomous Vehicles Surveillance," 2020 International

Conference on Unmanned Aircraft Systems (ICUAS), 2020, pp. 1093-1102, doi: 10.1109/ICUAS48674.2020.9214053.

[11] W. Tabib, K. Goel, J. Yao, C. Boirum and N. Michael, "Autonomous Cave Surveying With an Aerial Robot," in IEEE Transactions on Robotics, doi: 10.1109/TRO.2021.3104459.

[12] Z. Li, J. Li, S. Zhao, Y. Yuan, Y. Kang and C. L. P. Chen, "Adaptive Neural Control of a Kinematically Redundant Exoskeleton Robot Using Brain–Machine Interfaces," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 12, pp. 3558-3571, Dec. 2019, doi: 10.1109/TNNLS.2018.2872595.

[13] G. Ficht et al., "NimbRo-OP2X: Adult-Sized Open-Source 3D Printed Humanoid Robot," 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), 2018, pp. 1-9, doi: 10.1109/HUMANOIDS.2018.8625038.

[14] M. Langen and S. Heinrich, "Humanoid Robots: Use Cases as AI-Lab Companion : Can an empathic and collaborative digital companion motivate innovation?," 2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), 2019, pp. 1-6, doi: 10.1109/ICE.2019.8792614.

[15] K. Kim, S. Jeong, W. Kim, Y. Jeon, K. Kim and J. Hong, "Design of small mobile robot remotely controlled by an android operating system via bluetooth and NFC communication," 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2017, pp. 913-915, doi: 10.1109/URAI.2017.7992864.

[16] L. Pang, Z. Cao, J. Yu, P. Guan, X. Chen and W. Zhang, "A Robust Visual Person-Following Approach for Mobile Robots in Disturbing Environments," in IEEE Systems Journal, vol. 14, no. 2, pp. 2965-2968, June 2020, doi: 10.1109/JSYST.2019.2942953.

[17] J. Chen and W. Kim, "A Human-Following Mobile Robot Providing Natural and Universal Interfaces for Control With Wireless Electronic Devices," in IEEE/ASME Transactions on Mechatronics, vol. 24, no. 5, pp. 2377-2385, Oct. 2019, doi: 10.1109/TMECH.2019.2936395.

[18] W. Yuan, Z. Li and C. -Y. Su, "Multisensor-Based Navigation and Control of a Mobile Service Robot," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 4, pp. 2624-2634, April 2021, doi: 10.1109/TSMC.2019.2916932.

[19] Yang, Guang-Zhong, et al. "Combating COVID-19—The role of robotics in managing public health and infectious diseases." (2020): eabb5589.

[20] D. Conte, S. Leamy and T. Furukawa, "Design and Map-based Teleoperation of a Robot for Disinfection of COVID-19 in Complex Indoor Environments," 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2020, pp. 276-282, doi: 10.1109/SSRR50563.2020.9292625.

[21] M. Cardona, F. Cortez, A. Palacios and K. Cerros, "Mobile Robots Application Against Covid-19 Pandemic," 2020 IEEE ANDESCON, 2020, pp. 1-5, doi: 10.1109/ANDESCON50619.2020.9272072.

[22] Ackerman, Evan. "Autonomous robots are helping kill coronavirus in hospitals." IEEE Spectrum 11 (2020).

[23] Guettari, M., Gharbi, I. & Hamza, S. UVC disinfection robot. Environ Sci Pollut Res 28, 40394–40399 (2021). https://doi.org/10.1007/s11356-020-11184-2

[24] D. L. Marino et al., "AI Augmentation for Trustworthy AI: Augmented Robot Teleoperation," 2020 13th International Conference on Human System Interaction (HSI), 2020, pp. 155-161, doi: 10.1109/HSI49210.2020.9142659.

[25] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[26] S. Bhattacharya, T. K. Maiti, S. Dutta, A. Luo, M. Miura-Mattausch and H. J. Mattausch, "System Simulation for Robot Control Based on AI Approach," 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2018, pp. 1-4, doi: 10.1109/ICSICT.2018.8565654.

[27] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[28] Z. Dwiel, M. Candadai and M. Phielipp, "On Training Flexible Robots using Deep Reinforcement Learning," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 4666-4671, doi: 10.1109/IROS40897.2019.8968516.

[29] H. Jiang, H. Wang, W. -Y. Yau and K. -W. Wan, "A Brief Survey: Deep Reinforcement Learning in Mobile Robot Navigation," 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2020, pp. 592-597, doi: 10.1109/ICIEA48937.2020.9248288.

[30] T. Hiejima, S. Kawashima, M. Ke and T. Kawahara, "Effectiveness of Synchronization and Cooperative Behavior of Multiple Robots based on Swarm AI," 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2019, pp. 341-344, doi: 10.1109/APCCAS47518.2019.8953108.

[31] P. Kirsanov et al., "DISCOMAN: Dataset of Indoor SCenes for Odometry, Mapping And Navigation," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 2470-2477, doi: 10.1109/IROS40897.2019.8967921.

[32] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," in IEEE Communications

Surveys & Tutorials, vol. 22, no. 2, pp. 869-904, Secondquarter 2020, doi: 10.1109/COMST.2020.2970550.

[33] M. Goudarzi, H. Wu, M. Palaniswami and R. Buyya, "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1298-1311, 1 April 2021, doi: 10.1109/TMC.2020.2967041.

[34] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," in IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657-1681, thirdquarter 2017, doi: 10.1109/COMST.2017.2705720.

[35] S. Wang et al., "A Cloud-Guided Feature Extraction Approach for Image Retrieval in Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 20, no. 2, pp. 292-305, 1 Feb. 2021, doi: 10.1109/TMC.2019.2944371.

[36] Y. Liu, M. Peng, G. Shou, Y. Chen and S. Chen, "Toward Edge Intelligence: Multiaccess Edge Computing for 5G and Internet of Things," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6722-6747, Aug. 2020, doi: 10.1109/JIOT.2020.3004500.

[37] X. Liu, Q. Yang, J. Luo, B. Ding and S. Zhang, "An Energy-Aware Offloading Framework for Edge-Augmented Mobile RFID Systems," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 3994-4004, June 2019, doi: 10.1109/JIOT.2018.2881295.

[38] M. De Donno, K. Tange and N. Dragoni, "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog," in IEEE Access, vol. 7, pp. 150936-150948, 2019, doi: 10.1109/ACCESS.2019.2947652.

[39] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1738-1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.

[40] H. Khelifi et al., "Bringing Deep Learning at the Edge of Information-Centric Internet of Things," in IEEE Communications Letters, vol. 23, no. 1, pp. 52-55, Jan. 2019, doi: 10.1109/LCOMM.2018.2875978.

[41] S. N. Shirazi, A. Gouglidis, A. Farshad and D. Hutchison, "The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog From a Security and Resilience Perspective," in IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2586-2595, Nov. 2017, doi: 10.1109/JSAC.2017.2760478.

[42] A. Karimian and R. Tron, "Bearing-Only Navigation With Field of View Constraints," in IEEE Control Systems Letters, vol. 6, pp. 49-54, 2022, doi: 10.1109/LCSYS.2020.3048802.

[43] R. Benyoucef et al, "Towards Kinematics From Motion: Unknown Input Observer and Dynamic Extension Approach," in IEEE Control Systems Letters, vol. 6, pp. 1340-1345, 2022, doi: 10.1109/LCSYS.2021.3093067.

[44] P. K. Mohanty, A. K. Sah, V. Kumar and S. Kundu, "Application of Deep Q-Learning for Wheel Mobile Robot Navigation," 2017 3rd International Conference on Computational Intelligence and Networks (CINE), 2017, pp. 88-93, doi: 10.1109/CINE.2017.11.

[45] Y. Zhang, X. Sun, J. Gao and Q. Liang, "Visual Trajectory Tracking Control of Non-Holonomic Mobile Robots: A Cascaded Approach," 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM), 2021, pp. 817-822, doi: 10.1109/ICARM52023.2021.9536070.

[46] F. A. X. Da Mota, M. X. Rocha, J. J. P. C. Rodrigues, V. H. C. De Albuquerque and A. R. De Alexandria, "Localization and Navigation for Autonomous Mobile Robots Using Petri Nets in Indoor Environments," in IEEE Access, vol. 6, pp. 31665-31676, 2018, doi: 10.1109/ACCESS.2018.2846554.

[47] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," in Tsinghua Science and Technology, vol. 26, no. 5, pp. 674-691, Oct. 2021, doi: 10.26599/TST.2021.9010012.

[48] H. Guo, S. Wang, J. Fan and S. Li, "Learning Automata Based Incremental Learning Method for Deep Neural Networks," in IEEE Access, vol. 7, pp. 41164-41171, 2019, doi: 10.1109/ACCESS.2019.2907645.

[49] D. Wu, C. Wang, Y. Wu, Q. -C. Wang and D. -S. Huang, "Attention Deep Model With Multi-Scale Deep Supervision for Person Re-Identification," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 5, no. 1, pp. 70-78, Feb. 2021, doi: 10.1109/TETCI.2020.3034606.

[50] G. Lv, K. Wen, Z. Wu, X. Jin, H. An and J. He, "Nuclei R-CNN: Improve Mask R-CNN for Nuclei Segmentation," 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), 2019, pp. 357-362, doi: 10.1109/ICICSP48821.2019.8958541.

[51] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[52] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[53] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[54] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

[55] Wang, Chien-Yao, et al. "CSPNet: A new backbone that can enhance learning capability of CNN." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020.

[56] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "Scaled-yolov4: Scaling cross stage partial network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[57] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.

[58] Zheng, Zhaohui, et al. "Distance-IoU loss: Faster and better learning for bounding box regression." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 07. 2020.

[59] D. Diamantopoulos, B. Ringlein, M. Purandare, G. Singh and C. Hagleitner, "Agile Autotuning of a Transprecision Tensor Accelerator Overlay for TVM Compiler Stack," 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), 2020, pp. 310-316, doi: 10.1109/FPL50879.2020.00058.

[60] Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).

[61] Hu, Hengyuan, et al. "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures." arXiv preprint arXiv:1607.03250 (2016).

[62] H. Yan, H. R. Cherian, E. C. Ahn, X. Qian and L. Duan, "iCELIA: A Full-Stack Framework for STT-MRAM-Based Deep Learning Acceleration," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 2, pp. 408-422, 1 Feb. 2020, doi: 10.1109/TPDS.2019.2937517.