Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Pedestrian target detection based on infrared images

by
Heng Liu

Supervisor: Terry Walcott

Project submitted as part of the requirements for the award of MSc Data and Artificial Intelligence

September 2022

DECLARATION

I, Heng Liu, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Heng Liu

Date: 06/09/2022

FORM OF CONSENT

I, Heng Liu, hereby consent that my Project, submitted in candidature for the MSc Data and Artificial Intelligence, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Heng Liu

Date: 06/09/2022

Abstract

As the most popular object detection algorithm based on deep learning, YOLOv5 algorithm has the advantages of high detection accuracy and fast detection speed. However, the current target detection algorithm based on deep learning is mainly aimed at targets in visible images. In order to study the problem of low pedestrian target recognition rate after applying YOLOv5 algorithm to infrared pedestrian target detection, this project studies a large number of related literatures and reviews them. Moreover, the attention module is added to YOLOv5 to make it have a higher recognition rate for pedestrian targets in infrared scenes.

The main work of this project is as follows :(1) since the existing infrared pedestrian target data sets are few and most of them do not conform to the current road conditions in China, this project collects infrared videos through equipment, manually labels them, and produces a data set. (2) Three lightweight attention modules, SE, CBAM and CA, are introduced into the backbone network of the YOLOv5s model, which alleviates the shortcomings of the model such as difficulty in discriminating between occluded pedestrian targets and missing detection of small targets, and the accuracy of pedestrian detection in infrared images is improved by 0.8% at most to 95.8%. (3) The visual interface is developed based on PyQt5, and the test FPS is 40, which achieves the purpose of real-time.

Table of Contents

Abstract 3
Chapter 1 - Introduction 7
Chapter 2 - Literature Review 10
2.1 Traditional object detection method 10
2.2 Deep learning object detection method 12
2.3 Convolutional neural network theory 17
2.4 ResNet structure 21
2.5 Common Frameworks 23
2.6 YOLOv5 algorithm principle 24
2.7 Attention Mechanism 33
2.8 User interface design library 35
2.9 Literature review conclusions 36
Chapter 3 - Research methodology 37
3.1 Philosophy 37
3.2 Approach 38
3.3 Strategies 38
3.4 Risks 39
Chapter 4 - Data collection and processing 40
4.1 Data acquisition equipment and process 40
4.2 Rename the dataset images 41
4.3 Annotate dataset images 42
4.4 Pedestrian analysis in a dataset 44
Chapter 5 – Design & Implementation 45
5.1 Resources and planning requirements 45
5.2 Design 48
5.2.1 Algorithm design 48
5.2.2 Detection Interface Design 50
5.3 Implementation 51
Chapter 6 – Results and evaluation 53
6.1 Evaluation indicators 53
6.2 Results 54
6.3 Infrared image pedestrian detection system test evaluation 58
Chapter 7 – Conclusions and Recommendations 59
7.1 Conclusions 59
7.2 Recommendations for future research 59
Chapter 8 – Reflections 61
References 62
Appendices 67

List of Figures

Figure 1 – Far infrared image 8
Figure 2 - Max pooling and averaging pooling operations 19
Figure 3 - Sigmoid function 20
Figure 4 - ReLU function and Leaky ReLU function 21
Figure 5 - Comparison between standard neural net and neural net after applying dropout 21
Figure 6 - The basic structure of the residual block 22
Figure 7 - Structure diagram of YOLOv5 25
Figure 8 - Enhanced comparison between CutMix and Mosaic data 26
Figure 9 - Focus slicing operation 28
Figure 10 - Schematic diagram of Focus structure 29
Figure 11 - PN+PAN structure 30
Figure 12 - SENet schematic diagram 33
Figure 13 - CBAM structure 34
Figure 14 - Basic interface of QtDesigner 36
Figure 15 - Far infrared camera installation position 41
Figure 16 - renamed images 41
Figure 17 - Infrared images annotation 42
Figure 18 - Xml annotation content 43
Figure 19 - TXT annotation content 43
Figure 20 - Analysis of the pedestrian situation in the dataset 44
Figure 21 - Image detection results in the YOLOv5 environment 47
Figure 22 - SE code 49
Figure 23 - The network model after adding the SE module 49
Figure 24 - CBAM partial code 50
Figure 25 - The network model after adding the CBAM module 50
Figure 26 - Infrared image pedestrian detection system interface 51
Figure 27 - Initializing the hyperparameters 52
Figure 28 - PR curve graph 55
Figure 29 - AP50 experimental comparison result 56
Figure 30 - The pedestrian detection effect verification diagram of the model before the introduction of the SE module 57
Figure 31 - The pedestrian detection effect verification diagram of the model after introducing the SE module 57
Figure 32 - Schematic running 58

List of Tables

Table 1 - Features comparison of deep learning frameworks 24
Table 2 - Far infrared camera detailed parameters..... 40
Table 3 - System software configuration 45
Table 4 - System software configuration 46
Table 5 - Training hyperparameters 51
Table 6 - confusion matrix 53
Table 7 - Comparison of the results of adding different attention mechanisms..... 55

Chapter 1 - Introduction

Infrared imaging technology has been incorporated into more and more facets of our lives in recent years. The amount of data for infrared photos and videos has also greatly increased as more and more infrared equipment is being used. How to wring valuable data out of infrared photos and videos is one of the current research hotspots. Infrared imaging technology plays a special role in the detection of pedestrians in the field of security monitoring and unmanned operations. Unlike visible light imaging, which has very strict requirements on lighting conditions, infrared imaging technology can compensate for insufficient light and can operate continuously. As a result, an increasing number of researchers start to investigate the infrared target identification algorithm based on deep learning and apply it practically, which is very important to the advancement of security monitoring and autonomous driving.

A method known as pedestrian detection [1] precisely locates pedestrian targets in video or picture sequences. Rectangular boxes are typically utilised to locate pedestrians for pedestrian detection activities. It has significant social and economic significance and is widely used in the domains of intelligent security, autonomous driving, and intelligent transportation. Particularly, the use of deep learning in computer vision has substantially sped the commercialization of pedestrian detection technology and greatly aided its development. The majority of mainstream pedestrian identification algorithms today, however, concentrate mostly on Visible images. Visible images produce good imaging results when there is adequate lighting, but there are imaging problems when there is insufficient lighting or when it is dark. As a result, the infrared picture [2] of a pedestrian enters people's fields of vision. Infrared image pedestrian detection provides the following benefits over visible image pedestrian detection: (1) the imaging effect is still outstanding even in dim or fully dark conditions, and the image can be crisper without relying on ambient light. (2) Since infrared imaging is unaffected by the colour of the clothing worn by pedestrians, colour space interference due to background colours that are comparable

to the pedestrians' clothing can be avoided. Based on these benefits, infrared pedestrian detection has recently shown promising results in the areas of intelligent video surveillance, automobile assistance, and military early warning. Infrared imaging, in contrast to visible light imaging, has the drawbacks of single features and low resolution, which makes it more challenging to detect pedestrians. Designing a reliable and effective infrared pedestrian detection system is crucial for addressing the current major issues.

According to various imaging techniques, infrared imaging can be separated into active infrared imaging and passive infrared imaging. In order to acquire grayscale images, active infrared imaging uses an infrared light-emitting diode to actively emit infrared rays. The reflected infrared light is then collected by the image sensor of the camera. The basic idea behind passive infrared imaging is known as blackbody radiation, which states that any object above absolute zero can emit electromagnetic waves. The detector can pick up electromagnetic waves of various intensities, which it then transforms into infrared images that can be seen by human eyes. The military, automotive night vision, and industrial flaw detection industries all make extensive use of passive infrared imaging. Far-infrared images, also known as passive infrared images, are depicted in Figure 1 as imaging images. The two infrared imaging techniques each have their own benefits and drawbacks from the standpoint of imaging effect, but both have a much worse imaging effect than visible light imaging with adequate lighting. Therefore, it is still quite difficult to figure out how to effectively increase the performance of pedestrian recognition on infrared photos.

Figure 1 – Far infrared image

Infrared picture pedestrian detection is crucial to robotics, autonomous driving, and intelligent security. For instance, in the area of intelligent security, intelligent monitoring was done in areas with a high concentration of pedestrians at night in order to prevent incidents. When compared to the conventional video monitoring method, the intelligent video monitoring method greatly reduces the need for human resources because it effectively prevents crime through the use of automatic pedestrian detection and gesture recognition techniques. In an autonomous area, gathering information about pedestrians and traffic is crucial. Using infrared target detection technology for pedestrians is a good way to do this whether it's day or night, test pedestrians on the road, and ensure their safety in order to make driverless cars more intelligent and safe in traffic.

Chapter 2 - Literature Review

2.1 Traditional object detection method

Recent years have seen an increase in the popularity of image processing as a research topic, especially when it comes to object detection utilising computer vision. Visible photos make up the majority of the photographs used in their research. Infrared cameras are becoming more and more common, and some academics are beginning to focus on infrared picture target recognition as a result. The two primary categories of currently in use pedestrian detection systems are classical object detection and deep learning-based object detection. Deep learning has made considerable strides in resolving many existing object detection issues as artificial intelligence has advanced. It has assumed a dominant position and evolved into the standard approach for object detection due to its high detection accuracy and rapid detection time.

There are three steps in the traditional path of pedestrian detection: the first step is to determine and extract the candidate region from the scene and construct the detection model; The second step is to obtain the feature information from the candidate region and analyze the information statistically. The last step is to classify the information and refine the position of the valid candidate box [3].

The most common way to extract candidate regions is the window sliding method, which uses the preset classifier to detect whether the object exists in the window, determines the size parameter of the detection window according to the image input, and slides the window according to the step parameter until the detection of the whole image is completed. Before the research and application of deep learning, the feature extraction of target region is mainly done by manual extraction. For example, Histogram of Oriented Gradient (HOG) [4] feature is based on the Gradient in the direction of pixel points as the extraction standard, involving the Gradient direction, Gradient magnitude and other elements. The local gradient histogram of the image is

calculated, and the features are extracted. Then the results of these steps are summarized to construct the HOG feature of the whole image. David et al [5] jointly studied the scale-invariant feature transform algorithm (SIFT), which describes the local features of the image and obtains the total local feature descriptor of the whole image [6]. Later, the algorithm was improved and modified in 2004. Overall evaluation, the process of extracting key points and embedding local features in SIFT algorithm can be called SIFT feature generation.

After feature extraction, the classifiers commonly used in traditional pedestrian detection methods include adaptive enhanced Adaboost and Support Vector Machines (SVM) [7]. Adaboost is an optimization strategy, which strengthens the classifier by stacking. Its adaptability lies in that it will increase the weight of the misjudged samples after each round of training and continue to train the next initial optimizer after the weight iteration. Among them, many scholars have applied it to infrared image pedestrian detection. Olmeda [8] proposed a pedestrian detection system based on discrete features in infrared images. Search for unique key points in the image and find the points in the histogram with the same phase direction and then extract descriptors around them. These descriptors are used to match the specified area of the pedestrian's body. If matched, a region of interest is created in the image, which is classified as pedestrian or non-pedestrian by SVM.

Besbes [9] proposed a pedestrian detector with an on-board far-infrared camera. The system consists of three modules, each of which is based on SURF feature matching algorithm. The second module is used to generate the region of interest. In the far infrared image, the shape of the pedestrian image may change significantly, but the head usually appears as a bright area. The region of interest can be detected with high recall by layered SURF features located in the header region. The second module uses support vector machine to classify pedestrians' whole body, which can improve the accuracy with low computational cost. In the third module, the mean shift algorithm is skillfully combined with the scale-invariant SURF feature tracking method to enhance the robustness of the system.

Zhao [10] proposed an effective method for pedestrian detection and segmentation in far-infrared images using Shape Distribution Histogram(SDH) and Modified Sparse Representation Classification(MSRC). The algorithm first identifies candidate areas that may contain pedestrians based on contour saliency map. Then, the distance between random points on the pedestrian contour map in the candidate area is applied to obtain SDH features. SDH is a powerful and differentiated feature that accurately characterizes pedestrians. Finally, a robust and high-precision MSRC classifier is used to identify real pedestrians.

Above all: although foreign scholars from different Angle design of the pedestrian detection model, get a lot of strong functions, wide coverage, using a good algorithm, but these are not for deep in the traditional detection, feature extraction, the detection information only relies on the image information of shallow, and depends on the characteristic information of artificial mark is higher. Although the traditional pedestrian detection method can achieve good detection accuracy in some specific scenes, the detection speed is still difficult to meet the realistic detection requirements, and its detection accuracy is also difficult to be applied in complex scenes.

2.2 Deep learning object detection method

Deep learning is a branch of artificial intelligence, which enables computers to learn features and perform tasks. It is an indispensable research tool in computer vision, natural speech processing, image recognition and detection and other fields. In the last decade, deep learning has been widely used in our daily lives due to the significant increase in the availability of large amounts of data and the feasibility of computing resources. Target detection methods based on deep learning are mainly divided into two-stage method and one-stage method [11]. The two-stage method firstly generates the candidate region of pedestrian target, then obtains the features of the candidate region by convolution operation, and finally achieves the location of pedestrian target by classifier. In the detection method of the first stage, Anchor is set in advance, and then features are obtained by direct convolution operation for the

whole input image. Finally, pedestrian target position is located in the feature map [12] [13]. Two-stage detection usually has high detection accuracy, while one-stage detection has the advantage in detection speed.

1) two-stage

Grishick et al [14] proposed the Regions with Convolutional Neural Network (R-CNN) two-stage target detection algorithm, which requires the acquisition of candidate regions on the image, and then the classification and regression of the candidate regions. In the traditional method, the deep learning method is introduced. The detection accuracy is greatly improved. In order to solve the problem that R-CNN's budget for each independent candidate region easily causes redundant calculation of overlapping regions, In 2017, Purkait et al [15] proposed the Spatial Pyramid pooling-net (SPP-NET) algorithm to extract the features of candidate regions from the feature map of the original image after the convolutional layer, and extract the feature vectors after the Spatial Pyramid Pooling layer. According to the overall evaluation, SPP-NET algorithm is more impressive than R-CNN algorithm. It can not only detect images with different area sizes and realize normalized processing, but also has high detection efficiency and operation accuracy. However, the model volume of SPP-NET is still large and the steps such as candidate region extraction and detection are still carried out separately, which leads to the slow detection speed.

To solve a series of problems of SPP-NET, Girshick et al [16] proposed the Fast R-CNN algorithm, which added the region of interest pooling layer after the last convolutional layer of SPP-NET. Firstly, Fast R-CNN solves the previous storage space consumption problem, and secondly, it can change the parameters of all network layers at the same time. As mentioned above, the Fast R-CNN algorithm has outstanding advantages, but there are also limitations, such as: the selective search efficiency for extracting candidate regions is not high, and the time cost is large.

In 2015, Ren et al [17] proposed the Faster R-CNN target detection algorithm, which

replaces selective search to generate candidate regions in the region generation network. The region generation network determines that the anchor points are foreground or background through Softmax, and then revises by bounding candidate box regression. The Mask R-CNN algorithm proposed by He et al [18] is built on the Faster R-CNN framework, and transforms the region of interest pooling layer into a matching layer, and inputs the instance to split the branch after the transformation. After this optimization, the detection speed is improved.

Sheng proposed a new framework for pedestrian segmentation in infrared images [19]. The framework consists of a candidate region generation module and a classification module, both of which are implemented based on convolutional neural networks. Specifically, an effective segmentation threshold is learned by the deep learning method, and a saliency region self-growing algorithm is proposed to generate candidate objects. In addition, the literature classifies the candidate pedestrians with environmental perception to obtain the surrounding environmental conditions for auxiliary judgment, so as to reduce the false alarm rate. Cao proposed an Automatic Region Proposal Network (ARPN) to generate bounding boxes with confidence scores for far infrared pedestrian detection [20]. The model consists of two parts: First, the bounding box is predicted by L2 loss function, and the module is designed based on convolutional neural network. The module is simple, with only two layers, each with a 1×1 kernel. Secondly, the far infrared pedestrian is segmented based on the feature pyramid network to obtain the score map. The score is used as a confidence level for predicting bounding boxes. In order to obtain the score, a new labeling method for far infrared image segmentation is also introduced. This method can obtain the bounding box of each pixel directly and efficiently without any manually designed hyperparameters associated with the anchor box.

Devaguptapu proposes a "Pseudo Multimodal" object detector trained on visible light image data to help improve the performance of object detection in far infrared images [21]. Considering that there are currently large-scale visible light image datasets, while far-infrared image datasets are relatively few, this study attempts to

generate an equivalent pseudo-visible light image of a given far-infrared image using a well-known end-to-end image transformation framework, and then use the multimodal architecture is used to detect pedestrians in far infrared images.

2) one-stage

The first-stage detection algorithm mainly includes the You Only Look Once (YOLO) series target detection algorithm and the Single Shot Detector (SSD) target detection algorithm. In the two-stage detection algorithm, the network needs to classify the proposed bounding box several times in different regions of the input image. The one-stage detection algorithm does not need a separate network to propose and classify the bounding box. Instead, it calculates the position and category of the bounding box at the same time, which makes the one-stage detection have a fast inference speed.

In 2016, Redmon proposed YOLOv1 target detection algorithm [22], which aroused great response due to its fast detection speed. To solve the problem of poor visibility at night, Heo et al [23] combined YOLOv1 with the significant feature map constructed using adaptive Boolean diagrams, and successfully applied it to the thermal image dataset captured by moving vehicles, which greatly improved the detection accuracy while ensuring the detection speed. However, the pedestrian detection method based on YOLOv1 still has some shortcomings. For example, YOLOv1 divides the picture into 7×7 grids, and each grid can only detect one pedestrian, so when the detected pedestrians are close, the detection will be missed. In addition, the last two layers of YOLOv1 network structure are fully connected layers, and the model can only train images with the same resolution. When training images with different resolutions, the image must be scaled to make the pedestrians in the image deform or cause the loss of features.

Aiming at the deficiency of YOLOv1, Redmon proposed YOLOv2 target detection algorithm. YOLOv2 divides the picture into 13×13 grids for detection to improve the missed detection caused by the crowded pedestrians detected [24]; The last two

fully connected layers of YOLOv2 network structure are replaced by convolutional layers, so the network can input images of different sizes. In addition, the Anchor mechanism is introduced to improve the accuracy of frame regression in object detection [25]. Researchers proposed a series of pedestrian detection methods based on YOLOv2. For example, Hong et al [26], inspired by DenseNet[27], successively passed the low-level feature maps of YOLOv2 to the deep network. By combining the feature maps of different convolutional layers, more features are retained for network detection.

In 2018, Redmon designed the new network architecture Darknet-53 and proposed the YOLOv3 object detection algorithm [28]. Darknet-53 consists of 53 convolutional layers and utilizes multi-scale feature fusion. Hsu et al [29] proposed a new scale sensing mechanism based on the YOLOv3 algorithm, and used the multi-resolution fusion method to improve the false detection problem of small pedestrians in the image. Yang et al [30] introduced two subnets to detect pedestrians of different sizes, and then combined the outputs from the two subnets to generate reliable detection results. However, the pedestrian detection method based on YOLOv3 still has some shortcomings. For example, YOLOv3 uses convolution sampling feature maps of different scales for fusion, which reduces the detection speed. In addition, when IOU is greater than 0.5, the detection recall rate drops sharply.

In 2020, Redmon announced to stop all research work on computer vision, but other researchers are still improving YOLO algorithm, among which the most outstanding improvement is YOLOv4 proposed by Alexey [31] and YOLOv5[32] source code open source by Ultralytics team. YOLOv4 draws on the experience of CSPNet [33] and uses the CSPNet module to divide the feature mapping of Darknet-53 base layer into two branches, and then merge them to achieve the purpose of reducing the computation. The pedestrian detection model proposed by Wen et al [34] combines the novel Spatial Pyramid Pooling(SPP) network and K-means clustering algorithm with YOLOv4 model, which makes it easier to extract features. However, from a

practical point of view, the complexity of YOLOv4 prevents many hardware from fully supporting it, so its popularity is limited. Subsequently, the Ultralytics team opened source YOLOv5 [32], which was improved based on YOLOv4. Firstly, the adaptive anchor frame was introduced, and the length and width of the anchor frame were adaptively learned through training for border regression. In addition, the Focus structure is added, which realizes the size transformation of the feature map and reconstructs the high-resolution feature map into the low-resolution feature map, which can reduce the training parameters.

Although the infrared image targets detection algorithm based on depth of learning is not much, but more and more researchers have come to here, will be more excellent convolution neural network is applied to the infrared image targets detection, more suitable for the infrared image targets detection algorithm, infrared image targets detection algorithm of detection accuracy and speed will be improved.

To sum up, the research on pedestrian target detection technology is still in the stage of continuous development. Although a large number of relevant theories and algorithms have been proposed, there are still great challenges in the detection of pedestrian targets in infrared images at night. Considering the real-time requirements of automatic driving, time cost has become the most important consideration. In recent days, YOLOv6 [35] and YOLOv7 [36] have been updated successively, but due to the issue of release time, it is obvious that YOLOv5 is the most suitable for the task requirements of this project, so YOLOv5 will be introduced to this project.

2.3 Convolutional neural network theory

CNN is a kind of artificial neural network, which is used to process image-related tasks. The basic composition of CNN is composed of Convolutional layer, Pooling layer and Fully Connected layer, which is optimized by various Activation Functions, Dropout and Regularization to form a complex deep neural network [37].

1) Convolutional layer

Convolution is the most important operation in CNN, which is a linear operation that

multiplies the weight with the input matrix. Since the input image is two-dimensional, the weight is a two-dimensional matrix, that is, a convolution kernel. The linear operations is a dot product operations, its working principle is sliding on the input image convolution kernels, the overlapping area of convolution kernels from its input image multiplication under cover, and the sum to produce a single value, the convolution kernels is usually referred to as filter, its size can be adjusted, and the input image multiplied many times, produce characteristic figure.

In order to extract rich image features in CNN, a relatively large receptive field is usually required. Increasing the receptive field can be achieved by increasing the convolution kernel, but a large amount of image information will be lost if the convolution kernel is too large. Using small convolution kernel (3×3 or 5×5) for multiple Convolution can not only increase the receptive field to extract rich image information, but also avoid the loss of image information. In addition, the study shows that, because the activation function of each layer increases the nonlinear expression ability of the model, the use of small Convolution kernel stacking has better effect than the use of large convolution kernel stacking. The other hyperparameters that need to be set in the Convolutional layer are the step size of the Convolutional kernel sliding at the input (a larger step size will produce a smaller output feature map) and padding, so that the space size of the output feature map can be controlled and preserved relative to the input (usually using zero padding).

2) Pooling layer

The pooling layer is used to simplify the output of the convolution layer, which not only preserves the important information of the feature map, but also reduces the dimension of the feature map. For 2×2 pooling, the pooling layer obtains 2×2 pixel regions and summarizes them into a pixel value in the output feature map of the pooling layer, and then the summarized information can be provided to the next convolution layer. The common pooling methods are maximum pooling and average pooling. The output of maximum pooling is the maximum value of the input region, and its purpose is to extract features such as edges or basic shapes in the image

region. The output of average pooling is the average value of the input region, and its purpose is to make the extracted features smoother in the image region, as shown in Figure 2. After the pooling layer is used to reduce the dimension of the feature map, the convolution layer is usually used to increase the number of channels of the feature map, so that the size of the feature map keeps decreasing and the number of feature maps keeps increasing in the process of feature extraction.

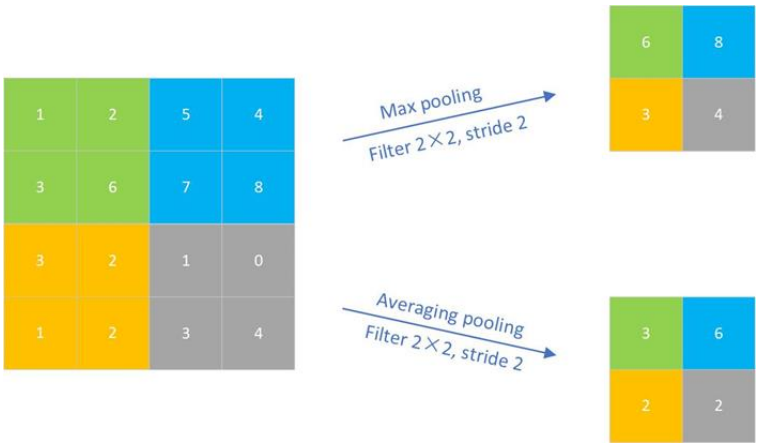


Figure 2 - Max pooling and averaging pooling operations

3) Fully Connected layer

The fully connected layer generally appears at the end of the network, which is used to integrate the previous features and use the feature information to obtain the final category label, playing the role of "classifier". In terms of connectivity, nodes in the fully connected layer are connected to every output in the previous layer, and its output is connected to every node in the next layer.

4) Activation Functions

In neural networks, the quality of activation function usually affects the performance of the network to a large extent. When activation functions are not used, the operations in neural networks are all linear relations, so it is not possible to train a model to fit the data in complex situations such as linear inseparability. The activation function brings nonlinear factors to the network and can express functions of any shape without previous restrictions. The most common activation functions are Sigmoid and Tanh. Their expressions are as follows:

$$f(x)\frac{1}{1+e^{-x}}$$

$$f(x)\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

As shown in Figure 3, the input range of the Sigmoid function is unlimited, and the output range is all from 0 to 1. After the activation function, it is equivalent to normalizing the output of each neuron, which can be well applied to the model that takes the predicted probability as the output. However, the exponential operation is introduced in this function, the calculation speed is relatively slow, and the gradient disappearance problem will occur.

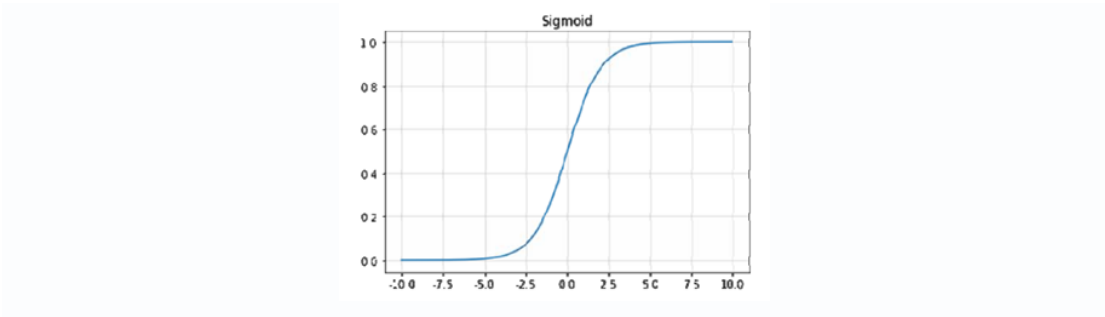


Figure 3 - Sigmoid function

In order to meet the requirements of real-time detection, we need to find a lightweight and fast activation function. In recent years, the activation function of the network has changed from Sigmoid to lightweight ReLU[38], and the performance has been greatly improved, and many activation functions beyond ReLU have emerged.

The ReLU function is simple but powerful. Because there is only linear relationship, it can only judge whether the input is greater than 0, and the computation speed and training convergence speed are very fast. In the case of input greater than 0, there is no gradient saturation problem. The Leaky ReLU function assigns a very small linear component to a value less than 0, solving the 0 gradient problem of negative values of the ReLU function, resulting in better performance than the previous activation function [39]. The Leaky ReLU function is widely used in YOLOv3, 4 and 5, which brings better performance and speed improvement and meets the requirements of

lightweight network. The expressions of the functions of ReLU and Leaky ReLU are as follows, and the shapes of the functions are shown in Figure 4.

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$$f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}$$

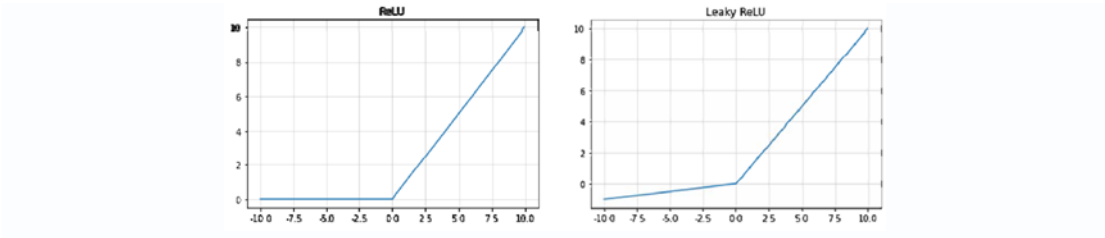


Figure 4 - ReLU function and Leaky ReLU function

5) Dropout

In 2012, Hinton et al [40] proposed Dropout algorithm, which combined multiple models by training and ignored half of the feature detectors to solve the problem of too many parameters but too few samples and prevent the occurrence of fitting phenomenon.

In every training, every feature detector with probability p retention (usually is 0.5, namely random half of the hidden layer node value of 0), the rest stop working, and does not depend on some local characteristics, play a regularization effect, reduced over fitting phenomenon.

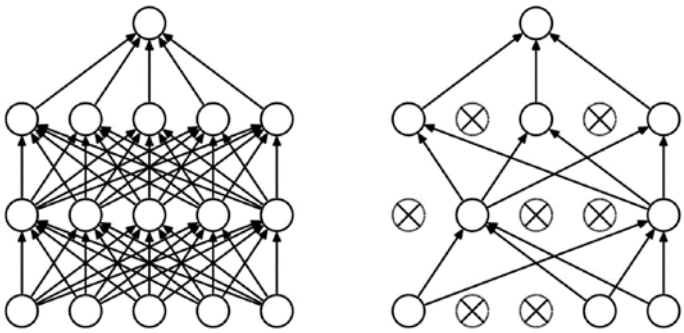


Figure 5 - Comparison between standard neural net and neural net after applying dropout

2.4 ResNet structure

With the continuous development of CNN, many researchers and research institutions have put forward many excellent network models. Some of these

networks have higher accuracy but are more complex and require strong computational power, while others have less accuracy but have obvious advantages in speed. This section will briefly introduce ResNet Structure.

ResNet (Residual Networks), also known as deep residual network, is composed of a series of residual blocks [41]. Its purpose is to solve the problem that deep neural networks are difficult to train. Figure 6 shows the basic structure of the residual block, which consists of two parts: the residual part (left side) and the direct mapping part (right side). The direct mapping part directly maps the input to the output layer, thereby preserving the integrity of the original input information. The Weight layer generally contains two or three convolution operations. In the learning process of residual network, the output value of the network is not fitting the input x to solve the output result $H(x)$, but defining the residual function $F(x) = H(x) - x$ from the input to the output, making the output result $H(x) = F(x) + x$. When $F(x) = 0$ and $H(x) = x$, it is also called congruent mapping.

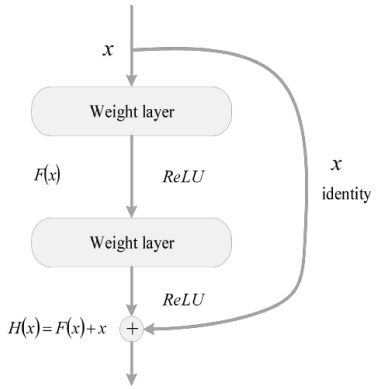


Figure 6 - The basic structure of the residual block

The underlying ResNet network consists of either 34 or 18 convolutional layers, all of which are located in a two-layer residual block. Without using residual structure, the error rate of 34-layer ResNet is higher than that of 18-layer ResNet on ImageNet classification dataset. The 152-layer ResNet, based on a residual block design, won three titles in the 2015 ILSVRC. Due to the residual design, the number of parameters and computational complexity of the deep neural network are significantly reduced, which greatly improves the learnability of the network. At the same time, the ResNet network can further reduce the number of parameters by eliminating the number of network layers, so that the model can work in lightweight applications.

2.5 Common Frameworks

Mainstream deep learning frameworks include TensorFlow, Keras, MXNet, PyTorch, etc., which are briefly introduced as follows:

- TensorFlow

TensorFlow is an opensource Python-based deep learning framework developed by Google and named for how it works. It is widely used in computer vision, audio processing, recommendation systems, natural language processing and other scenarios. TensorFlow mainly supports the form of static calculation graph, the structure of static calculation graph is very intuitive, but its debugging process is extremely complex, so the use of TensorFlow is too complicated has been a point of complaint of researchers. However, the high time complexity is exchanged for excellent performance. In the application scenarios with huge data volume, TensorFlow's GPU distributed computing capability is far superior to other frameworks, which is the main reason why it has been unanimously recognized in the industry.

- Keras

Keras is an opensource neural network library written in Python. It can run on TensorFlow, Theano, or MXNet. Compared with TensorFlow, Keras is simpler and achieves the goal of rapid neural network experiments. But Keras is essentially a high-level API for a base framework (such as TensorFlow), and when dealing with lower-level issues (such as modifications to existing models), Keras must be modified using the base framework.

- PyTorch

PyTorch was launched by Facebook AI researchers. It is a Python-based renewable computing package. PyTorch has two main characteristics :① it has a powerful GPU tensor computing ability; ② its calculation graph is a dynamic calculation graph, and the convolutional neural network built by PyTorch has the ability of automatic

derivation. PyTorch, formerly Torch, has the same underlying parts as the Torch framework, but PyTorch uses Python to rewrite everything else. PyTorch is a typical representative of dynamic calculation diagram, easy to debug, and has a high degree of modularity.

The characteristics of the three mainstream deep learning frameworks are shown in Table 1:

Table 1 - Features comparison of deep learning frameworks

Attribute	TensorFlow	Keras	PyTorch
Support Hardware	CPU, GPU	CPU, GPU	CPU, GPU
distributed	support	support	support
Imperative	nonsupport	nonsupport	support
Declarative	support	support	nonsupport
Computational Graph	static	static	dynamic
Learning difficulty	difficulty	easy	easy
Performance ranking	1st	3th	2rd
community integrity	perfect	perfect	perfect

According to the characteristics of this topic, we choose PyTorch deep learning framework to build YOLOv5 neural network model by integrating ease of use, performance, community activity and other factors.

2.6 YOLOv5 algorithm principle

The structure of YOLOv5 is shown in Figure 7, including input, Backbone, Neck and Head [42]. On the basis of YOLOv4, YOLOv5 has made some improvements to these four parts, which will be introduced in the following sections.

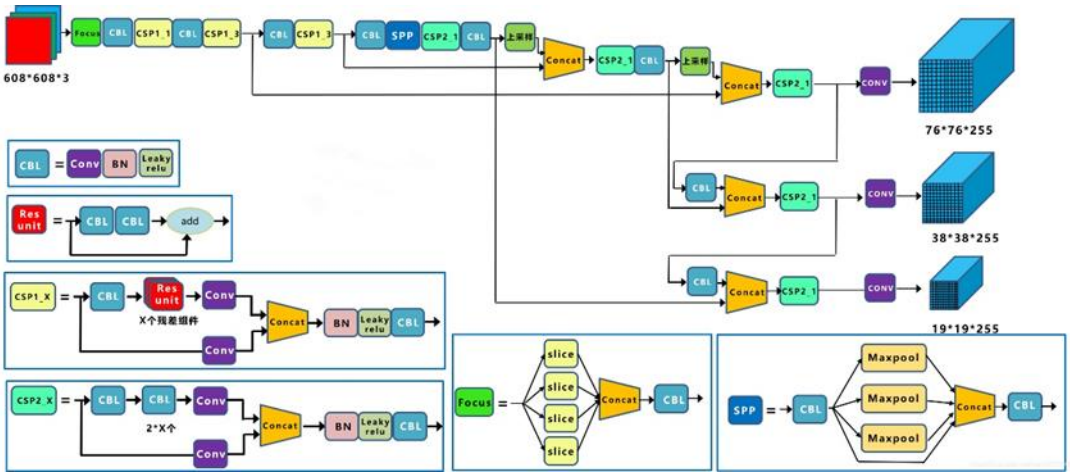


Figure 7 - Structure diagram of YOLOv5

1) Input

- Mosaic data enhancement

The input of YOLOv5 uses Mosaic data enhancement algorithm in the model training phase. This algorithm was first proposed in YOLOv4 paper, which is a very important data enhancement technique [43]. Mosaic is based in principle on the CutMix data enhancement method proposed in 2019, with minor improvements. CutMix uses two samples, while Mosaic uses four. Generally speaking, Mosaic adopts the method of randomly selecting four image samples from the data set, and forming a new image sample by means of random cropping, random scaling, arbitrary combination and stitching. The advantage of this method is that the detection set has enough sample data. More significantly, by means of random clipping, this method expands the data sets of the background and target objects to improve the stability of the network. At the same time, the data of four picture samples are calculated at one time, so that the size of mini-batch does not need to be increased and the expected effect can be achieved by a single GPU. However, it is not suitable for the case that there are many small targets in the data set, which will lead to smaller targets and reduce the generalization ability of the model. A comparison of the Mosaic and Cutmix data enhancement operations is shown in Figure 8.

Figure 8 - Enhanced comparison between CutMix and Mosaic data

- Adaptive anchor frame calculation

YOLOv5 directly nests the calculation of anchor frame into the whole process of training. In the process of model training, the way of output prediction frame is different from YOLOv3 and YOLOv4. The novelty of YOLOv5 is that it is carried out on the anchor frame set at the beginning, and then the loss value is obtained by comparing with ground-truth. This is then updated to continuously update the size of the anchor box. Through this adaptive form, YOLOv5 can arrive at the best anchor frame value even in different training sets. If in certain circumstances, the effect of the calculated anchor box is not ideal, you can choose to close the function of the calculated anchor box in the code to improve the final accuracy.

- Adaptive image zooming

Previous versions of YOLOv5 used images with an aspect ratio of one to one, such as 416×416, 608×608 pixel size. However, in the actual detection items, there are different proportions of picture length and width. If the data enhancement method is forced to fill the ratio as 1:1, then there will be the situation that the size of the black edge at both ends is not the same after scaling and filling. If the black edges on both

ends of the image height are reduced, the redundant information will also be reduced, and the speed of object detection will be further improved on this basis. The specific operation steps are successively calculated as contraction ratio, length and width of the image after contraction, pixel to be filled, and finally resize the image and fill the pixel.

2) Backbone

Backbone is the Backbone of the model. It extracts features from images and provides them to subsequent networks for target location and classification. YOLOv5 uses CSPDarknet structure as Backbone. Compared with YOLOv4 algorithm, its features are Focus as the benchmark network and CSP structure to improve the ability of feature extraction.

- Focus structure

Compared to YOLOv4, the Focus structure is unique to YOLOv5, where the key is the slicing operation. The Focus structure refers to that before the image enters the backbone network, four groups of images can be obtained by taking a value from every other pixel of the image, similar to the adjacent under-sampling operation. The Focus operation concentrates the width and height information of the image into the channel space, and the dimension of the channel is expanded by 4 times of the original, that is, the result after Mosaic is equivalent to the original RGB three-channel mode is changed into 12 channels. Finally, the obtained result is convolved again, and the final feature map with double under-sampling is obtained. The process of slicing operation is shown in Figure 9 below.

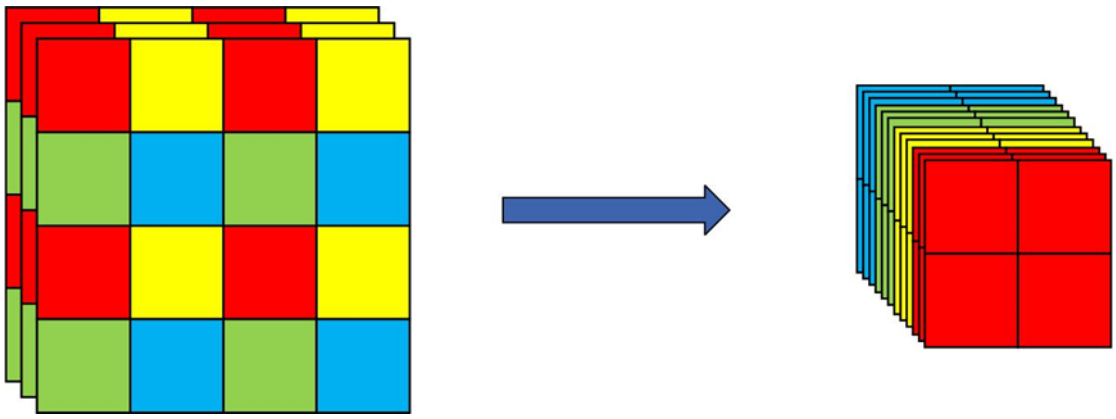


Figure 9 - Focus slicing operation

After the image passes the Focus structure, the most intuitive thing is that it plays the role of under-sampling, but the Focus operation is slightly different from the commonly used convolution under-sampling. We can compare the calculation amount of Focus with the down-sampling calculation amount of ordinary convolution, and the calculation amount is evaluated by FLOPs. Taking YOLOv5s as an example, the input image size is 608×608×3, and the comparison result of calculation amount is as follows:

① If ordinary convolution under-sampling is used, the convolution kernel size is 3×3, the step size is 2, and the number of output channels is 32, then the dimension of the feature map obtained after under-sampling is 304×304×32. Then the theoretical result of the calculation amount of ordinary convolution under-sampling without deviation is shown in the formula.

$$FLOPs(Conv) = 3 \times 3 \times 3 \times 32 \times 304 \times 304$$

② If the Focus structure is adopted, slicing operation is first adopted to change the size of the feature map to 304*304*12. After a convolution operation with a convolution kernel size of 3*3, the final output feature map size is 304*304*32, as shown in Figure 10. The theoretical results of the Focus structure without bias are shown in the formula.

$$FLOPs(Focus) = 3 \times 3 \times 3 \times 4 \times 32 \times 304 \times 304$$

It can be seen that the computational burden of the Focus structure is more than that of the ordinary convolution down-sampling, which is about 4 times that of the

ordinary convolution down-sampling. However, the benefit of the Focus structure is that the loss of information is reduced in the process of down-sampling. To sum up, the main function of the Focus structure adopted by YOLOv5 algorithm is that, in the process of image down-sampling, wide and high information is concentrated into the channel dimension, and the convolutional layer is used for feature extraction, making feature extraction more adequate. Although it brings some computation increase, it reduces the loss of information.

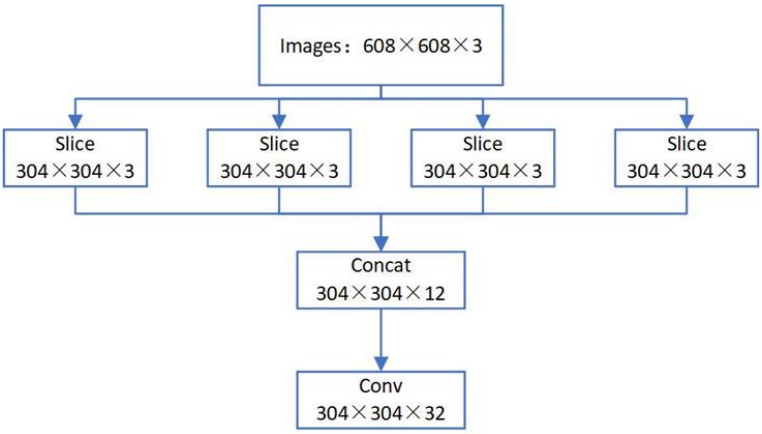


Figure 10 - Schematic diagram of Focus structure

- CSP structure

YOLOv4 uses the design idea of CSPNet for reference and adds the CSP structure to Backbone network Backbone. YOLOv5 inherits YOLOv4 and designs two CSP structures: CSP1_X is located on Backbone and CSP2_X is located in Neck. A convolution layer is set in front of the CSP1_X structure. The size of the convolution kernel is 3*3 and the step size is 2, which reduces the size of the feature map by half and plays the role of down-sampling. Backbone has 4 CSP1_X structures, and the final output feature map size is 19*19.

Deep learning network framework is prone to the problem of a large amount of computation in the inference process, which is generally due to the large number of repetitions of gradient information in the network. To solve this problem, CSP structure refers to Resnet network to merge feature maps across multiple Convolutional layers, enhance the information of deep feature maps, reduce the

amount of computation, and ensure the accuracy of detection.

YOLOv5 uses CSPDarknet as Backbone to extract the features of the input image, which solves the problem of the large amount of computation caused by the repetition of gradient information in the network. The gradient information is concentrated into the features through Focus and CSP structure, which reduces the required parameters of the model, improves the computational efficiency and ensures the accuracy of the model.

3) Neck

In deep learning, Neck is located between Backbone and Head. Generally speaking, the length and width of feature maps extracted by Backbone are large, while the positioning output results and classification dimensions are small. Therefore, the length and width of features need to be compressed. Neck can make better use of the features extracted by Backbone and further fuse and process the features to the Head module for positioning and classification. With the continuous development of deep learning, more Neck modules and updated technologies have been innovated. The Neck module of YOLOv5 adopts FPN+PAN structure, which combines down-sampling and up-sampling to generate feature pyramid, producing a network with the effect of non-deformation. The FPN+PAN structure is shown in Figure 11.

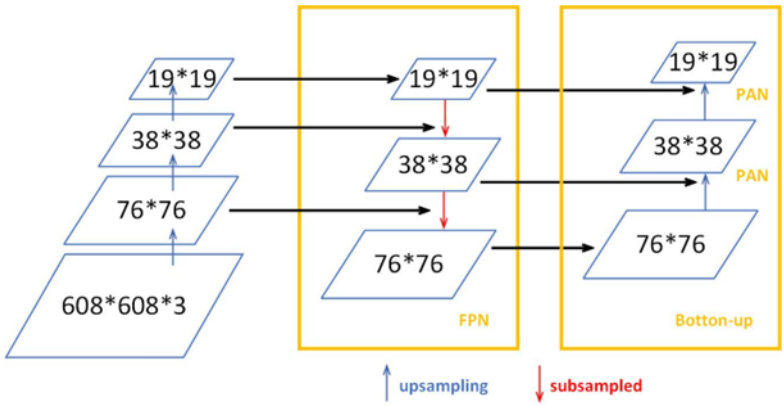


Figure 11 - PN+PAN structure

The feature pyramid generated by FPN fuses the up-sampling results with the features of the next layer. This method has excellent results when detecting small objects. In contrast to FPN, the feature pyramid generated by PAN is bottom-up. It

fuses the down-sampling result with the last feature. PAN can segment the shallow features of the network. Therefore, the importance of shallow network information in target detection is extraordinary. In YOLOv5, the structure combined with FPN+PAN is selected as the Neck module. The top-down direction is strengthened in the bottom-up direction, so that the top information or the location information of the bottom features can be obtained, and the detection effect of large targets can be enhanced.

4) Head

In YOLOv5 algorithm, Head is the final detection and output part. The features extracted by Backbone are compressed and fused by Neck to make classification and prediction. YOLOv5 adopts the same Head network as YOLOv3 and uses a convolution structure of 1×1. There are three sets of outputs, the dimensions of which are bs×255×76×76, bs×255×38×38 and bs×255×255×19×19 respectively, where BS is batch size. The calculation method of 255 is na×(nc +1+4), where na (number of anchors) is the number of scales of each group of anchors, YOLOv5 sets 3 anchors in each small cell, and each group has 3 scales, nc is the number of class. There are 80 COCO dataset categories, 1 is the confidence of foreground background, 4 is the coordinates of center point and width and height (x, y, w, h).

- Loss function

In the target detection task, the accuracy of the position of the prediction box is more important than the classification accuracy of the target in the box. The loss functions of the prediction box mainly include IOU_Loss, GIOU_Loss, etc. YOLOv5 used BECLogits loss function to calculate confidence loss, cross-entropy loss function was used for classification loss, and GIOU_Loss was used for prediction box. IOU_Loss is relatively simple, as shown in the following formula.

$$IOU_{Loss} = 1 - IOU = 1 - \frac{A}{B}$$

Where, A and B are the intersection and union of the predicted box and the real box respectively. This calculation method is simple, but there are two problems: the prediction box and the target box do not intersect, that is, A=IOU=0, and the loss function is not differentiable at this position, that is, IOU_Loss cannot optimize the disjoint situation of the two boxes. The prediction box overlaps with the target box and the IOU is the same, but the overlap position of the two boxes is not the same. IOU_Loss cannot distinguish between them. Therefore, GIOU_Loss was proposed to improve the prediction box in 2019. GIOU_Loss introduces the minimum surrounding rectangle of the prediction box and the real box, as well as the difference set (the difference between the minimum surrounding rectangle and the union), and has

$$GIOU_{Loss} = 1 - \mathrm{G}IOU = 1 - \left(IOU - \frac{|C \backslash (A \cup B)|}{|C|}\right)$$

- Non-Maximum Suppression

In the post-processing process of object detection, because the detection model will output more prediction boxes for the input image, most of the contents contained in these prediction boxes are the same, and only one prediction box with the best effect is needed to simplify the detection result. At the same time, after NNS suppression, the previously undetected occluded overlapping target can be identified.

NMS searches for local maxima by suppressing non-maximum elements. The specific steps are as follows: Firstly, all the selection boxes obtained are sorted according to the score in the iteration process, and the selection box with the highest score is extracted. Then overlap the remaining selection boxes with the selection box with the highest score. If IOU is greater than the initial threshold, the selection box will be deleted. Do this by sorting out the other box with the highest score among the unprocessed boxes.

2.7 Attention Mechanism

Attention Mechanism has gradually become one of the important methods and research hotspots in the field of deep learning. It is a special structure that can be embedded in deep learning models and has been successfully applied to many deep learning models to improve model performance.

1) SENet

SENet (squeeze-and-Excitation Networks) [44]is the ImageNet2017 Category winner, which is a plug and play networking module. SENet has been applied to some classical network models, which plays a significant role in improving the model effect.

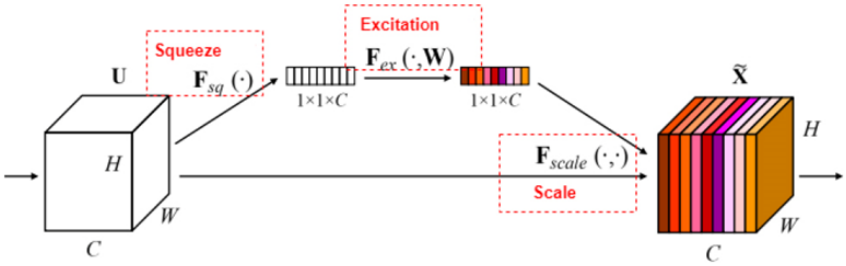


Figure 12 - SENet schematic diagram

As shown in Figure 12, SENet attention module performs the following three operations on the feature map obtained by convolution: Squeeze operation, Excitation operation, and Scale operation, among which the first two operations are the core content of the module. Squeeze operation is to reduce the dimensionality of the original image data. In the process of Squeeze operation, global pooling is used to convolve the feature map into a vector of 1×1×C (where C is the number of channels), so that compression is achieved in the spatial dimension. The corresponding weights are generated for each feature channel through training and learning. The weights are constantly updated in the network (i.e., feature redirection). The network learns to adjust which features need to be strengthened and which need to be weakened through the weight values. The first full-connection layer is applied for feature fusion between channels. The second one gets the weight of each feature

map on the channel. Then, the original feature map is weighted in the channel direction by Scale operation.

2) CBAM

Studies on lightweight networks show that channel attention can significantly improve model performance. However, channel attention only focuses on the relationship between channels, ignoring the relationship between the location information of different spatial regions. Therefore, Sanghyun Woo et al proposed CBAM, a hybrid attention structure integrating channel attention and spatial attention, for the first time in 2018 [45]. The innovative point of CBAM is not only to draw on the idea of SENet and design a new channel attention module through GAP and GMP, but also to put forward a hybrid attention module combining channel attention and spatial attention, which can be embedded in other existing network structures. Its specific structure is shown in Figure 13.

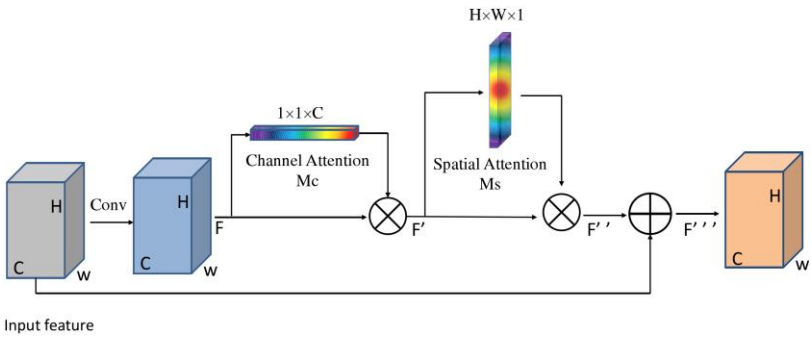


Figure 13 - CBAM structure

3) CA

CA attention module (Coordinate Attention) is a new mobile network attention mechanism proposed by Hou Q et al. in 2021, which embeds location information into channel attention [46]. The coordinate attention of the CA attention module decomposes the channel attention into two one-dimensional feature encoding processes, and aggregates features along two spatial directions respectively, one of which retains location information and the other obtains long-distance dependencies. relationship, which can effectively alleviate the loss of position information caused by 2D global pooling. The CA attention module is flexible and lightweight, it can not

only accurately retain the position information, but also capture its long-distance dependencies, which is very helpful for identifying the target object of interest, and can be easily inserted into the classic deep learning network model to improve the performance of the model.

The CA attention module uses two one-dimensional global pooling operations, and then transforms the input features into corresponding feature maps in the two spatial directions through aggregation operations along the vertical and horizontal directions respectively [46]. Then, for the characteristics of the vertical and horizontal direction after processing to get figure coding operation, make the two note to a position sensitive, retained the accurate location information, another direction sensitive to capture the reliance of the long distance relationship, then change after get attention to combining with characteristics of input figure processing, by emphasizing the interested information to improve the ability of feature extraction.

2.8 User interface design library

PyQt5 has a Graphical User Interface (GUI) based on Python programming language, which makes it easy to design software interfaces. It is widely used because of its drag-and-drop operation mode, support for cross-platform (Windows, Linux, etc.) transplantation, and enrich code base [47].

PyQt5 main modules are:

QtCore: includes core non-GUI functions, system time, files and folders, various data processes and thread blocks.

QtGui: Includes Windows, events, 2D images, basic paintings, fonts, and text classes.

QtWidgets: Includes all UI features for creating software applications.

The interface of QtDesigner is shown in Figure 14.

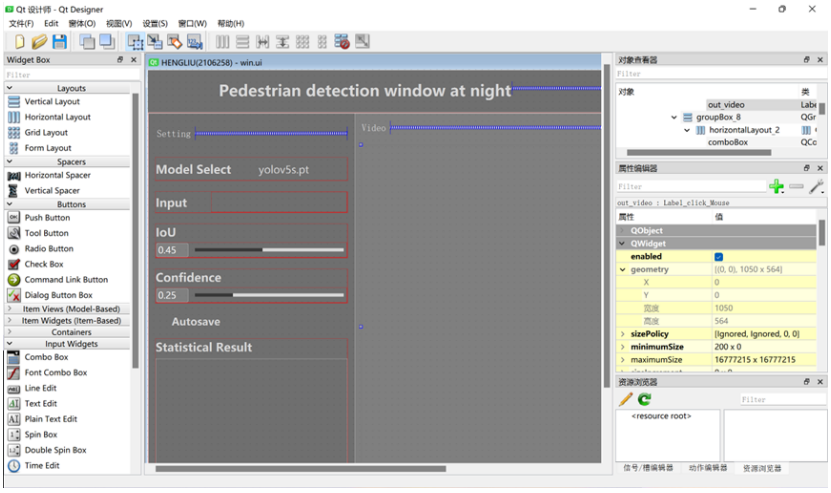


Figure 14 - Basic interface of QtDesigner

2.9 Literature review conclusions

This chapter reviews the previous studies involved in this project. First, different recognition methods for object detection are discussed, and then YOLOv5 is selected as the algorithm of this project, and the YOLOv5 algorithm is analyzed in detail. Then the Attention Mechanism is discussed, and in the following research, the algorithm will be improved through Attention Mechanism to achieve the goal of this project.

Chapter 3 - Research methodology

The aim of this project is to investigate different techniques and algorithms for pedestrian detection in infrared images. By the end of this project, the goal is to produce a program that uses the YOLOv5 algorithm and its three improved algorithms to detect pedestrians in infrared images. The procedure will be experimented to investigate the effectiveness of the four methods. The end result should be to collect accuracy information about each method and draw a conclusion about which algorithm is most suitable for pedestrian detection in infrared images.

3.1 Philosophy

The paradigm of philosophy is defined by research philosophy, a unique method of knowledge development [48]. In business research, there are four main research philosophies: pragmatism, positivism, realism, and interpretivism [49]. Positivism focuses on data gathering and analysis in order to comprehend the objective reality and to give people actual, practical, certain, and precise information. Positivism seeks to transcend or exclude value judgement, discloses just the intrinsic constituents of the objective reality and their universal interconnection, and summarises the essence and operating principles of the phenomenon. Understanding the external reality as well as researching the movement law and internal logic of the phenomenon itself are the goals of positivism. According to interpretivism, there is no such thing as absolute objectivity, and the meaning of a study is primarily derived from the individual's subjective personal experience and sentiments. According to presumptions about the scientific method of knowledge growth, realism is based on the notion that reality can be distinguished from the human mind.

Based on the above analysis and combined with the purpose of this project, the most appropriate philosophy can be viewed by positivism. The reason is that the purpose of the project is to collect the accuracy information of each method and conclude which algorithm is the most suitable for infrared image pedestrian detection, which

mainly focuses on the results and does not involve subjective judgment.

3.2 Approach

Deductive research methods, inductive research methods and abductive research methods are the three main types of research methods. Deduction involves the development of a theory, then through a sequence of propositions, and the concept must be run in a way that allows quantitative evaluation of facts and testing of presupposition hypotheses, which are first derived from general principles and then tested by testing the previously proposed hypotheses. Induction is a kind of reasoning from the individual to the general. Transition from a certain degree of views on individual things to a wider range of views, and deduce the general principles and interpretation methods of principles from special and specific cases [50]. Abductive method is a kind of information processing method to infer the cause of a phenomenon according to its characteristics.

In this project, the object detection algorithm is studied, and the best one is selected by collecting the accuracy of each algorithm. Therefore, the deductive method is the most suitable research method for this project to obtain and analyze conclusions from hypotheses.

3.3 Strategies

Experiments, literature research and interface development were mainly taken in this project. Experiments are a series of operations or activities designed to test a theory or confirm a hypothesis. This project verifies the effectiveness of YOLOV5 and its improved algorithm through multiple experiments. In order to fully and accurately comprehend the issue that has to be researched, literature research is a way for gathering information by looking into the literature in accordance with a specific research aim or topic. This project has carried out a detailed study on the algorithm principle and attention mechanism through a large number of literature research. This project also designed a user visual interface to perform pedestrian detection by

uploading videos or turning on the camera.

3.4 Risks

The data collection part of this project will involve the participation of volunteers, so the collected data must be strictly reviewed and processed. The data collection for this project was authorized by Wuhan University of Technology, so it will not break any laws. And it ensures that the data set does not contain personal information and other irrelevant information, which ensures the privacy of volunteers.

Chapter 4 - Data collection and processing

Andrew Ng, one of the most authoritative ML scholars in the world, answers the question "Are the advances in machine learning due to models or data?" In this debate, he stated that 80% data and 20% models equal better machine learning. Therefore, the importance of data is self-evident, so producing high-quality data sets is the first step to the success of deep learning algorithms. Therefore, in order to adapt to the road situation in China, a self-made data set is made in this project to lay a good foundation for the following experimental work.

4.1 Data acquisition equipment and process

Images were captured with an InfiRay Xtherm T3Pro thermal imaging camera, and Digital detail Enhancement (DDE) was used to see more important details and measure temperatures between -20C and 400C. Its technical parameters are shown in Table 2.

Table 2 - Far infrared camera detailed parameters

Parameter Name	Parameter Values
Resolution	934×708
Pixel Size	17μm
Spectral Band	8~14μm
Lens	13mm
Image Frame Rate	25Hz
Operating Temperature	-20℃~+ 50℃
Measuring Range	-20℃~+400℃
Accuracy	±2°C or ±2% of reading, the greater shall prevail
Image Enhancement	Digital detail enhancement (DDE)

The driving scene is collected by a position mounted on the top of the car's passenger seat, and the installation position is shown in Figure 15. Each image is from the continuous segment of 3min video acquisition data, a total of 103 segments were collected. Finally, the method of frame extraction was used to intercept the image of each video to obtain 1161 images to make the HL_FIR2022 data set. The videos are all from the geographical environment in and around Wuhan University of

Technology in July 2022. The weather conditions are sunny, and the collection time is day and night.

Figure 15 - Far infrared camera installation position

4.2 Rename the dataset images

In order to standardize the management, six digits of "000000" are adopted to rename the picture in an successively increasing manner with the growth rate of 1. The named picture is shown in Figure 16.

Figure 16 - renamed images

4.3 Annotate dataset images

For the annotation of pictures, there are many tools available on the network, such as LabelImg, Labelme, etc. After comparative analysis, LabelImg is finally used to complete the annotation of pictures. Enter Person in the predefined_classes.txt file in the data folder, and the Person preselection will appear during annotation to facilitate annotation operations. The annotation interface is shown in Figure 17.

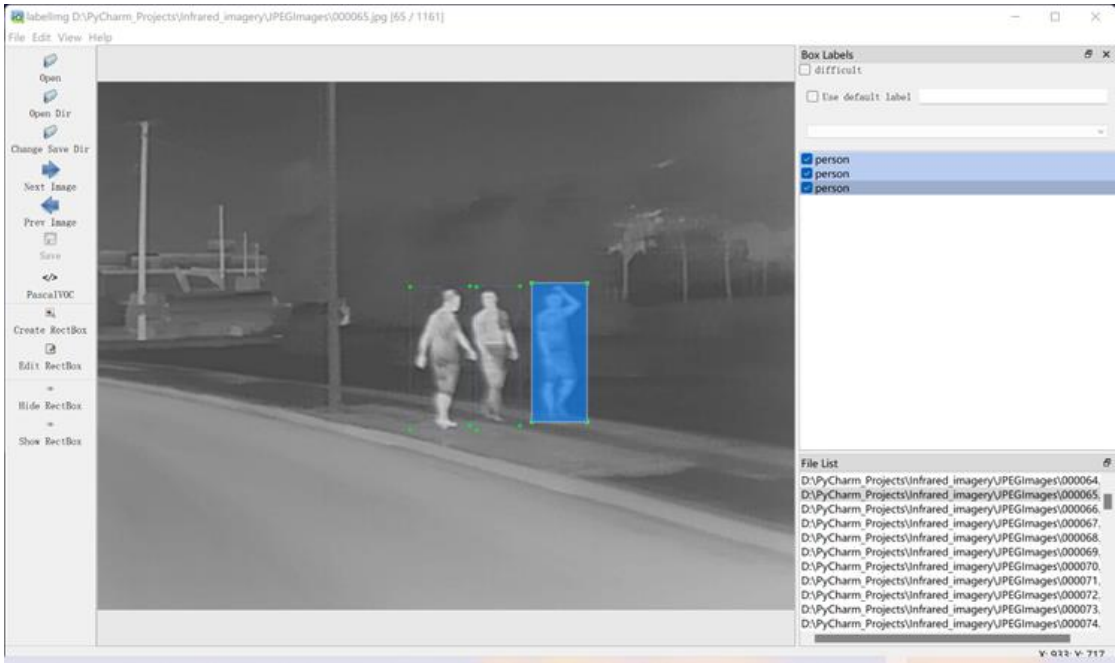


Figure 17 - Infrared images annotation

The dataset annotation format adopted by the YOLO family of algorithms is different from SSD and Faster R-CNN. The format of SSD and Faster R-CNN datasets is PASCAL VOC dataset format, and after annotation, XML files are generated to store information such as the size, target category and specific location of the image, as shown in Figure 18.

000065.xml

```xml
<annotation>
    <folder>images</folder>
    <filename>000065.jpg</filename>
    <path>D:\PyCharm_Projects\Infrared_imagery\images\000065.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>934</width>
        <height>708</height>
        <depth>1</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>person</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>419</xmin>
            <ymin>275</ymin>
            <xmax>508</xmax>
            <ymax>467</ymax>
        </bndbox>
    </object>
    <object>
        <name>person</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>499</xmin>
            <ymin>274</ymin>
            <xmax>566</xmax>
            <ymax>461</ymax>
        </bndbox>
    </object>
    <object>
        <name>person</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>582</xmin>
            <ymin>270</ymin>
            <xmax>656</xmax>
            <ymax>456</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 18 - Xml annotation content

After the YOLO dataset is annotated, a TXT file is generated to store the target class ID, the target's center x-coordinate (horizontal)/total image width, the target's center Y-coordinate (vertical)/total image height, the target's width/total image width, and the target's height/total image height. This is shown in Figure 19. Of course, these two data formats can be converted to each other using script code.

000065.txt

```txt
0 0.49518201284796576 0.5225988700564972 0.09528907922912207 0.2711864406779661
0 0.5690578158458245 0.5176553672316384 0.07173447537473233 0.2641242937853107
0 0.6616702355460385 0.5112994350282486 0.07922912205567452 0.2627118644067797
```

Figure 19 - TXT annotation content

The PyTorch framework used in this paper does not need to convert XML files to CSV format like other open source frameworks such as TensorFlow, and then generate TFRecord format files for training before training. We just need to proportionally split the images and generated TXT files into the training and validation sets. In this paper, a total of 1161 photos are collected, and the data set is randomly divided into training set and validation set according to the ratio of 7:3, in

which the training set contains 812 images and the validation set contains 349 images.

4.4 Pedestrian analysis in a dataset

As shown in Figure 20, the Kmeans algorithm is used to calculate the number, annotation frame, location and size of pedestrians in the data set produced this time. It can be seen that the data set contains more than 2000 pedestrians, and the targets are mainly concentrated in the lower part of the image. As can be seen from the bottom-right image in the figure, the targets are mainly small-sized targets, which conforms to the actual situation in the current research and is more conducive to the algorithm to meet the subsequent use. Therefore, this dataset is reasonable.

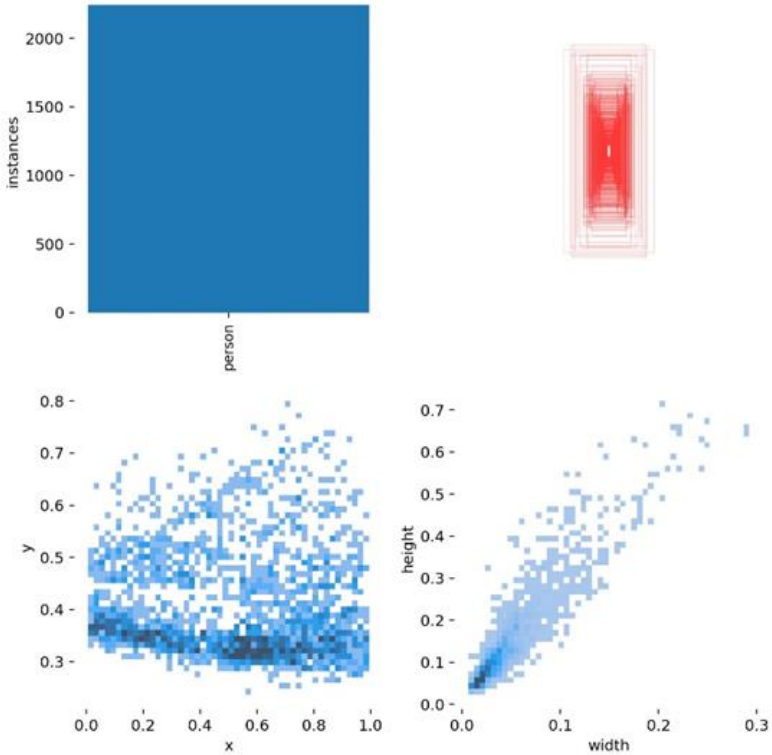


Figure 20 - Analysis of the pedestrian situation in the dataset

Chapter 5 – Design & Implementation

5.1 Resources and planning requirements

This project requires a variety of resources and plans to enable the authors to produce design and implementation in order to achieve the goals set by this project. This is followed by a detailed description of the hardware and software required for the project.

- Hardware

Deep learning algorithms frequently generate a huge number of parameters during the training and reasoning phases as deep learning research advances, necessitating a computer with more memory and processing capability. For the parallel processing of enormous amounts of data, NVIDIA has created a specific GPU inference module. As a result, the NVIDIA GeForce RTX 3050 graphics card is chosen when configuring the environment. The GPU is utilised to speed up data calculations, training speeds, increase model correctness, and satisfy training and reasoning requirements by setting the CUDA and CUDNN environment. Table 3 displays the hardware setup of the machine.

Table 3 - System software configuration

Operating system	Windows11
CPU	AMD Ryzen 7 5800H
Mainboard	LENOVO LNVNB161216
GPU	NVIDIA GeForce RTX 3050（4GB）
RAM	Samsung 16GB
HDD	WDC PC SN730 SDBPNTY 512G

- software

The running of YOLOv5 algorithm also depends on the construction of the

environment in the software. For example, image processing requires OpenCV and Pillow library functions. The SciPy and NumPy libraries need to be configured to perform high-end scientific computations and extensive multi-dimensional array and matrix operations. To use GPU to carry out large-scale parallel computing, CUDA and CUDNN should be configured in advance. Tensorboard must be installed in order to visualise the training process; Matplotlib must be used in order to annotate the detection results on an image; and so on. Table 4 displays the configuration of the system software.

Table 4 - System software configuration

Name	Versions	Function
PyTorch/torchvision	1.7.1/0.8.2	Accelerating deep neural networks on GPU
CUDA/CUDNN	11.2/8.1.0	To improve the large-scale parallel computing capability of GPU
Anaconda	3	Efficiently manage and create environments that work on multiple different projects
Pycharm	Community 2022.2	Program debugging, code improvement
Python	3.8.13	Programming language
opencv-python	4.6.0.66	Computer Vision Library
Pillow	9.2.0	Third-party image processing library
scipy	1.7.3	Scientific Computing Toolkit
numpy	1.22.3	Mathematical function library
matplotlib	3.5.1	Python 2D drawing library
tensorboard	2.6.0	Visualizing the training process

- Environment construction steps

Relying on Windows11 operating system, this paper finally builds a deep learning environment based on Anaconda virtual environment, Python3.8 as the basis, PyTorch as the open source framework, which can realize GPU acceleration and run YOLOv5 algorithm. The installation steps and installation precautions are as follows.

Installation steps

① Install Anaconda and Pycharm. Anaconda provides support for creating

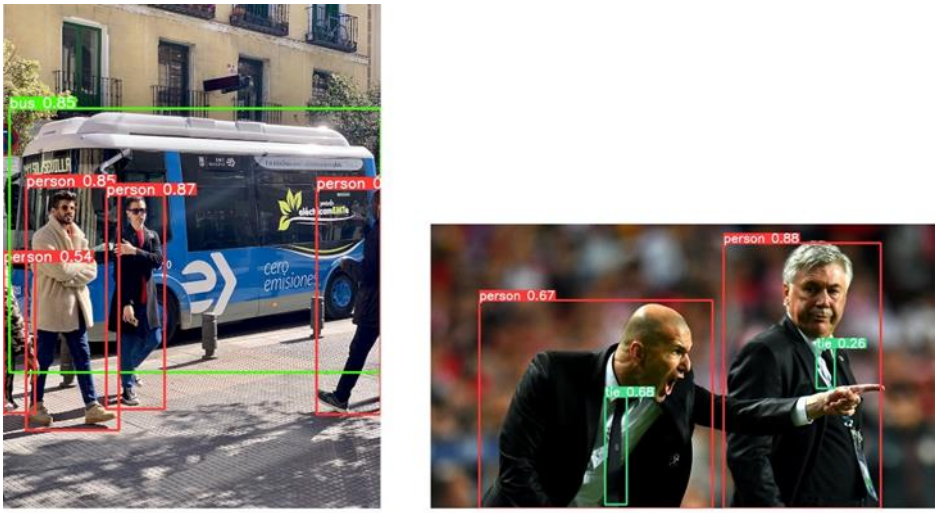
virtual environments and includes numerous Python packages for scientific computing and data analysis for later version management. After downloading the installation package from the official website, install it through the terminal and configure the environment variables.

② Install PyTorch. On the virtual environment created by Anaconda, install the CUDA and CUDNN versions corresponding to the PyTorch version, and configure the environment variables.

③ Clone the YOLOv5 project. Clone the YOLOv5 project on Github and install the dependencies in the requirements.txt file in the YOLOv5 directory.

④ Download the pre-trained weights file. Download the yolov5s.pt, yolov5m.pt, yolov5l.pt, Yolov5x.pt weights files on Github and place them in the weights folder.

⑤ Install tests. Image and video tests were conducted to verify the installation results. A successful installation is shown in Figure 21



bus.jpg zidane.jpg

Figure 21 - Image detection results in the YOLOv5 environment

Installation Precautions

① The unification of various versions. When building the PyTorch

environment, it is necessary to take a top-down approach to determine the Python version and PyTorch version needed by the YOLOv5 algorithm, and then determine the graphics card driver, CUDA version and CUDNN version in turn combined with the computer hardware equipment, which is the most core part.

② You can change the source of the in-China mirror in China. In order to facilitate the later download software, the domestic Tsinghua source can be changed to improve the download speed.

③ Run installation tests. After the installation is completed, you must perform installation tests to check whether the installation is successful, so as to ensure the subsequent successful use.

5.2 Design

5.2.1 Algorithm design

Three attention mechanisms, SE, CBAM and CA, are added to the original YOLOv5 algorithm, and the specific implementation is as follows:

- SE

First, add the implementation code of the SE attention module in the common.py file, then call the function added in common.py by modifying the parameter definition of the yolo.py model, and finally find the backbone network part in the parameter configuration file of the network structure. Add SE to the backbone network. The code implementation of SE in the YOLOv5s model and its added network structure are shown in Figure 22 and Figure 23.

```python
class SE(nn.Module):
    def __init__(self, c1, c2, ratio=16):
        super(SE, self).__init__()
        #c*1*1
        self.avgpool = nn.AdaptiveAvgPool2d(1)
        self.l1 = nn.Linear(c1, c1 // ratio, bias=False)
        self.relu = nn.ReLU(inplace=True)
        self.l2 = nn.Linear(c1 // ratio, c1, bias=False)
        self.sig = nn.Sigmoid()
    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avgpool(x).view(b, c)
        y = self.l1(y)
        y = self.relu(y)
        y = self.l2(y)
        y = self.sig(y)
        y = y.view(b, c, 1, 1)
        return x * y.expand_as(x)
```

Figure 22 - SE code

Run: train

	from	n	params	module	arguments
0	-1	1	3520	models.common.Focus	[3, 32, 3]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	1	156928	models.common.C3	[128, 128, 3]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	1	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	656896	models.common.SPP	[512, 512, [5, 9, 13]]
9	-1	1	1182720	models.common.C3	[512, 512, 1, False]
10	-1	1	524288	models.common.SE	[512, 512, 1]
11	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
12	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
13	[-1, 6]	1	0	models.common.Concat	[1]
14	-1	1	361984	models.common.C3	[512, 256, 1, False]
15	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
16	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
17	[-1, 4]	1	0	models.common.Concat	[1]

Version Control | Run | Python Packages | TODO | Python Console | Problems | Terminal | Services

Figure 23 - The network model after adding the SE module

- CBAM

First, add the implementation code of the CBAM attention module in the common.py file, then call the function added in common.py by modifying the parameter definition of the yolo.py model, and finally find the backbone network part in the parameter configuration file of the network structure, And replace the four C3 layers of the backbone network with the CBAMC3 attention module respectively. The code implementation of CBAM in the YOLOv5s model and its added network structure are shown in Figure 24 and Figure 25.

```python
class CBAMC3(nn.Module):
    # CSP Bottleneck with 3 convolutions
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5):  # ch_in, ch_out, number, shortcut, g
        super(CBAMC3, self).__init__()
        c_ = int(c2 * e)  # hidden channels
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c1, c_, 1, 1)
        self.cv3 = Conv(2 * c_, c2, 1)
        self.m = nn.Sequential(*[Bottleneck(c_, c_, shortcut, g, e=1.0) for _ in range(n)])
        self.channel_attention = ChannelAttention(c2, 16)
        self.spatial_attention = SpatialAttention(7)

        # self.m = nn.Sequential(*[CrossConv(c_, c_, 3, 1, g, 1.0, shortcut) for _ in range(n)])

    def forward(self, x):
        return self.spatial_attention(
            self.channel_attention(self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)), dim=1))))
```

Figure 24 - CBAM partial code

Run: train

	from	n	params	module	arguments
0	-1	1	3520	models.common.Focus	[3, 32, 3]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	19426	models.common.CBAMC3	[64, 64]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	3	229926	models.common.CBAMC3	[128, 128]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	914214	models.common.CBAMC3	[256, 256]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	656896	models.common.SPP	[512, 512, [5, 9, 13]]
9	-1	1	559202	models.common.CBAMC3	[512, 512, False]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]

Figure 25 - The network model after adding the CBAM module

- CA

The specific implementation of the CA attention module is similar to the steps in SE.

5.2.2 Detection Interface Design

The interactive interface of this project is designed by PyQt5. After the design is completed, a ui file is generated, and then the ui file is converted into a py file and embedded in yolov5. The main interface of the infrared image pedestrian detection system is shown in Figure 26. This system mainly includes several modules of "video display", "button operation" and "detection result display", among which "video display" is mainly used to display the infrared image pedestrian detection result image; "button operation" is mainly responsible for the model selection, Input video or image selection, IoU and confidence debugging, saving video after detection, detection start and pause, etc.; "Data result display" is mainly used to display the current system detection rate and the number of people detected.

Figure 26 - Infrared image pedestrian detection system interface

5.3 Implementation

This experiment is based on the previously built environment. Because the training network is used for pedestrian detection, only one label "person" is reserved in the category name table in the YOLO network, and the configuration file nc=1 is changed. The initialization hyperparameters are shown in Figure 27.

The hyperparameters during training are shown in Table 5. By controlling variables, we try to achieve the expected effect, and other hyperparameters use default values.

Table 5 - Training hyperparameters

Hyperparameter	value
Batch-size	8
Train-size	640
Val-size	640
Epoch	200
Workers	2

```yaml
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

Figure 27 - Initializing the hyperparameters

Chapter 6 – Results and evaluation

6.1 Evaluation indicators

- Precision and Recall

Precision and Recall are metrics used in machine learning to measure the accuracy of classifiers. Among them, the accuracy rate refers to the ratio of the number of correctly predicted data to the number of data predicted as positive examples, which can reflect the correct rate of the prediction results of the model. The recall rate refers to the ratio of the number of correctly predicted data to the real positive data, which can reflect the detection rate of the model for all targets. The target detection model usually judges the relationship between the predicted value of the sample and the true value according to the confusion matrix, and then calculates the precision and recall rate. Table 6 shows a confusion matrix.

Table 6 - confusion matrix

		Prediction	
		Positive	Negative
Actual	Positive	TP	FP
	Negative	FN	TN

The formulas for precision and recall are as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

- IoU

Intersection-over-unions (IoU) is the ratio of the area of the Intersection region to the area of the union of two boxes. In the field of object detection, it is often used to measure the overlap between the predicted box and the true box. A larger IoU indicates a higher degree of overlap between the two boxes; otherwise, a lower degree.

- F1score

The F1 value is an indispensable measure in classification problems, which can be regarded as a weighted average of precision and recall. The value is between 0 and 1, and the closer F1 is to 1, the higher the accuracy of the model, otherwise the lower. The specific formula is as follows:

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- Average Precision

Average Precision is the most commonly used indicator for evaluating model performance in the field of object detection, and its value ranges from 0 to 1. The calculation method is based on Detection and Ground Truth, traverses each prediction frame according to the confidence order, and determines whether the prediction frame belongs to TP or FP by whether the IoU is greater than the threshold, and then obtains the corresponding precision rate and recall rate. P-R curve is formed by plotting these two values as abscissa and ordinate as a curve. AP represents the area under the P-R curve, and the formula is as follows:

$$AP = \int_0^1 P(r)dr$$

Mean Average Precision (mAP) is the Mean Average Precision, which is the sum of aps divided by the number of classes. Because pedestrian detection is a single category of pedestrians and non-pedestrians, mAP and AP are equal in this experiment.

- FPS

Another key evaluation metric in object detection algorithms is the detection speed, which is the number of Frames Per Second (FPS) that can be detected.

6.2 Results

Figure 28 shows the PR curves of the original model and the three classical attention modules combined with SE, CBAM and CA. The corresponding FPS values are 85,

55, and 84, respectively, compared with the FPS values of the original model. It becomes smaller, which is also one of the shortcomings of introducing the attention mechanism into the model, but it basically maintains the target detection speed of the original model, and better improves the target detection accuracy of the model. The specific experimental results are shown in Table 7.

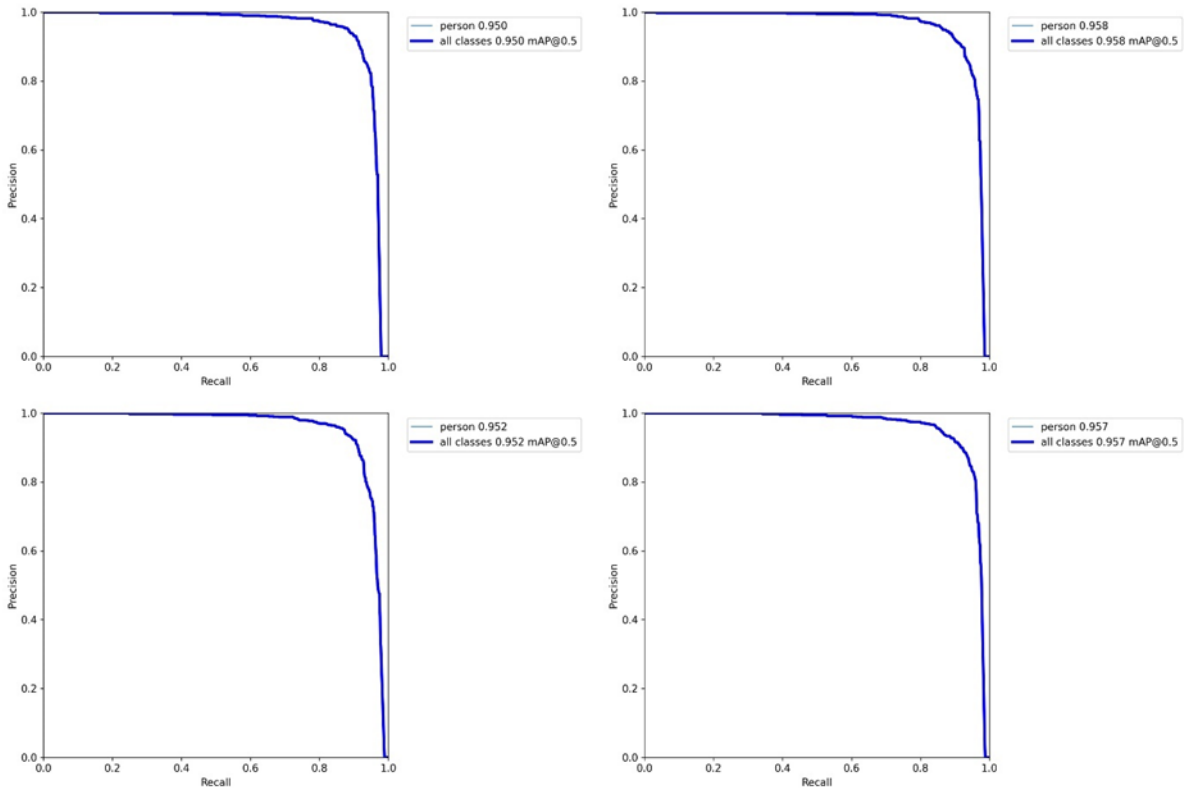


Figure 28 - PR curve graph

Table 7 - Comparison of the results of adding different attention mechanisms

Model	P/%	R/%	AP50/%	Training time/h
Base	92.9	90.7	95.0	2.951
+SE	93.9	88.9	95.8	3.004
+CBAM	92.2	90.5	95.2	3.582
+CA	91.4	90.9	95.7	2.999

Figure 29 shows the control experiment results of the original model combined with three classical attention modules, SENet, CBAM and CA. The convergence speed of the network model with the introduction of the attention mechanism is obviously slower than that of the original model, but the training accuracy of the improved network model is also higher than that of the original model. For the SE attention

module introduced in the model, the feature map obtained by convolution is subjected to Squeeze operation, Excitation operation and Scale operation to strengthen or weaken the features, thereby improving the training accuracy of the model. Compared with the original model, the accuracy is improved by 0.8 %. For the CA attention module introduced in the model, the position information is embedded into the channel attention, and the channel relationship and long-distance dependency are encoded by the precise position information, which can make the target detection network more accurately locate and identify the interested object. target object, the accuracy is improved by 0.7% compared to the original model. For the CBAM attention module introduced in the model, the effect of the SE and CA attention modules is worse. Compared with the original model, the accuracy is only increased by 0.2%, and the training time of adding the CBAM attention module will increase significantly. Through experiments, it is proved that the YOLOv5s model introducing the three classic attention modules of SE, CBAM and CA can improve the training accuracy of the model while basically maintaining the detection speed. Among them, the YOLOv5s model introducing the SE module has the highest training accuracy, compared with the original Model accuracy has improved.

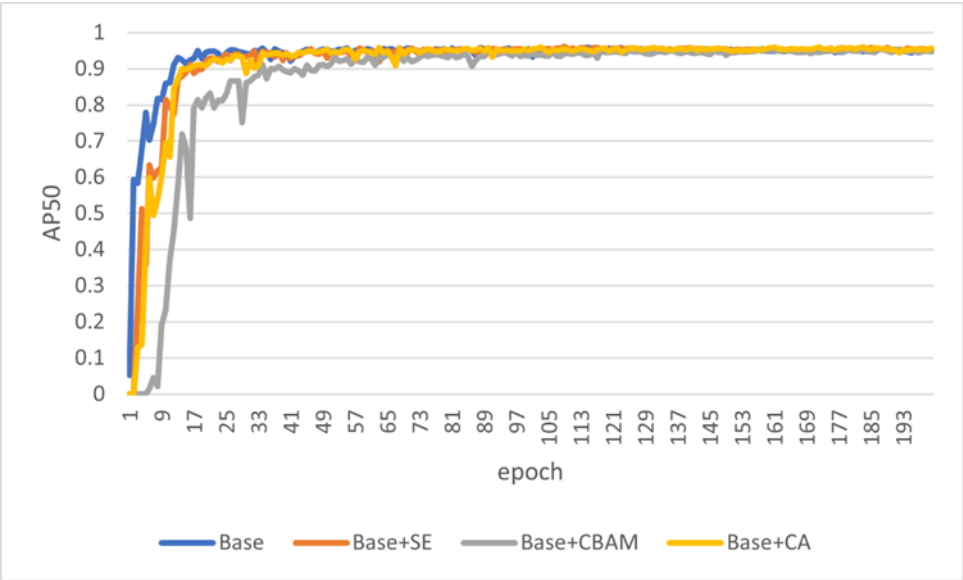


Figure 29 - AP50 experimental comparison result

Taking the YOLOv5s model introduced with the SE module as an example, the staged effects of the model before and after the improvement are shown. Figure 30 is

the pedestrian detection effect verification diagram of the model before the SE module is introduced, and Figure 31 is the pedestrian detection effect verification diagram of the model after the SE module is introduced. From the comparison of Figure 30 and Figure 31, it can be seen that after the introduction of the attention mechanism, the problem that the original model is not easy to identify each other and small target pedestrians at a distance is alleviated to a certain extent. There is no false detection of the non-pedestrian target (trash can), and it also improves the feature extraction ability of the model for the image, and the target detection accuracy has been improved to a certain extent.

Figure 30 - The pedestrian detection effect verification diagram of the model before the introduction of the SE module

Figure 31 - The pedestrian detection effect verification diagram of the model after introducing the SE module

6.3 Infrared image pedestrian detection system test evaluation

After the platform is built, the software interface designed by PyQt5 is tested and verified. Figure 32 shows the test run of the infrared image pedestrian detection system. Manually select the previously trained model with SE attention module added. After clicking the start button, the system obtains the test video in the disk, and the detection result is displayed in the video display area. It can be seen from the data result display on the left that there are 3 pedestrians in the current frame page, the system can accurately discriminate the pedestrians ahead in the video, and the system reliability is strong. The detection rate is 40 FPS, which basically meets the real-time requirements.

Figure 32 - Schematic running

Chapter 7 – Conclusions and Recommendations

7.1 Conclusions

On the basis of reading a large number of related literatures and based on the principle of deep learning, this project adds an attention module to the yolov5s model to carry out research on pedestrian detection algorithms in infrared images. The specific work content is as follows:

Create a dataset for deep learning. Aiming at the problem of the lack of a picture database suitable for pedestrians on the roads in China, the infrared pedestrian pictures collected in real scenes were manually annotated, and a dataset with 1161 pictures and more than 2000 pedestrian targets was independently produced.

In order to alleviate the problem that the model is difficult to distinguish the mutual occlusion of pedestrian targets and the missing detection of small targets, and further improve the accuracy of pedestrian detection in infrared images, a lightweight attention module is introduced into the backbone network of the YOLOv5s model. The three classical attention modules of SE, CBAM and CA are studied and controlled. Finally, the optimal SE attention module in the YOLOv5s model in this project is determined. After introducing the SE attention module, the accuracy of the pedestrian detection model is improved by 0.8% compared with the original model, and the purpose of optimizing the model is achieved.

This project also develops a visual interface based on PyQt5, and designs an infrared image pedestrian detection system. The system has high reliability and can achieve the purpose of real-time.

7.2 Recommendations for future research

Although the infrared image pedestrian detection method of this project has been experimentally verified in the infrared image dataset, the effectiveness of the method

is proved. However, the research content of this paper is only the tip of the iceberg in the field of infrared image pedestrian detection, and due to the limited personal time and energy, there are certain limitations in the practical application process.

- Limited by time, the data set produced in this project is small. If more infrared image data can be used to improve the data set and the model training effect, the accuracy of infrared image pedestrian detection can be improved.
- Although there are some improvements in the detection accuracy of the model, it still needs to be further improved and there is still a certain distance from the requirements of practical applications.
- This project algorithm based on vision sensors to collect digital image information source, the pedestrian detection average accuracy is 95.8%, but the infrared camera pixels lower defect, further improve the detection accuracy has the certain difficulty, if you can blend other sensors to collect information, such as laser radar, the pedestrian detection accuracy and reliability is expected to increase.
- This project only makes a visual interface of infrared image pedestrian detection through PyQt5, and it needs software encapsulation later, so that it can be transplanted and deployed to other platforms.

Chapter 8 – Reflections

At the beginning of this experiment, I had limited experience in Python coding and testing, so the study of the code wasted a lot of my time beyond the planned time. I think my lack of programming skills contributed to this frustration. Secondly, in the process of making the data set, I needed the support of the school when collecting the data, so I encountered some setbacks in the application, which led to the delay of the plan. This also warned me to plan and understand the application process in advance.

In this project, I would like to thank the team of YOLOv5 algorithm, the detailed instructions made it easier for me to use and improve it. Most importantly, I think the main things I gained from this project are as follows; 1) Python programming experience; 2) a deep understanding of the object detection process.

After completing this project, my understanding of dataset making and object detection has gone from a simple understanding to developing an infrared image pedestrian detection system. I gained quite a lot of knowledge from this project, but due to the limited time, there are many improvements to be made, which is my next work.

References

[1] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey, " *Neurocomputing (Amsterdam)*, vol. 300, pp. 17–33, 2018.

[2] A. Rogalski, "History of infrared detectors," *Opto-electronics review*, vol. 20, no. 3, pp. 279–308, 2012.

[3] D. Prasanna and M. Prabhakar, "An effiecient human tracking system using Haar-like and hog feature extraction," *Cluster computing*, vol. 22, no. Suppl 2, pp. 2993–3000, 2018.

[4] C. Shu, X. Ding and C. Fang, "Histogram of the oriented gradient for face recognition," in *Tsinghua Science and Technology*, vol. 16, no. 2, pp. 216-224, April 2011.

[5] G. Lowe, "Sift-the scale invariant feature transform," *Int. J*, vol. 2, no. 2, pp. 91-110, 2004.

[6] D. G. Lowe, "Object recognition from local scale-invariant features." *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee, 1999.

[7] L. Auria and R. A. Moro, "Support Vector Machines (SVM) as a Technique for Solvency Analysis," *IDEAS Working Paper Series from RePEc*, 2008.

[8] D. Olmeda, J. M. Armingol, and A. de la Escalera, "Discrete features for rapid pedestrian detection in infrared images," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3067–3072.

[9] B. Besbes, A. Rogozan, A.-M. Rus, A. Bensrhair, and A. Broggi, "Pedestrian detection in far-infrared daytime images using a hierarchical codebook of SURF," *Sensors (Basel, Switzerland)*, vol. 15, no. 4, pp. 8570–8594, 2015.

[10] X. Zhao, Z. He, S. Zhang, and D. Liang, “Robust pedestrian detection in thermal infrared imagery using a shape distribution histogram feature and modified sparse representation classification,” *Pattern recognition*, vol. 48, no. 6, pp. 1947–1960, 2015.

[11] J. Wang, T. Zhang, Y. Cheng, and N. Al-Nabhan, “Deep Learning for Object Detection: A Survey,” *Computer Systems Science and Engineering*, vol. 38, no. 2, pp.165–182, 2021.

[12] Y. Li., et al. “Multi-block SSD based on small object detection for UAV railway scene surveillance.” *Chinese Journal of Aeronautics*, vol. 33, no. 6, pp.1747-1755, 2020.

[13] S. Shinde, A. Kothari, and V. Gupta. “YOLO based human action recognition and localization.” *Procedia computer science,* vol. 133, pp.831-838, 2018.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv.org*, 2014.

[15] P. Purkait, C. Zhao, and C. Zach, “SPP-Net: Deep Absolute Pose Regression with Synthetic Views,” *arXiv.org*, 2017.

[16] R. Girshick, “Fast r-cnn,” *Proceedings of the IEEE international conference on computer vision*. 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *arXiv.org*, 2016.

[18] K. He, et al. “Mask r-cnn.” *Proceedings of the IEEE international conference on computer vision*. pp.2961-2969, 2017.

[19] H. Sheng, et al. “Context-aware Pedestrian Detection with Salient Region Self-growing in Far-infrared Images.” *International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)*. pp.72-76, 2018.

[20] Z. Cao, H. Yang, J. Zhao, X. Pan, L. Zhang, and Z. Liu, “A New Region

Proposal Network for Far-Infrared Pedestrian Detection," *IEEE access*, vol. 7, pp. 135023–135030, 2019.

[21] C. Devaguptapu, N. Akolekar, M. M. Sharma, and V. N. Balasubramanian, "Borrow from Anywhere: Pseudo Multi-modal Object Detection in Thermal Imagery," 2020.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[23] D. Heo, E. Lee, B. C. Ko, "Pedestrian detection at night using deep neural networks and saliency maps," *Journal of Imaging Science and Technology*, vol. 61, no. 6, 2017.

[24] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv.org*, 2016.

[25] R. Girshick, "R-CNN [C]. Proceedings of them the IEEE International Conference on Computer Vision," 2015, pp.1440-1448.

[26] Z. Hong, L. Zhang, and P. Wang. "Pedestrian detection based on YOLO-D network," *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS,* 2018, pp.802-806.

[27] F. Iandola, et al. "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869,* 2014.

[28] J. Redmon, and A. Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767*, 2018.

[29] W. Hsu, and W. Lin. "Ratio-and-scale-aware YOLO for pedestrian detection," *IEEE transactions on image processing*, 2020, pp.934-947.

[30] X. Yang, Y. Wang, and R. Laganière. "A scale-aware YOLO model for pedestrian detection," *International Symposium on Visual Computing*. Springer,

2020, pp.15-26.

[31] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv.org*, 2020.

[32] Yolov5 at "https://github.com/ultralytics/yolov5", Date Accessed June. 22, 2022.

[33] C. Y. Wang, H. Y. Mark Liao, et al. "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1571-1580.

[34] W. Boyuan, W. Muqing, "Check Study pedestrian detection-based on the an improved Algorithm," *Proceedings of 6th IEEE International Conference on Computer and Communications*, 2020, pp.1198-1202.

[35] Yolov6 at "https://github.com/meituan/YOLOv6", Date Accessed June. 25, 2022.

[36] Yolov7 at "https://github.com/WongKinYiu/yolov7", Date Accessed June. 25, 2022.

[37] Convolutional neural network at "https://en.wikipedia.org/wiki/Convolutional_neural_network", Date Accessed August. 5, 2022.

[38] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks," *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp.315-323.

[39] A. Maniatopoulos and N. Mitianoudis, "Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function," *Information (Basel)*, vol. 12, no. 12, p. 513–, 2021.

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of

feature detectors," *arXiv.org*, 2012.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," 2016.

[42] YOLOv5 structure at "https://www.pianshen.com/article/76631870813/", Date Accessed August. 13, 2022.

[43] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo and J. Choe, "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6022-6031.

[44] J. Hu, S. Li, and G. Sun. "Squeeze-and-excitation networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp.7132-7141.

[45] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," 2018, pp.3-19.

[46] Q. Hou, D. Zhou and J. Feng, "Coordinate Attention for Efficient Mobile Network Design," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13708-13717.

[47] PyQt5 at "https://pypi.org/project/PyQt5/", Date Accessed August. 15, 2022.

[48] M. Saunders, P. Lewis, and A. Thornhill. "Research methods forbusiness students," *Essex: Prentice Hall: Financial Times*, 2003.

[49] K. J. Sileyew, "Research design and methodology," Rijeka: IntechOpen, 2019.

[50] Inductive Reasoning at "https://baike.baidu.com/item/%E5%BD%92%E7%BA%B3%E6%8E%A8%E7%90%86/5502360?fromtitle=%E5%BD%92%E7%BA%B3%E6%B3%95&fromid=120098&fr=aladdin", Date Accessed August. 15, 2022.

Appendices

CA module code

```
    def __init__(self, inplace=True):
        super(h_sigmoid, self).__init__()
        self.relu = nn.ReLU6(inplace=inplace)
    def forward(self, x):
        return self.relu(x + 3) / 6
class h_swish(nn.Module):
    def __init__(self, inplace=True):
        super(h_swish, self).__init__()
        self.sigmoid = h_sigmoid(inplace=inplace)
    def forward(self, x):
        return x * self.sigmoid(x)

class CoordAtt(nn.Module):
    def __init__(self, inp, oup, reduction=32):
        super(CoordAtt, self).__init__()
        self.pool_h = nn.AdaptiveAvgPool2d((None, 1))
        self.pool_w = nn.AdaptiveAvgPool2d((1, None))
        mip = max(8, inp // reduction)
        self.conv1 = nn.Conv2d(inp, mip, kernel_size=1, stride=1, padding=0)
        self.bn1 = nn.BatchNorm2d(mip)
        self.act = h_swish()
        self.conv_h = nn.Conv2d(mip, oup, kernel_size=1, stride=1, padding=0)
        self.conv_w = nn.Conv2d(mip, oup, kernel_size=1, stride=1, padding=0)
    def forward(self, x):
        identity = x
        n, c, h, w = x.size()
        #c*1*W
        x_h = self.pool_h(x)
        #c*H*1
        #C*1*h
        x_w = self.pool_w(x).permute(0, 1, 3, 2)
        y = torch.cat([x_h, x_w], dim=2)
        #C*1*(h+w)
        y = self.conv1(y)
        y = self.bn1(y)
        y = self.act(y)
        x_h, x_w = torch.split(y, [h, w], dim=2)
        x_w = x_w.permute(0, 1, 3, 2)
        a_h = self.conv_h(x_h).sigmoid()
        a_w = self.conv_w(x_w).sigmoid()
        out = identity * a_w * a_h
        return out
```

Network structure after adding CA attention module

```
                 from  n    params  module                                  arguments
  0                 -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                 -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                 -1  1     18816  models.common.C3                        [64, 64, 1]
  3                 -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                 -1  1    156928  models.common.C3                        [128, 128, 3]
  5                 -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                 -1  1    625152  models.common.C3                        [256, 256, 3]
  7                 -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                 -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                 -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 10                 -1  1     25648  models.common.CoordAtt                  [512, 512]
 11                 -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 12                 -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 13            [-1, 6]  1         0  models.common.Concat                    [1]
 14                 -1  1    361984  models.common.C3                        [512, 256, 1, False]
 15                 -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 16                 -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 17            [-1, 4]  1         0  models.common.Concat                    [1]
 18                 -1  1     90880  models.common.C3                        [256, 128, 1, False]
 19                 -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 20           [-1, 15]  1         0  models.common.Concat                    [1]
 21                 -1  1    296448  models.common.C3                        [256, 256, 1, False]
 22                 -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 23           [-1, 11]  1         0  models.common.Concat                    [1]
 24                 -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 25       [18, 21, 24]  1     16182  models.yolo.Detect                      [1, [[10, 13, 16, 30, 3
Model Summary: 293 layers, 7089190 parameters, 7089190 gradients
```