Dense Convolutional Networks for Efficient Video Analysis

A DISSERTATION SUBMITTED TO THE CENTRE OF SIGNAL AND IMAGE PROCESSING, DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENFINEERING AND THE COMMITTEE FOR POSTGRADUETE STUDIES OF THE UNIVERSITY OF STRATHCLYDE IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY

By
Tian Jin

-2019

Declaration

I declare that this thesis embodies my research work and that it is composed by myself. Where appropriate, I have to make acknowledgments to the work of others.

Signed:

Date:

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof John J. Soraghan and Dr. Gaetano Di Caterina, for their guidance, support, and encouragement throughout my MPhil. They guide me in deep learning and image processing area, which made me feel brave when I have trouble learning. Besides, several reference chance they give me help me so much. Without their helpful advice and practical support to numerous personal concerns, I could not finish the work in this thesis.

I want to thank my family and my friends, for their patient and encouragement to me. When I have a problem with work or life, they always made me feel bright and brave again. In the hard day of study, I can't be without them. I also thank my University. The University of Strathclyde gives me a chance to study. Also, it is my pleasure to work and study with my colleagues in the Electronic & electrical engineering Centre for Signal & Image Processing (CeSIP) at the University of Strathclyde. Thanks for your help and support. I dedicate this thesis to my parents, my friends, and my supervisors.

Abstract

In the past few years, artificial intelligence has ushered in the third climax, which is due to the improvement of hardware computing power, so that deep learning algorithm can be applied on a large scale. Before the large-scale use of deep learning, image feature engineering is more dependent on the experience of engineers, and it can't get good results in the face of random and complex environment. Through a large number of manual features, through SVM training to classify, detect, segment images. With the success of Alexnet [15], people began to realize that feature engineering can rely on neural network to extract itself. It has been proved that the neural network trained by big data has very strong robustness to image feature extraction

This article is divided into four chapters. Chapter 1 mainly introduces the experimental significance and motivation of this article, and introduces the role of deep learning in computer vision. Chapter 2 introduces the application of deep learning algorithms, including image classification (AlexNet[8], VGGNet[34], ResNet[28]), target detection (FasterRCNN[35], YOLO[36], SSD[37]), video analysis (LSTM based, 3DCNN), and introduces some of the most advanced algorithms The structure and their experimental results, the experimental results of the new structure can reach 790 FPS on the Tesla P100. Chapter 3 mainly introduces the achievements I made this year and the algorithms designed for effective video understanding. Chapter 4 is a summary of the entire process and points out what improvements can be made in the future

Content

Figures

Tables

Acronym

SIFT	Scale-invariant Feature Transform
HOG	Histogram of Oriented Gradient
LBP	Local Binary Pattern
SVM	Support Vector Machine
YOLO	You Only Lock Once
SSD	Single Shot Multi Box Detector
LSTM	Long Short-Term Memory
Bi-LSTM	Bi-dircetion Long Short-Term Memory
CONVLSTM	Convolutional Long Short-Term Memory
RPN	RegionProposal Network
3DCNN	Three Dimension Convolutional Neural Networks
BiCONVLSTM	Bi-dircetion Convolutional Long Short-Term Memory
ECO	Efficient Convolutional Network for Online Video Understanding
GAN	Generative Adversarial Networks
FDDB	Face Detection Data Set and Benchmark
COCO	COCO Dataset
VOC	Pascal VOC Dataset

Chapter 1

Project Introduction

1.1 Preface

Computer vision is the science of how to make a machine "watch the world", which further means to use the camera and computer instead of human eyes to identify, track and measure the target, so that computer processing becomes more suitable than human eyes to observe or detect object. As a scientific discipline, computer vision is an artificial intelligence system through related theories and technologies try to build a system that capable of obtaining information from images or multi-dimensional data. Because perception can be seen as extracting information from sensory signals, computer vision can also be seen as the science how to make artificial systems perceive from images or multi-dimensional data. Computer vision is a challenging and important research field in both engineering and science. Computer vision is a comprehensive subject, which has attracted researchers from various disciplines to participate in it. These include computer science and engineering, signal processing, physics, mathematics and statistics, neurophysiology and cognitive science.

Computer vision is an integral part of a wide range of intelligent and autonomous systems such as manufacturing, inspection, document analysis, medical diagnosis, and military. Because of its importance, some advanced countries, such as the United States, computer vision has been listed as a major fundamental issue in science and engineering that has broad implications for economics and science, the so-called grand challenge. The challenge for computer vision is to develop vision capabilities on a human level for computers and robots. Machine vision requires image signals, texture, color modeling, geometric processing and reasoning, and object modeling. A competent visual system should integrate all these processes closely. As a subject, computer vision began in the early 1960s, but many important advances in basic research on computer vision were made in the 1980s.

The concept of deep learning originated from the research of artificial neural network, which was proposed by Hinton et al in 2006. Deep learning structure consist of multi-layer perceptron with multiple hidden layers. 2012 is the most important year in the development history of deep learning, Alexnet[8] won the ImageNet[16] image classification competition that year, the top-5 error rate decreased by 10 percentage points compared with the previous year's champion, and it was far higher than the second place in the year. The craze for deep learning was sparked by the emergence of Alexnet[8] in 2012. After that, deep learning has developed rapidly in the task of image classification. People are not satisfied with only applying deep learning to image classification, but start

to explore in a new field. With the proposal of algorithms such as R-CNN[35] series and YOLO[36] series, deep learning has also developed rapidly in the task of object detection. Because of the depth of neural network with strong signs extraction ability, people began to use neural network to extract image information and then construct new images, has the same characteristics of this image information that is Generative Adversarial Networks (GAN)[81]. As the space for deep learning on a single picture becomes smaller and smaller, people began to look to video analysis, However some new problems emerged, the neural network powerful ability of feature extraction is due to hardware computing power has reached a new level, so the deep learning display much powerful ability than other machine learning algorithm on image problems, However, compared to image analysis, video analysis is a huge project, a series of images need to be processed not like single image analysis. Traditional deep learning method transferred to video analysis is processing these frames one by one, but it will lose interframe information and the calculation will be the new bottleneck. Therefore, a group of people begin to explore the efficient deep learning analysis architecture for video.

In this thesis, an efficient video analysis architecture has been proposed, which is helpful to relieve the pressure of video analysis. The idea of this paper is inspired by two different famous architecture, one of them is Dense Convolutional Network (DenseNet)[38], and the other one is Efficient Convolutional Network for Online Video Understanding (ECO)[82]. The new architecture by using multiple block structure like DenseNet to reduce amount of calculation, and by using the ideas of the ECO to select the effective Video frames and reduce the computation redundancy. In addition, the new architecture has achieved the state-of-the-art result on the ucf-101 dataset. The new architecture has been accepted by the he 5th International Conference on Control, Automation and Robotics

1.2 Motivation of Our Research

With the development of technology, life will always become more and more convenient, and technology can always give birth to various new industries, especially with the development of artificial intelligence, in terms of image, voice, and natural language processing. Automatic driving technology, as a comprehensive system is also maturing. In March 2016, the team led by DeepMind developed AlphaGo to defeat the human professional Go player, which marked the culmination of another wave of artificial intelligence [1]. Artificial intelligence is consisting of many parts. Today, deep learning is undoubtedly one of the hottest directions in artificial intelligence, it is a branch of machine learning in artificial intelligence. Deep learning demonstrates far more power on classification and regression than other machine learning algorithms., which is also an important reason for it to become a hot spot. Due to the exploration of great scientists such as Andrew Ng, Geoff Hinton, Yann LeCun, Adam Gibson and Andrej Karpathy, this area has made significant progress over the years [2-4].

Although the algorithm has been greatly developed, especially image classification, object detection, and image segmentation have all made a qualitative leap, but in fact these algorithms will encounter many problems when they go industrial. In industrial production, people are more considering the cost performance of the algorithm, the biggest problem of deep learning is the support of big data sets and the experience computational cost. For example, in this Go match with human players, the Alpha Go used 1,202 CPUs and 176 GPUs [5]. Although it defeated the human player, such a product obviously has no meaning, the energy consume of Alpha Go is much higher than human player and this product is too cumbersome to carry. In another example, in 2015, Google Inc. proposed a very successful face recognition algorithm called facenet [6]. This algorithm spent 1000-2000 hours on 700 CPUs clusters to train. From 500 hours, the loss was greatly reduced, although the final effect was very Ok, the cost of training is extremely high.

In order to help the development of the industry, researchers began to turn the research direction to the compression of neural networks and the pruning of models. It has been found that many parameters of neural networks are close to zero, which means that a large number of parameters are useless. The parameter 0 is equivalent to the fact that this nerve branch has died and will not pass more information backwards, so people invented the method of pruning [7], cutting out the branches with these parameters close to 0, reducing the parameters and increasing the calculation speed. Taking the VGG[34] network as an example, in the VGG16[34], 90% of the weight parameters are in the fully connected layer, but these weight parameters increase the final result of the model by only 1%. Until recently, most of the work was studying how to pruned the fully connected layer. Pruning the fully connected layer is very effective in reducing the size of the model file. As the network layer is deeper, the degree of pruning is higher. This means that the last convolutional layer is pruned the most, which also leads to a significant reduction in

the number of neurons in the fully connected layer. The method of pruning the convolution kernel can also be to reduce the weight parameter in the convolution kernel or to discard a certain dimension of the convolution kernel. Pruning actually has the advantage of reducing memory overhead. In fact, there are a large number of papers on the pruning method of neural network models. However, in industry, we have not heard of actual projects using these pruning methods for the following reasons:

1. the current method of sorting, according to contribution to neurons is not good enough, resulting in too much loss of model accuracy;

2. the method is difficult to implement;

3. those who use neural network pruning technology may use it as a technical secret.

Another good idea is to build an efficient small model, which not only has fewer parameters, but also has good effects. This method has relatively low technical difficulty and is easy to generalize, it can be used for various problems. For example, SequeezeNet [8], MobileNet [9] is designed according to this idea. These networks do not rely on hardware acceleration, but directly from the algorithm, mainly through the splitting of the network structure, skillfully designing the convolution kernel, and improving efficiency. The biggest advantage of this algorithm is portability.

Our motivation is to design a fast and efficient neural network from the network structure。 The main design idea is to use Bi-convlstm as the main body to bring up the high-dimensional features of the video, and draw on the structure of the currently popular Densenet[80] to make the network easier to understand the content through various effective shortcuts. This structure can effectively reduce the convolutional layer. At the same time achieve a very good effect, and finally use 3DCNN for fusion features.

1.3 Summary of Original Contributions

The main research results of this paper are as follows:

The first contribution is to summarize the classic neural network models that have emerged. The classification network, as well as the object detection recognition network and video analysis algorithm, are studied. The structure, advantages, and disadvantages of traditional classification models, such as Alexnet[8], VGGNet[34], Resnet[28], are discussed. The classification network is still developing. In the imagenet competition, the classic classification model has achieved high accuracy. A well-built inside classification network needed in the object detection recognition network, which determines the performance of the net. This paper studies and summarizes the existing systems, including Faster-RCNN[35], YOLO[36], SSD[37], and video analysis methods based on convolutional neural network and their advantages are summarized

The second contribution is to introduce the problems that appear in model training and the concept of transfer learning. The quality of the model is inseparable from the setting of the parameters during training. The Loss function, the one-hot label, and the analysis of the training effects are skills that should know during the actual training model.

The third contribution is to use a new image preprocessing method to process this video. The traditional video analysis method takes each frame as an input and uses traditional visual processing methods or neural network methods to analyze. The original method consumes too much calculation and is very slow. The new method I proposed was inspired by DenseNet[38]. In the case of randomly extracting frames, LSTM is used to learn the semantics of the context, and Dense connect is used to ensure accuracy. The main design idea is to use Bi-convlstm as the main body to bring up the high-dimensional features of the video, and draw on the structure of the currently popular Densenet[80] to make the network easier to understand the content through various effective shortcuts. This structure can effectively reduce the convolutional layer. At the same time achieve a very good effect, and finally use 3DCNN for fusion features. Finally, our algorithm achieves 790 FPS on a Tesla P100 GPU on the utf-101 dataset, with an accuracy rate of 91.4%

1.4 Organization of the Thesis

This thesis is organized as follows.

Chapter 1 describes the purpose and motivation of the study and briefly summarizes the original contribution of this paper.

Chapter 2 is divided into five parts. The difficulty of each part is gradually deepened. The first part reviews the development of neural network in image processing and various usages in different problems. The second part introduces some well-known data sets in deep learning, and introduces the problems that these data sets focus. The third part introduces the development of different networks, explains in detail with several famous neural networks, and explains why they are excellent. The fourth part will introduce a very interesting and very difficult topic, object detection, and detail three well-known object detection algorithms. In the last part, I will raise the difficulty to the video processing algorithm. The previous algorithm is for picture, and the video not only has a large amount of frame information but also context information.

Chapter 3 will introduce my own video processing algorithms, as well as my conclusions and prospects for subsequent algorithms, and detail my design ideas, as well as various experimental comparison results. My algorithm consists of three main ideas, a Dense 2D network, BiConvLstm, and 3D network.

Chapter 2

Deep Learning Based Computer Vision Algorithms

2.1. Introduction

This chapter mainly introduces the background of computer vision and some very famous data sets, and introduces what is deep learning and convolutional neural networks. In the image classification algorithm will introduce AlexNet[8], VGGNet[34], ResNet[28], object detection will introduce Faster-RCNN[35] , YOLO[36], SSD[37], video analysis will introduce commonly used methods based on LSTM and 3DCNN-based methods

2.2. Background

Before the deep learning algorithm comes out, for the visual algorithm, it can be roughly divided into the following five steps: feature perception, image preprocessing, feature extraction, feature selection, inference prediction and recognition. In the early machine learning, the dominant machine learning group did not care much about the features. Before using these machine learning methods, people have to design the first four parts, but this is a difficult task for anyone. In traditional computer recognition, feature extraction and classifier are two independent parts, and puts them together when applying. For example (Fig 2.1.1), if the input is a motorcycle image, first, there must be a feature expression or feature extraction process, and then put the features in to the classification algorithm for learning.

Figure 2.1.1: *Traditional method for image recognition, feature extraction and classifier are designed independently of each other. Normally SIFT, HoG algorithms are used as Feature representation, and the result of these algorithms are used for training classifier*

There have been many excellent ideas in the past 20 years, such as the most famous one SIFT(Scale-invariant feature transform) [10]. It is widely used in image comparisons, especially in the applications of structure from motion. The other is the HoG(Histogram of Oriented Gradient) [11], this method can be used to detect edges. Before deep learning, HoG [11] feature representation and SVM is the best algorithm in the field of object detection. Other feature representation algorithms include Textons [12], Spin image [13] and GLOH [14], all of which dominate the mainstream of visual algorithms before the popularity of deep learning. In fact, designing features representation algorithm require designer has a lot of experience, and very familiar with this field and dataset. Designer also need spent a mass of time on fine-tuning on parameter. Another difficulty is that the designer not only needs to design the features representation algorithms, but also they have to choose a suitable classifier algorithm based on feature. Designing a good feature representation algorithm and selecting a classifier, which combine to achieve the optimization, is almost impossible to accomplish.

The neural network solves many problems very well. Before the neural network, it is very difficult to work on feature recognition, it also leads the preference of many problems like image classification and object detection is unsatisfactory. Therefore,

people think that if the method of extracting features can also be left to the computer to finish, it will greatly reduce the difficulty of the algorithm, or even get better results. The neural network is an algorithm that combines feature recognition and classifiers, it also got very good results. a convolutional neural network (CNN) called AlexNet [15] Won the championship in the ImageNet [16] 2012 Challenge, Alexnet[15] pushed the neural network to a climax in an instant. In the traditional algorithm, these feature recognition algorithms are actually looking for the underlying information of the edges and corners in an image, and the effect is unsatisfied. After the visualization of the feature map of the neural network, its ability far exceeds the traditional algorithm, the feature visualization shows that the lower dimension network is extracting features of the edges and corners, and the higher dimension network is extracting global features (Fig 2.1.2), in terms of feature extraction capabilities the efficiency of the neural network is much higher than normal algorithm.

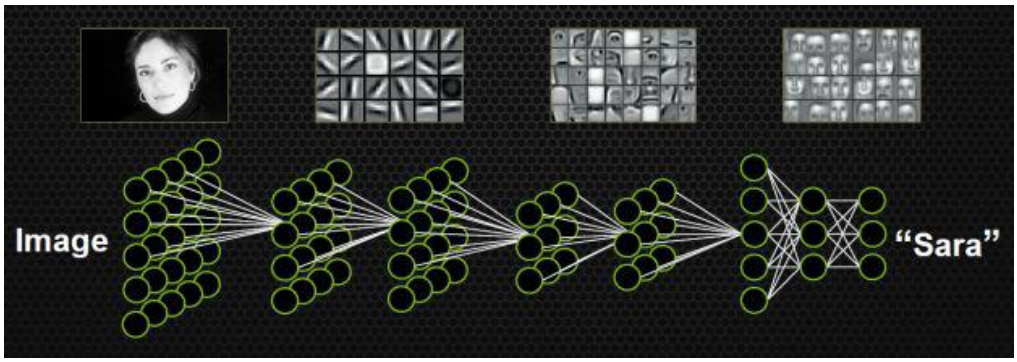

Figure 2.1.2: *This is an example of classification. When Sara's photo is input to the network, the initial network focus information based on a few pixels, so some information such as edges and corners is obtained. As the network continues to deepen, the feature maps continue to shrink, and any pixel in the higher dimension feature map can represent the region in the original image. The network begins to focus on global information. Finally, this information was classified as Sara.*

Since the neural network has shown extremely powerful capabilities in image feature recognition, people have made great progress in object detection and image semantic segmentation. On this basis, face recognition and optical character detection have been well developed.

2.3. Popular datasets for deep learning challenge

Neural networks can have such amazing results, thanks to big data, here I introduce some of the most popular datasets, and the role of these datasets. There are four different types of data sets

The first type is generic large-scale dataset, the typical large-scale dataset like MS-COCO [17], ImageNet [16], Pascal [18]. The role of this data set is more to provide a banchmark, so that everyone can test the algorithm on this open and approved data fairly, making the effect more convincing. and because of the huge amount of data, you can provide a Good data pre-trained model for better fine-turning. For example, ImageNet project is a large visual database for visual object recognition. More than 14 million images are annotated manually [16], at least a million images' bounding boxes are also provided. ImageNet contains more than 20,000 categories [16]. The dataset and image annotation are available free of charge from the ImageNet website. Since 2010, the ImageNet project has held an annual software competition, the ImageNet large-scale visual recognition challenge (ILSVRC), in which software programs compete to correctly classify, detect objects and segmentation. Other two datasets are similar to ImageNet the different between them is number of images in the dataset, the reason of these datasets very famous and useful is the quality of these datasets is very high, and the amount of picture is also very large. Tasks of these datasets are involving image classification, object detection, and image segmentation (Fig 2.2.1). Many famous neural networks are trained based on these three datasets. Normally, people use these datasets to train a network, and

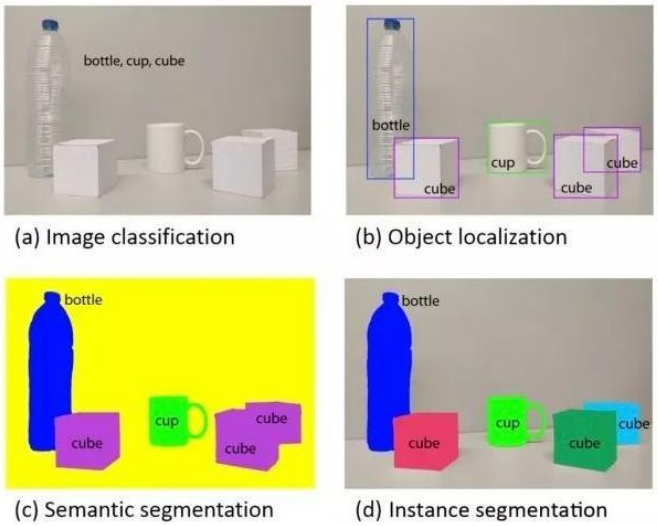

Figure 2.2.1: *there figure shows classification, object detection, image segmentation, the difficulty of these issues is gradually improved, classification problem needs to distinguish what kind of photo it is. Object detection need to draw the object bounding box in the figure. Semantic segmentation is the classification of pixel, each pixel has a category. Instance segmentation is a task combination by object detection and segmentation*

then fine-tuning this pre-training model on themselves problem, because these datasets contain a large number of images, the trained models have very powerful feature extraction capabilities. Fine-tuning with such parameters can greatly reduce training time.

The second type is human face dataset. Like FDDB [19]. This data set is a very important data set for face recognition, very convincing, and an important benchmark for measuring the quality of the algorithm. These datasets only have human face, FDDB is extracted from news images and headlines. The images contain various poses, lighting, and backgrounds. The face contains various expressions, movements and occlusion, so it is very close to the recognition scene in reality, FDDB is contains:2,845 images, including 5,171 faces [19], this is a of object detection problem, the only objects tested were faces, so normal object detection algorithms can be used on this dataset, an example of FDDB show in Fig 2.2.2

Figure 2.2.2: *Example images, for some image regions, deciding whether or not it represents a "face" can be challenging. Several factors such as low resolution (green, solid), occlusion (blue, dashed), and pose of the head (red, dotted) may make this determination ambiguous.*

The third type is autopilot dataset. The most famous one is KITTI [20]. The KITTI dataset is a joint venture between the karlsruhe institute of technology and the Toyota institute of technology in the United States. This dataset is used to evaluate the performance of computer vision technologies, such as stereo, optical flow, visual odometry, object detection and 3D tracking, and Simulate algorithm in real environment. KITTI includes real image data from urban, rural and highway scenes, with up to 15 cars and 30 pedestrians per image, as well as varying degrees of occlusion and truncation. The entire data set is composed of 389 pairs of stereo images and optical flow diagrams, 39.2 km visual ranging sequence and over 200k 3D labeled objects images. Because the data provided by this dataset are not only images and video, but also many optical flow graphs and 3D images, so KITTI is a very practical dataset in optical flow CNN and 3DCNN field, its role in the field of 3D images is similar to ImageNet

2.4. Deep neural network

This section will introduce some of the latest neural network model design ideas and experimental comparison results in detail, and introduce some basic theories of neural network.

Neural Network, in the field of machine learning and cognitive science, is a mathematical model or computational model that mimics the structure and function of biological neural networks (the central nervous system of animals, especially the brain). Used to estimate or approximate a function. The neural network is calculated by a large number of artificial neuronal connections. In most cases, artificial neural networks can change the internal structure based on external information. It is an adaptive system. In general, it has a learning function. Modern neural networks are a kind of nonlinear statistical data modeling tools.

The neural network is inspired by the perceptron. The neural network can be understood as a nonlinear combination of multiple sets of perceptrons. A simple perceptron is a model with several inputs and one output, as shown in Fig 2.3.1:

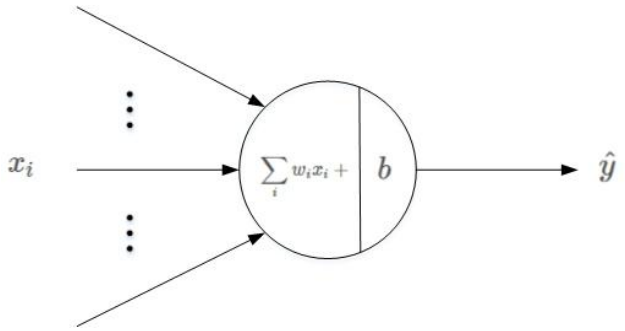

Figure 2.3.1 : *x_i is input, w_i is weight, b is bias of a perceptron*

The images (Fig 2.3.1) can be simple to understand, we have a result it called y, and the result influenced by multiple different factors x, different factors have the different effect on the final y, so each x corresponds to an own weight w, add these factors together and finally give a general bias. This formal of perceptron is in 2.3.1

$$y = \sum_{i=1}^{m} w_i x_i + b \qquad 2.3.1.1$$

The perceptron model is too simple and can't learn the complex nonlinear model, so it can't be used in industry. Neural networks extend the perceptron model. There are three major improvements

1. The hidden layer is added. The hidden layer can have multiple layers to enhance the expressive ability of the model. As shown in Fig 2.3.2

2. The neurons in the output layer can also have multiple outputs (shown in Fig 2.3.2), so the model can be flexibly applied to classification regression, and other machine learning fields such as dimensionality reduction and clustering.

3. Extended activation functions, using other activation functions like Sigmoid , tanx, softmax, and ReLU. The expression ability of neural network is further enhanced by using different activation functions

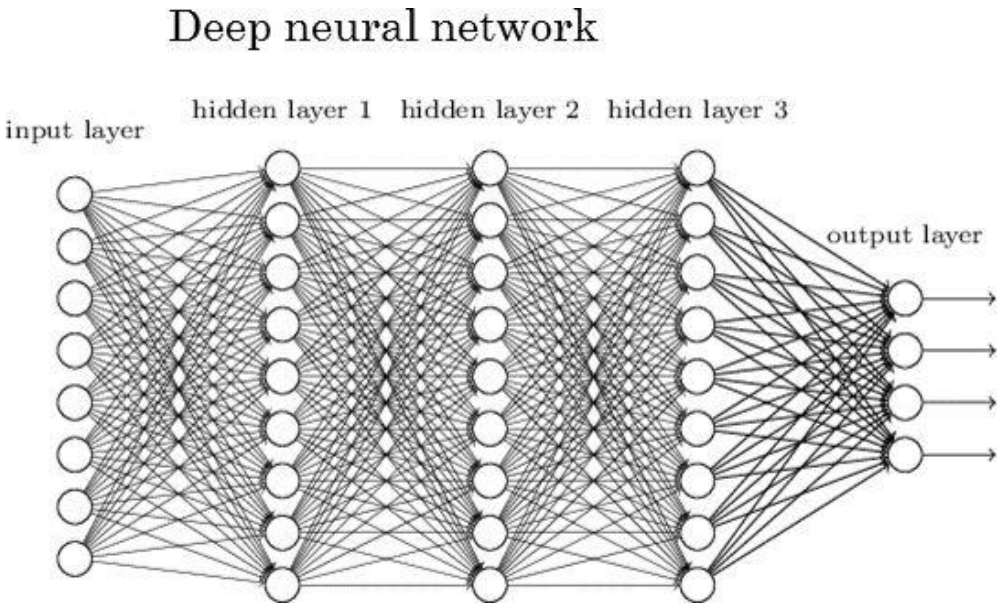

Figure 2.3.2 : *multiple hidden layers can enhance the expressive ability of the model. Multiple outputs instead of the one output*

We have known the definition of the linear relation coefficient w and the bias b of each layer. Suppose the activation function we choose is sigma (z) and the output value of the hidden layer and the output layer is sigma (a). For the three layers of DNN in the fig 2.3.3, using the same idea as perceptron, we can use the output of the upper layer to calculate the output of the next layer, which is the so-called DNN forward propagation

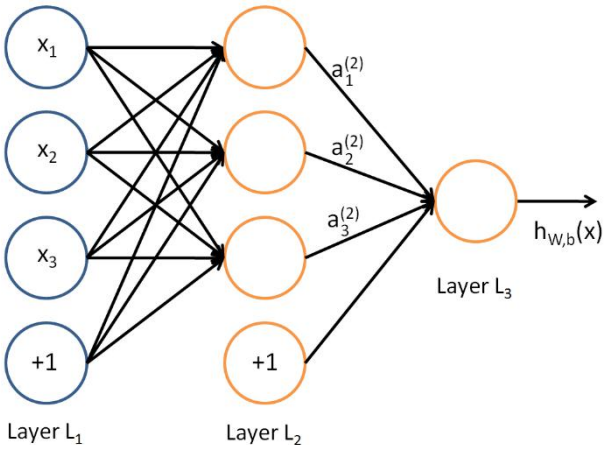

Figure 2.3.3: *This figure is an example to show forward propagation algorithm*

algorithm. This formal of forward propagation in 2.3.2-2.3.4

$$a_1^2 = \sigma(z_1^2) = \sigma(w_{11}^2 x_1 + w_{12}^2 x_2 + w_{13}^2 x_3 + b_1^2) \quad 2.3.1.2$$

$$a_2^2 = \sigma(z_2^2) = \sigma(w_{21}^2 x_1 + w_{22}^2 x_2 + w_{23}^2 x_3 + b_2^2) \quad 2.3.1.3$$

$$a_3^2 = \sigma(z_3^2) = \sigma(w_{31}^2 x_1 + w_{32}^2 x_2 + w_{33}^2 x_3 + b_3^2) \quad 2.3.1.4$$

Assuming that the neurons in the last layer of the neural network are our own classification items, then each neuron will have a score when the first forward propagation is completed. However, in fact all the parameters in this model are randomly defined, which brings up a problem, that is, this score is not what we expected, so we need to optimize this network. For example, a picture of a cat was fed into the neural network, by forward propagation, the pixel matrix of the image is multiplied by its corresponding W coefficient matrix, and after adding bias, we consider that each element in the output matrix is set as a score value of classification (show in Fig 2.3.4).

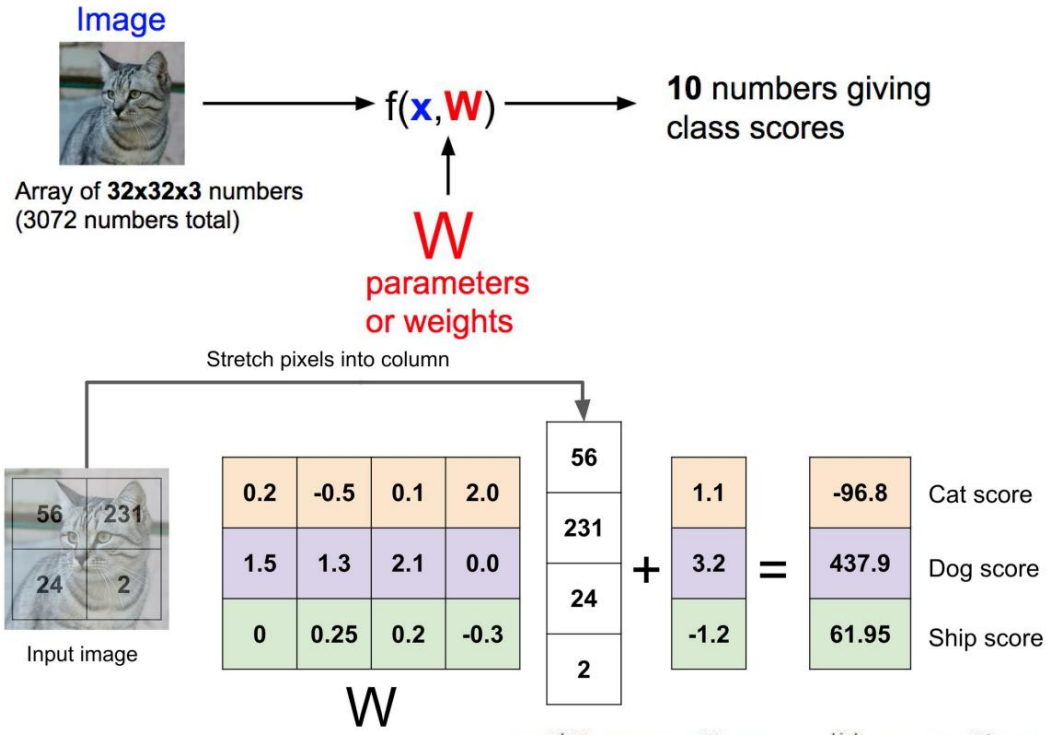

Figure 2.3.4 : *This image shows the result of a forward propagation. The input is a picture, and the output value is the score of classification. Obviously, this score value is not what we expect*

In order to solve this problem, people define the loss function and invent the backpropagation. The most famous loss function is the cross-entropy function. In the

general classification network, the activation function used in the last layer is always the softmax function, because the original output is not a probability value, it just a value which is the input value after a complex weighted sum and nonlinear processing. The function of softmax is to make this value as a probability value the formula of softmax show in 2.3.5. Suppose the original output of the neural network is y1, y2...., yn, then the output after softmax regression processing is The percentage of each value to the total value. Here we

$$\text{softmax(y)} = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_i}} \qquad 2.3.1.5$$

seem to have a clear direction of optimization, so that the ratio of the score of the target neuron to the total proportion is higher. However, this seemingly simple question for people, is still very difficult for mathematicians, because in the optimization problem an effective method in the gradient descent, this method needs a good model makes the gradient becomes more intuitive, cross - entropy can help to solve this problem very well， it is a function describing the difference between the expected value and the true value. The formal of cross – entropy is show in 2.3.6

$$L_i = -\log\left(\frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_i}}\right) \qquad 2.3.1.6$$

After having the loss function describing the task, we need to optimize the artificial neural parameter w. Gradient descent is a very good method in the optimization problem. The calculation method is to obtain partial derivatives for each neuron. This process is show in Fig 2.3.5. In the process of repeated iterations, the neural network gradually approaches our desired results and achieves the training effect.

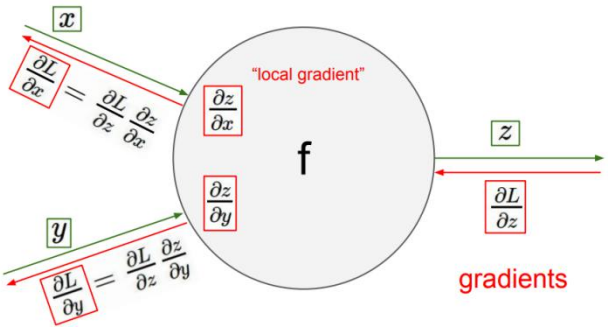

Figure 2.3.5 : *This figure shows the update process of a neuron in the back propagation of the neural network. It will get a partial derivative from former layer and pass the gradient to each next step through the partial derivative*

This section is only an introductory part. The purpose is to give the reader some understanding of the neural network. In the next few sections, the structure and calculation process of various neural networks will be introduced in detail.

2.3.1 Convolutional neural network

The idea of convolutional neural networks originates from the visual processing unit in living organisms. In 1958, David H. Hubel and Torsten N. Wiesel studied the visual cortex of cats and monkeys and found that each neuron responded only to objects in a small part of the visual range [39]. This small area is called Receptive Field. In addition, adjacent neurons have similar and overlapping receptive fields. The receptive field of neurons in the visual cortex changes with its own position, forming a complete field of vision. The visual cortex of the left and right brains is responsible for the right and left views, respectively. In 1968 they pointed out the basic visual cells in both brains [21]. Among them, the S cell (Simple Cell) has the strongest response to the straight edge of the receptive field. The C cell (Complex Cell) receptive field is larger than the simple cell, but it is not sensitive to the actual position of the edge in the receptive field. This finding provides a solid theoretical basis for convolutional neural networks.

Inspired by this observation, Kunihiko Fukushima developed a new perceptive machine in 1980 (Neocognitron) [22], whose topology is shown in Fig 2.3.6. The network consists of many types of similar parts to the basic visual cells, such as s-cell and c-cell. The network also USES cascade structure, in which the neurons at the higher level can acquire larger receptive fields and specific local features than those at the lower level. This is thought to be the inspiration for the convolutional neural network.

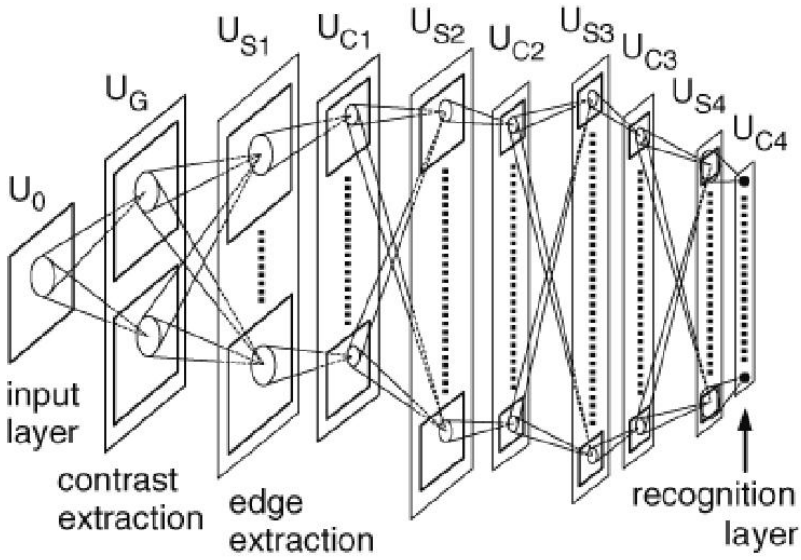

Figure 2.3.6: *new perceptive machine with different cell and cascade structure*

In 1998, Yann LeCun et al. proposed the first practical Convolution Neural Network (CNN) -- LeNet-5[23], whose Network topology is shown in Fig 2.3.7. The network is capable of sorting 32x32 pixels of handwritten digital images. At that time, many traditional image classification algorithms could achieve basically consistent or even

higher classification accuracy with much less computational cost. Limited by the computing power of hardware, the convolutional neural network at that time was shallow in depth, small in scale and unable to process high-resolution images. However, the expansibility of convolutional neural network is much stronger than those traditional algorithms, and its carrier computer is also developing rapidly, which lays a solid foundation for future research.

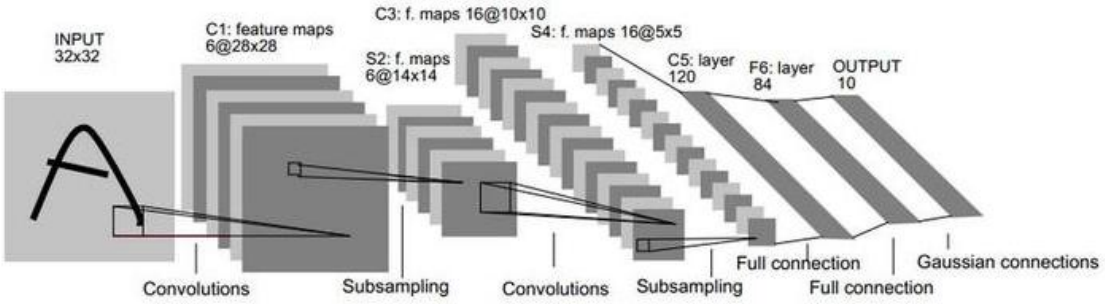

Figure 2.3.7: *LetNet-5 Convolution Neural Network*

In 2005, Dave Steinkraus et al. discovered the unique advantages of Graphics Processing Unit (GPU) in Machine Learning (ML) [24] and proposed an efficient CNN training method based on GPU. This idea liberates computing power and makes the development of CNN more free.

In 2012, Alex Krizhevsky et al. proposed AlexNet [15]. The network uses the GPU to perform parallel training using packet convolution, and uses the ReLU (Rectified Linear Unit) activation function for the first time. In addition, the dropout technique is used in the network to effectively alleviate the problem of network overfitting. In the end, they obtained the accuracy of the traditional image classification method in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [16] 2012 competition by AlexNet[15]. Since then, CNN has become the mainstream of image classification methods and shines in the field of machine vision.

In 2014, Christian Szegedy et al. proposed GoogLeNet [25]. The biggest contribution of this network is the Inception architecture. The discovery of this structure stems from the idea of finding the optimal local network structure. The authors believe that for different network levels, the size of the convolution to achieve the best results is different, so they let each layer of input through the processing of multiple filters (different kinds of convolution aggregation) And concatenate the results of the next layer of feature selection. They also use 1x1 convolution to reduce the amount of computation required for subsequent 3x3 and larger convolutions. In the end, the network has achieved significant improvements in classification results and computational efficiency.

In 2015, Sergey Ioffe et al. proposed Batch Normalization [26]. This structure acts before each activation function in the network. In order to eliminate the internal covariate

shift (Internal Covariate Shift) reflected in the output distribution of each layer in the network, the structure linearly maps the net output (NetOutput) of each layer with trainable parameters to ensure the output of each layer in the network (The Activation) is basically stable and the output amplitude is kept at the same scale. At the same time, it greatly reduces the dependence of the network on the initial parameters, improves the efficiency of gradient transmission in the network, and greatly improves the network speed. And its corrective effect on the network output at each layer makes the training of the deep network really possible. .

In the same year, Kaiming He et al. proposed the Residual Network (ResNet) [28]. The network's creative use of Residual Learning has ingeniously alleviated the problem of gradient instability, greatly increasing the number of information transmission paths in the network [29], and pushing the training network depth from dozens of layers to thousands. Floor. This kind of thinking is a milestone. With it, the deep neural network can reflect its performance advantages relative to the shallow neural networks

In 2016, Gao Huang et al. proposed the Densely Connected Convolutional Network (DenseNet) [30]. The design of the network is based on the idea of feature reuse, and it has achieved the ultimate in the field of network topology. Each layer in DenseNet is directly connected to all of its previous layers. The output of each layer is combined to be the input of the next layer. Such a topology connection method enables each layer in the network to have a data transfer path directly connected to the final fully connected layer, which greatly improves the problem of gradient transfer in the network. At the same time, the special structure of the network makes it much higher in parameter efficiency and computational efficiency than other networks.

In 2017, Jie Hu et al. proposed Squeeze-and-Excitation Networks (SENet) [31]. The network makes a more specific assumption about the input image signal: the input of the convolutional neural network has the same prior knowledge in the spatial dimension, and the same in the depth dimension. In this network, a path is added next to each convolutional layer, allowing the convolutional neural network to learn the strength of each channel autonomously and use it in the linear mapping of the channel.

The convolutional network is mainly composed of the convolutional Layer, the Pooling Layer and the full connection Layer. However, with the development of neural network, these structures have also been changed, and the development trend of the network is more towards the full convolutional network

A convolutional layer is a collection of filters with trainable parameters. The spatial scale (height and width) of these filters will not be too large, but the depth and input data should be consistent. In the network's Infer, forward propagation process, each filter slides in the spatial dimension of the input data and calculates the filtering result of the filter at that location. The filtering results of each filter at different positions will form a unique feature map, and the feature maps of all filter outputs are superimposed in the depth direction to obtain the final output of the convolution layer. According to the results

of the visualization of convolutional neural networks [32], shallow-layer filters tend to respond to simple set features, such as object edges or patches, while deep filters can produce advanced features of the image. Reacts and is not sensitive to the location of the feature. This is very similar to the corresponding function of S cells and C cells in the mammalian visual cortex [21].

The convoluted neurons are distributed in three dimensions. These three dimensions are width, height (space dimension), and depth. The depth of a certain neuron depends on the filter number corresponding to the neuron, and its spatial position depends on the spatial position of the filter in which the neuron is located. However, the size of the feature map output by the convolutional layer has many variables. The normal filtering operation takes as input the gray values of all pixels in the center pixel and its neighborhood. According to the above idea, if the filter size is larger than 1, the size of the filtered image should be smaller than the original image. In a convolutional neural network, if the output size of each convolutional layer is smaller than the input, then in theory, the number of layers in the entire network is directly limited by the size of the input image, which is for constructing deep neural networks. It is a huge limitation. Filling the input image with 0, zero-padding the input image to ensure that the size after convolution remains a viable solution to the above problem. Although this will cause the features of the edge of the original image to be difficult to detect, the depth of the convolutional neural network can be theoretically unrestricted (of course, it should be used according to actual needs in the actual network). The convolution layer also has a parameter called the step. This parameter determines the distance the filter slides. For example, when the step size is 1, the filter moves one pixel at a time in the width or height direction of the feature map. When the step size is not 1, the filter moves two pixels in the width or height direction of the feature map each time, and the size of the output is scaled down. This is usually used as a means of down sampling. For any convolutional layer, when the spatial size (width and height) of the input feature map is W, the filter size is F, the step size is S, and the padding fill width is P, the spatial size of the output feature map should be $\frac{W-F+2P}{S}+1$. This size should be an integer, otherwise it has no practical meaning. A complete picture of the convolution calculation process can refer to Fig 2.3.8

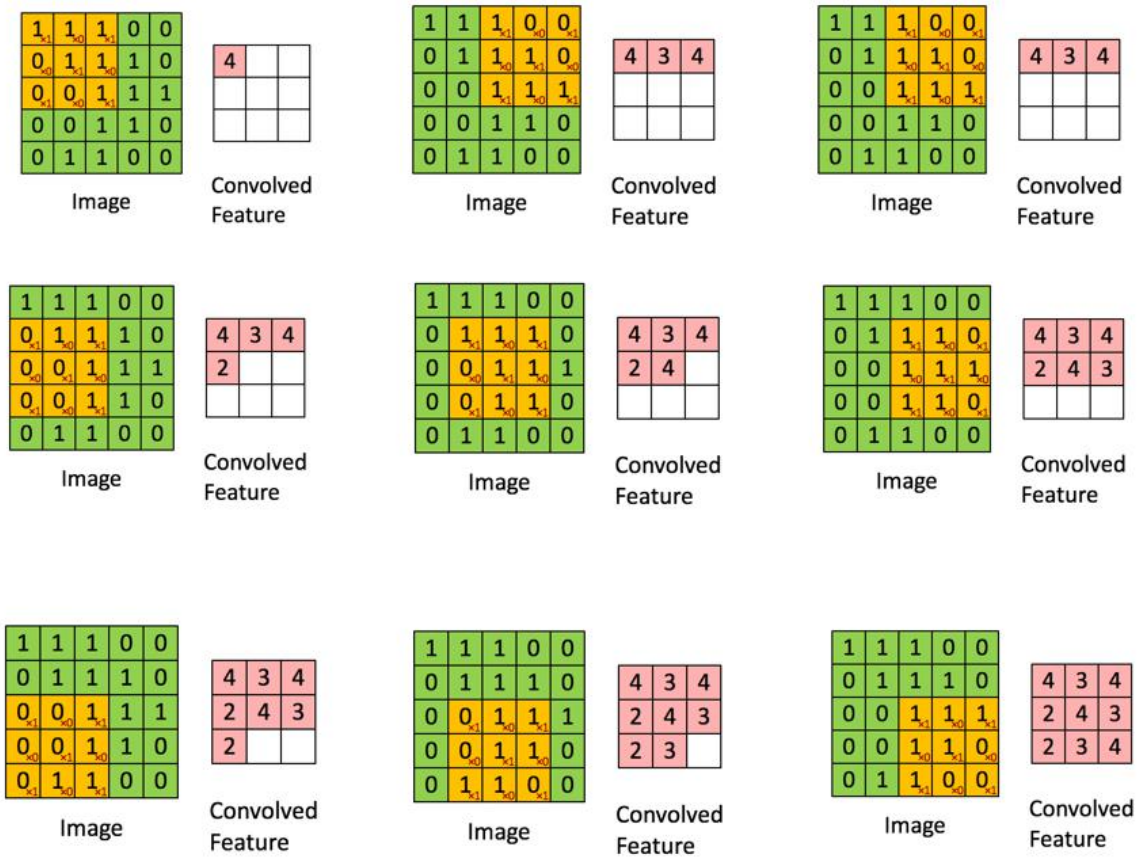

Figure 2.3.8: *process of image convolution*

Parameter sharing is a means of controlling the number of other parameters in a convolutional network. In a neural network, assume that the first convolutional layer has a filter space size of 3x3, an input size is 28x28x3, and an output depth of 32. if there is an independent filter in the position of each pixel of the original image, the required parameters of the convolution layer are $28x28x3x3x3x32 = 677376$. The number of parameters seems acceptable, but once the image size or filter size increases, the number of parameters will quickly increase to an unacceptable level. If it is assumed that an identical feature appears equivalent at any position in the image, that means the filters used in the same output feature map are identical at any position, the required parameters are 3x3x3x32=864. The shared parameters of the same layer filter also make each filter in the convolutional neural network equivalent to a group of S cells with the same function but different positions. In addition, the convolutional layer based on local connection and parameter sharing can be regarded as the input feature map after the block is convoluted with the filter weight, which is also the origin of the convolutional neural network name.

2.3.2 AlexNet

AlexNet is a significant neural network in the history of computer vision. This network was completed by the University of Toronto's Hinton and his students. This network participated in the ImageNet LSVRC-2010 classification competition in the 1.2 million sample 1000 category classification they achieved top-1 and top-5 error rates of 37.5% and 17.0% respectively, this is a state-of-the-art result at that time. This network proposes a variety of effective training methods to make neural networks more efficient, such as dropout, and Relu function. in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry

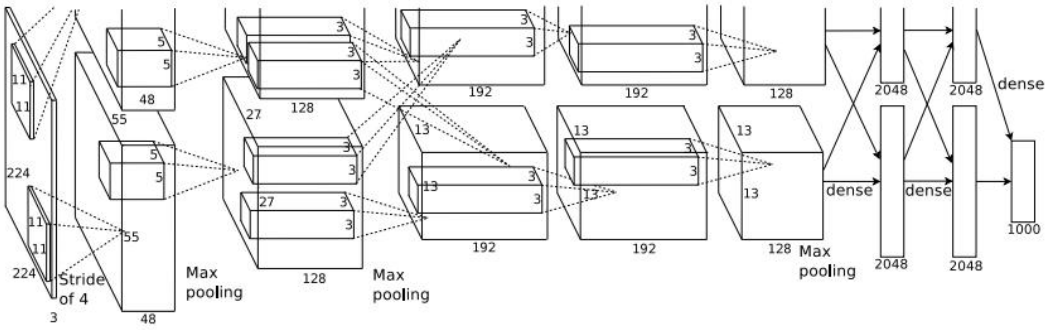

Figure 2.3.9: *The structure of Alexnet*

The architecture of Alexnet is summarized in Fig 2.3.9. It contains eight learned layers five convolutional and three fully-connected. It actually looks more like a larger LeNet-5. The network structure contains a total of 8 weight layers. The first five are convolutional layers for down sampling, the last three are fully-connection layers, and the last layer is fully-connection with 1000 neurons for classification.

An important factor in Alexnet is to use Relu function [33] the formula is $f(x) = \max(0,x)$, instead of traditional Tanh $f(x) = \frac{sinhx}{coshx} = \frac{e^x - e^{-x}}{e^x + e^x}$ or Softmax $f(x) = \frac{1}{1+e^{-x}}$ for activation. This is because considering the time cost of training, the convergence rate of using Relu function is much faster than other functions, Fig 2.3.10 shows an experiment using ReLUs and tanh as the activation function of the typical four-layer network on CIFAR-10s dataset, the error rate converges to the convergence curve at 0.25, and the convergence speed difference can be clearly seen. The dotted line is tanh and the solid line is ReLUs.

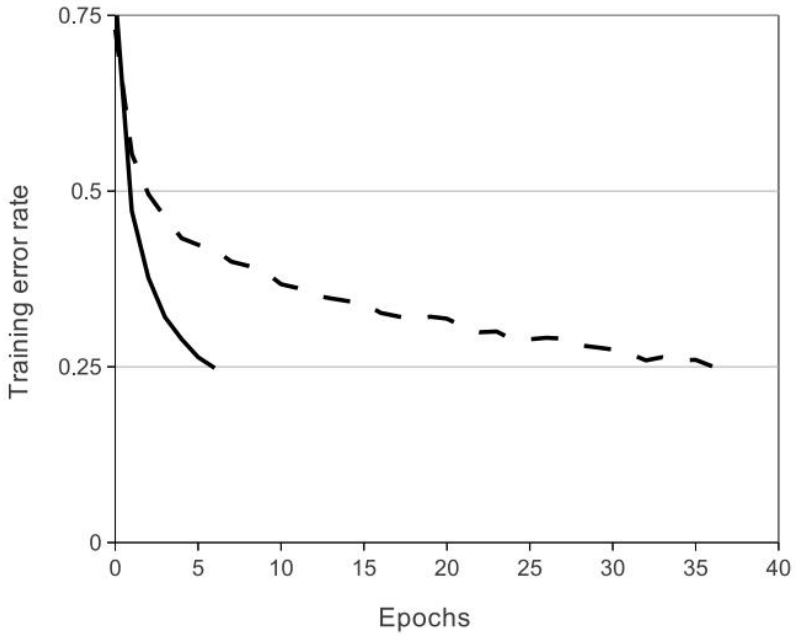

Figure 2.3.10: *Comparison of relu function and tanh function, solid line is relu, dotted line is tanh*

The activation function makes the output of the neuron a non-linear mapping, but the range of the traditional activation functions of tanh and sigmoid is ranged, but the range obtained by the ReLU activation function does not have an interval, so the result to be obtained for ReLU Normalize. That is, Local Response Normalization. The local response normalization method is as formula 2.3.2.1:

$$b^i_{(x,y)} = \frac{a^i_{(x,y)}}{(k+\alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^i_{(x,y)})^2)^\beta} \qquad 2.3.2.1$$

The details of this network is follows: input pictures are all 224x224x3. The Kernel size of first convolutional layer is 11x11 and the number of feature map is 96， stride is 4. the kernel size of other convolutional layer is 3 × 3 and the number of feature map in second layer is 256， in third and fourth layer is 384, in fifth layer is 256, the number of parameter in fully-connection layer is 4096,4096 and 1000. The output of each convolution layer will have a relu activation function, and after the second, third and sixth convolution layer there will be Max pooling, the total parameters show in Table 2.3.1

Layer	Output Size	Kernel Size/Stride/Feature Map/Padding
INPUT IMAGE	227×227×3	
CONV1+RELU1+LRN1	55×55×96	K=11×11, S=4×4, F=96, P=0
MAX POOL1	27×27×96	K=3×3, S=2×2, F=96, P=0
CONV2+RELU2+LRN2	27×27×256	K=5×5, S=1×1, F=256, P=2
MAX POOL2	13×13×256	K=3×3, S=2×2, F=256, P=0
CONV3+RELU3	13×13×384	K=3×3, S=1×1, F=384, P=1
CONV4+RELU4	13×13×384	K=3×3, S=1×1, F=384, P=1
CONV5+RELU5	13×13×256	K=3×3, S=1×1, F=256, P=2
MAX POOL5	6×6×256	K=3×3, S=2×2, F=256, P=0
FC6+RELU6+DROP OUT6	4096	Probability 0.5
FC7+RELU7+DROP OUT7	4069	Probability 0.5
FC8	1000	

Table 2.3.1: *the total parameters of Alexnet*

2.3.3 VGGNet

After Alexnet[15], neural network-based image tasks have become easier, and the black box mechanism of neural networks makes it difficult to design a perfect network, relying more on the designer's design experience, through a large number of trial and error methods. Finding the most appropriate result, people have been studying the effects of various parameters on neural networks since 2012. VGG[34] is one of the most important results. The VGG network was proposed by the Visual Geometry team at Oxford University, who investigated the effect of the depth of the convolutional neural network on its accuracy in large-scale image recognition settings. Their main contribution was the use of very small convolution filters (3*3) for deeper network effects, which showed significant improvements in the experiment by pushing the depth to 16-19 convolutional layers. The network details are as follows

During training, the image input is a fixed-size 224 x 224 RGB image. The only pre-processing done by the network is to subtract the average RGB value from each pixel and calculate it on the training set. The image passes through a stack of convolution (conv.) layers, each kernel size is very: a 3x3 filter (which is the smallest size that captures the left/right, up/down, and medium concepts).

The convolution stride is fixed to 1 pixel, and the spatial resolution after convolution is kept constant by padding, that is, if the 3×3 convolution kernel image is filled with 1 pixel. Downsampling is achieved by 5 maxpooling. Maxpooling is executed on a 2×2 kernel with a strike of 2.

The volume base layer is then connected to three fully connection (FC) layers: the first two layers each have 4096 channels, and the third layer performs 1000 ILSVRC classifications, thus containing 1000 channels (one for each class). The last layer is the soft-max layer. The configuration of the fully connected layer is the same in all networks.

VGG experimented with multiple sets of data. The convolution configuration at various depths is shown in Table 2.3.2. The name of the network is (A-E). The size of these network convolution kernels and the way they are pooled are exactly the same, differing only in depth: from 11 weight layers (8 conv. And 3 fc layers) in network a to 19 weight layers in network e (16 conv. And 3 fc layers).

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					

Table 2.3.2: *ConvNet configurations (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "convhreceptive fiela sizei-hnumber of channelsi". The ReLU activation function is not shown for brevity*

The training process of VGG is similar to Alexnet. That is, the minimum batch gradient descent with momentum (based on back propagation is used to optimize the polynomial Logistic regression target. The batch size is set to 256 and the momentum is 0.9. Training Weight decay (L2 penalty multiplier is set to 5.10^{-4}), the first two fully connected layers use Dropout regularization (the probability of dropout is 0.5). The learning rate is initially set to 10^{-2}, when the accuracy of the validation set When the increase stopped, the learning rate decreased by 10 times. After 370 k iterations (74 cycles), the learning rate decreased 3 times and the learning stopped.

The initialization of network weights is very important, and incorrect initialization will hinder learning. To avoid this problem, VGG starts training from the A structure because the network is shallow enough to be trained by random initialization. Then, when

training deeper structures, the parameters that have been trained before loading are initialized for the next field, and the extra layers are randomly initialized. For those layers that are randomly initialized, VGG samples from a normal distribution with a mean of 0 and a variance of 10^{-2}

The experimental results and comparison with other networks in the same period are listed in Table 2.3.3. In the classification task of the ILSVRC-2014 challenge, the "VGG" team used 7 combinations of models and won the second place with a test error of 7.3%. After submission, through the integration of the two models, VGG reduced the error rate to 6.8%.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

Table 2.3.3： *Comparison with the state of the art in ILSVRC classification. Our method is denoted as "VGG". Only the results obtained without outside training data are reported.*

It can be seen from Table 2.3.3 that the very deep VGG model is significantly better than previous model classifications, and has achieved the best results in terms of ILSVRC-2012 and ILSVRC-2013 capabilities. VGG is also competitive with the champion network of classification tasks (GoogLeNet with error 6.7%), and greatly exceeds the results submitted by the ILSVRC-2013Clarifai structure. In terms of single-network performance, VGG's architecture achieved the best results (7.0 % Test error), which is 0.9% higher than a single Google network. It is worth noting that VGG did not deviate from the classic ConvNet result, but only improved it by greatly increasing the depth.

Summary: VGG evaluates deep convolutional networks (up to 19 layers) for large-scale image classification. The results show that representation depth is conducive to classification accuracy, and using traditional convolutional network architecture, the most advanced performance can be obtained on the ImageNet challenge dataset.

2.3.4 ResNet

Deep networks integrate low / medium / high-level features [40] and classifiers in an end-to-end multi-layer manner, and the “levels” of features can be enriched by the number of stacked layers. The previous network [25, 34] showed that the depth of the network has a great impact on the final results. The leading results on the challenging ImageNet dataset have adopted the "very deep" [40] model, with depths ranging from 16 [34] to 30 [26]. Many other important visual recognition tasks [41, 42, 43, 35, 44] also benefit greatly from very deep models.

Driven by the importance of depth, a question arises: is it easier to learn a better network than to stack more layers? One obstacle to answering this question is the well-known problem of vanishing gradients / explosion [45, 46, 47], which hinders convergence from the beginning. However, this problem has been largely solved by standard initialization [48, 47, 49, 50] and intermediate normalization layers [26], which enables networks with tens of layers to pass through stochastic gradient descent (SGD with back propagation) Begins to converge.

However, when deeper networks can begin to converge, a degradation problem is exposed: as the network depth increases, the accuracy rate saturates and then decreases rapidly. This decline is not caused by overfitting, and adding more layers on the appropriate depth model results in higher training errors, as reported in [51, 52], and is fully confirmed by experiments by the resnet authors. Fig 2.3.11 shows a typical example

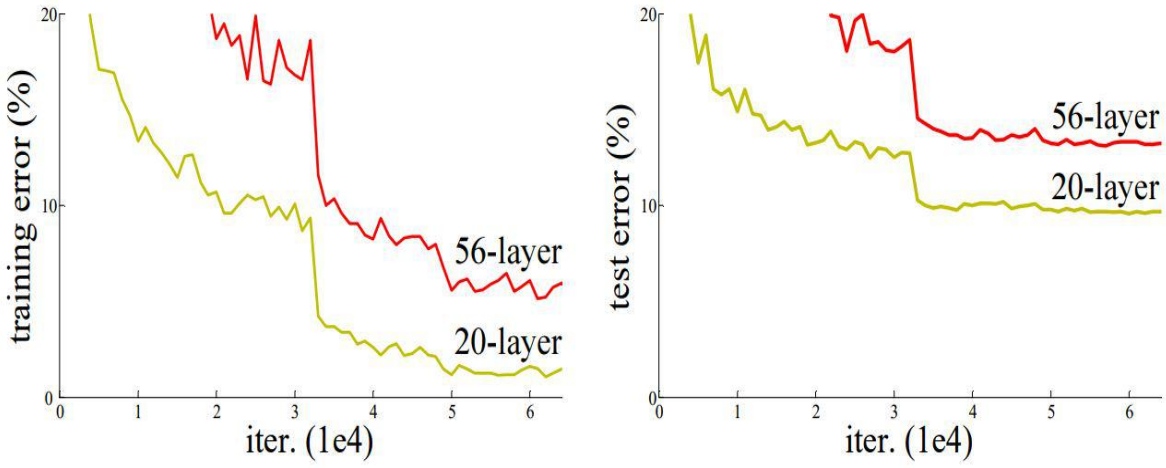

Figure 2.3.11: *Training error (left) and test error (right) on CIFAR-10 with 20-layer ana 56-layer “plain” networks. The deeper network has higher training error, and thus tes*

In order to solve the problem of degradation, the author proposes the structure of resnet as shown in Fig 2.3.12. Suppose the output is H (x). In the previous pure network, H (x) = F (X), after the non-linear representation of the residual H (x) = F (X) + x, x = αX, If the depth of the network exceeds the appropriate depth, the weight of the previous layer can be affected by changing the value of α

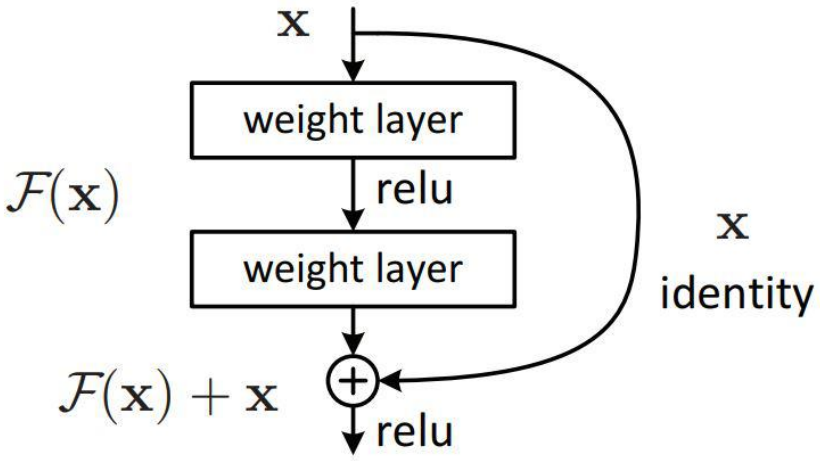

Figure 2.3.12： *Residual learning: a building block*

The authors tested various simple / residual networks and observed a consistent phenomenon. In order to provide an example for discussion, the author describes two models of ImageNet as shown in Fig 2.3.13. The simple network as the benchmark (Fig 2.3.13, middle) is mainly inspired by the philosophy of VGG network [40] (Fig 2.3.13, left). The convolutional layer mainly has 3 × 3 filters and follows two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; (ii) if the feature map size is halved, The number of filters is doubled in order to maintain the time complexity of each layer. authors directly perform downsampling through a convolutional layer with a step size of 2. The network ends with a global average pooling layer and a 1000-dimensional fully connected layer with softmax. The total number of weighted layers in Fig 2.3.13 (middle) is 34.

Residual Network. On the basis of the ordinary network in the middle, insert shortcut connections (Fig 2.3.13, right). When the input and output dimensions are the same, the features of the previous layer can be directly added to the features of the next layer (solid line in Figure 3). When the input and output sizes are not the same (the dashed shortcut in Figure 3), a 1 × 1 convolution kernel is used with a step size of 2 to achieve feature map matching.

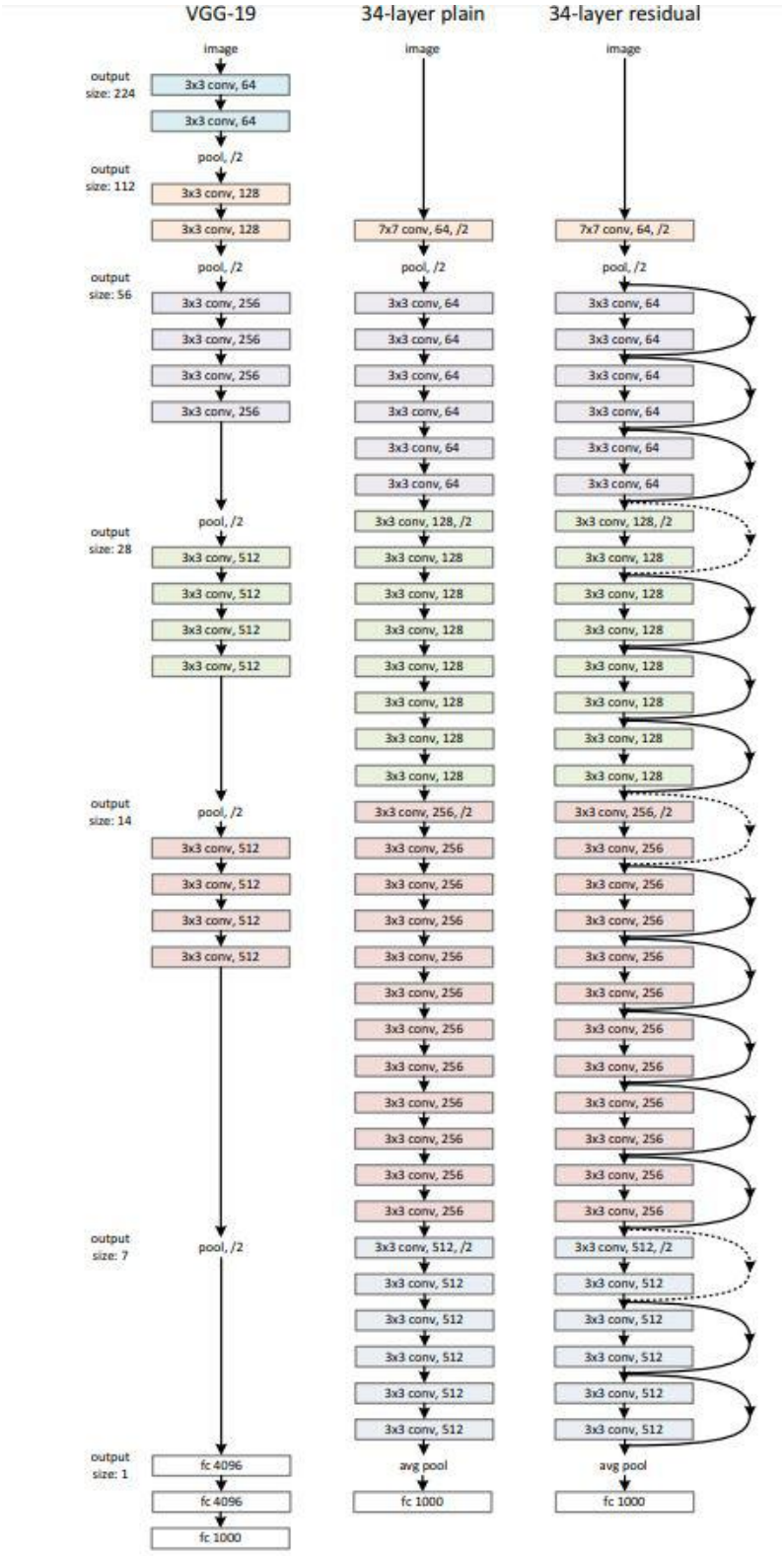

Figure 2.3.13 : *Example network architectures for ImageNet. Left: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs). Right: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. Table 1 shows more details and other variants.*

The author tested multiple groups of networks with different depths. The parameters of their network structure are shown in Table 2.3.4.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 2.3.4: *Architectures for ImageNet. with the numbers of blocks stacked. Down sampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2.*

Implementation details: The author's implementation follows the practice of [15, 34]. Resize the image to a certain range, with the shorter sides randomly sampled between [256,480] and the long sides scaled proportionally. This is used for scale enhancement [34]. Then an area of 224 × 224 size is randomly cropped from the image and horizontally flipped randomly. The cropped image is subtracted from the mean value pixel by pixel [15]. After each convolution and before activation, batch normalization is adopted [26]. The author initializes the weights according to the method of [50] and trains all simple / residual networks from scratch. SGD method with a batch size of 256. The learning speed starts from 0.1, the learning rate is divided by 10 when the error is stable, the weight decay rate is 0.0001, and the momentum is 0.9. According to the practice of [26], the author did not use dropout [53].

authors evaluated the two structures (plain, resnet) in the ImageNet 2012 classification data set [35], which consists of 1000 categories. These models were trained on 1.28 million training images and evaluated on 50,000 validation images. authors also obtained the final results reported by the test server on 100,000 test images. The result show in Tabel 2.3.5

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2.3.5: *Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have nc extra parameter compared to their plain*

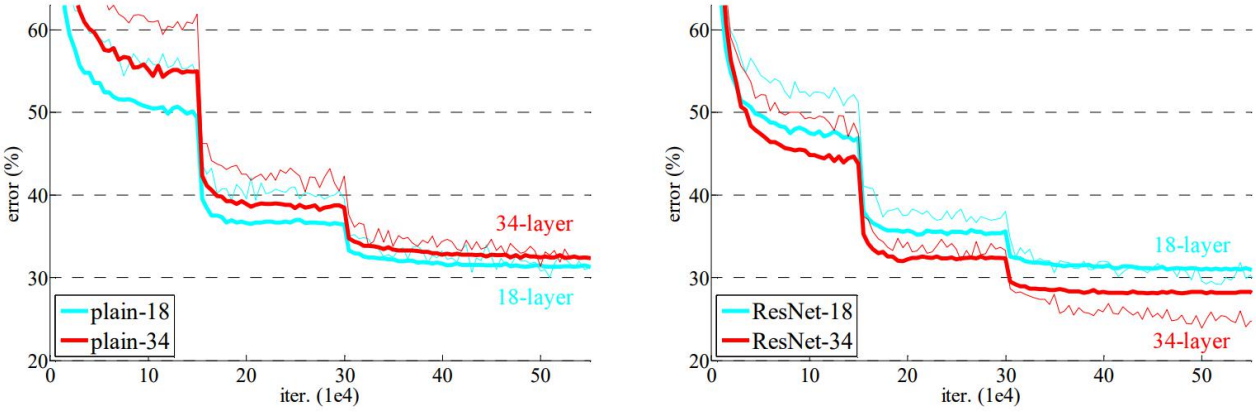

Figure 2.3.14: *Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18*

The training process show in Figure 2.3.14, the deeper 34-layer simple networks have higher verification errors than shallower 18-layer simple networks. The author believes that this optimization difficulty cannot be caused by the disappearance of the gradient. However, after using the residual block, whether it is layer 18 or layer 34, the network error rate has been significantly reduced, which effectively proves that the residual block is an efficient and simple design.

2.5. Deep learning-based object detection algorithms

Object detection has always been a fundamental problem in computer vision, and it has stagnated around 2010. Since the publication of a 2013 paper[83], object detection has changed from the original traditional manual feature extraction method to feature extraction based on convolutional neural networks, and it has been out of control since then. Prior to the formal intervention of deep learning, the traditional "target detection" methods are all trilogy of region selection, feature extraction, and classification and regression, so there are two problems that are difficult to solve. One is that the strategy of region selection is poor and the time is complicated. The degree is high; the second is that the features extracted by hand are less robust.

After the advent of the deep learning era, the large family of object detection algorithms is mainly divided into two factions, one is one stage, and the other is the two stage.

The one stage is directly returning the predicted target object. The regression to solve the problem is simple and fast, but too rude. The main representatives are YOLO and SSD. The two-stage is divided problem into two parts: 1. Generate Region Proposal and CNN to extract features, 2. Put it into a classifier to classify and correct the position. The main representative is the R-CNN system.

2.5.1. Faster-RCNN

Object detection is one of the most important basic disciplines in computer vision. After years of development of traditional computer vision, object detection has always been a flyover that has not been achieved. With the continuous development of deep learning, a new round of object detection technology has been obtained. The rapid increase. The Faster-Rcnn algorithm is the most classic algorithm in the two-stage series of target detection algorithms based on convolutional neural networks. Even today, this algorithm is still widely used in various academic and industrial scenarios.

Faster-Rcnn is the third generation of the R-CNN series of algorithms. The biggest achievement of Faster-Rcnn is the huge increase in speed and accuracy, which makes it possible to identify targets in images in real time. The advances made in object detection prior to the birth of Faster-Rcnn are all based on region suggestion methods (eg [54]) and region-based convolutional neural networks (R-CNN) [41]. The early methods based on the CNN series had a huge power consumption and time problem. The early algorithms could only stay at the academic level and could not be commercialized. Later, people proposed a method based on shared convolution. Fortunately, this consumption passed the proposal box and shared convolutions [42,43] are greatly reduced, making the second-generation algorithm Fast R-CNN [43] use a very deep network [34] to achieve a near real-time detection rate, but this rate ignores the generation region suggestion box time. In the third generation of Faster-Rcnn, it is to solve the calculation bottleneck of generating suggestions.

The most popular solution for region proposes is Selective Search (SS) [54], and this post-selection region generation algorithm based on texture and low-level features can only be processed by the CPU. It is not applicable to the scene used for real-time object detection. One of the solutions to this problem is to use a more powerful GPU to complete

The biggest contribution point of the algorithm is the Region Proposal Networks. This improvement makes the calculation of the suggestion box hardly consume the calculation of the detection network. At the time of testing, the marginal cost of calculating the suggestion box is very small (for example, 10ms per image) through shared convolution. And RPN is a fully-convolutional network (FCN) [44], which can be trained end-to-end for the task of generating detection suggestion frames. Make the whole algorithm not only efficient and smooth.

Region Proposal Networks can take an image of any size as a set of input and output rectangular target suggestion boxes, and each anchor has a target score. A full convolutional network [14] is used to build a model for this process, and the first 13 layers of the VGG network are used as shared convolutional layers for all anchors. In order to generate Region Proposal boxes, a small network is sliding on the convolution feature map output by the last shared convolutional layer. This network is fully connected

to the $n \star n$ spatial window of the input convolution feature map. This vector is output to two fully connected layers at the same level-the regression layer (reg) and the classification layer (cls). At the position of each sliding window, k Region Proposals are predicted simultaneously, so the reg layer has 4k outputs (4 points * number of anchors). The cls layer outputs 2k (foreground and background) scores, that is, the estimated probability that each suggestion box is a target / non-target (for simplicity, it is a cls layer implemented with two types of softmax layers, and it can also be generated using logistic regression k scores). The k suggestion boxes are parameterized by the corresponding k boxes called anchors. Each anchor is centered on the current sliding window center and corresponds to a scale and aspect ratio. We use 3 scales and 3 aspect ratios, so that there are k = 9 anchors at each sliding position. For a convolution feature map of size $w \star h$ (typically about 2,400), there are $w \star h \star k$ anchors in total. An important feature of this approach is translation invariance. The RPN process shouw in Fig 2.4.1

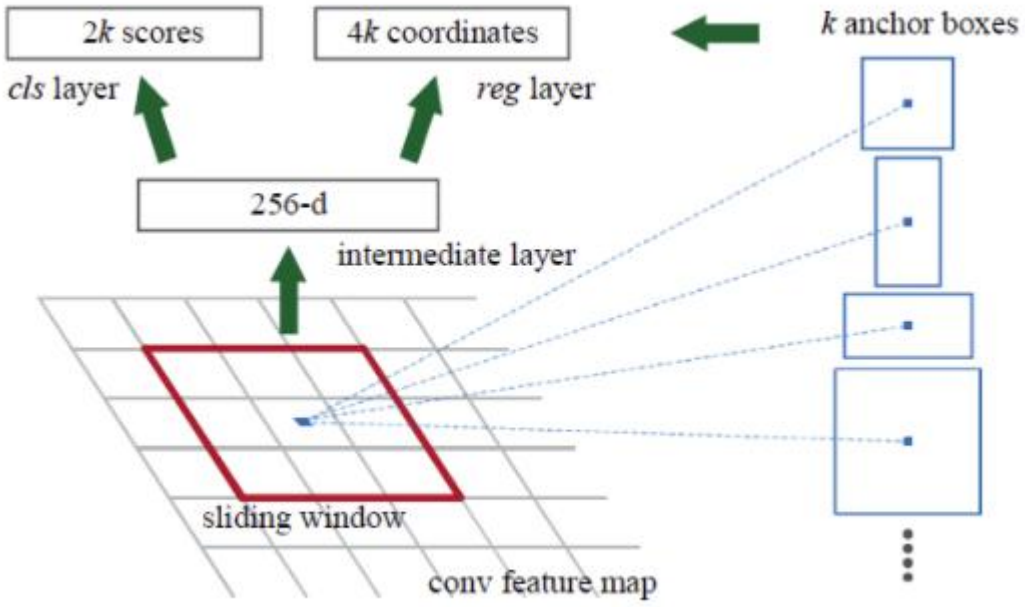

Figure2.4.1: *Faster-Rcnn anchors generate layer*

To train RPN, each anchor is assigned a binary label (is it a target). Positive labels are given to two types of anchors: (i) anchors with the highest IoU (Intersection-over-Union, ratio of intersection and union) that overlap with a certain ground truth (GT) bounding box (maybe less than 0.7), (ii) An anchor that overlaps with any GT bounding box with an IoU greater than 0.7. Note that a GT bounding box may

assign positive labels to multiple anchors. We assign negative labels to anchors with IoU ratios below 0.3 for all GT bounding boxes. Non-positive and negative anchors have no effect on training goals. With these definitions, the loss function for an image is defined as formula 2.4.1.1

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \quad 2.4.1.1$$

Here, i is the index of the anchor in a mini-batch, and p_i is the anchor, i is the predicted probability of the target. If the anchor is positive, the GT label p_i^* is 1, and if the anchor is negative, p_i^* is 0. t_i is a vector representing the four parameterized coordinates of the predicted bounding box, and t_i^* is the coordinate vector of the GT bounding box corresponding to the positive anchors. Classification loss L_{cls} is the log loss of two categories (target vs. non-target) the formula shows in 2.4.1.2

$$L_{cls}(p_i,) = -\log\left[p_i^* p_i + (1-p_i^*)(1-p_i)\right] \quad 2.4.1.2$$

For the regression loss L_{reg}, the defined calculation method is formula 2.4.1.3

$$L_{reg}(t_i,t_i^*) = R\,(t_i - t_i^*)] \quad 2.4.1.3$$

Where R function is smooth L1 function formula 2.4.1.4

$$smooth_{L1} = \begin{cases} 0.5x^2 & if \quad |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad 2.4.1.4$$

The term $p_i \star L_{reg}$ means that only a positive anchor (p_i^ ∗= 1) will have a regression loss, otherwise it will not (p_i^ ∗= 0). The output of the cls layer and the reg layer are respectively composed of $\{p_i\}$ and $\{t_i\}$. These two items are respectively normalized by N_{cls} and N_{reg} and a balance weight λ = 10. The size, that is, N_{cls} = 256, the normalized value of the reg term is the number of anchor positions, that is, N_{reg} ~ 2,400, so the cls and reg terms are almost equally weighted.

For regression, we learn to use 4 coordinates formula 2.4.1.5 and 2.4.1.6:

$$t_x = \frac{(x-x_a)}{w_a},\ t_y = \frac{(y-y_a)}{h_a},\ t_w = \log\left(\frac{w}{w_a}\right),\ t_h = \log\left(\frac{h}{h_a}\right) \quad 2.4.1.5$$

$$t_x^* = \frac{(x^*-x_a)}{w_a},\ t_y^* = \frac{(y^*-y_a)}{h_a},\ t_w^* = \log\left(\frac{w^*}{w_a}\right),\ t_h^* = \log\left(\frac{h^*}{h_a}\right) \quad 2.4.1.6$$

x, y, w, h refer to (x, y) coordinates, width, and height of the center of the bounding box. The variables x, x_a , x^* refer to the x coordinate of the predicted bounding box, anchor's bounding box, and GT's bounding box (the same is true for y, w, h). It can be understood as the regression from the anchor bounding box to the surrounding GT bounding box.

RPN is naturally implemented as a fully convolutional network [44], trained end-to-end through backpropagation and stochastic gradient descent (SGD) [55]. Each mini-batch consists of a single image containing many positive and negative samples. This can optimize the loss function of all anchors, but because the number of negative samples is much larger than the positive samples, using this method directly results in biased negative samples. Therefore, 256 anchors are randomly sampled in an image to calculate the mini-batch loss function. The ratio of the positive and negative anchors sampled is 1: 1. If the number of positive samples in an image is less than 128, the mini-batch is filled with negative samples. The initialization of the last convolutional layer and subsequent layers uses a Gaussian distribution with a zero mean standard deviation of 0.01. All other layers (ie, shared convolutional layers) are initialized by a pre-trained model on ImageNet classification [2]. RPN network overall structure can be seen in Fig 2.4.2

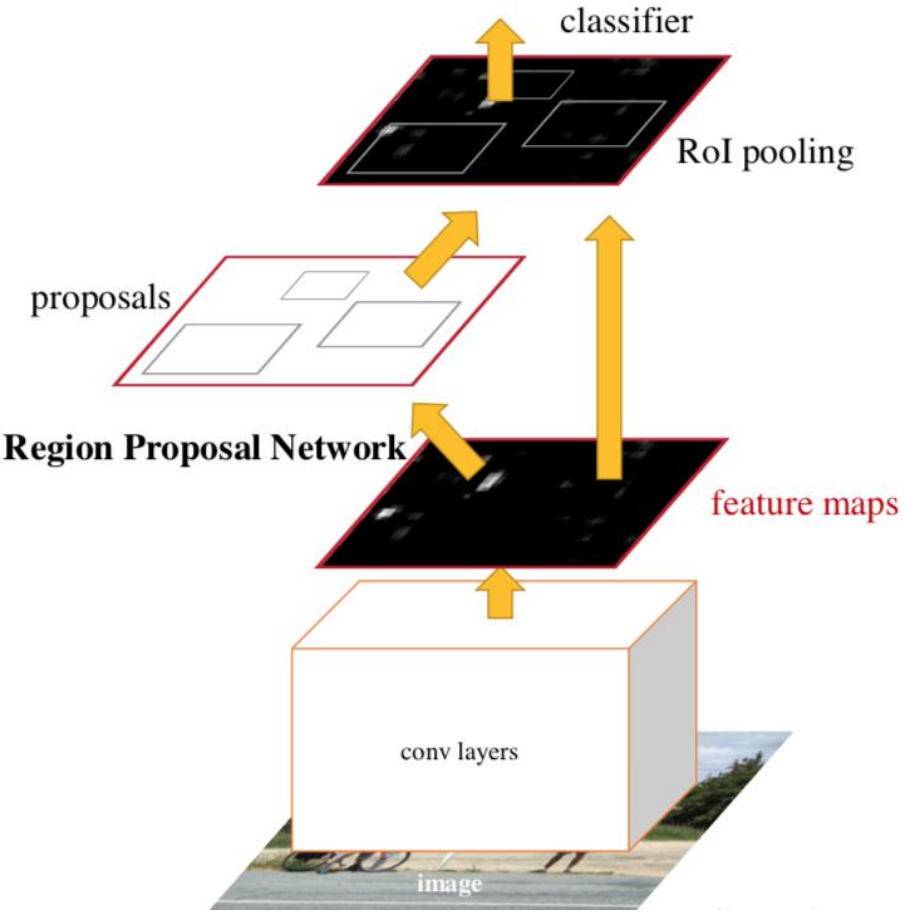

Figure 2.4.2: *Faster Rcnn RPN layer*

Network training, range proposal layer, and target detection networks are all on a fixed scale image [42, 43]. Each image of different sizes will be scaled before entering the network, so that their short sides are s = 600 pixels [43]. Multi-scale feature extraction

may improve accuracy but is not good for the trade-off between speed and accuracy [43]. For example, in the VGG network, one pixel on the output feature map of the last convolution layer for the scaled image corresponds to the receptive field of the original image of 16 * 16 pixels, which is equivalent to about a typical PASCAL image (~ 500x375). 10 pixels (600/16 = 375/10). Even this large step size has achieved good results, and if the downsampling multiple is smaller, the accuracy may be further improved.

For anchor, Faster-Rcnn uses 3 simple scales, the bounding box area is 128x128, 256x256, 512x512, and 3 simple aspect ratios, 1: 1, 1: 2, 2: 1. Note that the algorithm was designed using an anchor bounding box that is larger than the receptive field. With this design, our solution does not require multi-scale features or multi-scale sliding windows to predict large regions, saving considerable runtime. Figure 2.4.1 (right) shows the algorithm's ability to handle multiple scales and aspect ratios. Table 2.4.1 shows the average suggested box size (s = 600) learned by the ZF network for each anchor.

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

Table2.4.1: *Faster-Rcnn anchor in ZF network*

In actual processing, some anchors will exceed the boundaries of the image. During training, all anchors that cross image boundaries are ignored so that they do not affect the loss. For a typical 1000x600 image, there are almost 20k (~ 60x40x9) anchors in total. After ignoring the anchors that cross the border, there are only 6k anchors left for each image to be trained. If the anchor that crosses the boundary is not ignored during training, it will make the network very difficult to train, because these anchors that exceed the boundary of the image do not have any features, and the results cannot be fitted based on the image features. When testing, we still apply the full convolution RPN to the entire image, which may generate a suggestion box that crosses the boundary, and we crop it to the edge of the image.

Some RPN suggestion boxes overlap a lot with other suggestion boxes. In order to reduce redundancy, we use non-maximum suppression (NMS) based on the cls score of the proposed area. The IoU threshold of NMS is 0.7, so that only 2k suggested regions are left for each image. NMS does not affect the final detection accuracy, but greatly reduces the number of suggestion boxes.

Detection benchmark in PASCAL VOC2007 [56]. This dataset includes 20 target categories, approximately 5k trainval images and 5k test images. For ImageNet pre-trained networks, two networks were evaluated. One is the ZF network [40], which has 5 convolutional layers and 3 fc layers, and the second is the public VGG-16 model [34], which has 13 convolutional layers and 3 fc layers. It is mainly evaluated by mean Average Precision, mAP.

Table 2.4.2 shows the results of Faster R-CNN when trained and tested using various regional recommendations. These results use a ZF network. For the method of selective search (SS) [54], about 2k SS suggestion boxes are generated. For EdgeBoxes (EB) [57], we adjusted the default EB setting to 0.7IoU to generate suggestion boxes. The mAP of SS was 58.7% and the mAP of EB was 58.6%. RPN and Fast R-CNN achieved competitive results, with mAP of 59.9% when using 300 suggested frames. Using RPN achieves a faster detection system than using SS or EB, because there are shared convolution calculations; fewer suggestion boxes.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2k	SS	2k	58.7
EB	2k	EB	2k	58.6
RPN+ZF, shared	2k	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2k	RPN+ZF, unshared	300	58.7
SS	2k	RPN+ZF	100	55.1
SS	2k	RPN+ZF	300	56.8
SS	2k	RPN+ZF	1k	56.3
SS	2k	RPN+ZF (no NMS)	6k	55.2
SS	2k	RPN+ZF (no *cls*)	100	44.6
SS	2k	RPN+ZF (no *cls*)	300	51.4
SS	2k	RPN+ZF (no *cls*)	1k	55.8
SS	2k	RPN+ZF (no *reg*)	300	52.1
SS	2k	RPN+ZF (no *reg*)	1k	51.3
SS	2k	RPN+VGG	300	59.2

Table 2.4.2: *Test results of PASCAL VOC2007 test set (trained in VOC2007 trainval). The detector is Fast R-CNN and ZF, but is trained and tested using various suggestion box methods.*

The experiments also evaluated the effect of a more robust VGG network on the RPN suggestion box. And still use the above SS + ZF detector for comparison. mAP increased from 56.8% (using RPN + ZF) to 59.2% (using RPN + VGG). This is a satisfactory result because it shows that the quality of the suggested box of RPN + VGG is better than that of RPN + ZF. Since the RPN + ZF suggestion box is competitive with SS (58.7% when used consistently in training and testing).

VGG-16 detection accuracy and running time. Table 2.4.3 shows the result of VGG-16 suggestion box and test. Using RPN + VGG, the result of Faster R-CNN on unshared features is 68.5%, which is slightly higher than the SS benchmark. As shown above, this is because the suggestion box generated by RPN + VGG is more accurate than SS. Unlike the pre-defined SS, RPN is trained in real time and can benefit from a better network. For the feature-sharing variant, the result is 69.9%-better than a strong SS benchmark, and the suggestion box is almost lossless. Further training RPN on the union of PASCAL VOC2007 trainval and 2012 trainval, mAP is 73.2%. When training on the

union of VOC 2007 trainval + test and VOC2012 trainval as in [5], there was 70.4% mAP on the PASCAL VOC 2012 test set (Fig 2.4.6-Fig2.4.9).

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9†	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**198**
RPN+VGG, shared	300	07+12	**73.2**	**198**

Table 2.4.3: *shows the detection results on the PASCAL VOC 2007 test set. The detectors are Fast R-CNN and VGG16. Training data: "07": VOC2007 trainval, "07 + 12": Union of VOC 2007 trainval and VOC 2012 trainval.*

This efficient RPN algorithm shares the convolutional features with the detection network that follows, and the range proposal is almost lossless. This deep learning-based object detection system runs at 5-17 fps, which is a very important progress in the field of object detection. Here are some renderings

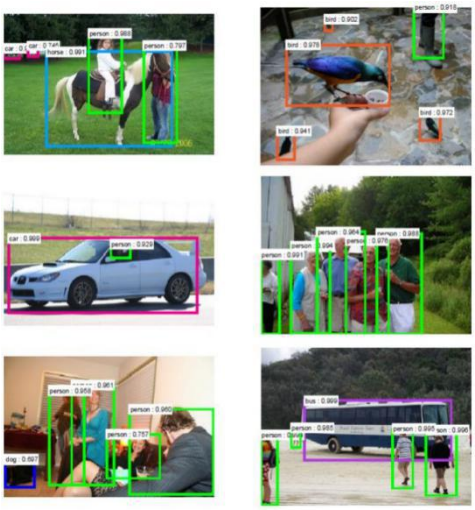

Figure 2.4.6: *PASCAL VOC 2012 test set test results. The detectors are Fast R-CNN and VGG16. Training data: "07": VOC 2007 trainval, "07 ++ 12": Union of VOC 2007 trainval + test and VOC 2012 trainval.*

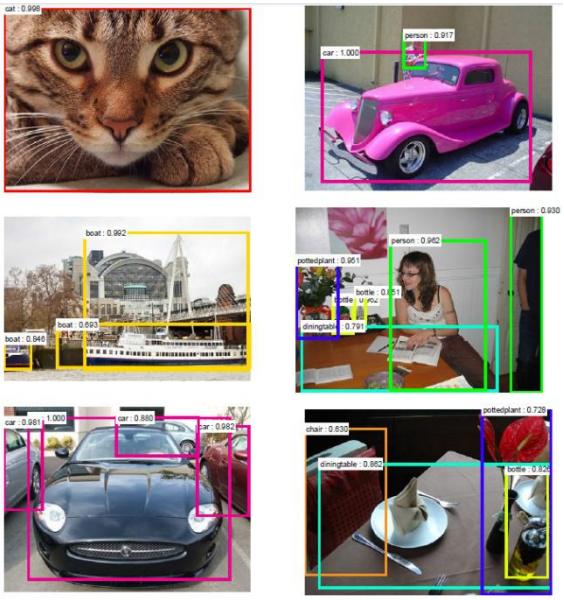

Figure 2.4.7: *PASCAL VOC 2012 test set test results. The detectors are Fasi R-CNN and VGG16. Training data: "07": VOC 2007 trainval, "07 ++ 12": Union of VOC 2007 trainval + test and VOC 2012 trainval.*

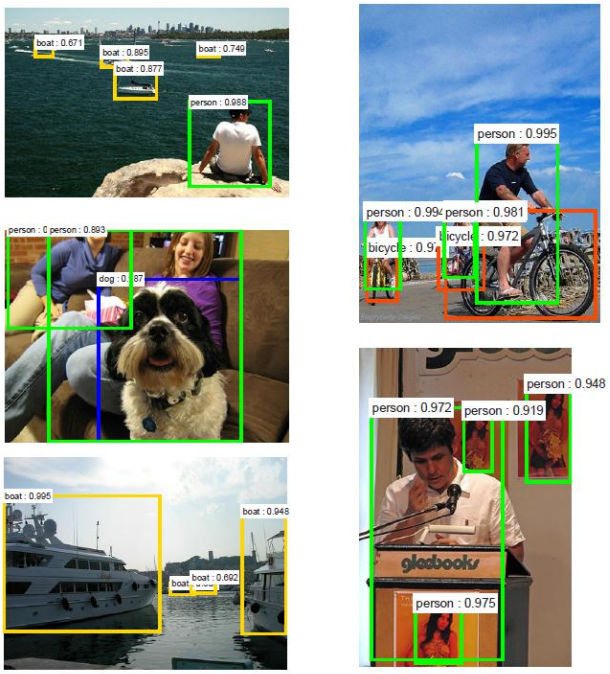

Figure 2.4.8: *PASCAL VOC 2012 test set test results. The detectors are Fasi R-CNN and VGG16. Training data: "07": VOC 2007 trainval, "07 ++ 12": Union of VOC 2007 trainval + test and VOC 2012 trainval.*

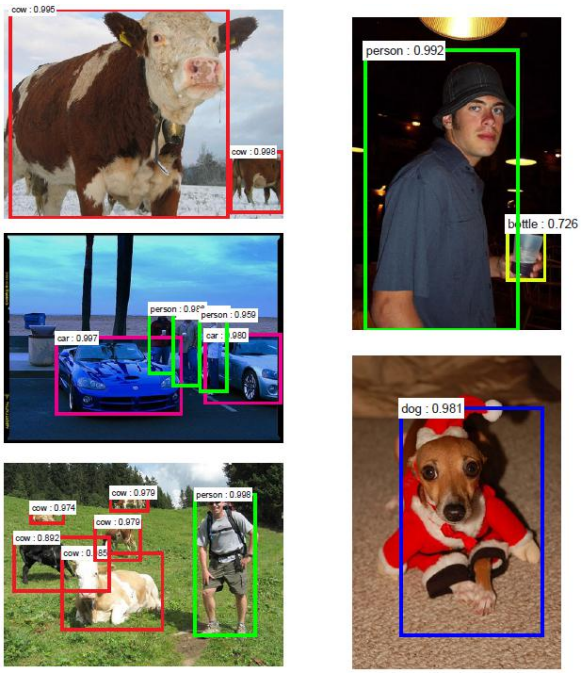

Figure 2.4.9: *PASCAL VOC 2012 test set test results. The detectors are Fast R-CNN and VGG16. Training data: "07": VOC 2007 trainval, "07 ++ 12": Union of VOC 2007 trainval + test and VOC 2012 trainval.*

2.5.2. You only look once

YOLO[36] is an algorithm born in the same era as Faster-Rcnn[35]. When it was born, the RCNN series of algorithms have developed to the third generation. From the technical characteristics, YOLO[36] is undoubtedly another significant algorithm in the target detection algorithm. The biggest contribution of this algorithm is that it truly achieves real-time. When the Faster-Rcnn algorithm achieves an FPS of 5-17, YOLO has reached 45, and Fast YOLO[36] has even reached 155 FPS. Although Faster-Rcnn[35] algorithms are already fast, they are still insufficient in many fields, such as autonomous driving, which has very high requirements for speed. However, in order to achieve such a fast speed, YOLO also sacrificed high accuracy. The accuracy of YOLO may only reach the accuracy of the first-generation RCNN. Fortunately, it has been 4 years since YOLO v1[36] was proposed, and YOLO has already developed to the third generation. The accuracy of YOLO v3[58] has reached the level of Faster Rcnn, and it still maintains the speed of v1.

Object detection systems before YOLO used classifiers to complete object detection tasks. In order to detect an object, these object detection systems need to use the classifier on different locations and different sizes of bounding boxes to evaluate whether there is an object. For example, a DPM[59] system uses a sliding window to slide uniformly over the entire image, and uses a classifier to evaluate whether there is an object.

Other methods proposed after DPM, such as the R-CNN[60] method, use a region proposal to generate potential bounding boxes that may contain objects to be detected in the entire image, then use a classifier to evaluate these boxes, and then use post-processing to improve bounding boxes , Eliminate duplicate detection targets, and re-score boxes based on other objects in the entire scene. The entire process is slow to execute, and because these steps are trained separately, it is difficult to optimize detection performance.

YOLO treats the object detection task as a regression problem, and directly obtains the coordinates of the bounding box, the confidence of the object contained in the box, and class probabilities through all pixels of the entire picture. With YOLO, each image only needs to be input to the neural network to find out which objects are in the image and the locations of these objects.

YOLO is very simple as Fig 2.5.1, a single convolutional network can simultaneously predict multiple bounding boxes and the class probabilities of these boxes. This unified model has many advantages over traditional object detection methods.

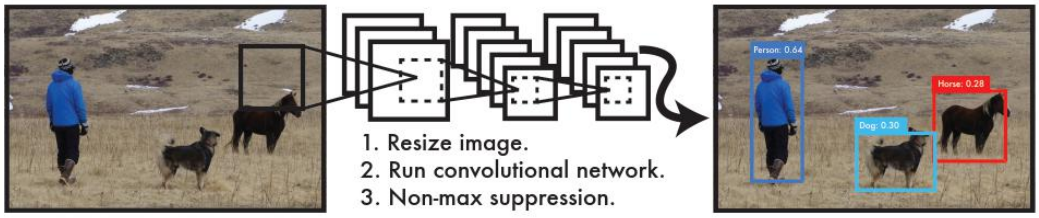

Figure 2.5.1: *YOLO detection system. Processing images with YOLO is simple and straightforward. Our system (1) adjusts the size of the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the result detection by the confidence of the model.*

YOLO has several advantages: 1. YOLO is very fast. Because there is no complicated detection process, you only need to input the image to the neural network to get the detection result. YOLO can complete the object detection task very quickly. The standard version of YOLO can reach 45 FPS on the Titan X GPU. Faster Fast YOLO detection can reach 155 FPS. Moreover, YOLO's mAP is more than double that of other previous real-time object detection systems. 2. YOLO can avoid background errors. Unlike other object detection systems that use a sliding window or region proposal, the classifier can only get local information about the image. YOLO can see the information of an entire image during training and testing, so YOLO can make good use of context information when detecting objects. Compared with Fast R-CNN [43], the background error of YOLO is less than half that of Fast R-CNN[43]. 3. YOLO can learn the generalization characteristics of objects.

Despite these advantages, YOLO also has some disadvantages: 1. The accuracy of YOLO is lower than other state-of-the-art object detection systems. 2. YOLO is prone to object positioning errors. 3. YOLO is not good at detecting small objects (especially dense small objects, because a grid can only predict 2 objects).

YOLO uses features of the entire image to predict each bounding box. It also predicts all bounding boxes of all classes simultaneously. This means that YOLO's network can fully understand all the images in the image and all objects in the image. YOLO design can achieve end-to-end training and real-time speed, while maintaining high average accuracy. YOLO divides the input image into S * S grids, and each grid is responsible for detecting objects whose center falls in the grid. Each grid predicts B bounding boxes and the confidence scores of these bounding boxes. The confidence scores reflect the model's predictions for this grid: whether the grid contains objects, and how accurate the coordinates of this box are predicted. The formula is 2.5.1.1

$$\text{confidence} = P_r(Object) * IOU_{pred}^{truth} \quad 2.5.1.1$$

If there are no objects in this grid, the confidence score should be 0. Otherwise, the confidence score is the IOU (intersection over union) between the predicted bounding box and the ground truth box.

YOLO has 5 predictions for each bounding box: x, y, w, h, and confidence. The coordinates x, y represent the relative values of the predicted bounding box center and the grid boundary. The coordinates w, h represent the ratio of the predicted width and height of the bounding box to the width and height of the entire image. Confidence is the predicted IOU value of the bounding box and ground truth box. Each grid also predicts C conditional class probability ($P_r(Class_i|Object)$). That is, under the premise that a grid contains an Object, the probability that it belongs to a certain class. We only predict a set (C) of class probabilities for each grid, regardless of the number of boxes B. The flow of the entire YOLO algorithm is shown in Fig 2.5.2.

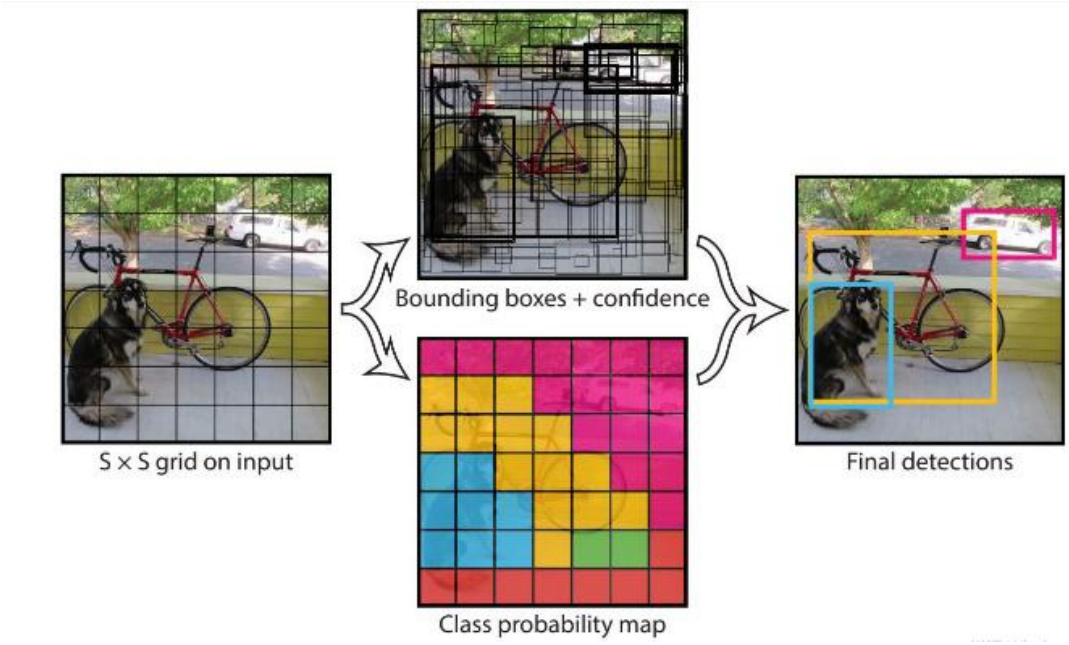

Figure 2.5.2: *Our YOLO system models detection as a regression problem. It divides the image into S × S grids, and each grid unit predicts B bounding boxes, the confidence of these boxes, and the class C probability. These predictions are encodea as S × S × (B * 5 + C) tensor. To evaluate YOLO on PASCAL VOC, we use S = 7 and B = 2. PASCAL VOC has 20 tag classes, so C = 20. Our final prediction is a 7 × 7 × 30*

The YOLO network borrows from GoogLeNet classification network structure. The difference is that YOLO does not use an inception module, but uses a 1x1 convolutional layer (here the 1x1 convolutional layer exists for cross-channel information integration) + 3x3 convolutional layers simply replace. The complete network structure is shown in Fig 2.5.3. The final output is a 7 * 7 * 30 tensor.

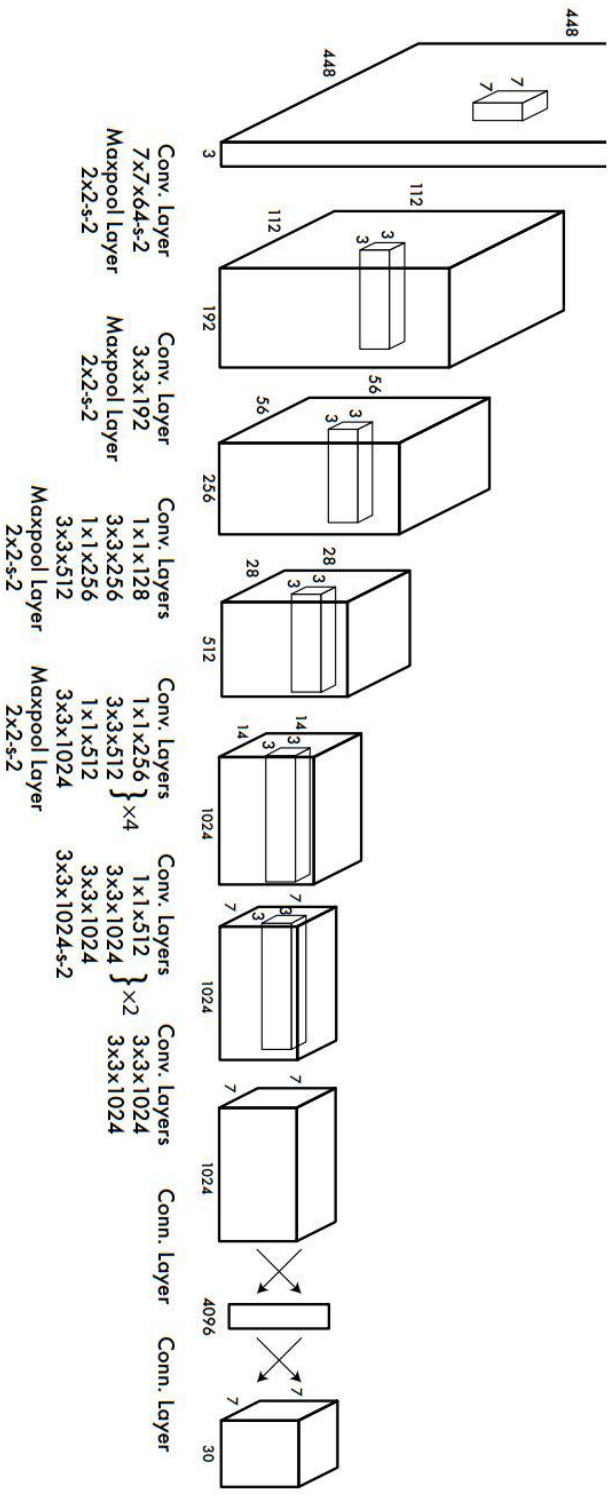

Figure 2.5.3: *The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1， convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification， task at half the resolution (224 × 224 input image) and then double the resolution for detection*

The training strategy of the YOLO network is to first use the ImageNet 1000-class classification task dataset Pre-train convolutional layer (darknet). Use the first 20 convolutional layers in the above network, add an average-pooling layer, and finally add a fully connected layer as the Pretrain network. After training for about a week, the accuracy of the Top-5 validation data set in ImageNet 2012 reached 88%. This result is comparable to the effect of GoogleNet. The first 20 convolutional layers of the result of Pretrain are applied to Detection, and the remaining 4 convolutional layers and 2 fully connected are added. At the same time, in order to obtain more refined results, the resolution of the input image was increased from 224 * 224 to 448 * 448. All prediction results are normalized to 0 ~ 1, and Leaky RELU is used as the activation function. Leaky RELU can solve the problem of gradient disappearance of RELU. Leaky RELU's formula is 2.5.1.2:

$$\Phi(x) = \begin{cases} x & if\ x > 0 \\ 0.1x & otherwise \end{cases} \qquad 2.5.1.2$$

The design goal of the loss function is to achieve a good balance between the three aspects of coordinates (x, y, w, h), confidence, and classification. Simply using sum-squared error loss to do this will have the following disadvantages:

a) An 8-dimensional localization error is as important as a 20-dimensional classification error, which is obviously unreasonable.

b) If there are no objects in some grids (there are many such grids in a picture), then the confidence of the bounding box in these grids will be set to 0. Compared with fewer grids with objects, these The contribution of the grid containing objects to the gradient update will be far greater than the contribution of the grid containing objects to the gradient update, which will cause the network to be unstable or even divergent. To solve these problems, the definition of YOLO's loss function is shown in Fig 2.5.4

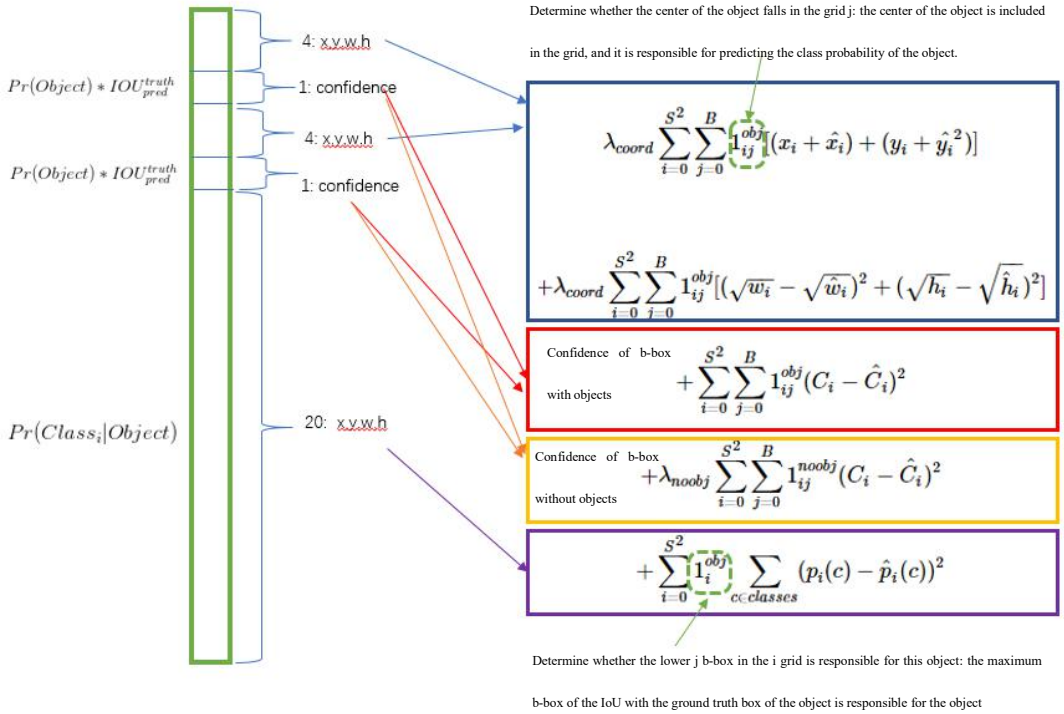

Figure 2.5.4: *YOLO LOSS*

More attention is paid to 8-dimensional coordinate prediction, and a larger loss weight is given to these losses, denoted as λcoord. (The blue box in the Fig 2.5.4) For the confidence loss of the b-box without an object, a small loss weight is given, and it is recorded as λnoobj. (Orange box in the figure above) The confidence weight of the b-box with the object (the red box in the figure above) and the loss weight of the category (the purple box in the figure above) are normally taken as 1. For different sizes of b-box predictions, compared with large b-box predictions, the smaller box predictions have the same impact on IOUs. The sum-square error loss is the same for the same offset loss. In order to alleviate this problem, the author used a clever method, which is to take the square root of the width and height of the box instead of the original height and width. As shown in the figure 2.5.5: The small b-box has a small horizontal axis value. When the offset occurs, the loss on the y axis (green in the Fig 2.5.5) is larger than the big box (red in the Fig 2.5.5).

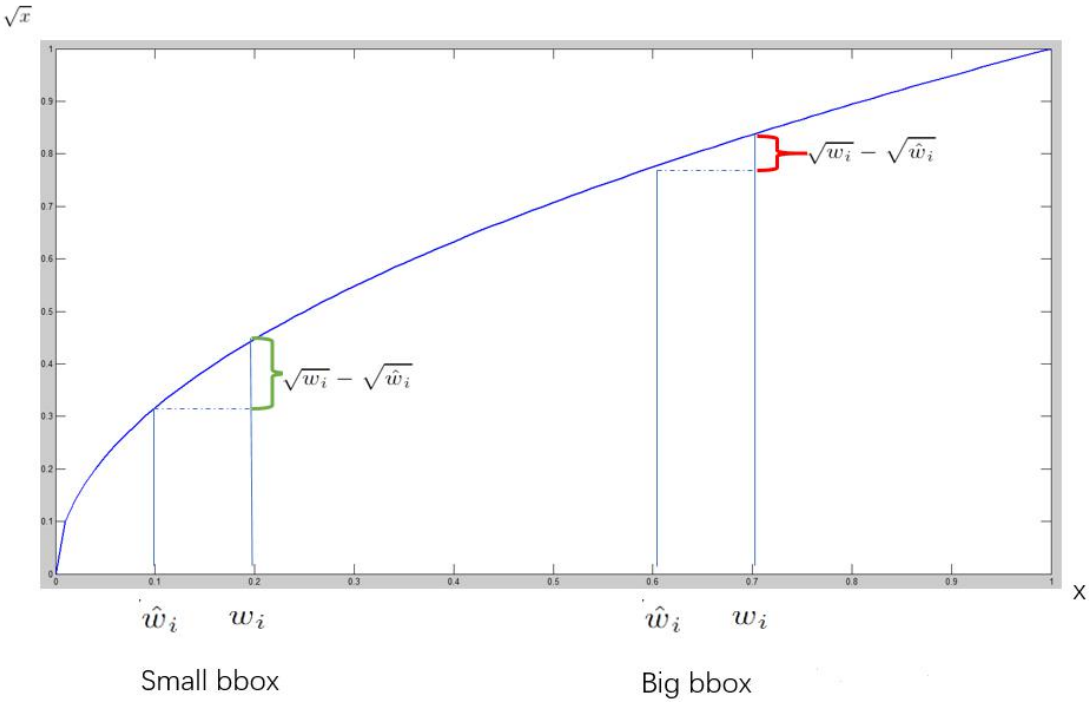

Figure 2.5.5: *Mean Square Error*

In YOLO, each grid predicts multiple bounding boxes, but in the training of the network model, it is hoped that each object is finally predicted by a bounding box predictor. Therefore, which predictor currently has the largest IOU for the bounding box and ground truth box, this predictor is responsible for predicting the object. This allows each predictor to be specifically responsible for specific object detection. As the training progresses, each predictor will get better and better at predicting the types of objects with specific object sizes and aspect ratios.

The author also gives the recognition error ratio of YOLO and Fast RCNN in various aspects, as shown in Fig 2.5.6. YOLO's false positive rate of background content (4.75%) is much lower than that of fast rcnn (13.6%). However, the positioning accuracy of YOLO is poor, accounting for 19.0% of the total error rate, while fast rcnn is only 8.6%. The results of the top algorithms of the time on the VOC2012 dataset are shown in Table 2.5.1.

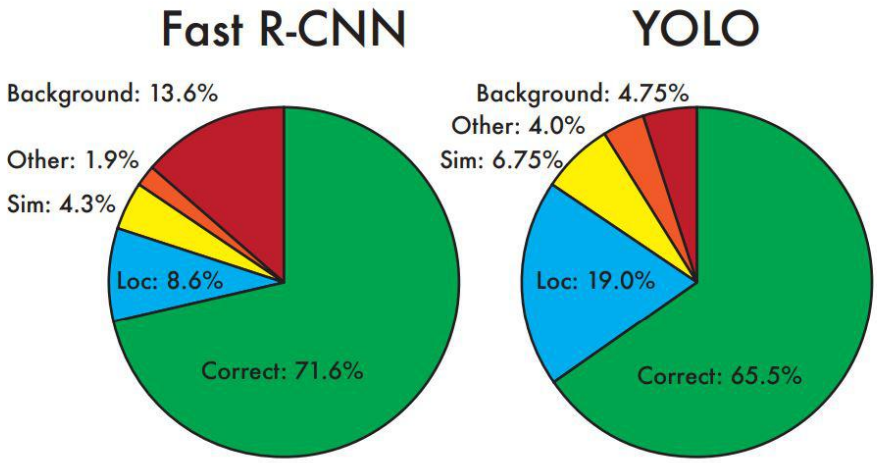

Figure 2.5.6: *result of YOLO and fast R-CNN*

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Table 2.5.1: *different state-of-the-art algorithm result*

Summary: YOLOv1's most groundbreaking contribution is to solve object detection as a regression problem. After an inference of the input image, we can get the position of all objects in the image, their category and the corresponding confidence probability. The rcnn-based algorithm solves the detection result in two parts: the object category (classification problem), and the object position is the bounding box (regression problem), so YOLO's target detection speed is very fast. YOLO is still a speed-for-accuracy algorithm, and the accuracy of target detection is not as good as RCNN.

2.5.3. Single shot multibox detector

SSD[37] is a method to detect objects using a single deep neural network. SSD is a combination of YOLO and Faster-Rcnn. This algorithm combines the high speed of YOLO and the anchor mechanism of Faster-Rcnn, but simplifies the RPN layer, so the SSD gets a very high accuracy and still very fast. This makes this algorithm quickly popular in the field of object detection. For VOC2007, at 300 × 300 input, the SSD reached 72.1% mAP at 58 FPS on Nvidia Titan X[37], and 500 × 500 input SSD reached 75.1% mAP[37], which is better than similar prior art Faster R-CNN models

After the rapid development of deep learning-based object detection algorithms, more target detection ideas are variations of the following methods: assuming bounding boxes, resampling pixels or features for each box, and then applying a high-quality classifier. After the selective search[54] method, Faster R-CNN[35] achieved leading results in the detection of PASCAL VOC, MSCOCO, and ILSVRC. This process became a milestone in the field of detection with deeper features, as described in resnet[28]. Although accurate, these methods are too computationally expensive for embedded systems, and even for high-end hardware, they are too slow for real-time or near-real-time applications. The detection speed of these methods is usually measured in frames per second, and the high-precision detector (basic Faster R-CNN) runs as fast as 7 frames per second (FPS). But so far, the significantly increased speed has only come at the cost of significantly reduced detection accuracy (YOLO).

SSD does not resample pixels or features assumed by the bounding box, but it is as accurate as this. This significantly improves the high-precision detection speed (in the VOC2007 test, 72.1% mAP at 58 FPS, 73.2% at Faster R-CNN 7 FPS, and 63.4% at YOLO 45 FPS). The fundamental improvement in speed comes from the elimination of the bounding box proposal and the subsequent pixel or feature resampling phase. Improvements in SSD include using filters with different aspect ratios, predicting object categories and offsets in the bounding box, and applying these filters to multiple feature maps later in the network in order to perform multi-scale detection. With these modifications, people can use a relatively low-resolution input to achieve high-precision detection, further improving processing speed. Although these contributions may seem small independently, author notice that the resulting system improves the accuracy of high-speed detection of PASCAL VOC, from 63.4% mAP of YOLO to 72.1% mAP of our proposed network. Compared with that era, this is a great improvement in detection accuracy. In addition, significantly increasing the speed of high-quality inspections can broaden the useful range of computer vision.

SSD is based on a feedforward convolutional network, which generates a fixed-size bounding box set and the score of object categories in the box, which are suppressed by non-maximum values to produce the final detection. The head network is based on the standard architecture of high-quality image classification (for example: vgg, resnet). The author of the SSD calls it the basic network (the author's experiments used the VGG-16

network as the basis. Other networks should also produce good results). The author adds auxiliary structures to the network, resulting in detections with the following main features:

Multi-scale feature maps for detection: Select an excellent convolutional neural network to intercept the part before the full connection as the basic network. The size of these layers is gradually reduced to obtain the predicted values of multiple scale detection.

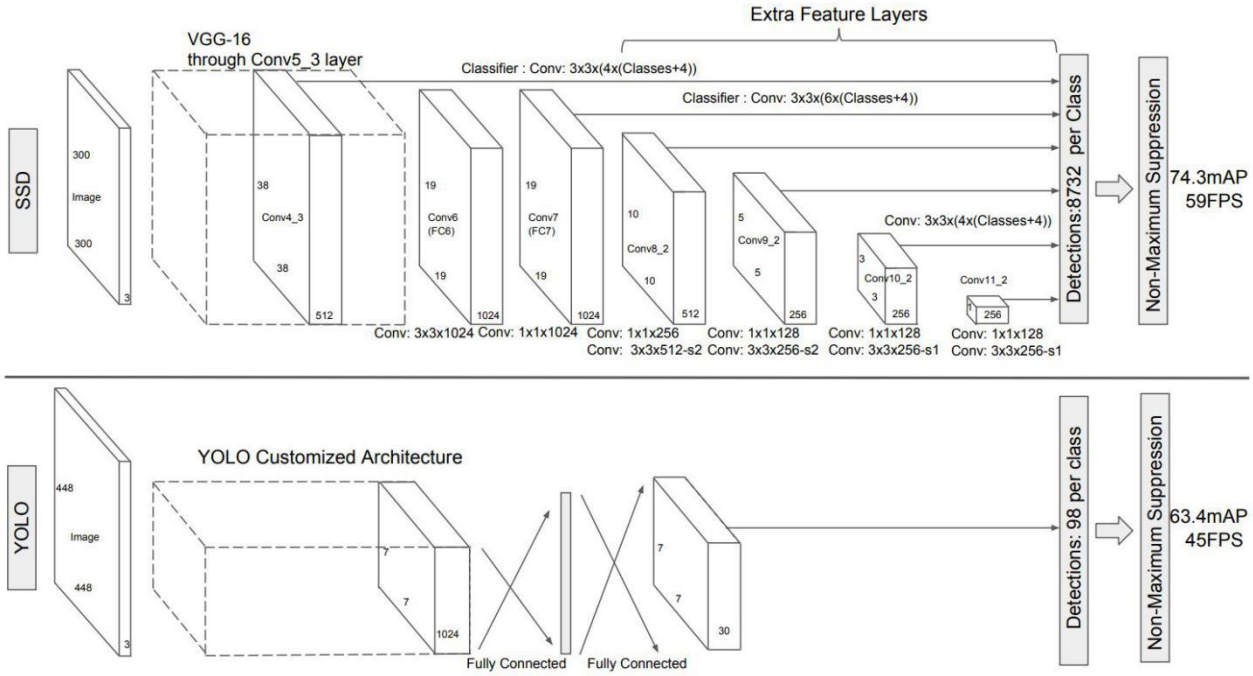

Figure 2.6.1: *A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 × 300 input size significantly outperforms its 448 × 448 YOLO counterpart in accuracy on VOC2007 test while also improving the speed*

Convolutional predictors for detection: These are indicated in the SSD network architecture in Fig 2.6.1. For m × n feature layers with p channels, a 3 × 3 × p convolution kernel convolution operation is used to generate scores of different classes and coordinate offsets from the default box. At each m × n size location using a convolution kernel operation, an output value is generated. The bounding box offset output value is measured relative to the default box, and the default box position is relative to the feature map (see the architecture of YOLO [36], using a fully connected layer in the middle instead of the convolution filter used in this step).

Default boxes and aspect ratios: The SSD associates a set of default bounding boxes with each feature map unit in the last layers of the convolutional network. The default box performs a convolution operation on the feature map, so that the position of each box relative to its corresponding cell is fixed. In each feature mapping unit, the SSD predicts the offset from the default box shape in the cell, and the classification score of the object in each box. Specifically, for each box in the k boxes at a given position, a class c score and 4 offsets from the original default box are calculated. This requires a total of (c + 4) k filters at each position in the feature map, and produces (c + 4)* k*m*n outputs for the m × n feature map. For a description of the default box, see Figure 2. The default box of SSD is similar to the anchor boxes used in Faster R-CNN [2]. Using different default box shapes in multiple feature maps can effectively disperse the possible output box shape space.

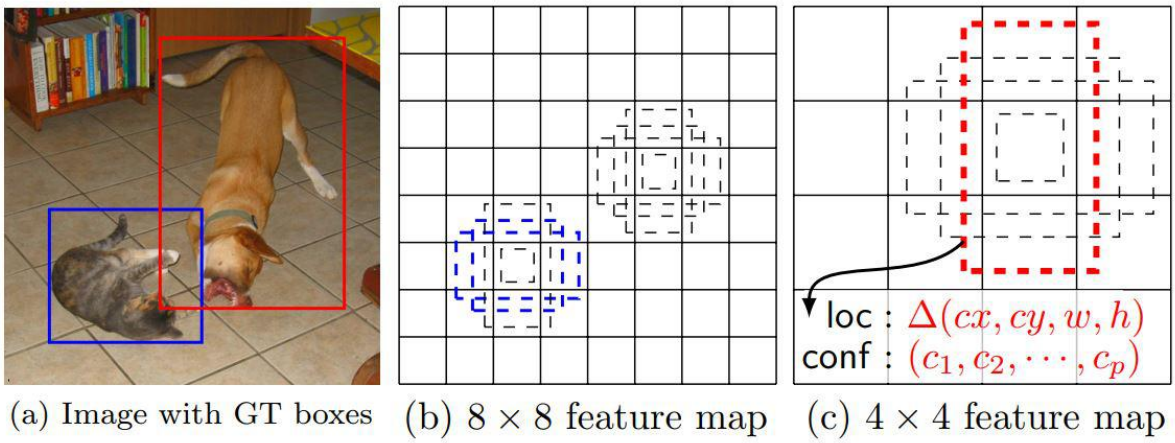

Figure 2.6.2 : *SSD framework. (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8 × 8 and 4 × 4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ((c1, c2, · · · , cp)). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [43]) and confidence loss (e.g. Softmax).*

The key difference between training SSD and training a typical classifier using region proposal and pooling is that the real label information needs to be assigned to a specific output in a fixed detector output set. Faster R-CNN [35] 's region proposal stage and YOLO [36]' s training stage also need similar labels.

Matching method: During training, the SSD needs to establish the correspondence between the real label and the default box. For each real label box, the SSD selects from the default box. SSD matches each real label with the largest overlap of anchor. It ensures that each real label box has a matching default box. The overall target loss function is the

weighted sum of the position loss (loc) and the confidence loss (conf) the formula is 2.6.1.1

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right) \qquad 2.6.1.1$$

Where N is the number of matching default boxes, and position loss is the smooth L1 loss between the prediction box (l) and the true label value box (g) parameters [43]. Similar to Faster R-CNN [35], the offset of the center (cx; cy) of the default border (d)

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m) \qquad 2.6.1.2$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

and its width (w) and height (h) the formula is 2.6.1.2

Confidence loss is a cross-validation of softmax loss on multi-category confidence (c)

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \qquad 2.6.1.3$$

and weighting term α set to 1, the formula is 2.6.1.3.

Generally. SSD integrates the thoughts of YOLO and Faster-RCNN

2.6. Deep learning-based video analysis architecture

The main topic of AI application in video is video understanding, which tries to solve the problem of video semantic understanding, including:

a). Video structural analysis: video is segmented into frames, shots, scenes, stories, etc., so that it can be processed and expressed at multiple levels.

b). Target detection and tracking: for example, vehicle tracking is mostly used in the field of security.

c). Character recognition: recognize the characters in the video.

d). Action recognition: recognize the actions of the characters in the video.

e). Emotional semantic analysis: that is, what kind of psychological experience the audience will have when watching a video.

Most of the content information in the video and live broadcast is people + scene + action + voice. How to express the content with effective features is the key to video understanding. There are a lot of traditional manual features. At present, IDT (improved dense trajectories) has a good effect. Deep learning is very good for the expression of image content, and there are corresponding methods for the expression of video content. Here are the main technical methods in recent years.

1. Recognition method based on single frame:

One of the most direct methods is to cut the video frame, and then carry out deep learning expression based on a single frame. A certain frame of the video gets a recognition result through the network. However, a single frame is a small part of the whole video, especially when the frame is not very distinguishable, or some images are irrelevant to the video theme, the classifier will recognize the error. Therefore, learning the expression of video time domain is the main factor to improve video recognition. Of course, it can only be distinguished in the video with strong motion, and only by the features of the image in the more static video.

2. Recognition method based on CNN extended network:

Its overall idea is to find a pattern in the time domain to express local motion information in CNN framework, so as to improve the overall recognition performance. For example, a convolution network convolutes the image sequence with $M * N * 3 * T$ (where $M * N$ is the resolution of the image, 3 is the channels of the image, and T is the number of frames involved in the calculation), which is equivalent to taking into account the total T frames of the surrounding image in a single image feature extraction. This CNN has one more time dimension than the ordinary CNN, which is called 3dcnn. Its overall accuracy is improved by about 2% compared with single frame CNN, especially

in motion rich videos, such as wrestling, climbing pole and other strong motion video types, which also proves that the motion information in the feature contributes to the recognition. In the implementation, the network architecture can add multi-resolution processing methods, which can improve the speed.

3. Recognition method of two-way CNN

This is actually two independent neural networks. Finally, average the results of the two models. One model is image-based CNN model. These CNN models are usually pre train on Imagenet data, and then adjust parameters on the last layer on video data. Another CNN network is to stack the optical flow information of several consecutive frames as CNN input. In addition, it uses multi task learning to overcome the problem of insufficient data. This method is the most accurate method to solve video analysis, but its disadvantage is that the speed is very slow.

4. Recognition method based on LSTM

Its basic idea is to integrate the single frame image into CNN and then use LSTM to integrate the last layer features. CNN here does not use the features of the full connection layer for fusion, because the features after the full connection layer have lost the information on the timeline. Compared with method 2, on the one hand, it can fuse CNN features for a longer time to express longer videos; on the other hand, method 2 does not consider the sequence of frames entering the network at the same time, and the network can effectively express the sequence of frames through the memory unit introduced by LSTM.

2.6.1. 3D ConvNets

3DCNN first appeared for Human Action Recognition [61]. Most video analytics rely on the expertise of designers to build complex human features. The actual application scenario must be controllable. Convolution neural network can directly act on the original input, so the image features are generated by the network itself. However, before 3dcnn is proposed by the author, the convolution network model is usually limited to two-dimensional images. In this paper, a new 3dcnn is proposed to deal with motion recognition. In this model, time and space information are extracted simultaneously by three-dimensional convolution, and motion information between consecutive frames is captured. It achieves better performance without relying on manual functions.

Before the emergence of convolutional neural network, video analysis mostly follows the traditional pattern recognition pattern, which consists of two steps. The first step is to calculate complex manual features from the original video frame. The second step is to classify learning according to these characteristics. In a real scene, it is difficult to know which feature is important to the current task, and the selection of feature is closely related to this problem. Especially in human motion recognition, different motion categories may have great differences due to their different appearance and motion mode.

CNN is mainly used for two-dimensional images. In the video recognition based on 2D CNN, a video frame is directly regarded as a still image, and CNN operates on a single frame. In fact, this method has been applied to video analysis of embryo development. However, this method does not consider the motion information between consecutive frames. In order to combine information effectively in video motion analysis, the author proposes to apply 3D convolution technology to CNN in order to capture the dimensions with time and space characteristics for recognition. 3dcnn proposes a framework for generating multi-channel information between adjacent video frames, and performs convolution and desampling operations on each channel. The final feature of multi-channel is composed of the obtained information. Another advantage is that the efficiency of the identification stage is particularly high because of the feedback characteristics of CNN model.

The author's experiments show that on the TRECVID dataset [62], the 3DCNN model is better than other methods, and on the KTH dataset [63], it does not rely on manual features and achieves better performance. These indicate that the 3DCNN model is in the real world environment is more effective. This experiment also suggests that for most tasks, the 3DCNN model performs significantly better than the frame-based 2DCNN. The author also observes the difference in performance between 3DCNN and other methods. When the positive training samples are unbalanced (when the number of positive samples is small), the difference is larger, and 3DCNN is better.

2D convolution operation is to extract features in the local neighborhood of the previous feature map. Then apply an added bias value and pass the result to a sigmoid

function. At the position (x, y), on the i layer and the j feature map, its unit is expressed as v_{ij}^{xy} and is given by the following formula 2.7.1.1

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+y)(y+q)}\right) \quad 2.7.1.1$$

Among them, tanh is a hyperbolic tangent function, b_{ij} is an offset value for this feature map, m is the coordinates of a collection of i-1 feature maps connected to the current feature map, and w_{ijm}^{pq} is the first m feature maps, weights at positions (p, q), P_i and Q_i are the height and width of the kernel. In the downsampling layer, on the feature map of the previous layer, the resolution of the feature map is reduced through pooling, thereby increasing the invariance of the input distortion, a CNN architecture can stack multiple layers of convolution and desampling in an alternating manner. The parameters of the CNN, such as the bias value b_{ij} and the kernel weight w_{ijm}^{pq}, are usually trained by the method of directional propagation.

In 2DCNN, the convolution operation only computes features on a two-dimensional space. When dealing with video analysis problems, you need to capture motion information between consecutive frames. For this purpose, the 2D convolution is changed to a 3D convolution operation to capture features from both time and space dimensions. The 3D convolution operation is realized by convolving a cube formed by stacking multiple consecutive frames at the same time with a 3D convolution kernel. Through this construction, the feature map on the convolutional layer is connected to multiple consecutive frames of the previous layer to capture motion information. Formally, at the (x, y, z) position of the i and j feature maps, the value is given by the following formula 2.7.1.2

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \quad 2.7.1.2$$

Among them, R_i represents the size of the 3D kernel in the time dimension, and Among them, R_i represents the size of the 3D kernel in the time dimension, and w_{ijm}^{pqr} is the value of the (p, q, r) convolution kernel connected to the m feature map in the previous layer. A comparison of 2D and 3D convolution operations can be seen in Fig 2.7.1 is the value of the (p, q, r) convolution kernel connected to the m feature map in the previous layer.

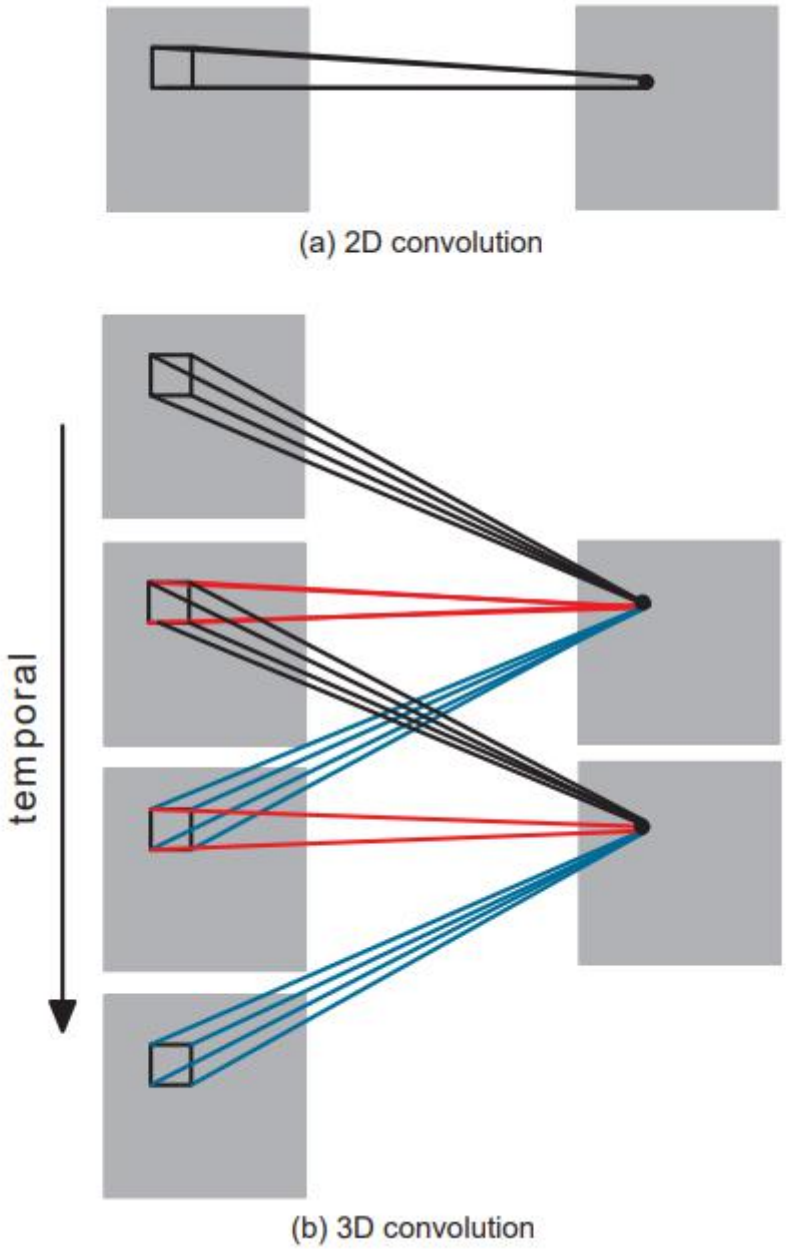

Figure 2.7.1: *Comparison of 2D (a) and 3D (b) convolutions. In (b) the size of the convolution kernel in the temporal dimension is 3, and the sets of connections are color-coded so that the shared weights are in the same color. In 3D convolution, the same 3D kernel is applied to overlapping 3D cubes in the input video to extract motion features*

A 3D convolution kernel can only extract one feature from a frame cube, so the kernel weights can be applied repeatedly to the entire cube. The general design principle of CNN is to increase the number of feature maps by generating multiple types of features from the same set of low-level feature maps. Similar to the 2D convolution operation, this can be achieved by using multiple 3D convolution operations at the same position in the previous layer and with different kernels (Fig 2.7.2).

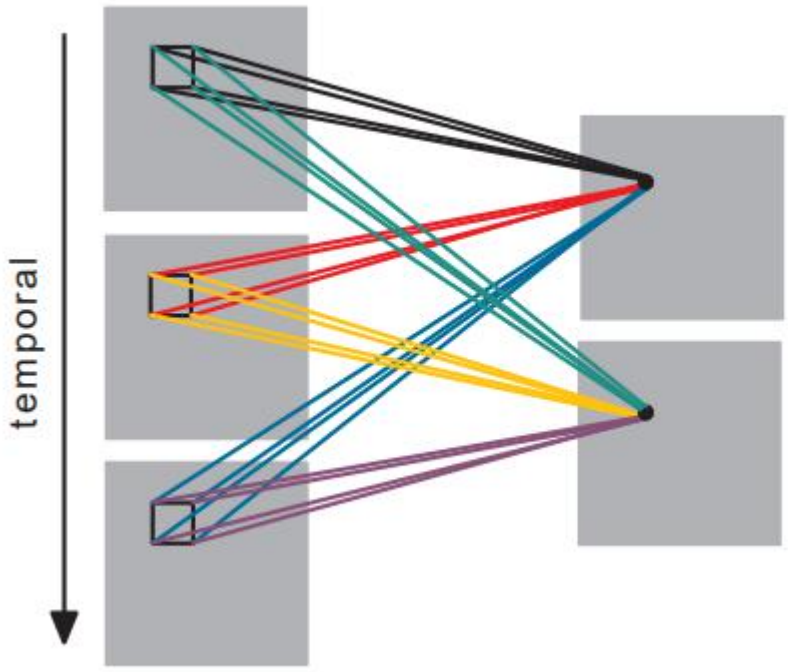

Figure 2.7.2: *Extraction of multiple features from contiguous frames. Multiple 3D convolutions can be applied to contiguous frames to extract multiple features. As in Figure 1, the sets of connections are color-coded so that the shared weights are in the same color. Note that all the 6 sets of connections do not share weights, resulting in two different feature maps on the right.*

Based on the three-dimensional convolution operation described above, different CNN structures can be designed. In the following, the author describes a 3dcnn architecture to deal with human motion recognition on TRECVID data set. In the architecture shown in Fig 2.7.3, a 7-frame image with a size of 60x40 centered on the current frame is used as the input of 3dcnn. Through the hardwired core, the information of multiple channels is generated from the input frame, and then 33 characteristic images are obtained in the second layer. There are 5 different channels, including gray level, abscissa gradient x, ordinate gradient y, optical flow X and optical flow y, among which the gray level channel contains the gray level values of 7 input frames. The gradient X

and Y channels are obtained by calculating 7 frames along the horizontal and vertical directions respectively. The optical \low X and Y channels are calculated from adjacent input frames along the horizontal and vertical directions, respectively. Hardwired layer is used to encode the content of known features, which can bring better performance than random initialization

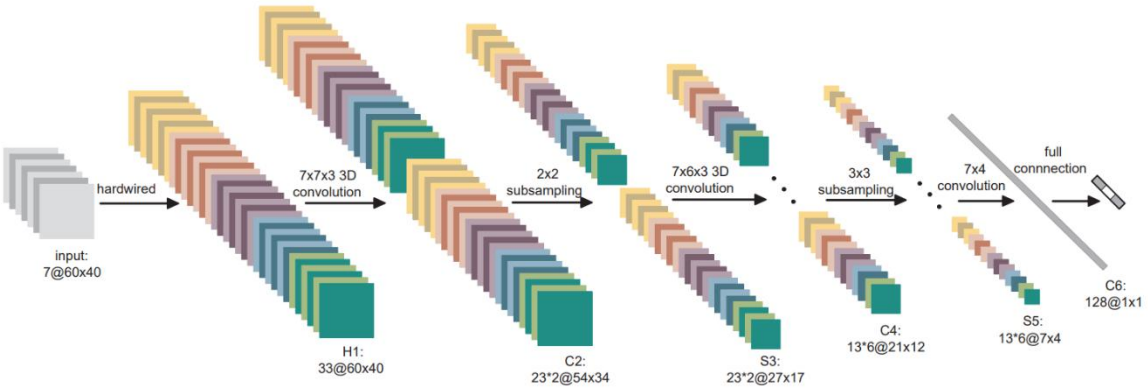

Figure 2.7.3: *A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer. Detailed descriptions are given in the text.*

Then, the author used 3D convolution operations on five channels. The convolution kernel size is 7x7x3, 7x7 is the spatial dimension, and 3 is the time dimension. To increase the number of feature maps, two different convolution sets are used at each location. This results in 2 feature maps on the C2 layer, each producing 23 feature maps. This layer contains 1480 training parameters. In the final downsampling layer S3, the author uses a 2X2 sliding window to apply to each feature map of the C2 layer. This results in the same number of feature maps, but reduces spatial resolution. There are 92 training parameters on this layer. The next convolution layer is C4. The size of the convolution kernel is 7x7x6, and the 3D convolution operation is performed on the five channels of the two feature maps. In order to increase the number of feature maps, the author uses 3 different convolution kernel operations at each position, which results in 6 different feature map sets on the C4 layer, each of which contains 13 feature maps. This layer contains 3810 training parameters. The next layer is S5, on each feature map of the C4 layer. Downsampling with a 3x3 window will result in the same number of feature maps, but its resolution will decrease. The training parameters for this layer are 156. At this stage, the size of the time dimension is relatively small (3 gray levels, gradient x, gradient y, 2 optical flow x, optical flow y), so at this layer, the author only applies convolution to Spatial dimension. The size of the convolution kernel is 7x4, so the size of the output feature map is reduced to 1x1, each of which is connected to all 78 feature maps of the S5 layer, which will have 289536 training parameters.

Through multi-layer convolution and downsampling operations, the 7-frame input

image is converted into a 128D feature vector that captures motion information in the input frame. The output layer contains the same number of units as the number of actions, and each unit is connected to 128 neurons in the C6 layer. In this design, the author essentially performs linear classification on 128D feature vectors for motion recognition. For a three-category action recognition problem, there are 384 training parameters in the output layer. There are 295,458 training parameters on this 3DCNN model, all of which are randomly initialized and trained using the online error back-propagation algorithm. The author designed and verified other 3DCNN architectures, and combined multi-channel information at different stages. The authors' results show that this architecture gives the best performance.

Experimental results: The RECVID 2008 development dataset contains 49 hours of video collected from London Gatwick Airport, from 5 different cameras, with a specification of 720 × 57, 25 fps. The video taken by camera 4 is excluded because almost no events occur in this scene. In this experiment, the author focuses on three action categories (CellToEar, ObjectPut, and Pointing). Each action is classified in a one-to-many manner. A large number of negative samples generated in the action do not belong to these three categories. This data set collected data for five days (20071101, 20071106, 20071107, 20071108, and 20071112). The experimental data is summarized in Table 2.7.1. The 3DCNN model used in this experiment uses the network introduced earlier and is also described in Fig 2.7.3.

Date\Class	CellToEar	ObjectPut	Pointing	Negative	Total
20071101	2692	1349	7845	20056	31942
20071106	1820	3075	8533	22095	35523
20071107	465	3621	8708	19604	32398
20071108	4162	3582	11561	35898	55203
20071112	4859	5728	18480	51428	80495
Total	13998	17355	55127	149081	235561

Table 2.7.1: *The number of samples per category drawn from the TRECVID 2008 development dataset. Shows the total number of samples for each category in the sample for each date*

Because it is a video recorded in a real-world environment, and each frame contains multiple characters, the author uses a human detector and tracker to locate the person's head. Some human detection and tracking samples are shown in Fig 2.7.4. Based on the results of detection and tracking, each person has a bounding box to perform the calculation of the action. The acquisition of the multi-frame image required by the 3DCNN model is obtained from the bounding box at the same position in the current frame and adjacent frames, which will result in a cube containing actions. In the author's experiments, the time dimension of this cube is set to 7, and it has been confirmed [64]

that 5-7 frames can achieve one, similar to the performance obtained from the entire video sequence. The frame extraction step size is 2 steps. Assuming the current frame is the number 0, author extract the bounding box from the same position in the frame with the numbers -6, -4, -2,0,2,4,6. The bounding box of each frame is scaled to 60x40 pixels

Figure 2.7.4: *Sample human detection and tracking results from camera numbers 1, 2, 3, and 5, respectively from left to right.*

Performance of the four methods under multiple false positive rates (FPR). The performance is measured in terms of precision, recall, and AUC. The AUC scores are multiplied by 103 for the ease of presentation. The highest value in each case is highlighted. Result show in table 2.7.2

Method	FPR	Measure	CellToEar	ObjectPut	Pointing	Average
3D CNN	0.1%	Precision	**0.6433**	**0.6748**	0.8230	**0.7137**
		Recall	**0.0282**	**0.0256**	0.0152	**0.0230**
		AUC($\times 10^3$)	**0.0173**	**0.0139**	0.0075	**0.0129**
	1%	Precision	**0.4091**	**0.5154**	0.7470	**0.5572**
		Recall	**0.1109**	**0.1356**	0.0931	**0.1132**
		AUC($\times 10^3$)	**0.6759**	**0.7916**	0.5581	**0.6752**
2D CNN	0.1%	Precision	0.3842	0.5865	**0.8547**	0.6085
		Recall	0.0097	0.0176	**0.0192**	0.0155
		AUC($\times 10^3$)	0.0057	0.0109	**0.0110**	0.0092
	1%	Precision	0.3032	0.3937	0.7446	0.4805
		Recall	0.0505	0.0974	**0.1020**	0.0833
		AUC($\times 10^3$)	0.2725	0.5589	**0.6218**	0.4844
SPM^{CUBE}_{GRAY}	0.1%	Precision	0.3576	0.6051	0.8541	0.6056
		Recall	0.0088	0.0192	0.0191	0.0157
		AUC($\times 10^3$)	0.0044	0.0108	**0.0110**	0.0087
	1%	Precision	0.2607	0.4332	0.7511	0.4817
		Recall	0.0558	0.0961	0.0988	0.0836
		AUC($\times 10^3$)	0.3127	0.5523	0.5915	0.4855
SPM^{CUBE}_{MEHI}	0.1%	Precision	0.4848	0.5692	0.8268	0.6269
		Recall	0.0149	0.0166	0.0156	0.0157
		AUC($\times 10^3$)	0.0071	0.0087	0.0084	0.0081
	1%	Precision	0.3552	0.3961	**0.7546**	0.5020
		Recall	0.0872	0.0825	0.1006	0.0901
		AUC($\times 10^3$)	0.4955	0.4629	0.5712	0.5099

Table 2.7.2: *Performance of the four methods under multiple false positive rates (FPR)*

Conclusion: the author proposes a 3DCNN model to handle motion recognition. This model constructs features from the spatial and temporal dimensions through 3D convolution operations. This deep architecture generates multi-channel information from adjacent input frames, and performs convolution and downsampling operations on each channel separately. The final feature representation is calculated by combining the information on all channels.

2.6.2. ConvNet+LSTM

The video analysis method based on 3dcnn puts forward a neural network video analysis algorithm for the first time, which considers the space-time structure. However, 3dcnn has a huge drawback, that is, it has a huge amount of calculation, and in exchange for a limited accuracy improvement, so 3dcnn still has a lot to improve. The biggest reason for this huge amount of calculation is the repeated calculation of the same content of the repeated frames, so convolution network based on LSTM is born

There are many methods for video analysis based on LSTM, but the main solution is to extract the information of the video frame through CNN, and then learn the time domain connection through LSTM. LRCNs[65] is a very successful method. A video model can handle variable-length input sequences and also support variable-length output, including full-length sentence description generation. Long-term Recurrent Convolutional Networks (LRCNs), is a structure for image recognition and description , Which combines convolutional layers and long-term recursion, and is end-to-end trainable. The following will introduce this structure through specific video behavior recognition, picture description generation, and video description tasks. The structure of the network is similar to Fig 2.8.1, Convolutional neural network to extract features from different frames, and send the extracted high-dimensional image features to LSTM to predict the sentence.

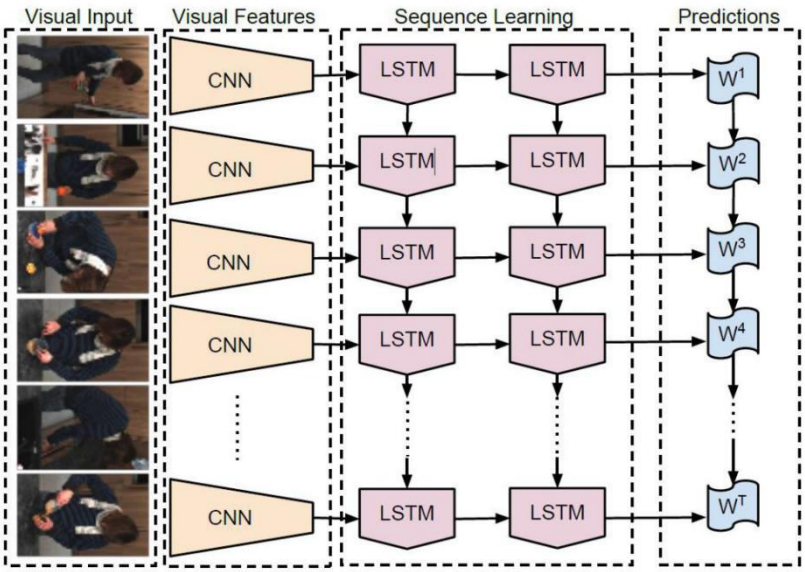

Figure 2.8.1: *We propose Long-term Recurrent Convolutional Networks (LRCNs), a class of architectures leveraging the strengths of rapid progress in CNNs for visual recognition problems, and the growing desire to apply such models to time-varying inputs and outputs. LRCN processes the (possibly) variable-length visual input (left) with a CNN (middleleft), whose outputs are fed into a stack of recurrent sequence models (LSTMs, middle-right), which finally produce a variable-length prediction (right). Both the CNN and LSTM weights are shared across time, resulting in a representation that scales to arbitrarily long*

LRCNs got this model through three experiments

1. The traditional convolution model is directly connected to the deep LSTM network, which can capture the time series dependencies in the video recognition model. As shown in Figure 2, each video stream passes the shared CNN network, and the convolutional features obtained are input into the LSTM. The output units of each LSTM are combined to take the average value. Finally, the final classification result is obtained through a full connection

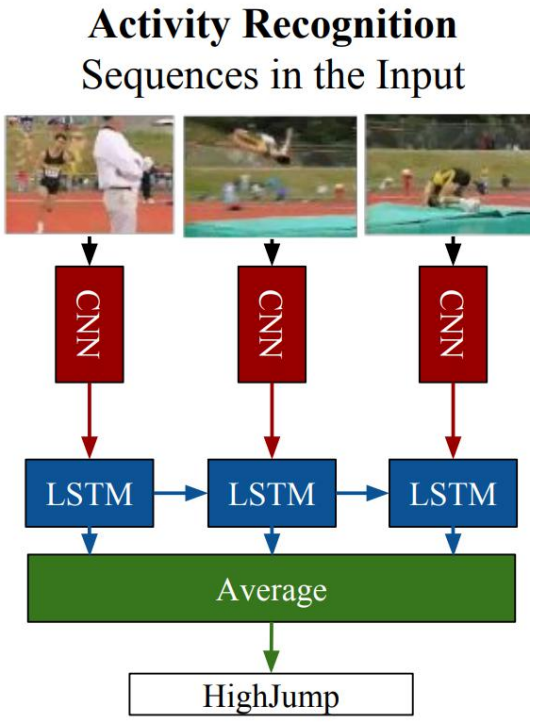

Figure 2.8.2： *LRCN model for activity recognition*

2. Authors researched an end-to-end trainable mapping from images to semantics. Machine translation has recently achieved a lot of results, and such models are based on LSTM encoding-decoding pairs. The author proposed a similar method, which uses a picture's convolutional neural network to encode a deep state vector, and then decodes the vector into a natural language string through an LSTM decoding. The final model can be used for end-to-end training of large-scale picture and text datasets. Even incomplete training, compared with the existing methods, it can also obtain a better generated result. As shown in Figure 2.8.3, a sequence of single-frame picture convolution is sent to the LSTM and the previously predicted LSTM result is also used as the input of the next unit

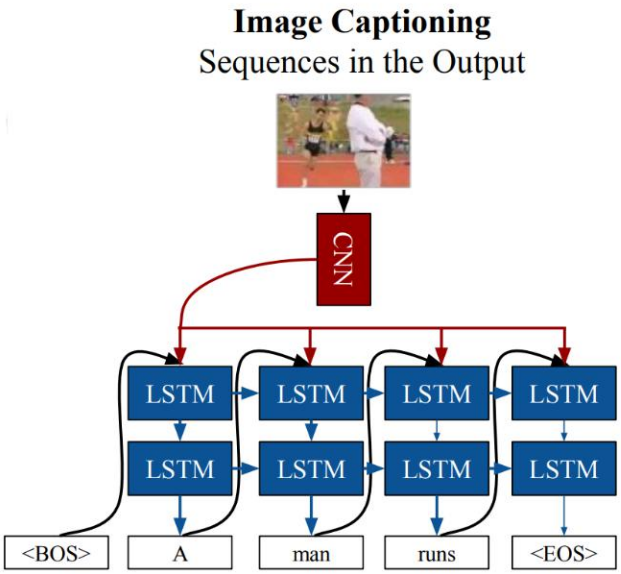

Figure 2.8.3： *LRCN model for image description*

3. LSTM decoder can directly convert high-level features into corresponding high-level language tags, such as semantic video role array prediction. This type of model is superior in structure and performance to the original statistical machine translation-based methods. The biggest difference between this experiment and experiment 2 in Fig 2.5.9 is that from a single frame image to a sequence image, using CRF to reasonably select the input image name and give semantic understanding through LSTM to generate subtitle, show in Fig 2.8.4

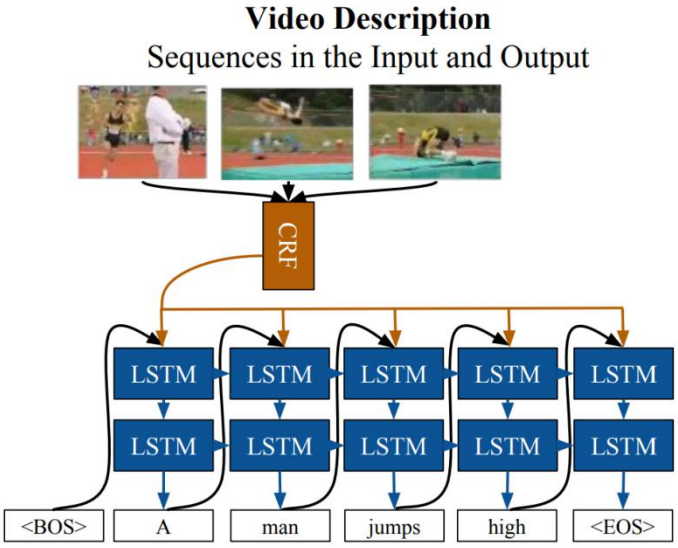

Fig 2.8.4： *LRCN model for video description*

Chapter 3

Dense Convolutional Networks for Efficient Video Analysis

3.1. Introduction

Over the past few years various Convolutional Neural Networks (CNNs) based models exhibited certain human-like performance in a range of image processing problems. Video understanding, action classification, gesture recognition has become a new stage for CNNs. The typical approach for video analyse is based on 2DCNN to extract feature map from a single frame and through 3DCNN or LSTM to merging spatiotemporal information, some approaches will add optical flow on the other branch and then post-hoc fusion. Normally the performance is proportional to the model complexity, as the accuracy keeps improving, the problem is also evolved from accuracy to model size, computing speed, model availability. In this paper, we present a lightweight network architecture framework to learn spatiotemporal feature from video. Our architecture tries to merge long-term content in any network feature map. Keeping the model as small and as fast as possible while maintaining accuracy.

Thanks to larger datasets and multiple deep learning algorithms, video understanding and, specifically, action classification which have reported outstanding performance in recent years. Meanwhile, efficient and low cost video analysis algorithm has become a meaningful subject, especially for real time human-robot computer interaction. Short time action recognition is depending on the present action within a short time window, however video understanding is concerned with longer-term frames information. Several architectures have been proposed to capture spatiotemporal information between frames, However the speed and efficiency always remained at the bottleneck. Such as 3DCNN [61,66], theoretically time window should cover whole video, due to the cost of calculation people have to use a small window, each convolution computation contains several frames, the information is merged by convolution and pooling computation layer by layer. Obviously, this algorithm is suboptimal for exploiting relationship between frames. In fact, ordinary LSTM [67,68] structure still cannot get satisfactory results, even if it has outstanding performance in several sequence problems, some algorithms choose to sacrifice speed to improve accuracy, these ideas are through add an optical flow branch to achieve ‘Two-Stream’ structure [70,71] and the get excellent results, however the speed of ‘Two-Stream’ is too slow that these algorithms lack of practicality.

Our approach is inspired by DenseNet [38] and ECO [69], we proposed a simple end-to-end trainable architecture to address previously mentioned dilemma, DenseNet provides a good idea to reduce calculation, normally feature map will increase as the number of layers increase, but in this way only a few number of feature maps are calculated per layer and through concat operation to merge them, on the other hand unlike other combine operations, concat can pass the results of each computation to the output without change. Effectively processing the information in each

frame is the contribution of ECO. Temporal neighbourhood of a frame almost redundant information. So normal 3D CNN method is very inefficient to slide from the first frame to the last. Most calculation will produce similar results and don't carry any useful information. It will waste a lot of computing resources. Select a single frame of a temporal neighbourhood and use bidirectional ConvLSTM to yield a representation for each sampled frame, each representation will have a strong ability to extract spatiotemporal information. Not like ECO, ECO yield the feature map with a 2D convolutional architecture and then to capture the contextual relationships between distant frames, re-stacked feature maps and feed into 3D CNN. The overview of our approach architecture is illustrated in Fig 3.1.1. Consequent network will become faster and still have satisfactory accuracy. This makes more sense for real time video retrieval.

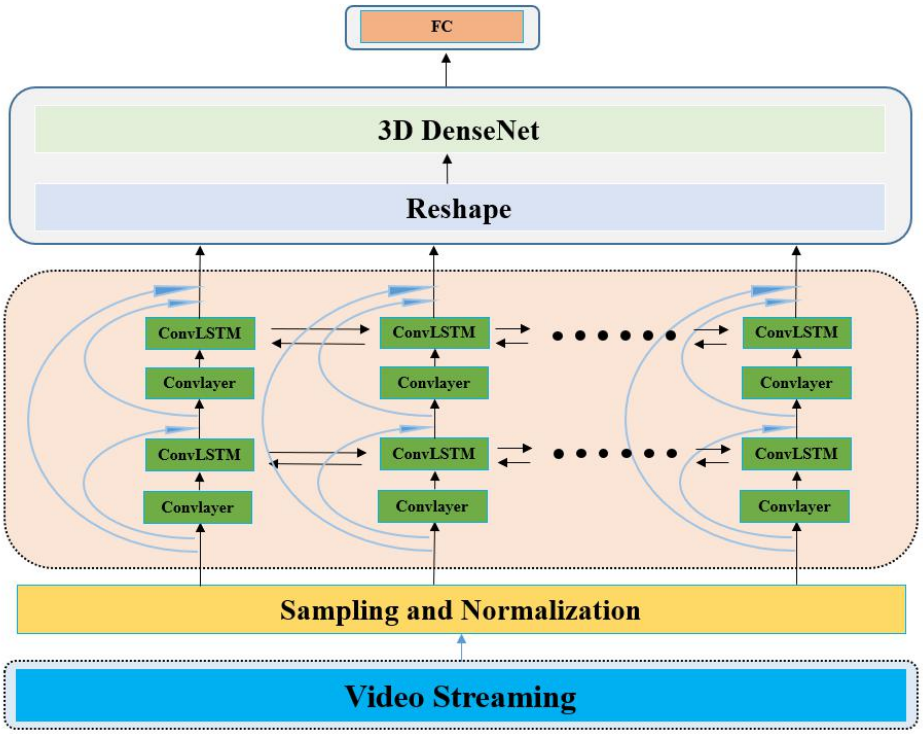

Fig 3.1.1： Overview of Dense Convolutional Networks, Dense bidirectional ConvLSTM are utilized to learn long-term spatiotemporal features and sent original information to the high-level structure, all feature will be re-stacked and fed to 3D DenseNet which classifies the action.

3.2. Related Work

Although the development of deep learning is very rapid, but there is still not a unified video analysis framework, the current architecture is mainly based on convolutional layers and layers operators with 2D or 3D convolution kernel. Also, either the network is only for analysing RGB videos or includes pre-computed light flow. 2D convolution merges the space and time information which is evident from LSTM network structure in 2D convolution, spatiotemporal information (can be learned by LSTMs), or aggregation over time.

For video classification problems, there are some generally accepted algorithm based on deep learning method [71,65,72,73,74]. In order to extract more spatiotemporal features from a sequence of frames 3D convolution network is introduced. 3D kernel can learn the spatiotemporal feature. Tran et al. [67] proposed a 3D architecture using 3D convolution kernels which slide whole video. They studied the 3D structure of Resnet system and improved the structure of C3D [74]. In addition, recurrent networks are also a good approach to extract temporal relation between frames [75,76,65]. Donahue et al. [65] proposed an approach using LSTM to integrate features from CNN. In fact, LSTM-based network to extract the features of spatiotemporal is inefficient and unsatisfactory. In action recognition field, the performance of LSTM-based method is always behind the CNN-based methods. Recently introduced 3D CNN has some new approach and new methods emerge one after another [77,73,78]. The characteristic of these methods is the use of sliding window to obtain short term temporal context. However, such methods consume all computing resources because the average score of these Windows needs to be calculated before final fusion.

In fact, these methods do not utilize video's spatiotemporal information very efficiently and require very large computing resources. Especially through the post-hoc fusion, not only inefficient, but also reduces the operation speed which cannot be used in the scene of fact work.

3.3. Algorithm architecture

The network architecture is shown in Fig 3.3,1. a whole video will be split into N subsections N_n n = 1, 2,……, N each subsection containing the same number of video frames and then randomly samples a frame from each subset to represent the entire subset. Since there are a lot of duplicate frames in whole video, it is very wasteful to send the whole video into the network for calculation, and it will get a lot of repeated useless information. By this method, redundant information can be avoided and useful information can be extracted, and spatiotemporal information of video can be extracted effectively through repeated random sampling iteration in training.

After sampling, N frames will be obtained and will be sent to network. The network has two main components, the first is the DenseConvLSTM, this part is used for learning simple spatiotemporal information. And second is the 3D DenseNet, this part is for high dimensional spatiotemporal information fusion. Concat is also used in the DenseConvLSTM block, which means that the original information can be passed on to the next structure irrespective of whether the results of the feature are good or not.

Since convolution layer is very useful in learning the features of image, it is necessary to extract the features of video using the convolution block before bi-direction ConvLSTM. Each convolution block has three different convolution layers, and connect to ConvLSTM block to learn spatiotemporal features. ConvLSTM consists of two bi-direction LSTM. All features will be re-stacked using DenseNet architecture with 3D convolutions. The DenseNet contains 4 dense blocks with growth rate of 32 and 3D DenseNet network yields the final action class label. It is a straightforward architecture and it can be trained end-to-end on action classification and large data base.

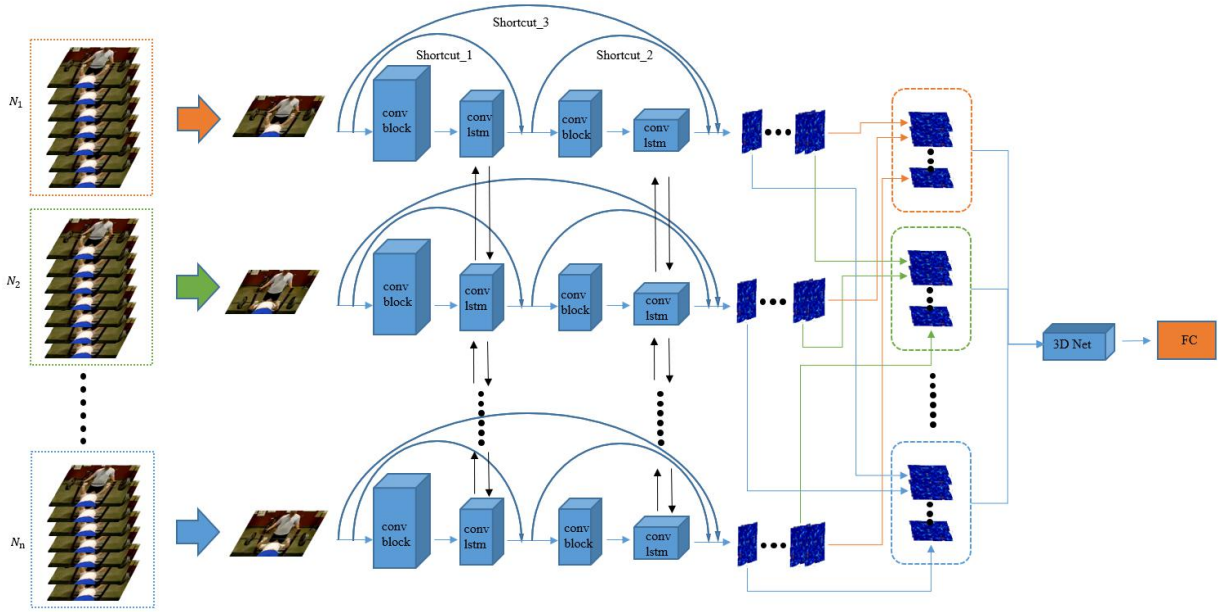

Fig 3.3.1： *The network architecture details*

3.3.1. 2D Layer

Each Conv block contains three convolution layers and two bi-direction LSTM layers. The parameters of convolution layers and shortcuts are shown in Table 3.3.1. the input image will be sized into 224*224, though six different convolution layer, the out put size of 2D layer will be 26*26. The kernel size of conv1_1 is 7, the step size is 2, the output channel is 16, padding is 0, the kernel size of the first conv1_2 is 3, step size is 2, output channel is 16, padding is 0, the difference between the second conv1_2 is that the step size is 1. The kernel size of conv2_1 is 3, step size is 1, output channel is 16, padding is 1, and the kernel size of the first conv2_2 is 3, Step size is 1, output channel is 16, padding is 1, the parameters of second conv2_2 is the same as first conv2_2. The kernel size of shortcuts_1 is 7 , step size is 2, output channel is 32, padding is 0, The kernel size of shortcuts-2 is 3, step size is 2, output channel is 32, padding is 0， shortcut-3 has two different kernel， The first kernel size of shortcuts_3 is 11, step size is 2, output channel is 32, padding is 0, The second kernel of shortcuts_3 is 3, step size is 2, output channel is 32, padding is 0.

Layer Name	Output size	Parameter
Conv1_1	109 × 109	7 × 7 conv, stride 2, padding 0, 16
Conv1_2	54 × 54	3 × 3 conv, stride 2, padding 0, 16
Conv1_2	54 × 54	3 × 3 conv, stride 1, padding 1, 16
Conv2_1	26 × 26	3 × 3 conv, stride 1, padding 1, 16
Conv2_2	26 × 26	3 × 3 conv, stride 1, padding 1, 16
Conv2_2	26 × 26	3 × 3 conv, stride 1, padding 1, 16
Shortcut_1	54 × 54	11 × 11 conv, stride 4, padding 0, 32
Shortcut_2	26 × 26	3 × 3 conv, stride 2, padding 0, 32
Shortcut_3	26 × 26	11 × 11 conv, stride 4, padding 0, 32 3 × 3 conv, stride 2, padding 0, 32

Table 3.3.1： *The proposed Architecture*

3.3.2. ConvLSTM Layers

In the traditional fully connected LSTM, input features will be vectorized, which is very unfavorable for learning space-time features. In the transmission process of each layer, vectorized features will lose spatial relevance, however, object posture and position is significance for recognition. So ConvLSTM [19] is used to learn long-term temporal and spatial properties, and convolution and recursion can take full advantage of spatiotemporal information. The formula is between 3.3.2.1 to 3.3.2.5, The feature map of hidden layer we adopt 32 as each signal LSTM output

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ct} \circ C_{t-1} + b_i) \quad 3.3.2.1$$

$$f_t = \sigma\left(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \quad 3.3.2.2$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c\right) \quad 3.3.2.3$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \quad 3.3.2.4$$

$$H_t = o_t \circ \tanh\left(C_t\right) \quad 3.3.2.5$$

3.3.3. 3D Layer

For the 3D network, we adopted the 3D structure based on DenseNet. DenseNet itself has many advantages, reducing model size and increasing classification effect. This is a very effective structure and the network parameters are shown in Table 3.3.2. 3D Layer has 4 3D-Dense Block, each has the same kernel size, each block has two different kernels, the first one is 1*1*1 convolution layer, the second one is 3*3*3, the stride of them is both 1 and padding is 1 as well. However, the number of kernels of each block is different, the first one is 4, the second one is 8, the third one is 12 and the last one is 8. Transition Layer is used to reduce the competition, with kernel size is 1*1*1, and average pooling, final use global average pooling to represent the global feature.

Layer Name	3d-DenseNet (growth rate = 32)
3D-Dense Block (1)	$\begin{bmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv} \end{bmatrix} \times 4$, stride 1, padding 1
Transition Layer (1)	$1 \times 1 \times 1$ stride 1, padding 0
	$3 \times 3 \times 2$ average pool, stride $2 \times 2 \times 2$, padding 1
3D-Dense Block (2)	$\begin{bmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv} \end{bmatrix} \times 8$, stride 1, padding 1
Transition Layer (2)	$1 \times 1 \times 1$ stride 1, padding 0
	$3 \times 3 \times 2$ average pool, stride $2 \times 2 \times 2$, padding 1
3D-Dense Block (3)	$\begin{bmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv} \end{bmatrix} \times 12$, stride 1, padding 1
Transition Layer (3)	$1 \times 1 \times 1$ stride 1, padding 0
	$3 \times 3 \times 2$ average pool, stride $1 \times 1 \times 2$, padding 1
3D-Dense Block (4)	$\begin{bmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv} \end{bmatrix} \times 8$, stride 1, padding 1
Classification Layer	7×7 global average pool
	FC, softmax

Table 3.3.2： *Network parameters for 3D-Net*

3.4. Experiment

3.4.1. Training Detail

We used mini-batch of SGD to train our network and used dropout in each fully connected layer. Each video is divided into 10 segments and randomly selected one frame from each segment. The purpose of this sampling is to avoid redundant information and improve network robustness. Furthermore, we applied the data augmentation technique introduced in resizing the frame of dataset to 240 × 320 and it adopts the scale jittering with horizontal flipping. Then we use per-pixel mean subtraction and randomly crop a 224×224 figure from frames. The initial learning rate is 0.001. Appearance-and-relation networks for video the weight attenuation is 0.0005, and the minus-sized is 32. All the training was done on a tesla P100 GPU.

Test time inference: In order to improve the accuracy, many start-of-the-art algorithms prefer to run some post-processing. For instance, TSN [79] and ARTNet [77], each segment video collects 25 individual frames. Each frame/volume sample crops 10 areas of corner and center and their horizontal flipping. Then based on the average scores of the 250 samples results are obtained. It is clear that methods for calculation are very expensive and that is not the best way for fast video analysis algorithm. However, our algorithm without too much additional operations, is based on many widely recognized ideas to solve the problem of speed. During the test period, our algorithm in video is divided into N subsection and randomly extracts a frame from each section, these frames only perform center cropping and send the cropped part to the network. This is an end-to-end prediction.

3.4.2. Result

In order to test the generalization ability of our method, we evaluated it on different data sets to make it more convenient and intuitive to compare with other excellent methods. We tested UCF101[11] on the original RGB video as input.

We also tested speed of our approach at the same time. A lot of methods always select fps (frames per second) as a standard comparison for speed. Actually such comparisons have problems because of the different algorithms applied on same video to calculate the number of frames which is always different. So here we are more concerned with the processing speed of each video. Our approach can reach 790 FPS on a Tesla P100 GPU. All the results are show in Table 3.3.3.

Method	**Pre-train**	**Accuracy(%)**	**Speed(VPS)**
Our Approach	Kinetics	**91.4**	**69.3**
HOG[20]	-	72.4	-
ConvNet+LSTM (AlexNet)[21]	ImageNet	68.2	-
C3D (VGGNet-11)[3]	Sports-1M	82.3	-
Res3D[15]	Sports-1M	85.8	<2
TSN Spatial Network[9]	ImageNet	86.4	21
TSN Spatial Network[9]	ImageNet + Kinetics	91.1	21
RGB-I3D[22]	ImageNet + Kinetics	**95.6**	0.9
$\boldsymbol{ECO_{12F}}$**[2]**	Kinetics	92.4	**52.6**
ARTNet w/o TSN[10]	Kinetics	93.5	2.9

Table 3.3.3: *The simulation Results*

3.5. Summary

We have put forward a structure which is highly efficient for video analysis. And that is evident form the result obtained. The proposed structure can be used not only in video classification but also can be used in various video understanding tasks. The advantage is that network only needs an RGB image and no other pre-treatment. ConvLSTM replaces the traditional LSTM which is good for extracting spatiotemporal information. High density of residual can make the network robust as well.

Chapter 4

Conclusion and Future work

4.1 Conclusion

In this paper, a new video preprocessing method is proposed, which is applied to construct a new fast video understanding algorithm and achieves satisfactory results in a variety of data sets.

Many video processing methods already exist, and the relationship between frames is considered to be the key of video analysis technology. However, the connection between video frames has never been used effectively in deep learning. Using a kind of dense connect, this paper thinks that when people produce facial expression, the facial movement pattern can be used for the feature construction of human expression recognition.

In the third chapter, the video preprocessing method is to use the dense connect method to effectively propagate the features of each layer of the video frame backward. In order to avoid the repeated calculation of the video frame with the same content, the video stream of any length is divided into multiple equal parts. When it is used for neural network training, a frame of video in each cell is randomly selected, so that It can avoid too many invalid calculations. The experimental results based on Python show that this method is very fast and has high accuracy. However, the network still has some improvements

1. The video understanding based on this architecture is to send the original frame directly to the network for training. However, the information we really need to consider only accounts for a small part of the video picture, which is equivalent to too many negative samples sent to the network for learning, which is not conducive to video understanding. Referring to the method of human face recognition, we can first use the method of key point detection to key the foreground of each frame of video Point out, in the network learning, avoid too many negative samples

2. The video quality needs to be high-quality， a low-quality video is not conducive to feature extraction. The super-resolution reconstruction algorithm based on neural network can be used to process the video stream first, so as to improve the resolution.

3. Using the attention structure to optimize the channel of the convolution layer.

In short, this article introduces some basic knowledge of image processing, video processing, neural networks, model training. A new video processing scheme is proposed, and the scheme is implemented and verified to be used in video analysis problems. Finally published the paper “ ***Dense Convolutional Networks for Efficient Video Analysis***” in conference ICAAR2019

4.2 Future work

Although this paper puts forward a new algorithm in video understanding, there are still many problems to be solved, such as video understanding in low resolution, super long video understanding, such as dozens of minutes or even hours of video understanding, which are very valuable in real generation activities, but this paper does not propose a solution. In this paper, the video stream is divided into several equal parts violently, this method also needs to be optimized, because when the video stream is too long, a quick action cannot be recognized.

Reference

1. *Borowiec, S. (2016). AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol. The Guardian, 15.*

2. *O. Russakovsky et al., "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, 2015.*

3. *R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-Recursive Deep Learning for 3D Object Classification," in NIPS, 2012, vol. 3, no. 7, p. 8.*

4. *Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.*

5. *Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484.*

6. *Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).*

7. *Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.*

8. *Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.*

9. *Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.*

10. *Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.*

11. *Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In international Conference on computer vision & Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE Computer Society.*

12. *Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5), 898-916.*

13. *Johnson, A. E. (1997). Spin-images: a representation for 3-D surface matching.*

14. *Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence, 27(10), 1615-1630.*

15. *Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).*

16. *Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.*

17. *Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.*

18. *Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338.*

19. *Jain, V., & Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings (Vol. 2, No. 4, p. 5). UMass Amherst Technical Report.*

20. *Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research, 32(11), 1231-1237.*

21. *HUBEL D H, WIESEL T N. Receptive fields and functional architecture of monkey striate cortex [J]. The Journal of physiology, 1968, 195(1): 215-43.*

22. *FUKUSHIMA K, MIYAKE S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition [M]. Competition and cooperation in neural nets. Springer. 1982: 267-85.*

23. *LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.*

24. *STEINKRAUS D, BUCK I, SIMARD P. Using GPUs for machine learning algorithms; proceedings of the Document Analysis and Recognition, 2005 Proceedings Eighth International Conference on, F, 2005 [C]. IEEE.*

25. *SZEGEDY C, LIU W, JIA Y Q, et al. Going Deeper with Convolutions; proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, F Jun 07-12, 2015 [C]. Ieee: NEW YORK, 2015.*

26. *IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift; proceedings of the 32nd International Conference on Machine Learning (ICML), Lile, France, F July 6, 2015 - July 11, 2015, 2015 [C]. International Machine Learning Society (IMLS).*

27. *Ioffe, S. (2017). Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In Advances in neural information processing systems (pp. 1945-1953).*

28. *HE K M, ZHANG X Y, REN S Q, et al. Deep Residual Learning for Image Recognition; proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, F Jun 26-Jul 01, 2016 [C]. Ieee: NEW YORK, 2016.*

29. *VEIT A, WILBER M, BELONGIE S. Residual networks behave like ensembles of relatively shallow networks; proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, F December 5, 2016 - December 10, 2016, 2016 [C]. Neural information processing systems foundation.*

30. *Huang, G., Liu, S., Van der Maaten, L., & Weinberger, K. Q. (2018). Condensenet: An efficient densenet using learned group convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2752-2761).*

31. *Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).*

32. *Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). springer, Cham*

33. *V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010.*

34. *Simonyan, K. , & Zisserman, A. . (2014). Very deep convolutional networks for large-scale image recognition. Computer Science.*

35. *Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis & Machine Intelligence, 39(6), 1137-1149*

36. *Joseph Redmon, Santosh Divvala, Ross Girshick, & Ali Farhadi. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*

37. *Liu, W. , Anguelov, D. , Erhan, D. , Szegedy, C. , Reed, S. , & Fu, C. Y. , et al. (2015). Ssd: single shot multibox detector.*

38. *Huang, Gao, Liu, Zhuang, van der Maaten, Laurens, & Weinberger, Kilian Q. . Densely connected convolutional networks*

39. *Hubel David H., & Wiesel Torsten N. . Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. Journal of Neurophysiology, 28(2), 229-289.*

40. *M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In*

ECCV, 2014.

41. *R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.*

42. *K.He, X.Zhang, S.Ren, and J.Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.*

43. *R. Girshick. Fast R-CNN. In ICCV, 2015.*

44. *J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.*

45. *S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, TU Munich, 1991.*

46. *Y.Bengio,P.Simard,andP.Frasconi.Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.*

47. *X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.*

48. *Y.LeCun,L.Bottou,G.B.Orr,and K.-R.Muller. Efficient back prop. In Neural Networks: Tricks of the Trade, pages 9–50. Springer, 1998.*

49. *A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv:1312.6120, 2013.*

50. *K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV, 2015.*

51. *K.Heand J.Sun. Convolutional neural networks at constrained time cost. In CVPR, 2015.*

52. *R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. arXiv:1505.00387, 2015.*

53. *G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.*

54. *J. R. Uijlings, K. E. van de Sande, T. Gevers, and A.W. Smeulders. Selective search for object recognition. IJCV, 2013.*

55. *Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard,W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989.*

56. *M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.*

57. *C. L. Zitnick and P. Doll´ar. Edge boxes: Locating object proposals from edges. In ECCV,*

2014.

58. *Redmon, J. , & Farhadi, A. . (2018). YOLOv3: an incremental improvement.*

59. *P. Felzenszwalb, D. McAllester, D. Ramaman, A Discriminatively Trained, Multiscale, Deformable Part Model IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008*

60. *R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014*

61. *Ji, S. , Xu, W. , Yang, M. , & Yu, K. . (2013). 3d convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1), 221-231.*

62. *TRECVID(2008,2010), Guidelines*

63. *KTH Dataset(2004)*

64. *Ivan Laptev and Tony Lindeberg; ECCV Workshop "Spatial Coherence for Visual Motion Analysis"*

65. *Donahue, Jeff, Hendricks, Lisa Anne, Rohrbach, Marcus, Venugopalan, Subhashini, Guadarrama, Sergio, & Saenko, Kate 等. (2014). Long-term recurrent convolutional networks for visual recognition and description. IEEE Transactions on Pattern Analysis & Machine Intelligence, 39(4), 677-691.*

66. *Taylor, Graham W., et al. "Convolutional learning of spatio-temporal features." Proceedings of the 11th European conference on Computer vision: Part VI. Springer-Verlag, 2010.*

67. *Tran, Du, et al. "Learning Spatiotemporal Features with 3D Convolutional Networks." Computer Vision (ICCV), 2015 IEEE International Conference on. IEEE, 2015.*

68. *Ng, Joe Yue-Hei, et al. "Beyond short snippets: Deep networks for video classification." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015.*

69. *Zolfaghari, M., Singh, K., & Brox, T. (2018). ECO: Efficient Convolutional Network for Online Video Understanding. arXiv preprint arXiv:1804.09066.*

70. *Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016*

71. *Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 1. MIT Press, 2014.*

72. *Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Largescale video*

classification with convolutional neural networks. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR'14, Washington, DC, USA, IEEE Computer Society (2014) 1725–1732

73. *Zolfaghari, M., Oliveira, G.L., Sedaghat, N., Brox, T.: Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. In: IEEE International Conference on Computer Vision (ICCV).*

74. *Tran, D., Ray, J., Shou, Z., Chang, S., Paluri, M.: Convnet architecture search for spatiotemporal feature learning. CoRR abs/1708.05038 (2017)*

75. *Lev, G., Sadeh, G., Klein, B.,Wolf, L.: Rnn fisher vectors for action recognition and image annotation. In Leibe, B., Matas, J., Sebe, N., Welling, M., eds.: Computer Vision – ECCV 2016, Cham, Springer International Publishing (2016) 833–850*

76. *Li, Z., Gavrilyuk, K., Gavves, E., Jain, M., Snoek, C.G.: Videolstm convolves, attends and flows for action recognition. Comput. Vis. Image Underst. 166(C) (January 2018) 41–50*

77. *Wang, L., Li, W., Li, W., & Van Gool, L. (2017). Appearance-and-relation networks for video classification. arXiv preprint arXiv:1711.09125.*

78. *Diba, A., Fayyaz, M., Sharma, V., Karami, A.H., Arzani, M.M., Yousefzadeh, R., Gool, L.V.: Temporal 3d convnets: New architecture and transfer learning for video classification. CoRR abs/1711.08200 (2017)*

79. *Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016, October). Temporal segment networks: Towards good practices for deep action recognition. In European Conference on Computer Vision (pp. 20-36). Springer, Cham.*

80. *Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).*

81. *Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).*

82. *Zolfaghari, M., Singh, K., & Brox, T. (2018). Eco: Efficient convolutional network for online video understanding. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 695-712).*

83. *Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).*