EVALUACIÓN DE ALGORITMOS DE MACHINE LEARNING PARA CONDUCCIÓN

MACHINE LEARNING ALGORITHM EVALUATION ON ADVANCED DRIVER ASSISTANCE

TRABAJO FIN DE GRADO
CURSO 2020-2021

BY
WENBO SUN

DIRECTOR
CARLOS GARCÍA SÁNCHEZ
GUILLERMO BOTELLA JUAN

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

THANKS

I am very grateful to the two teachers for their help in this tfg. From the beginning, I didn't know and didn't know how to do it, not only gave me the general direction of the project, but also gave me some necessary information. In the later period, Carlos pointed me to the shortcomings of the project and encouraged me to complete the project. For this I must say that the teachers are very good. I am very grateful and grateful to them. At the same time, I am very grateful to the school for the help I learned during the epidemic.

ABSTRACT

Machine learning algorithm evaluation

on advanced driver assistance

In this research and development project, our main purpose is to study four deep learning architectures for real-time object detection of people and bicycles encountered in front of driving.

We use 4 different algorithms for the same data set, and compare the mAPs obtained after training. And discuss which method is the most accurate, but also consider the time it takes to get what is suitable for what kind of scene.

The project I came up with would like to be used in a driving assistance system. The system uses camera sensors to get input, and then uses algorithms to assist, so that the safety of the car is guaranteed when driving. At the same time, it can run on a low-performance version of the machine and compare the fps of different algorithms.

Keywords

Computer vision, autonomous driving, convolutional neural network, object detection, deep learning, machine learning, feature selection and extraction, driving assistance

INDEX OF CONTENTS

Chapter1 - Introduction

Pedestrian Detection has always been a hot and difficult point in computer vision research. The problem to be solved by pedestrian detection is to find all pedestrians in an image or video frame, including their position and size, which are generally represented by rectangular boxes, similar to face detection, which is also a typical target detection problem.

Pedestrian detection technology has a strong use value. It can be combined with pedestrian tracking, pedestrian re-identification and other technologies. It can be used in automotive unmanned driving systems (ADAS), intelligent robots, intelligent video surveillance, human behavior analysis, passenger flow statistics systems, and intelligence Transportation and other fields.

Since the human body is quite flexible, there will be various postures and shapes, and its appearance is greatly affected by wearing, posture, viewing angle, etc., and it also faces the influence of factors such as occlusion and illumination. This makes pedestrian detection a computer vision A very challenging subject. The main problems to be solved in pedestrian detection are:

The appearance is very different. Including viewing angle, posture, clothing and attachments, lighting, imaging distance, etc. Looking at the past from different angles, the appearance of pedestrians is very different. Pedestrians in different postures have very different appearances. Due to the different clothes that people wear, as well as the influence of umbrellas, hats, scarves, luggage and other attachments, the appearance is very different. The difference in lighting also caused some difficulties. The human body at a distance and the human body at a close distance are also very different in appearance.

Occlusion problem. In many application scenarios, pedestrians are very dense and there are serious occlusions. We can only see a part of the human body, which brings serious challenges to the detection algorithm.

The background is complicated. Whether indoor or outdoor, pedestrian detection generally faces very complicated backgrounds. The appearance and shape, color, and texture of some objects are very similar to human bodies, which makes the algorithm unable to distinguish accurately.

Detection speed. Pedestrian detection generally uses a complex model with a large amount of calculations. It is very difficult to achieve real-time and generally requires a lot of optimization.

Chapter1.1 - Motivation

One day in the future, new energy vehicles will eventually replace fuel vehicles, and autonomous driving will also replace the driver one day in the future.

The current level of autonomous driving is formulated by the SAE International[27], which is divided into 6 levels (L0-L5).

-Level L0: The driver is in full control of the vehicle;

-Level L1: The automatic system can sometimes assist the driver to complete certain driving tasks;

-Level L2 assisted driving: The automatic system can complete certain driving tasks, but the driver needs to monitor the driving environment and complete the rest, while ensuring that problems occur and take over at any time. At this level, the wrong perception and judgment of the automatic system can be corrected by the driver at any time, and most car companies can provide this system. L2 can be divided into different usage

scenarios based on speed and environment, such as low-speed traffic jams on the loop, fast driving on highways, and automatic parking by the driver in the car;

-Level L3 semi-autonomous driving: The automatic system can not only complete certain driving tasks, but also monitor the driving environment under certain conditions, but the driver must be ready to regain driving control (when the automatic system requests it). Therefore, at this level, the driver still cannot sleep or take a deep rest. After the completion of L2, the research field of car companies is extended from here. Due to the particularity of L3, the most meaningful deployment currently seen is to upgrade on the high-speed L2; the difference between L3 and L2 is that the vehicle is responsible for peripheral monitoring, and the human driver only needs to maintain attention for emergencies.

-Level L4 highly automated driving: Automated systems can complete driving tasks and monitor the driving environment under certain environments and specific conditions; currently, the deployment of L4 is mostly based on city use, which can be fully automated valet parking. It can also be directly combined with taxi services. At this stage, within the scope of autonomous driving, all tasks related to driving have nothing to do with the driver and passengers. The perception of external responsibility lies in the autonomous driving system, and there are different design and deployment ideas here;

-Level L5 fully automated driving: all driving tasks that the automated system can complete under all conditions.

What we currently call Autonomous Driving System (ADS) is usually at level 3 to 5. We are now between 3-4. So, for this part, it belongs to conditional automation. Safety is very important in the process of automatic driving. One of the functions that you want to accomplish through target recognition is called automatic emergency braking.

Summary:

1. How to make real-time information feedback in a permitted manner

2. How to determine the most suitable system in terms of efficiency

Chapter1.2 - Objective

I did this project to make a contribution to the safety system of auto-driving cars. So, in this project, what needs to be clearly analyzed is to identify pedestrians and bicycles in a specific scene, such as a street.

Based on modeling and machine learning methods, we can already use technology to achieve better pedestrian detection efficiency or accuracy under specific conditions. The features learned by deep learning have strong level expression ability and good feasibility, which can better solve some visual problems. Therefore, we choose to use a deep convolutional neural network to solve the problem of pedestrian and bicycle detection.

Using general object detection frameworks based on deep learning, such as SSD, YOLO, FASTER-RCNN+fpn, etc., can be directly applied to pedestrian and bicycle detection tasks. We need to use the appropriate data set, use analogies, and use these frameworks mentioned above to achieve the target detection of pedestrians and bicycles, and compare their real-time training results on the same data set.

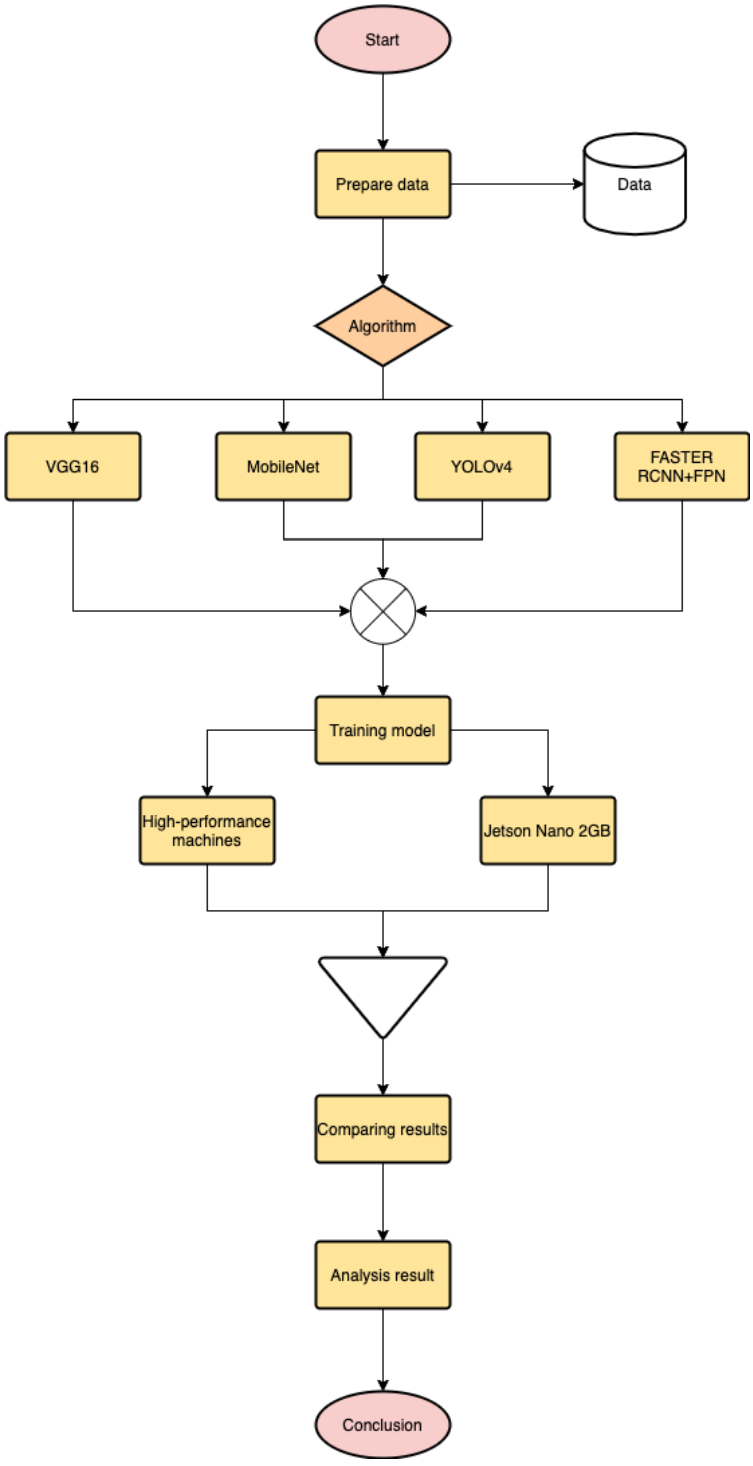
When completing this project, I hope to be able to use different algorithms to identify pedestrians and bicycles on the street. And it is possible to compare which algorithm is most in line with expectations under the conditions (hardware, scene, etc), which means that it can detect enough targets.

Chapter2 - Contribution of student

Chapter2.1 - Plan of Works

- Introduction and motivation of the development project
- Search for available data sets
- Analyze the data set and make it available
- Analysis and selection algorithm
- Read technical articles on object detection
- Read papers on 4 algorithms: vgg16_ssd, mobilev3_ssd, yolov4, faster_rcnn_r50+fpn
- Different structural systems and models
- Understand the architecture of the algorithm
- Download and install the environment to implement the algorithm
- Edit the data set so that the algorithm vgg16_ssd and mobilev3_ssd can be used
- Implement the algorithm code of vgg16_ssd and mobilev3_ssd
- Edit the data set so that the algorithm yolov4 can be used
- Implement the algorithm code of yolov4
- Edit the data set so that the algorithm faster_rcnn_r50+fpn can be used
- Implement the algorithm code of faster_rcnn_r50+fpn
- Adjust the technical parameters of each model to increase the accuracy
- Development of low-performance machine (jetson nano 2GB)
- Configure a low-performance version of the environment
- Configure mobilev3_ssd in jetson nano
- Configure vgg16_ssd in jetson nano
- Configure yolov4 in jetson nano
- Configure faster_rcnn_r50+fpn in jetson nano
- Plot the images of mAPs of various models
- Analyze the comparison of the results obtained after different algorithm training
- Develop an objective and conclusions of the work
- Communicate with the teacher and make a complete contribution to the work

Chapter2.2 - Flow Chart



Figure[2.2.1] Flow Chart

Chapter3 - Advanced and architecture

Chapter3.1 - Convolutional Neural Network, CNN

-Convolutional neural network (CNN) has been widely used in the field of computer vision in recent years due to its powerful feature extraction capabilities. In 1998, Yann LeCun et al. proposed the LeNet-5 network structure, which allows convolutional neural networks to be trained end-to-end and applied to document recognition. The LeNet-5 structure is the most classic network structure of CNN, and the later developed convolutional neural network structures are derived from this version.

Convolutional Neural Networks (CNN) is a type of Feedforward Neural Networks (Feedforward Neural Networks) that includes convolution calculations and has a deep structure. It is one of the representative algorithms of deep learning [1][2]. Convolutional neural network has the ability of representation learning, and can perform shift-invariant classification of input information according to its hierarchical structure, so it is also called "shift-invariant artificial neural network (Shift-Invariant Artificial Neural Network). Neural Networks, SIANN)" [3]

A typical CNN consists of 3 parts:

1.Convolutional layer

2.Pooling layer

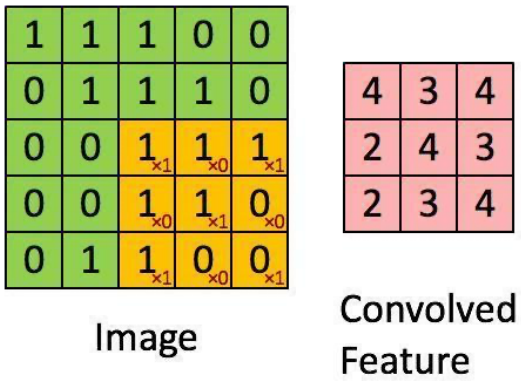
3.Fully connected layer

If you briefly describe it:

The convolutional layer is responsible for extracting local features in the image; the pooling layer is used to greatly reduce the parameter magnitude (dimensionality reduction); the fully connected layer is similar to the part of the traditional neural network to output the desired result.

1.Convolution-extracting features

The operation process of the convolutional layer is shown in the following figure, and a complete picture is scanned with a convolution kernel:

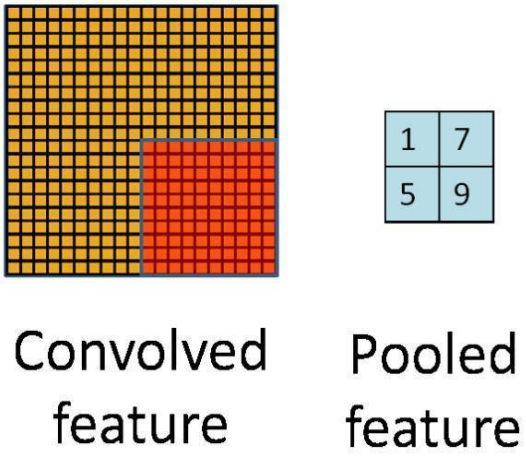


Figure[3.1.1] Convolutional layer extraction features

This process can be understood as we use a filter (convolution kernel) to filter each small area of the image, so as to obtain the feature value of these small areas.

2.Pooling layer (downsampling)-data dimensionality reduction to avoid overfitting

The pooling layer is simply downsampling, which can greatly reduce the dimensionality of the data. The process is as follows:



Figure[3.1.2] Pooling layer data dimensionality reduction

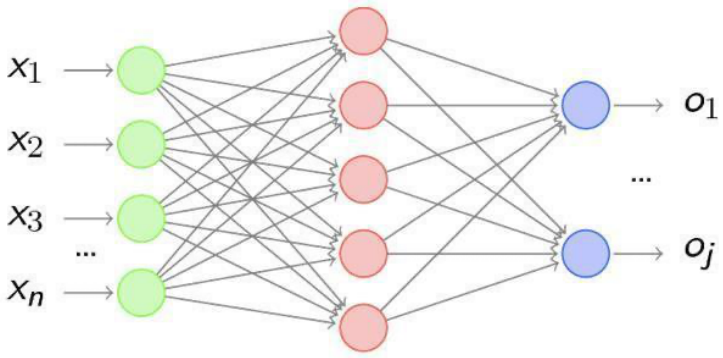
In the above figure, we can see that the original picture is 20×20, we downsample it, the sampling window is 10×10, and finally it is downsampled into a 2×2 feature map.

The reason for this is that even after the convolution is done, the image is still very large (because the convolution kernel is relatively small), so in order to reduce the data dimension, downsampling is performed.

3.Fully connected layer-output results

This part is the last step. The data processed by the convolutional layer and the pooling layer is input to the fully connected layer to obtain the final desired result.

After the data reduced by the convolutional layer and the pooling layer, the fully connected layer can "run", otherwise the amount of data is too large, the calculation cost is high, and the efficiency is low.



Figure[3.1.3] Fully connected layer output result

Chapter3.2 - Target detection network

-Commonly used detection can be roughly divided into two categories: two stage and one stage. Two stage means that the detection network is divided into two steps. The first step is to frame the object, and the second step is to determine the classification of the object. One stage is to directly use the regression network to get its classification and detection frame based on the extracted features. Among them, Faster R-CNN belongs to two stage, SSD and YOLO belong to one stage.

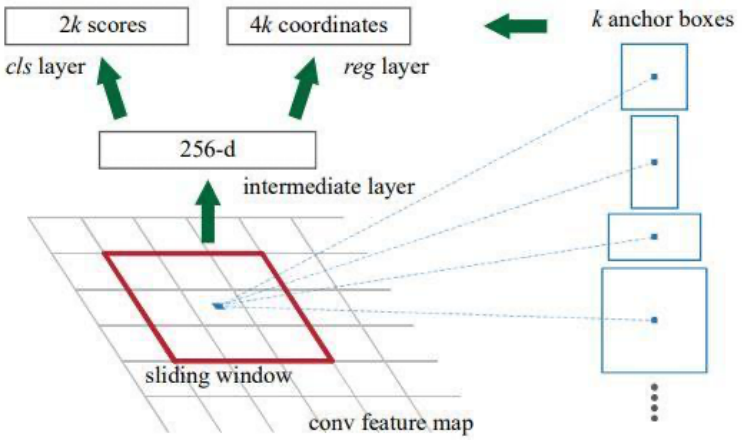
-The advantage of one stage is fast, and the advantage of two stage is accuracy. After getting the detection frame, it is much easier to classify this task than directly returning to the network to get the classification.

Chapter3.2.1 - Two stage

1. Faster-RCNN[4]

The Faster-RCNN model introduces RPN (Region Proposal Network) to directly generate candidate regions. Faster-RCNN can be seen as a combination of RPN and Fast RCNN models, that is, Faster-RCNN = RPN + Fast-RCNN.

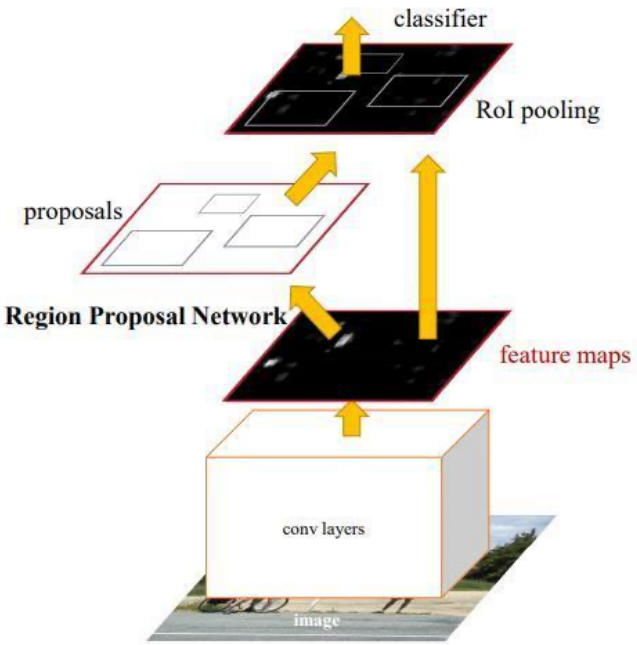
For the RPN network, a CNN model (generally called a feature extractor) is first used to receive the entire picture and extract the feature map. Then an N×N (3×3 in the text) sliding window is used on this feature map, and a low-dimensional feature (such as 256-d) is mapped to each sliding window position. Then this feature is sent to two fully connected layers, one for classification prediction and the other for regression. Generally, k a priori boxes (anchors, default bounding boxes) of different sizes or proportions are set for each window position, which means that k candidate regions (region proposals) are predicted for each position. For the classification layer, its output size is 2k, which means that each candidate area contains the object or background probability value, while the regression layer outputs 4k coordinate values, which indicate the position of each candidate area (relative to each a priori box). For each sliding window position, these two fully connected layers are shared. Therefore, RPN can be implemented using convolutional layers: first, an n×n convolution to obtain low-dimensional features, and then two 1×1 convolutions, which are used for classification and regression respectively.



Figure[3.2.1.1] CNN model (generally called a feature extractor)

RPN uses two classifications, only distinguishing background and objects, but does not predict the category of the object, that is, class-agnostic. Since the coordinate values need to be predicted at the same time, during training, the a priori box must be matched with the ground-truth box. The principle is: (1) the a priori box with the highest

IoU of a certain ground-truth box; (2) and A ground-truth box with an IoU value greater than 0.7 a priori box, as long as one is satisfied, the a priori box can match a ground-truth, so that the a priori box is a positive sample (belonging to the object), and the ground-truth For the return goal. For those a priori boxes whose IoU value with any ground-truth box is lower than 0.3, they are considered as negative samples. The RPN network can be trained separately, and the separately trained RPN model gives many region proposals. Due to the large number of a priori boxes, many of the candidate regions predicted by RPN overlap. We must first perform NMS (non-maximum suppression, IoU threshold is set to 0.7) to reduce the number of candidate regions, and then arrange them in descending order of confidence, select top -N region proposals are used to train the Fast R-CNN model. The role of RPN is to replace the role of Selective search, but it is faster, so Faster R-CNN can accelerate both training and prediction.



Figure[3.2.1.2] RPN structure

Faster-RCNN follows the following training process:

Step 1: Use the pre-trained model on ImageNe to initialize the feature extraction network and train the RPN network;

Step 2: Use the model pre-trained on ImageNet to initialize the Fast-RCNN feature extraction network, and use the candidate box generated by the trained RPN network in step one as input to train a Fast-RCNN network. So far, each of the two networks The parameters of the first layer are not shared at all;

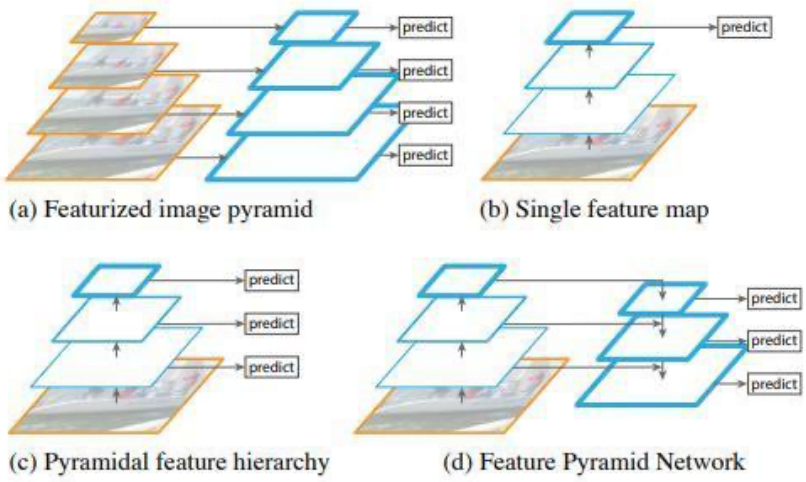
Step 3: Use the Fast-RCNN network parameters of step two to initialize a new RPN network, but set the learning rate of the feature extraction network parameters shared by RPN and Fast-RCNN to 0, even if the unique parameters of the RPN network are learned, it is fixed Feature extraction network. At this point, the two networks have shared all common convolutional layers;

Step 4: For those network layers that are still shared, add Fast-RCNN-specific network layers, continue training, and fine-tune Fast-RCNN-specific network layers. Up to this point, RPN and Fast-RCNN networks completely share parameters. Use Fast-RCNN to complete candidate frame extraction and target detection functions at the same time.

2. Fpn[5]

In the past faster rcnn for target detection, no matter it is rpn or fast rcnn, roi acts on the last layer. This is no problem for the detection of large targets, but there are some problems for detection of small targets. Because for small targets, when convolutional pooling is performed to the last layer, the semantic information is actually gone, because we all know that the method for mapping a roi to a feature map is to directly divide the

underlying coordinates by stride, Obviously, the later, the smaller the map will be, and it may even disappear. Therefore, in order to solve the problem of multi-scale detection, a feature pyramid network is introduced.



Figure[3.2.1.3] Multi-scale pyramid

Figure[3.2.1.3] (a) is a fairly common multi-scale method called a feature image pyramid. This method was widely used in the earlier artificial design features (DPM), and it was also used in CNN. It is to multi-scale the input image by setting different zoom ratios. This can solve multiple scales, but it is equivalent to training multiple models (assuming a fixed input size is required). Even if the input size is not allowed, it also increases the memory space for storing images of different scales.

Figure[3.2.1.3] (b) is CNN. Compared with artificially designed features, cnn can learn more advanced semantic features by itself. At the same time, CNN is robust to scale changes. Therefore, as shown in the figure, the features calculated from the input of a single scale can also be used to identify , But when encountering obvious multi-scale target detection, the pyramid structure is still needed to further improve the accuracy.

Judging from some of the leading methods on the imageNet and COCO data sets, the feature image pyramid method is used in the test, that is, the combination of Figure[3.2.1.3] (a) and Figure[3.2.1.3] (b). Explains that the advantage of each level of the characterized image pyramid is that it produces a multi-scale feature representation,

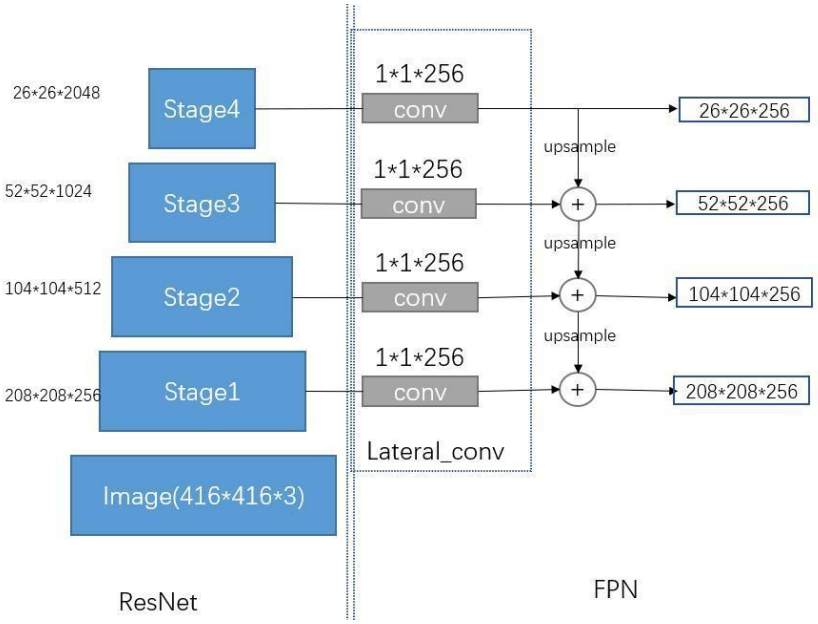
and the features of each level have strong semantics (because all features generated by cnn), including the high-resolution level (The largest scale input image).

However, this mode has obvious drawbacks. Compared with the original method, the time has increased by 4 times, and it is difficult to use in real-time applications. Similarly, it also increases the storage cost, which is why the image pyramid is only used in the testing phase. But if it is only used in the testing phase, then training and testing will be inconsistent in inference. Therefore, some recent methods have simply abandoned the image pyramid.

Figure[3.2.1.3] (c), SSD tried to use CNN pyramid-shaped hierarchical features earlier. Ideally, the SSD-style pyramid reuses the multi-scale feature maps from multiple layers calculated by the forward process, so this form does not consume additional resources. However, in order to avoid the use of low-level features, SSD abandoned the shallow feature map, but started building pyramids from conv4_3, and added some new layers. Therefore, SSD has given up on reusing higher-resolution feature maps, but these feature maps are very important for detecting small targets. This is the difference between SSD and FPN.

Figure[3.2.1.3] (4) is the structure of FPN. FPN is a pyramid form for natural use of CNN hierarchical features, while generating feature pyramids with strong semantic information at all scales. Therefore, the structure of FPN is designed with a top-down structure and a horizontal connection to integrate a shallow layer with high resolution and a deep layer with rich semantic information. In this way, it is possible to quickly build a feature pyramid with strong semantic information on all scales from a single input image at a single scale, without incurring significant costs.

FPN is not a complete target detection network, but a feature pyramid network. The FPN mentioned is actually a feature pyramid extraction feature. Therefore, the idea of Faster RCNN plus FPN essentially changes the feature extraction part because there are more feature layers. So ROIpooling has also increased.



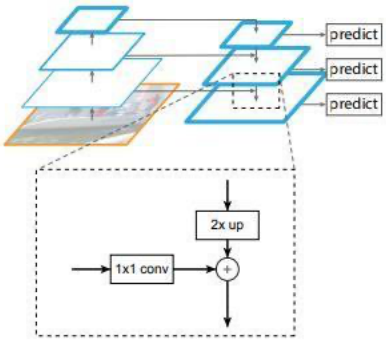
Figure[3.2.1.4] faster_rcnn+fpn structure

① The backbone network generates four-scale feature maps, and then sequentially passes through their respective Lateral_conv to make their channel numbers consistent.

②The deep feature maps are down-sampled to adjacent layer scales step by step, and then the two are added and fused.

③Output the feature map after fusion

The low-level feature semantic information is relatively small, but the target location is accurate; the high-level feature semantic information is richer, but the target location is relatively rough. Fpn is independently predicted in different feature maps.



Figure[3.2.1.5] Independent prediction in different feature maps

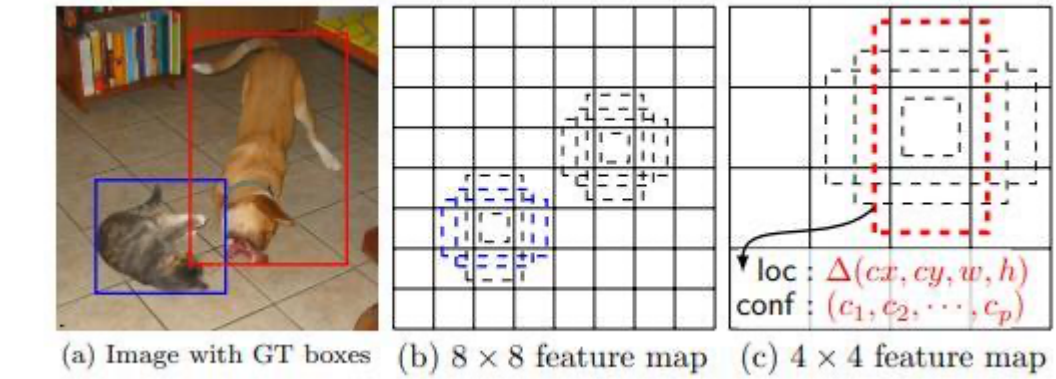
Chapter3.2.2 - One stage

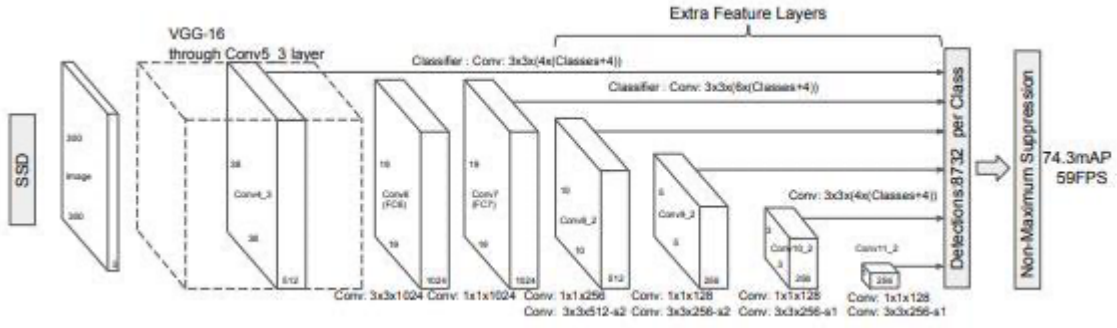
1. SSD[6]

SSD uses the idea of meshing, and unlike Faster RCNN, it integrates all operations into a convolutional network. In order to detect targets of different scales, SSD performs sliding window scanning on the feature images of different convolutional layers; small targets are detected in the feature images output by the previous convolutional layer, and large targets are detected in the feature images output by the subsequent convolutional layer. The goal. Its main features are:

1.1. Detection based on multi-scale feature images: Prediction on multi-scale convolution feature maps to detect targets of different sizes, which improves the detection accuracy of small target objects to a certain extent.

1.2. Drawing on the idea of Anchor boxes in Faster R-CNN, sampling candidate regions on feature maps of different scales, which improves the recall rate of detection and the detection effect of small targets to a certain extent. The following figure shows the principle of SSD:





Figure[3.2.2.1] Schematic of SSD

2. *VGG16[8]*

VGG was proposed in 2014 by the Visual Geometry Group, Department of Science and Engineering, Oxford University. The main work is to prove that increasing the depth of the network can affect the final performance of the network to a certain extent. VGG has two structures, namely VGG16 and VGG19. Except for the difference in network depth, there is no difference in essence between the two. Compared with AlexNet in 2012, a high advance of VGG is to use continuous 3x3 small convolution kernels to replace the larger ones in AlexNet (AlexNet uses 11x11, 7x7 and 5x5 convolution kernels). The superposition of two 3x3 convolution kernels with a step length of 1, its receptive field is equivalent to a 5x5 convolution kernel. However, the use of stacked small convolution kernels is due to the large convolution kernel, because the increase in the number of layers

increases the nonlinearity of the network, which allows the network to learn more complex models, and the small convolution kernel has fewer parameters.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure[3.2.2.2] VGG16 model

VGG16 contains:

13 convolutional layers (Convolutional Layer), respectively represented by conv3-XXX

3 fully connected layers (Fully connected Layer), respectively represented by FC-XXXX

5 pooling layers (Pool layer), respectively represented by maxpool

The outstanding feature of VGG16 is simplicity, which is reflected in:

3. The convolutional layers all use the same convolution kernel parameters

The convolutional layers are all expressed as conv3-XXX, where conv3 indicates that the kernel size of the convolutional layer used by the convolutional layer is 3, that is, the width and height are 3, and 3*3 is very Small convolution kernel size, combined with other parameters (stride=1, padding=same), so that each convolutional layer (tensor) can maintain the same width and width as the

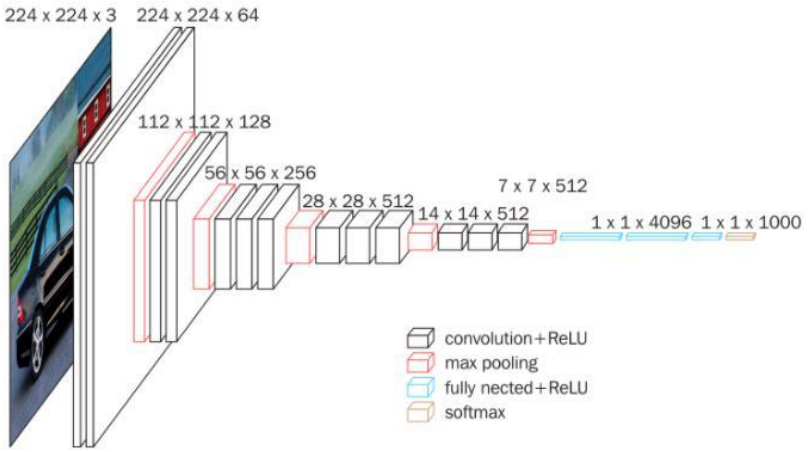
previous layer (tensor) high. XXX represents the number of channels of the convolutional layer.

3. Pooling layers all use the same pooling core parameters

The parameters of the pooling layer are all 2×2, stride=2, max pooling method, so that the width and height of each pooling layer (tensor) can be 12 of the previous layer (tensor) .

3. The model is composed of a number of convolutional layers and pooling layers stacked (stack), which is easier to form a deeper network structure (in 2014, 16 layers were considered to be very deep).

Based on the above analysis, the advantages of VGG can be summarized as: Small filters, Deeper networks



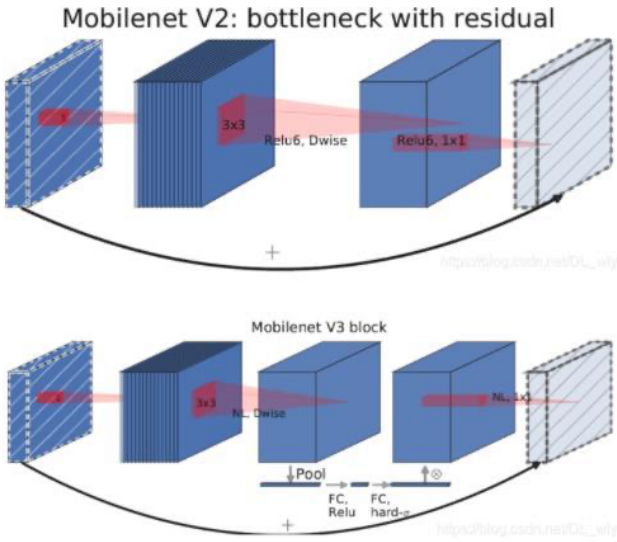
Figure[3.2.2.3] VGG16 structure

3. *MobileNetV3[9]*

MobileNetV3 has two major innovations

(1) Complementary search technology combination: resource-constrained NAS performs module-level search, and NetAdapt performs local search.

(2) Network structure improvement: The average pooling layer of the last step is moved forward and the last convolutional layer is removed, and the h-swish activation function is introduced.



Figure[3.2.3.1] mobilenetV3 network block structure

The above two pictures are the network block structure of MobileNetV2 and MobileNetV3. It can be seen that MobileNetV3 combines the ideas of the following three models: MobileNetV1's depthwise separable convolutions, MobileNetV2's inverted residual with linear bottleneck (the inverted residual with linear bottleneck) and MnasNet based Lightweight attention model of squeeze and excitation structure.

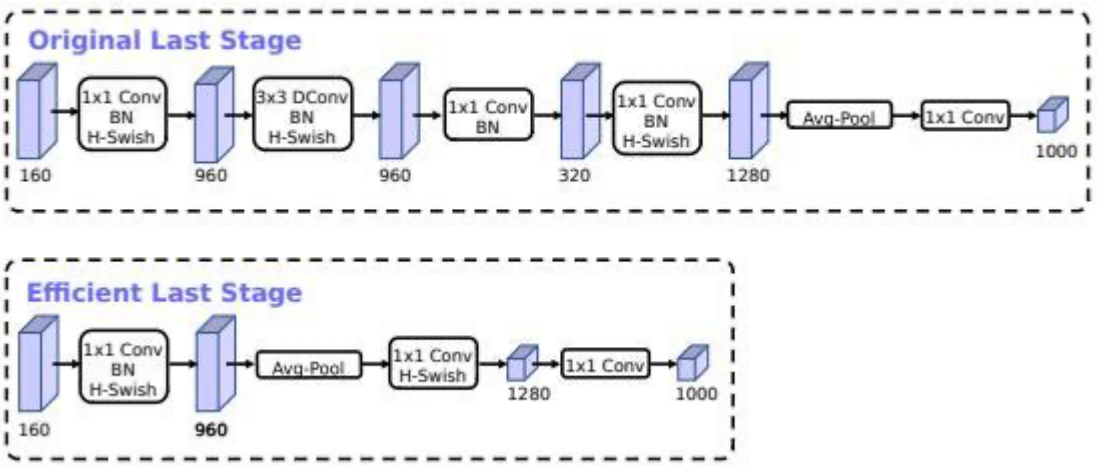
- Complementary search technology portfolio

(1) Resource-constrained NAS (platform-aware NAS): Search for various modules of the network under the premise of limited calculation and parameter amount, so it is called Block-wise Search.

(2) NetAdapt: used to fine-tune the network layer after each module is determined.

For the exploration and optimization of model structure, web search is a powerful tool. The researchers first used the neural network search function to build a global network structure, and then used the NetAdapt algorithm to optimize the number of cores in each layer. For the global network structure search, the researchers used the same RNN-based controller and hierarchical search space as in Mnasnet, and optimized the precision-delay balance for a specific hardware platform. The target delay (~80ms) Search within the range. Then use the NetAdapt method to tune each layer in a sequential manner. While maintaining accuracy while optimizing model delay as much as possible, reduce the size of the bottleneck in the expansion layer and each layer.

- Improvement of network structure



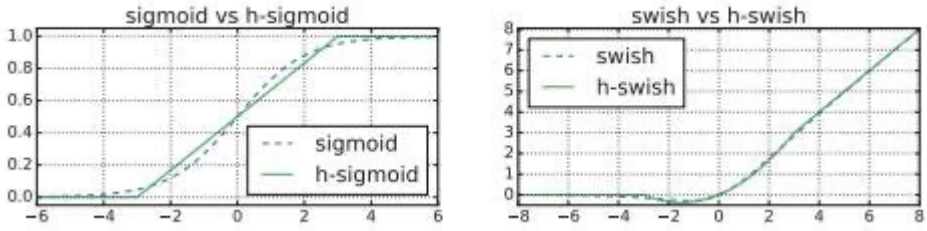
Figure[3.2.3.2] Mobilenet network structure improvement

In the MobileNetV2 model, the inverted residual structure and variables use 1*1 convolution to construct the final layer, so as to expand to the high-dimensional feature space. Although it is very important to extract rich features for prediction, it introduces the calculation Overhead and delay. In order to reduce the delay while preserving high-dimensional features, the layer before average pooling is removed and 1*1 convolution is used to calculate the feature map. After the feature generation layer is removed, the layer previously used for bottleneck mapping is no longer needed. This will reduce the

overhead of 10ms and reduce the number of operations by 30m while increasing the speed by 15%.

The swish activation function can effectively improve the accuracy of the network. However, swish is too computationally expensive. The author proposes h-swish (hard version of swish)

$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$



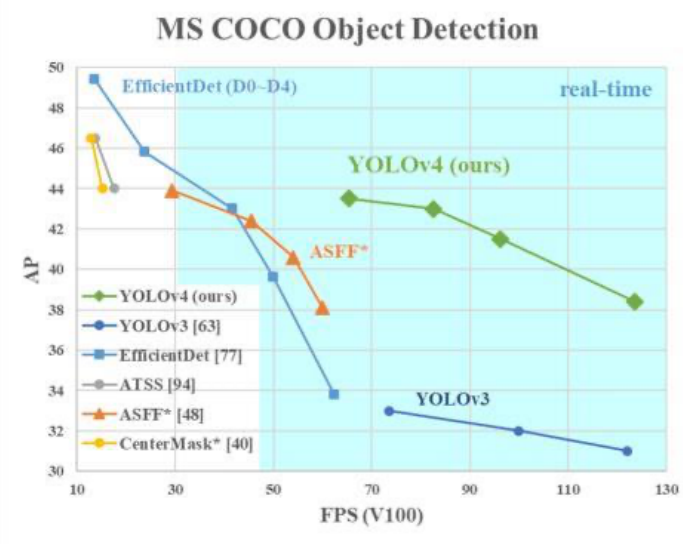
Figure[3.2.3.3] hwish formula and comparison

This nonlinearity brings many advantages while maintaining accuracy. First, ReLU6 can be implemented in many software and hardware frameworks, and secondly, it avoids the loss of numerical accuracy during quantization and runs fast. This non-linear change increases the delay of the model by 15%. However, the network effect it brings has a positive boost to accuracy and delay, and the remaining overhead can be eliminated by fusing the nonlinearity with the previous layer.

4. YOLOv4[7]

YOLOv4 has made improvements in all parts of YOLOv3 to ensure the speed, while greatly improving the detection accuracy of the model and reducing the requirements for the use of hardware. From the figure below, you can see that YOLOv4 obtained 43.5% AP value (65.7% AP50) on the MS COCO data set. With the same performance as

EfficientDet, YOLOv4 is twice as fast as EfficientDet,compared with YOLOv3, YOLOv4's AP and FPS are increased by 10% and 12%.



Figure[3.2.4.1] yolov4 algorithm comparison

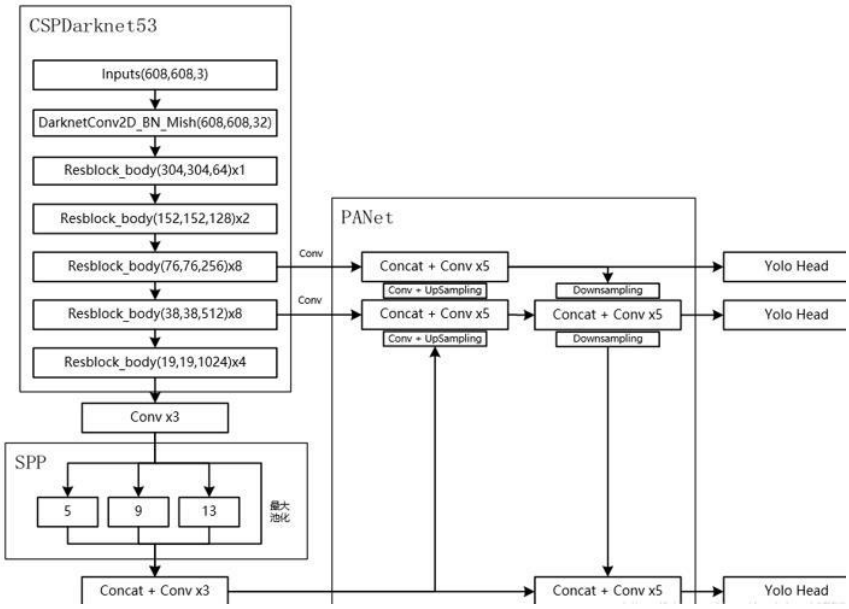
- Network structure

Backbone：CSPDarknet53

Neck：SPP，PAN

Head：YOLOv3

YOLOv4 = CSPDarknet53+SPP+PAN+YOLOv3



Figure[3.2.4.2] yolov4 network architecture

(1)CSPDarknet53

The full name of CSPNet[10] is Cross Stage Partial Network, which was proposed by Chien-Yao Wang and others in 2019 to solve the problem of the previous network structure that required a lot of inference calculations. The author attributed the problem to repeated gradient information in network optimization. CSPNet has good test results on ImageNet dataset and MS COCO dataset. At the same time, it is easy to implement and can be used in ResNet, ResNeXt and DenseNet network structures.

The main purpose of CSPNet is to be able to achieve a richer combination of gradients while reducing the amount of calculation. This goal is achieved by dividing the feature map of the basic layer into two parts, and then merging them through a cross-stage hierarchical structure.

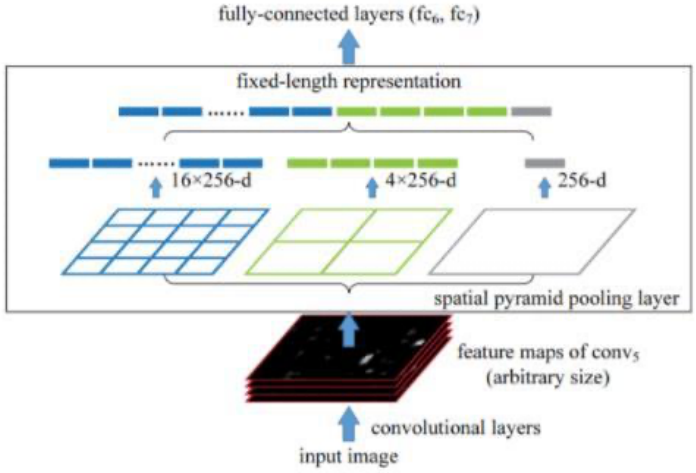
In YOLOv4, the original Darknet53 structure was replaced with CSPDarknet53, which made two main changes on the original basis:

a)Combine the original Darknet53 with CSPNet. YOLOv3 is composed of a series of residual structures. After the combination, the main job of CSPnet is to split the stack of the original residual block, and split it into two parts: the main part continues to stack the original residual block, and the branch part is equivalent to a residual edge. After a small amount of processing, it is directly connected to the end.

b)Use the MIsh activation function to replace the original Leaky ReLU. In YOLOv3, each convolutional layer includes a batch normalization layer and a Leaky ReLU. In the backbone network CSPDarknet53 of YOLOv4, Mish is used instead of the original Leaky ReLU.

(2)SPP[11]

The original design purpose of SPP is to make the convolutional neural network not restricted by the fixed input size. In YOLOv4, the author introduced SPP because it significantly increases the receptive field, isolates the most important context features, and hardly reduces the running speed of YOLOv4. As shown in the figure below, it is the classic spatial pyramid pooling layer in SPP.

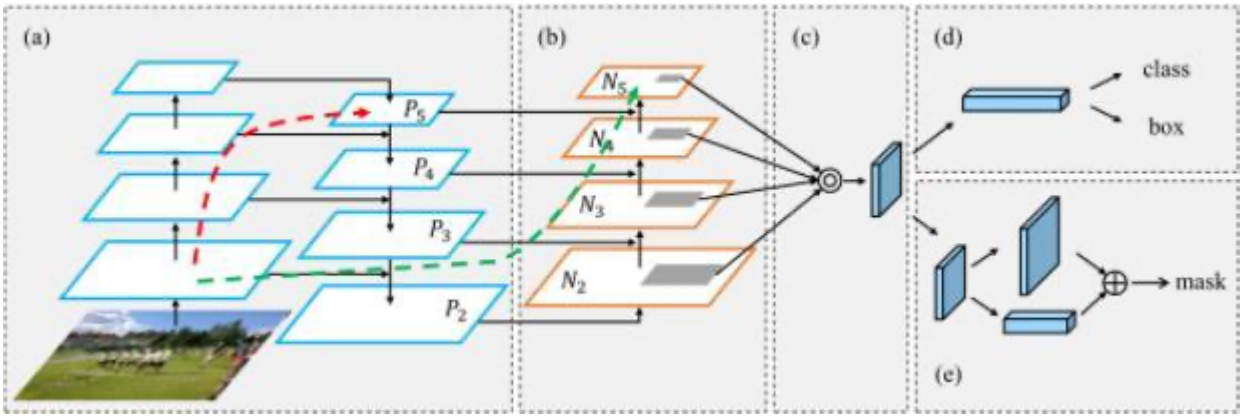


Figure[3.2.4.3] spp space pyramid pooling layer

In YOLOv4, the specific method is to use the maximum pooling of four different scales to process the feature map output by the upper layer. The maximum pooling core size is 13x13, 9x9, 5x5, 1x1, and 1x1 is equivalent to not processing.

(3)PANet[12]

PANet as a whole can be seen as a number of improvements on Mask R-CNN, making full use of feature fusion, such as introducing Bottom-up path augmentation structure, making full use of network shallow features for segmentation; introducing Adaptive feature pooling to make the extracted ROI Features are more abundant; Fully-conneFcted fusion is introduced, and a more accurate segmentation result is obtained by fusing the output of a front-background binary classification branch.



Figure[3.2.4.4] PANet schematic

(a)FPN backbone,(b)Bottom-up path augmentation,(c) Adaptive feature pooling,(d)Box branch,(e)Fully-connected fusion

In YOLOv4, the author uses PANet instead of FPN in YOLOv3 as the method of parameter aggregation, and performs parameter aggregation from different backbone layers for different detector levels. And modified the original PANet method, using tensor connection (concat) instead of the original shortcut connection (shortcut connection).

(4) YOLOv3 Head

In YOLOv4, the Head that inherits YOLOv3 performs multi-scale prediction, which improves the detection performance of targets of different sizes.YOLOv4 learns the

YOLOv3 method, uses three different levels of feature maps for fusion, and inherits the Head of YOLOv3.

(5)Tricks

To obtain better accuracy without increasing the inference cost, but only change the training strategy or only increase the training cost method, the author calls it "Bag of freebies"; it only increases a small amount of inference cost but can significantly improve the accuracy of target detection Plug-in modules and post-processing methods, called "Bag of specials"

- Bag of freebies

Random zoom Flip, rotate Image disturbance, noise, occlusion Change brightness, contrast, saturation, random erase(Cutout MixUp CutMix)

Common regularization methods are: DropOut DropConnect DropBlock

The methods to balance positive and negative samples are: Focal loss OHEM (online hard-to-separate sample mining)

In addition, there are improvements in return loss: GIOU DIOU CIOU

- Bag of specials

Increase the receptive field skills: SPP ASPP RFB

Attention mechanism: Squeeze-and-Excitation (SE) Spatial Attention Module (SAM)

Feature fusion integration: FPN SFAM ASFF BiFPN (from the famous EfficientDet)

Better activation function: ReLU LReLU PReLU ReLU6 SELU Swish hard-Swish

Post-processing non-maximum suppression algorithm: soft-NMS DIoU NMS

(6)Ways to improve

In addition to the various Tricks mentioned above, in order to make the target detector easier to train on a single GPU, the author also proposes 5 improved methods:

- Mosaic

This is a new data enhancement method proposed by the author, which draws on the idea of CutMix data enhancement method. CutMix data enhancement method uses two pictures for stitching, but Mosaic uses four pictures for stitching.

- SAT

SAT is a self-adversarial training data enhancement method, which is a new adversarial training method. In the first stage, the neural network changes the original image without changing the network weights. In this way, the neural network conducts an adversarial attack on itself, changing the original image to create a deception that there is no desired object on the image. In the second stage, the normal method is used to train the neural network to detect the target.

- CmBN

The full name of CmBN is Cross mini-Batch Normalization, which is defined as Cross Mini-Batch Normalization (CmBN). CmBN is an improved version of CBN, which is used to collect statistical data in multiple mini-batches in a batch.

- SAM

The author modified the original SAM (Spatial Attention Module) method, changing SAM from spatial attention to point attention. As shown in the figure below, for the conventional SAM, the maximum pooling layer and the average pooling layer act on the input feature maps respectively to obtain two sets of feature maps with the same shape, and then input the results into a convolutional layer, followed by Sigmoid function to create spatial attention.

- PAN

The author modified the original PAN (Path Aggregation Network) method, using tensor connection (concat) instead of the original shortcut connection (shortcut connection)

Chapter4 - Data Set

Chapter4.1 - Tshingua-Daimler[13]

Tsinghua-Daimler Cyclist Detection contains 9741 images with annotations only for "cyclist". Only cyclists which are fully visible (occlusion<10%) and higher than 60 pixels have been labeled here.

Chapter4.2 - INRIA Person Dataset[14]

This dataset has cropped the image of pedestrians so that the person can be more prominent, and the posture is not particularly strange. Only a few pictures are obtained from Google.

Chapter4.3 - Created data set

I did not select pedestrians from Tsinghua's data set, because the size of pedestrian images in Tsinghua's data set is 64x64, which is too small for experiments, so I did not choose. For the KITTI data set, when I got their data set, I found that it was all cars, so I judged that it did not meet my topic.

Finally.

Cyclist data set adopts Tsinghua-Daimler Cyclist Detection part of the data set ,1507 image resolution 2048*1024.

The Person data set uses the 614 training set and 288 test set of the INRIAPerson pedestrian data set.

A total of 2088 sheets of all data sets are divided into training set (1671) and test set (417) at a ratio of 8:2.

All the pictures used need to use the .jpg format. The created data set uses codes to distinguish the training set and the validation set, and then uses the code to convert the VOC format.

They are

1.Annotations (.xml, each picture corresponds to a file, which stores the name of the picture, the long section, the upper left and lower right coordinates of the target frame, the category name of the target frame)

2.ImagesSet(.txt, save The name of the image that needs to be trained)

3.JPEGImage (the original image)

Chapter5 - Hardware

The hardware I used to complete this project. Knowing this is necessary, because different hardware will affect the training process, especially the training time, or efficiency.

Chapter5.1 - Intel(R) Core(TM) i7-9700F CPU

- Cores: 8
- Threads: 8
- Processor Base Frequency: 3.00GHz
- Max Turbo Frequency: 4.7GHz
- TDP: 65W

Chapter5.2 - GEFORCE RTX 2080 Ti

- NVIDIA CUDA ® Cores: 4352
- Boost Clock (MHz): 1545
- Base Clock (MHz): 1350
- Standard Memory Config: 11 GB GDDR6
- Graphics Card Power (W): 250W

Chapter5.3 - Jetson Nano 2GB Developer Kit[28]

- CPU Quad-core ARM® A57 @ 1.43 GHz
- GPU 128-core NVIDIA Maxwell™
- MEMORY 2 GB 64-bit LPDDR4 25.6 GB/s

Since its release in 2019, NVIDIA Jetson Nano has created a frenzy in the field of AIOT edge computing applications worldwide, He was also named "Best AI Processor" in the 2020 Best Visual Products award list by Edge AI and Vision Alliance. This is mainly because the NVIDIA Jetson Nano chip fully meets the six challenges of AIOT chip: high performance, small size, low power consumption, full interface, ecological integrity, and excellent cost.

The reason for using this edge device was to test various models on a low-power machine. For comparison, test the FPS on a low-performance machine.

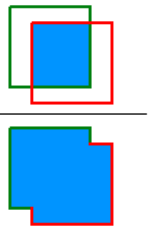
Chapter6 - Target detection evaluation

The classification task will be judged by the precision/recall curve. The principal quantitative measure used will be the average precision (AP).[15] There are some important things to know about calculating AP.[16]

Chapter6.1 - Intersection Over Union (IOU)

$$\text{IOU} = \frac{\text{area}\,(B_p \cap B_{gt})}{\text{area}\,(B_p \cup B_{gt})}$$

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



Figure[6.1.1] IOU formula

To put it simply, we require that the IOU of the two boxes is divided by the overlapping part of the two boxes by the union of the two boxes. It is clear from the second formula above. Pay attention to the two boxes when calculating. The area of the union is equal to the sum of the areas of the two minus the area of the intersection of the two.

Chapter6.2 - True Positive, False Positive, False Negative and True Negative

True Positive (TP): A correct positioning result is that the IOU between your predicted box and our groundtruth can be greater than our specified threshold, we generally take this threshold to be 0.5

False Positive (FP): It is a wrong result, that is, the IOU of the box and groundtruth you predicted is less than the threshold

False Negative (FN): We originally had an object here, so there should be a box in our place, but you did not predict it, then this groundtruth is a FN for your model

True Negative (TN): This is that our groundtruth has found a prediction box whose IOU is greater than the threshold, then the ground is considered to be successfully detected

Chapter6.3 - Precision

Precision is the ability of a model to identify only the relevant objects. It is the percentage of correct positive predictions and is given by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}$$

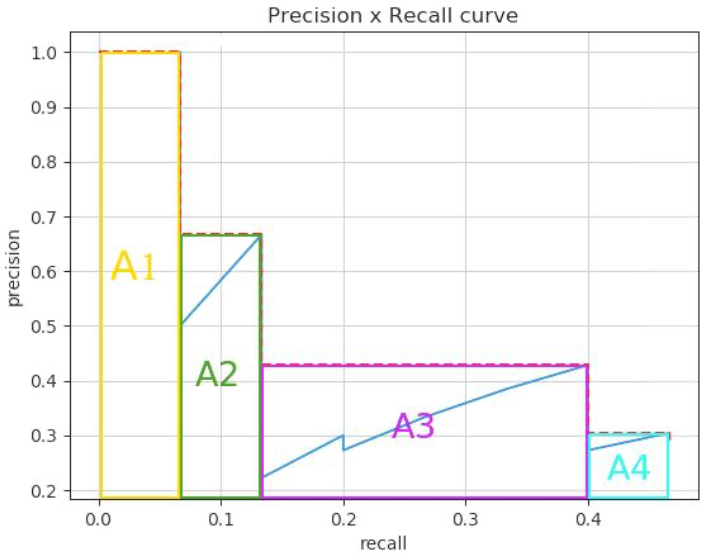
Chapter6.4 - Recall

Recall is the ability of a model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths and is given by:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}$$

Chapter6.5 - AP calculation

(1) Before VOC2010, you only need to select the maximum Precision when Recall >= 0, 0.1, 0.2, ..., 1 total 11 points, and then AP is the average of these 11 Precision.



Figure[6.5.1] AP calculation

Calculating the total area, we have the AP.

(2) In VOC2010 and later, for each different Recall value (including 0 and 1), select the maximum Precision when it is greater than or equal to these Recall values, and then calculate the area under the PR curve as the AP value. Finally the average AP value of all classes is mAP.

Chapter7 - Frameworks

Chapter7.1 - Cuda

CUDA (an acronym for Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by Nvidia.[17]It allows software developers and software engineers to use a CUDA-enabled graphics

processing unit (GPU) for general purpose processing – an approach termed GPGPU (general-purpose computing on graphics processing units).

Figure[7.1.1] Nividia cuda icon

The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.[18]

In my project, the version I use is 10.2.89

Chapter7.2 - Pytorch

PyTorch is an open source machine learning library based on the Torch library,[19][20][21] used for applications such as computer vision and natural language processing,[6] primarily developed by Facebook's AI Research lab (FAIR).[22][23][24] It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.[25]

Figure[7.2.1] PyTorch icon

In the framework of deep learning, the public now has two choices, generally TensorFlow and pytorch. TensorFlow is a static neural network, and pytorch is a dynamic neural

network, which means that I can change the structure of the network and adjust its own structure and parameters with zero delay. And the design idea of pytorch is linear, intuitive and easy to use (python first). For my project, GPU can also be supported, which is the best choice in my opinion.

In my project, the version I use is 1.8.1

Chapter7.3 - MMCV

MMCV is a foundational library for computer vision research.[26]

Figure[7.3.1] Nividia cuda icon

I installed mmcv-full, which has all the functions of mmcv, specifically for gpu

Chapter8 - Results of the experiment

For the experimental results, I will use the aforementioned mAP to record the mAP image drawn in each epoch. For the setting of each parameter:

Divide the complete data set into 12 (epoch=12)

The number of samples selected for one training is equal to 8 (batch size=8)

The initial learning rate is set to 1e-3 (LR=1e-3)

[80000,100000] each down 10 times. Means Iterate 80,000 times, become 0.0001, 100,000 times and then drop ten times

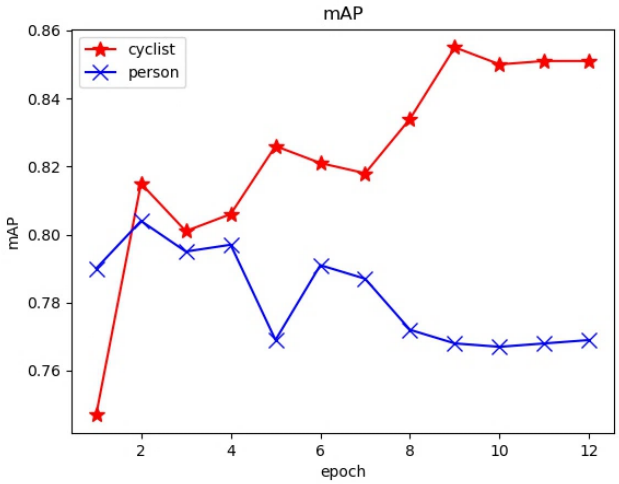
After updating the network in each batch, I will also record the change in loss

Chapter8.1 - Results of VGG16

epoch	mAp	cyclist	person
1	0.769	0.75	0.79
2	0.810	0.82	0.80
3	0.798	0.90	0.80
4	0.802	0.81	0.80
5	0.798	0.83	0.77
6	0.806	0.82	0.79
7	0.803	0.82	0.79
8	0.803	0.83	0.77
9	0.811	0.86	0.77
10	0.809	0.85	0.77
11	0.810	0.85	0.77
12	0.810	0.85	0.77

Table[8.1.1] SSD-VGG16

In order to easily view the effect, I will directly display it together with a line chart. The rest of the tables can be viewed in the appendix.

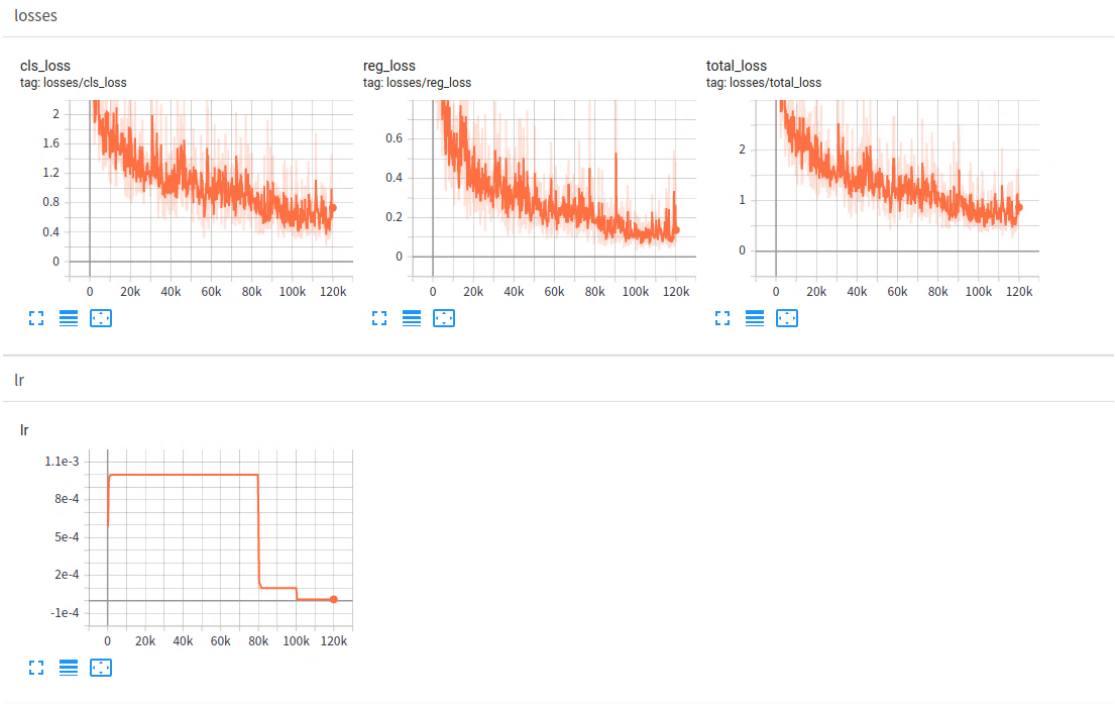


Figure[8.1.2] mAP line chart of SSD-VGG16

Although it can be found that bicycles are more recognizable than people, don't forget that for our data set, the number of bicycles is far greater than people. And the final confidence is greater than 75%, and the mAP has converged from the image, which means that 75% of the objects can be basically recognized.

We can see that the back half of the blue pedestrian line is actually falling. In this case, it is called "overfitting". The reason for this result is that people have fewer data sets. To improve this situation, we can use the method of increasing the amount of data. Such as data enhancement (random cropping, color jitter, grayscale...) Collect more data for scenarios. We can also stop training early. Or use as high a learning rate as possible, such as 0.1.

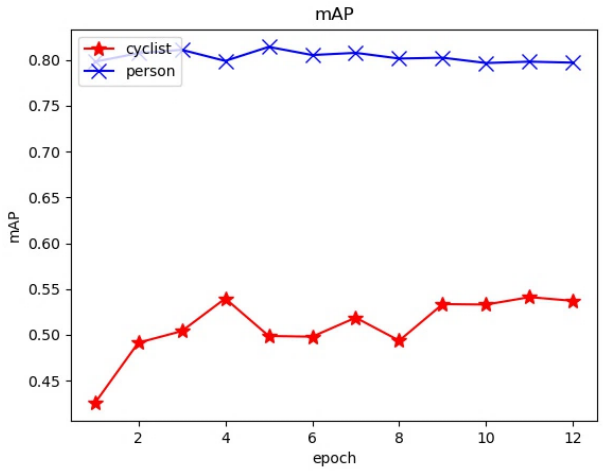
Similarly, record the change in loss after each batch.



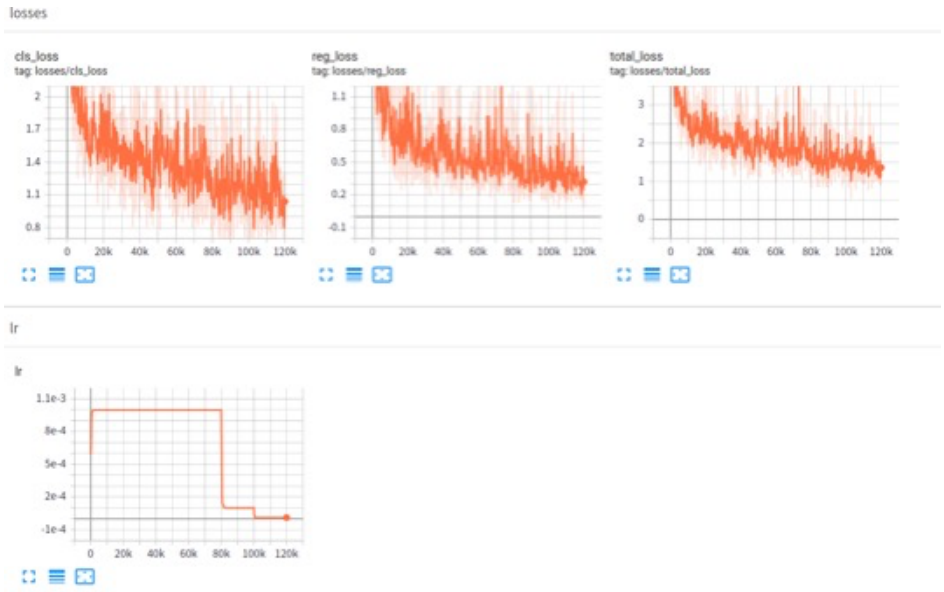
Figure[8.1.3] loss line chart of SSD-VGG16

From the change of loss, it can be seen that our initial learning rate design is reasonable, and at the end of learning, it can basically meet the convergence.

Chapter8.2 - Results of SSD MobileNet



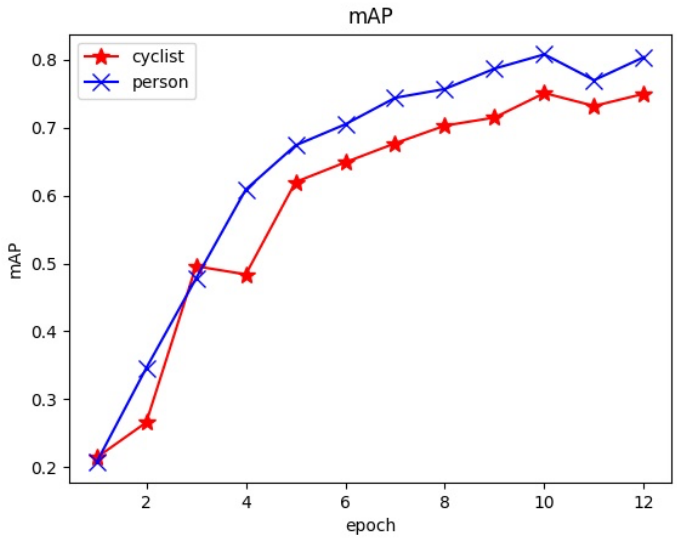
Figure[8.2.1] mAP line chart of SSD-MobileNet



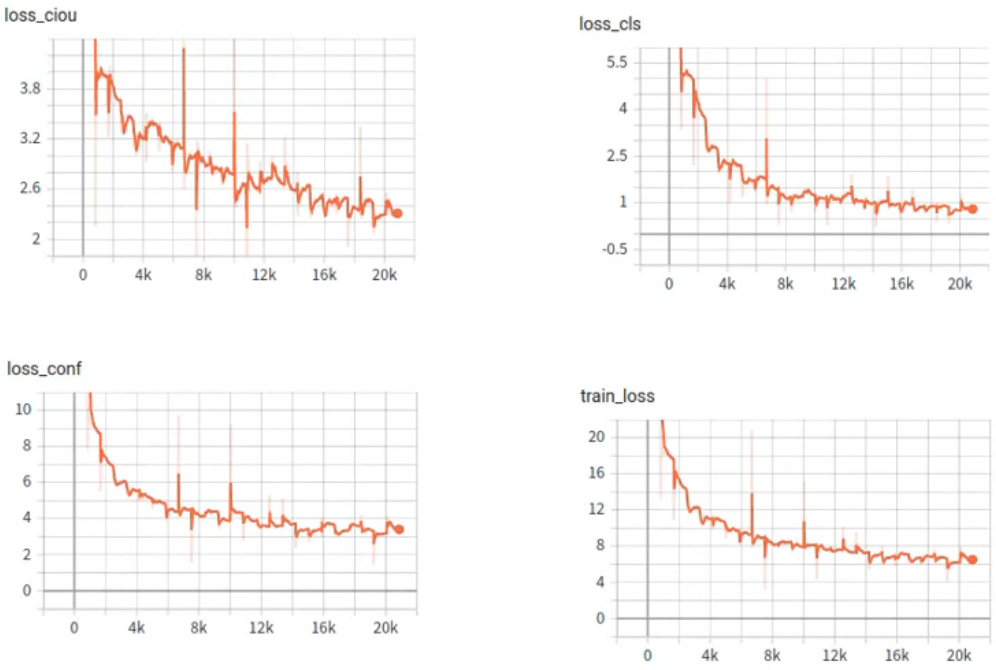
Figure[8.2.2] loss line chart of SSD-MobileNet

From the experimental results, it seems that MobileNet is not as good as Vgg16. I will explain in detail in the subsequent comparison of the results.

Chapter8.3 - Results of YOLOv4



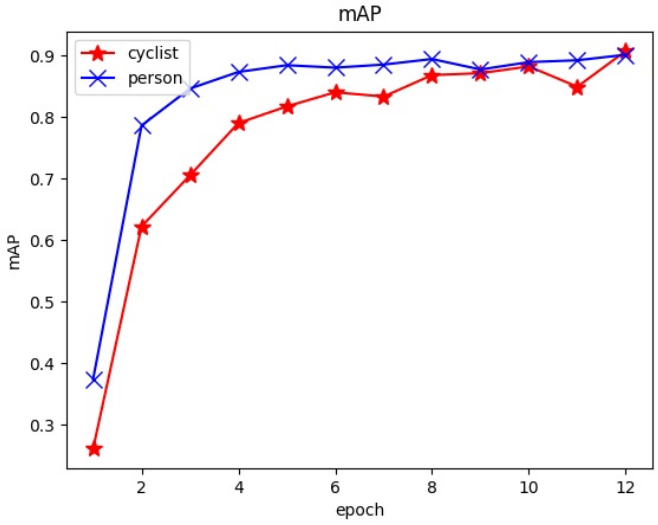
Figure[8.3.1] mAP line chart of YOLOv4



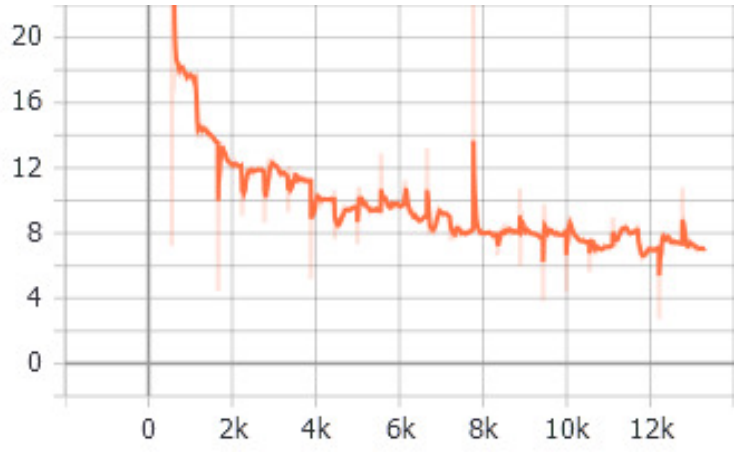
Figure[8.3.2] loss line chart of YOLOv4

The same is a single-layer target detection, yolov4's performance is significantly better than before.

Chapter8.4 - Result of FASTER RCNN+FPN



Figure[8.4.1] mAP line chart of FASTER RCNN+FPN



Figure[8.4.2] loss line chart of FASTER RCNN+FPN

I am very satisfied with the performance of faster rcnn+fpn, because the same data set has a very big difference through the algorithm. Below I will compare all the above results together.

Chapter9 - Comparison of experimental results

Chapter9.1 - Comparison of results of high-performance machines

Model	Input Size	FPS	mAP	Cyclist	Person
VGG16	512*512	47.3	0.81	0.85	0.77
MobileNetv3	320*320	43.1	0.67	0.54	0.80
YOLOv4	608*608	19.6	0.78	0.75	0.81
FASTER RCNN+FPN	1333*800	12	0.9	0.899	0.902

Table[9.1] Comparison of algorithm results-high

The table shows us the various data of each algorithm in the high-performance computer.

From the experimental results, we can see that FASTER RCNN+FPN performs best in terms of detection accuracy, but the shortcomings are also very obvious. The fps only reached 12. This is due to its two-stage network. It has a region proposal network RPN, this part uses CNN, and the region proposal network and classification network share convolutional features. And there is FPN, independent prediction of different feature layers. These can effectively target detection, but the speed is slower. We can easily find this from the table.

The above SSD-MobileNet input resolution is relatively small, and the recognition effect of the person class with the original image size of about 800 is better. The input resolution of SSD-VGG16 is relatively large, and it can better recognize the cyclist class with the original image size of 2048*1024.

The reason why mobileNet's mAP is too low is because of overfitting in the person part. It is because the data set of the person part is too small. The way to solve this problem and get the performance it should be is to increase the amount of data, as I explained in the results section.

YOLOv4 is already very good as a non-lightweight performance. It shows a very average value, and there are some aspects that must be explained. Compared with its previous version, it has added many improvements, such as adding feature pyramids (SPP, PAN) to the last convolutional layer, and modifying the activation function to Mish, learning rate annealing algorithm. From the structural point of view, YOLOv4 also has an FPN structure. Compared with FASTER RCNN+FPN, YOLOv4 has a higher fps performance. Although mAP did not reach the two stage, I got a better result in the one stage.

Chapter9.2 - Comparison of results of low-performance machines

Model	FPS
VGG16	0.98
MobileNetv3	7.61
YOLOv4	--
FASTER RCNN+FPN	--

Table[9.2] Comparison of algorithm results-low

On low-performance computers, I focus on the performance of fps. Because the machine I used is the jetson nano 2gb version of NVIDIA. It means that in the case of supporting a series of environments, the theoretical video memory of the machine is only 2gb. In actual use, the video memory is only about 1.3gb. This has actually reached a limit value. From the table, we can see that in the case of very low video memory, only lightweight algorithms can actually run normally and perform target recognition. The normal version of YOLOv4 cannot run on a machine with only 2gb of video memory. The two-stage FASTER RCNN+FPN is even more impossible.

Therefore, for a low-performance machine, it is very important to choose a correct algorithm. From the table, as a simple convolutional neural model, VGG16 can operate correctly and recognize targets normally. But I think its fps is not good enough. As a lightweight neural network, MobileNet can run normally and has a fps of about 7 I think it is understandable. There may be doubts that this fps is not very high, why is it still a lightweight neural network? This is because we are using the Large version, which has 3

bottleneck layers and a point-wise convolutional layer more than the Low version. Its Depthwise Separable Convolution is very suitable for low-performance versions.

Chapter10 - Conclusions and future work

Chapter10.1 - Conclusions

For these four algorithms and the selected data set, from the results, if the machine performance is not considered, then FASTER RCNN+FPN is undoubtedly the best auxiliary system. Because it has higher accuracy. It is undeniable that two stage is better than one stage in terms of accuracy. But the shortcomings are also very obvious. The training time of the two stage is much longer than that of the one stage. And in actual use, the fps of the two stage application on high-performance machines is not high enough, and it cannot run smoothly on low-performance machines.

The speed of yolov4 is much faster than faster-rcnn on high-performance machines. And because the feature extraction layer of yolov4 uses a feature pyramid structure, it can also get very good accuracy.

SSD-VGG16 uses feature maps of different scales for detection. This may also be the reason why in this data set, for small targets such as people and bicycles, and there may be obstructions on the road, it has a higher accuracy rate than the YOLO algorithm.

For this time, MobileNetv3, which performed best in low-performance machines. I think the lightweight neural network can be used as a solution in solving business problems, or in vehicle safety systems. Because the speed of the car is fast and the neural network needs to be fast, then a lightweight neural network is a very good choice. Video memory, memory, and cpu will be relatively small. Of course, this is just a structural point of view of these four algorithms.

I think the choice of the training data set is very important. From the final result, the choice of the picture size is actually very important, because there are actually restrictions on the pictures on the Internet or the pictures taken by yourself. Today's camera equipment is getting better and better, for example, the pixels of mobile phones are getting better and better. The requirements for the training set are also very high. Maybe a certain target detection algorithm is excellent, but due to the inappropriate choice of the data set, the final result is bad. So in the end, because of the different models of different models and the different camera options in the actual use, the final results will appear to be different.

I finally used all the algorithms to make actual detections for assisted driving. Using the video of learning to drive in a driving school, a novice may not notice pedestrians and bicycles on the road. 4 algorithms are used for the screenshots of the video in the appendix. I am satisfied with the training results and can achieve target detection for most of the scenes.

Chapter10.2 - Future Work

For future work, from the target detection framework. From the Faster R-CNN related series focusing on accuracy to the YOLO series focusing on speed, the future target detection research direction will focus more on the balance of accuracy and speed. Therefore, many network structures are generated on the SSD framework. I think that if target detection is to be used as an in-vehicle safety system, various lightweight neural networks are the most preferred in recent years. There are reports that YOLOv4-tiny performs very well, we can also try it in low-performance machine. The new YOLOv5 has appeared, we can try to use it in high-performance machine.

I have seen some research papers that show that the rotation invariance of convolutional neural networks is obtained through data enhancement and large sample learning, and it does not have this feature itself. Maybe continue to study this will change the basic framework of the target detection algorithm.

The use of 3D lidar will be more refined than the image detection. Better sensors will make target detection more refined, and now more autonomous driving applications on the market are 3D lidar.

Of course, some conventional ideas are to improve the loss function, activation function, and feature fusion, which can also be considered.

BIBLIOGRAPHY

[1] Goodfellow, I., Bengio, Y., Courville, A.. Deep learning (Vol. 1). Cambridge：MIT press，2016

[2] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G. and Cai, J., 2015. Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

[3] Zhang, W., 1988. Shift-invariant pattern recognition neural network and its optical architecture. In Proceedings of annual conference of the Japan Society of Applied Physics.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun.Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

[5] Tsung-Yi Lin1,2, Piotr Dollar´ 1 , Ross Girshick1 , Kaiming He1 , Bharath Hariharan1 , and Serge Belongie.Feature Pyramid Networks for Object Detection

[6] Wei Liu1 , Dragomir Anguelov2 , Dumitru Erhan3 , Christian Szegedy3 , Scott Reed4 , Cheng-Yang Fu1 , Alexander C. Berg.SSD: Single Shot MultiBox Detector

[7] Alexey Bochkovskiy,Chien-Yao Wang,Hong-Yuan Mark Liao.YOLOv4: Optimal Speed and Accuracy of Object Detection

[8] Simonyan,Zisserman.Very Deep Convolutional Networks for Large Scale Image Recognition

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen,Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig Adam,Searching for MobileNetV3

[10] Chien-Yao Wang,Hong-Yuan Mark Liao,I-Hau Yeh,Yueh-Hua Wu,Ping-Yang Chen,Jun-Wei Hsieh,CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

[12] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia,Path Aggregation Network for Instance Segmentation

[13] Y. Yang H. Xiong M. Braun S. Pan K. Li X. Li, F. Flohr and D. M. Gavrila. A new benchmark for vision-based cyclist detection, June 2016.

[14] Navneet DALAL,Finding People in Images and Videos

[15] Mark Everingham, John Winn, The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit

[16] https://github.com/rafaelpadilla/Object-Detection-Metrics

[17] https://developer.nvidia.com/cuda-zone

[18] Abi-Chahla, Fedy (June 18, 2008). "Nvidia's CUDA: The End of the CPU?". Tom's Hardware. Retrieved May 17, 2015.

[19] Yegulalp, Serdar (19 January 2017). "Facebook brings GPU-powered machine learning to Python". InfoWorld. Retrieved 11 December 2017

[20] Lorica, Ben (3 August 2017). "Why AI and machine learning researchers are beginning to embrace PyTorch". O'Reilly Media. Retrieved 11 December 2017

[21] Ketkar, Nikhil (2017). "Introduction to PyTorch". Deep Learning with Python. Apress, Berkeley, CA. pp. 195–208. doi:10.1007/978-1-4842-2766-4_12. ISBN 9781484227657.

[22] Patel, Mo (2017-12-07). "When two trends fuse: PyTorch and recommender systems". O'Reilly Media. Retrieved 2017-12-18.

[23] Mannes, John. "Facebook and Microsoft collaborate to simplify conversions from PyTorch to Caffe2". TechCrunch. Retrieved 2017-12-18. FAIR is accustomed to working with PyTorch – a deep learning framework optimized for achieving state of the art results in research, regardless of resource constraints. Unfortunately in the real world, most of us are limited by the computational capabilities of our smartphones and computers."

[24] Arakelyan, Sophia (2017-11-29). "Tech giants are using open source frameworks to dominate the AI community". VentureBeat. Retrieved 2017-12-18.

[25] "The C++ Frontend". PyTorch Master Documentation. Retrieved 2019-07-29.

[26] https://github.com/open-mmlab/mmcv/blob/master/README.md

[27] SAE. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. https://www.sae.org/misc/pdfs/automated_driving.pdf,2018.

[28] https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit

Appendix

Epoch	mAP	Cyclist	person
1	0.612	0.425	0.798
2	0.649	0.491	0.807
3	0.658	0.504	0.811
4	0.669	0.540	0.799
5	0.657	0.499	0.814
6	0.652	0.498	0.805
7	0.663	0.519	0.808
8	0.648	0.494	0.802
9	0.668	0.534	0.803
10	0.665	0.533	0.797
11	0.670	0.541	0.798
12	0.667	0.537	0.797

Table[1] SSD-MobileNet mAP

```
epoch     mAP                    cyclist              person
0:map:0.01511390840793961 AP: 0.011090592415383717  0.019137224400495503
1:map:0.12560629908052207 AP: 0.1355249720122446  0.11568762614879952
2:map:0.2106696195498335 AP: 0.213738450974705  0.20760078812496197
3:map:0.324931794722253 AP: 0.34805093502969325  0.30181265441481264
4:map:0.3064225665143996 AP: 0.26586389785583664  0.34698123517296253
5:map:0.48682805722909855 AP: 0.49566685419178796  0.47798926026640914
6:map:0.5489306371101657 AP: 0.5431001653748844  0.5547611088454469
7:map:0.546354252939313 AP: 0.4840780864136871  0.6086304194649388
8:map:0.5442693310894489 AP: 0.5529452305541747  0.5355934316247231
9:map:0.6470028267997745 AP: 0.6201007093681469  0.6739049442314021
10:map:0.6619819928573629 AP: 0.6312606163568658  0.6927033693578601
11:map:0.648031269346276 AP: 0.6040986735031665  0.6919638651893855
12:map:0.6768092663444434 AP: 0.6488908346814168  0.7047276980074699
13:map:0.7102364143895287 AP: 0.6765287098342889  0.7439441189447684
14:map:0.7143273072917977 AP: 0.6918312923323826  0.7368233222512129
15:map:0.7303403270436799 AP: 0.7032220686678539  0.7574585854195061
16:map:0.7361686348222793 AP: 0.7098234080225243  0.7625138616220344
17:map:0.7508686322173195 AP: 0.714527173819899  0.7872100906147399
18:map:0.7474784126086281 AP: 0.7132917333810811  0.7816650918361752
19:map:0.7585915030156352 AP: 0.7276023482385396  0.7895806577927307
20:map:0.7794352437586441 AP: 0.7510502717644872  0.807820215752801
21:map:0.749392438152801 AP: 0.7293594718460554  0.7694254044595465
22:map:0.7507952018610912 AP: 0.731737093544001  0.7698533101781814
23:map:0.7768674817976782 AP: 0.7496150187589092  0.8041199448364471
```

Figure[1] YOLOv4's mAPs

Figure[2] Faster Rcnn+fpn actual detection