Person Re-identification Research Based on Deep Learning

Abstract

Person re-identification (ReID) is used to judge whether the pedestrian image under different cameras is the same person. ReID is one of the core issues in intelligent video surveillance, and it has great application value in public security and other fields. At present, with the improvement of computer performance, deep learning has achieved excellent results in the field of computer vision. Compared with the traditional manual feature extraction method, the deep neural network learns better image features and classifies them through a large number of data, which greatly improves the accuracy of ReID.

In this paper, an end-to-end four-branch neural network based on ResNet-50 is designed. In this model, besides extracting global features, DeeperCut network is used to detect the key features of pedestrian's body. Four key points of forehead, chin, left hip and right hip are detected. Then, pedestrians are divided into three parts by using these four key points. Each part is extracted features, and four branch networks are trained separately. Then, fuse the four features and measure the distance, match pedestrian images. At the same time, this work also compares the performance differences between Softmax loss function and Triplet loss function for this model, and finally uses Triplet loss function. This work's model is trained and tested on two datasets, Market 1501 and CUHK03. The results are compared with the published model performance, which proves that the performance of this model has reached the state-of-art results.

Key Words: Pedestrian recognition; Key feature point detection; Component segmentation

Contents

1 Introduction

1.1 Research Background and Significance

As the development of the society, the video surveillance system is more and more popular. Nowadays, 30 millions surveillance cameras have been deployed in China. We can find them in various public places such as market, station, hospital, school, elevator and so on. When the cases like theft and robbery happen, the police want to find criminal suspect or when someone is missing and the police want to find him, the videos captured by surveillance cameras can provide clues for the police. Because as above-mentioned happens, the target object will appear in many surveillance videos in the process of moving. What the police need to do is to find the target object in a huge number of surveillance videos. The traditional method is to watch the videos, seek target person manually my eyes. However, this method is not only expensive and time-consuming but also has low accuracy. Because the video viewer will have visual fatigue due to watching the video for a long time. For example, to investigate a criminal suspect, the police need retrieve the surveillance videos for all time periods in all possible places. Then they watch the videos frame-by-frame and identify the criminal suspect among people in the videos. The process will consume lots of time and labor. Even if they identify the suspect finally successfully, the suspect probably has already fled to other provinces. Therefore, it is urgent to develop the intelligent monitoring technology for video surveillance system, which is person re-identification.

1.2 The introduction of person re-identification

1.2.1 Concept

Person re-identification (Abbreviation: ReID) is a technology which rises in the field of computer vision recent years. The technology identifies the same person in multiple different non-overlapping surveillance cameras. A complete person re-identification monitoring system can be divided into three modules: pedestrian detection, pedestrian tracking, pedestrian identification and retrieval. Pedestrian detection and pedestrian tracking are two independent computer vision problems. Most pedestrian re-identification research focuses on pedestrian identification and retrieval.

1.2.2 Basic framework [1]

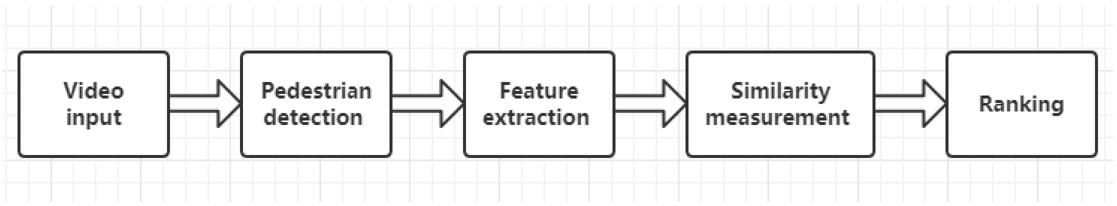


Figure 1-1 ReID Basic framework

As shown in the figure 1-1, ReID is based on the pedestrian detection because it is the first step in a ReID monitoring system. The task of the pedestrian detection is to detect the position of the pedestrian in the surveillance videos generally using a rectangle. For example in Figure 1-2.

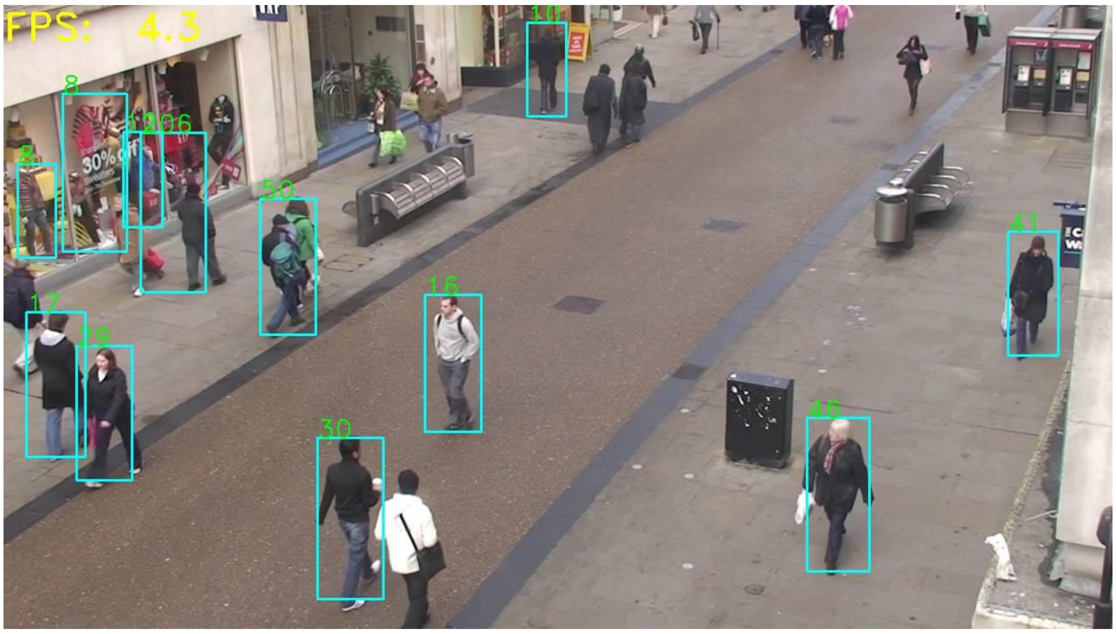


Figure 1-2 Examples of pedestrian detection results

After the pedestrian detection, features in the different images will be extracted. And then measure the similarity of these images features. Finally, given the query set, for each query image, find the image that is most likely to be the same pedestrian from the gallery set. The query set and gallery set are both boundary boxes of single pedestrian images which are segmented by the pedestrian detection algorithm. As shown in the figure 3, the blue boundary box represents the same person and the red boundary box represents different person.

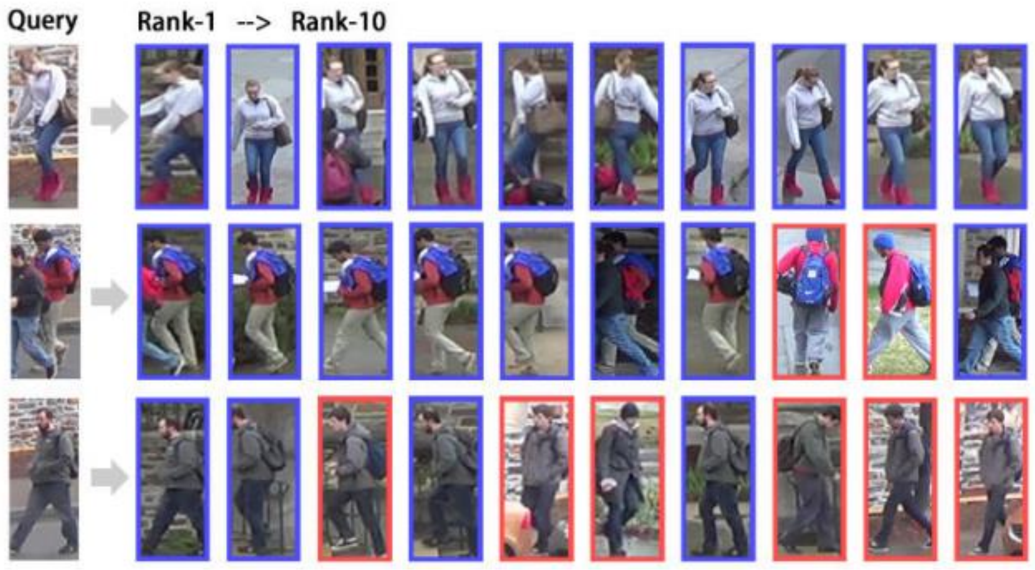


Figure 1-3 Examples of ReID results

1.2.3 Problems and Difficulties

Person re-id problem has three main difficulties.

(1) The first challenge is the need of the demand for large amounts of training data. Compared with other computer vision tasks, the dataset size of the ReID is very small. The large-scale image recognition dataset ImageNet has 1250k images, the pedestrian detection dataset Caltech has 350k pedestrian boundary boxes with labels. However, the datasets for the ReID now only have about 30k images. In that it is difficult to collect the pedestrian data from multiple different non-overlapping surveillance cameras because of the privacy issues. Besides, pedestrian data labeling is hard.

(2) The second challenge is the huge variations of the pedestrian's appearance. Pedestrians may have different poses, background, lighting and resolution. Achieving person re-id should get over the inter-class diversity(Different pedestrians look like similar) and intra-class diversity(The same pedestrian looks like different.), as shown in the figure

1-4.

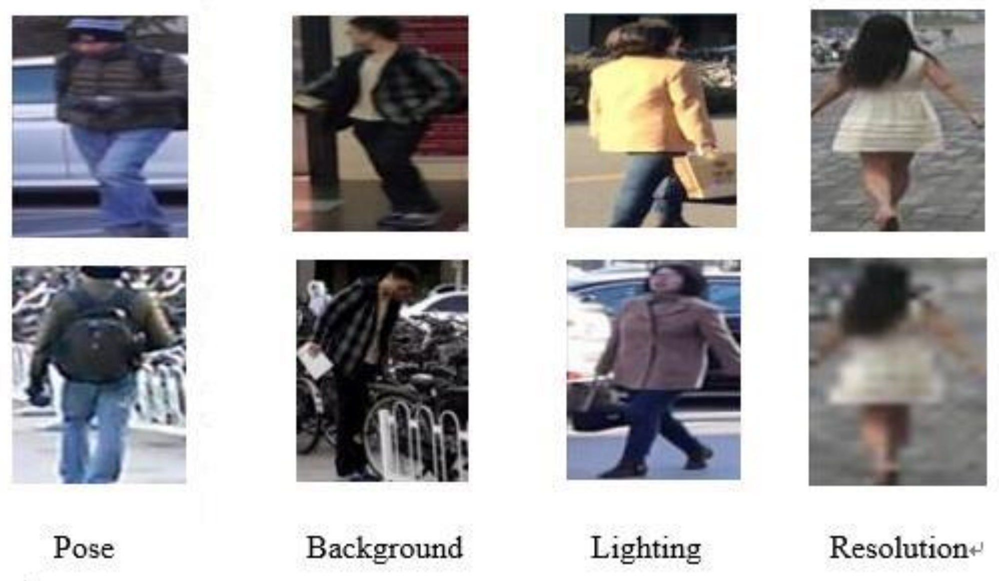


Figure 1-4 Examples of huge variants of pedestrian appearance

(3) The third challenge is the non-ideal scene. There are some problems in the image like misalignment, occlusion and low image quality which will affect the ReID, as shown in the Figure 1-5.

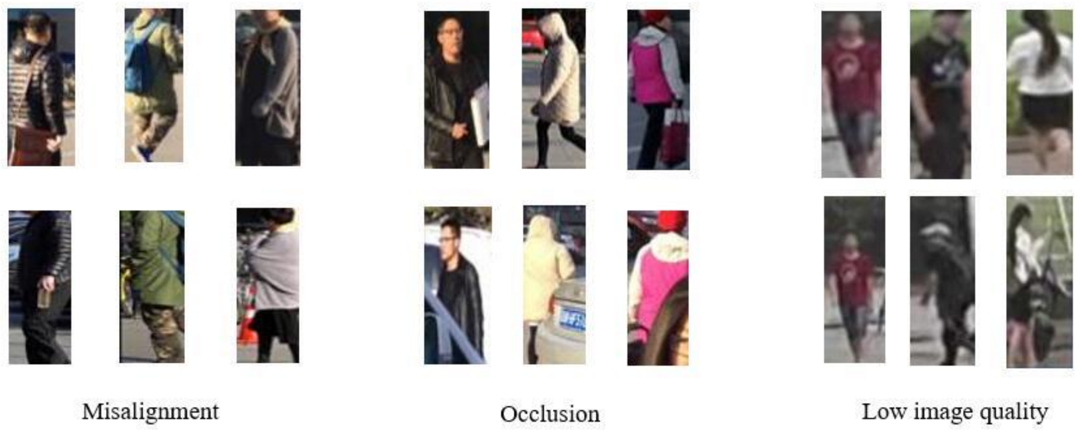


Figure 1-5 Examples of non-ideal scene

1.3 Research History and Present Situation Related work

1.3.1 Research History

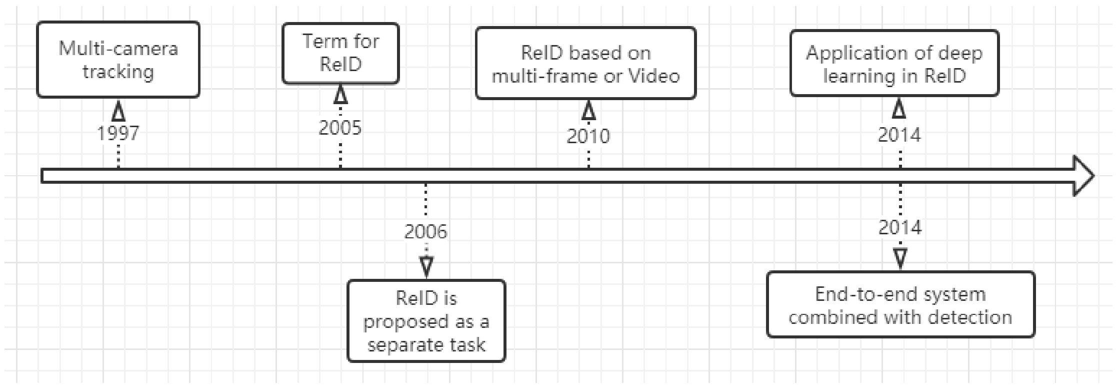

Multi-camera tracking
1997
Term for ReID
2005
ReID based on multi-frame or Video
2010
Application of deep learning in ReID
2014
2006
ReID is proposed as a separate task
2014
End-to-end system combined with detection

Figure 1-6 Development history of ReID

ReID is originated from muti-camera tracking problem. As shown in the figure 1-6, in 1997, Russell and Huang proposed a method which used the Bayesian formula based on the appearance characteristics of the object to estimate the posterior probability of an object captured by a camera appearing on other cameras. [2] [3] proposed the ReID term to identify the person when he come back the target area. In their method, everyone has an unique ID. In 2006, [4] proposed the first ReID method based on the images. They used the space-time segmentation algorithm to detect the foreground area of the pedestrian, and then extracted the visual features of the pedestrian, finally did the features matching. This paper is a symbol that ReID is divided from the multi-camera tracking task. In 2010, [5][6] proposed an innovative method of multi-frame image for ReID. In 2012, Hinton team joined the ImageNet image recognition competition, constructed AlexNet CNN and won the championship. After this competition, deep leaning became popular. In 2014, [7] firstly used deep learning in ReID. [8] proposed a end-to-end system combined detecting.

1.3.2 Present Situation Related Work

For ReID, there are four main research directions. One is based on feature extraction, one is based on similarity measurement which is metric learning essentially, one is based on video sequence, one is based on GAN mapping.

The traditional feature extraction is handcrafted features. [9] uses explicit polynomial kernel to generate the feature mapping to describe the similarity between blocks corresponding to two images. [10] proposed a effective feature representation method based on local maxima named LOMO. With the convolutional neural network (CNN) has achieved great success in various fields of computer applications, the latest feature extraction methods for ReID are based on CNN. And besides the global feature, researches begin to work on the local features. [11] cut the picture vertically. [12] proposed Spindle Net which extracts feature using 14 body key points. [13] proposed GLAD net which divides the body into head, upperbody and lowerbody three parts.

Metric learning is aimed to learn the similarity between two images through the network. On the issue of ReID, it means that the similarity of different pictures of the same pedestrian is greater than the different pictures of different pedestrians. The common metric learning methods include of Triplet loss[14-16], Contrative loss[17], TriHard loss[18] and Margin sample mining loss (MSML)[19].

The information of a single frame image is limited, so some recent works focused on video sequences in ReID [20-27]. These methods not only consider the contents information of the image, but also consider the motion information between the frame and the frame.

As the first difficulty mentioned above, the datasets for ReID are small nowadays. [27] first proposed using GAN in ReID to expand the datasets. [28] proposed CycleGAN to decrease the camera style disparities. [29] proposed that when a single frame image encounters occlusion, etc., it can be compensated by other information of multiple frames. Directly induce the network to reduce the importance of the poor quality frame.

1.4 Research Content and Structure Arrangement

1.4.1 Research Content

This article revolves around the issue of ReID. This work proposes to use several body parts with the global body to identify the pedestrians. Because this way not only uses the global body, but also uses the complementary information of different body parts.

This work uses a CNN model to detect four key points of the person and divides a person picture in three parts. Then input each part to the CNN model respectively to extract the feature. Because some existing deep learning models have achieved excellent results in feature extraction, this work uses several layers of the existing model as the first few layers of the model to extract features directly, and fuse those features to form a feature vector. And then use Euclidean distance to measure the similarity. Finally re-identify the pedestrian. This method results of accuracy achieves the state-of-art level.

1.4.2 Structure Arrangement

This article has five chapters. The contents of each chapter are as follows:

The first chapter introduces the background and significance of this work, explains the concept, basic frame work and difficulties of person re-identification.

2 The theoretical and technical foundations involved

2.1 The introduction of deep learning

In recent years, the term ‘Artificial Intelligence’ (AI) is popular in our life: Robotics, self-driving cars, AlphaGo and son on. Prof. Winston in MIT proposed that AI field is to do the study how to make computers do intelligent work that only human beings could do in the past. For example, heavy scientific computations in the past are undertaken by the human brain. However, nowadays, computers has developed a lot and it can do more accurately and faster than the human brain. From the 1950s to the end of the 1980s, the mainstream of artificial intelligence was symbolic AI which was classical programming design. In the symbolic AI, people input the rule and the data. The machine output the answer. Then machine learning appeared. People input the data and the expected answer. The machine output the rule. Then the rule can be applied in new data and generate new answers. Deep learning is a part of machine learning. It emphasizes learning from successive layers and it tries to simulate the brain of humans to extract features. The initial version of deep learning is artificial neural networks.

2.2 Artificial Neural Networks

2.1.1 Neuron model

In Artificial neural networks, a neuron is the most basic unit. The term is from the human’s brain because they have the same principal. In the math model of the neuron, the signal (eg.x_0) along the axon will do the multiplication interaction with other neuron’s based on the strength of the synapses which is w_0. The strength of the synapses which is weight w is learnable and can control the intensity of the influence of one neuron on another. In the model, dendrites pass the signals to the cell body, then the signals are added in the cell body. If the sum is larger than the threshold, the neuron will be activated and output a peak signal to the dendrite. The activation rate of a neuron is defined as an activation function f. The model is shown in the figure 2.1. The left is the neuron and the right is the mathematical model.

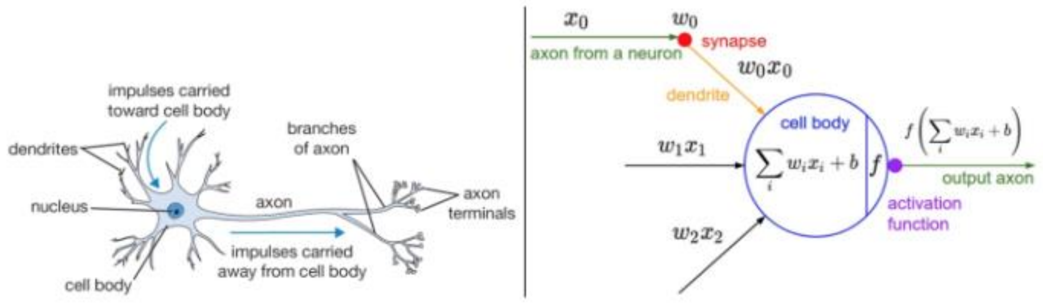


Figure 2-1 The left is the neuron and the right is the mathematical model

2.1.2 Forward propagation

Neural networks is the set of the neurons and the neurons. They are connected with the acyclic gragh. Generally, neurons in a neural network model are layered. And the most common layer is the fully connected layer. The neurons in the fully connected layer are connected with the neurons in the front and back layers completely. However, there is no connection between the neurons in the same fully connected layer. However, if there is only forward propagation, the previous weight cannot be adjusted through the current state which limits practicality. To solve this problem, back propagation appeared.

2.2.3 Back propagation

Through forward propagation, we can get predicted values. To judge the predicted values, loss function is defined. The loss function is used to express the difference between the forecast and the actual data. The smaller the loss function value is, the better the model is. So we need calculate the partial derivative of the weighting function of the loss function, which is a complex composite function. In order to calculate the gradient, the chain derivation rule is introduced. For example, for $f(x,y,z)=(x+y)z,$ let $q=x+y,$ so $f=qz.$ And than calculate their differentials separately $\frac{\partial f}{\partial q}=z, \frac{\partial f}{\partial z}=q.$ The problem we care about is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$. The chain derivation rule is that equation (2-1).

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} \qquad (2\text{-}1)$$
$$\frac{\partial f}{\partial z} = q$$

If we need to derive the elements, we can derive them layer by layer, and then multiply the results. This is the core of the chain rule and the core of the back propagation algorithm.

2.3 Convolutional Neural Networks

CNN was proposed by Yann Lecun in 1998 and it was designed to process the data stored in multidimensional arrays. A simple CNN consists of layers arranged in sequence. Each layer in the network uses a differentiable function to transfer activation data from one layer to another. Convolutional neural networks are mainly composed of three types of layers: convolution layer, pooling layer and full connection layer (the same as in conventional neural networks). By superimposing these layers, a complete convolutional neural network can be constructed.

2.3.1 Convolution layer

When we process the high dimensional input like image, if we use fully connected model, it will use lots of weights. To reduce the numbers of weights, the definition of receptive field appeared. Every neuron only connected with partial area of input data. For example, imagine that the input image is $32\times32\times5$, if we use fully connected model, every neuron will have $32\times32\times3$, connection. However, is we use receptive field, every neuron will only have $3\times3\times3$, connection. In the CNN, depth, stride, zero-padding, these three parameters control the output size. The output depth of convolution layer is the same as the number of the filters. What each filter do is to find a kind of feature in the input data. The stride determines the number of the pixels when moving the filters every step. Zero-padding can control the size of the output data in space to ensure that the input and output are spatially uniform. The size of the output can be calculated by a formula: $\frac{W-F+2P}{S}+1$.

W represents the size of the input data. F represents the size of the receptive field in convolution layer. S represents the stride. P represents the numbers of the zero in zero-padding. Sometimes, zero values are used to fill the edges of the input matrix, so that we can filter the edges of the input image matrix. Zero padding can allow us to control the size of the feature map. From the above formulas, we know that the selection of step size is not random. For example, the input size W = 10, if zero-padding is not used, which is P = 0, filter size F = 3. Then stride S = 2 is not desirable. Because $\frac{10-3+0}{2}+1=4.5$, and the result is not an integer, which means that neurons can not uniformly and symmetrically pass the input data. So such super parametric settings are invalid. When PyTorch is used, it will report errors. In this case, if the zero padding is used, the settings can be made reasonable. So we need to design the size of the network reasonably so that all dimensions can work properly.

Besides, sharing parameters in the convolution layer can also reduce the number of the parameters. It means that consider a single 2D slice in the depth dimension as a depth slice. The reason why this works is that the features introduced before are the same, that is to say, the same filter can detect the same features in different locations. For example, if the output of a convolution layer is $30\times30\times42$, then the number of neurons is $30\times30\times42=37800$. If the window size is 3×3, and the input data depth is 10, then each neuron has $3\times3\times10=900$ parameters. Thus, there are $37800\times900=34020000$ parameters in a single layer convolution, so the operation speed is obviously very slow. According to the previous introduction, a filter can detect the features of space, (x_1, y_1) then it can detect the features of (x_2, y_2). so we can use the same filter to detect the same features. In the example above, there are 42 filters, which makes the output's depth be 42. Each filter parameter has the number of $3\times3\times10=900$. So the total number of parameters is $42\times900=37800$ which greatly reduces the number of parameters.

2.3.2 Pooling

The pooling layer can reduce the spatial size of the data gradually to reduce the number of parameters in the network and computational resource cost. Besides, pooling can control over-fitting. The pooling layer has the window like the convolutional layer. Generally, select the maximum value in the window as the final result, and then slide the window.

2.3.3 Fully connected layer:

Same as the fully connected layer in the structure of the neural network previously introduced. Convolution extracts local features, and full connection is to assemble the previous local features into a complete graph through weight matrix. Because all local features are used, it is called fully connected layer.

2.3.4 Activation function

Activation function is used to add non-linear factors, because the expression of linear model is not enough. And the properties of the activation function are as follows:

(1) Nonlinearity. Linear activation layer has no effect on deep neural network, because it is still various linear transformations of input after linear activation.

(2) Continuous differentiability. The requirement of gradient descent method.

(3) Unsaturated range is better. When there is a saturated interval, if the system optimization enters the interval, the gradient approximates zero, and then the learning of the network will stop which is not expected.

(4) Monotonicity. When the activation function is monotonic, the error function of the single-layer neural network is convex and easy to optimized.

(5) It is approximate linearity at the origin, so that when the weight is initialized to a random value close to 0, the network can learn faster, without adjusting the initial value of the network.

However, at present, the commonly used activation functions only have some of the above properties, and none of them has all of them. The activation functions of neural networks include Sigmoid, Tanh, ReLU and so on. ReLU is the most commonly used activation function in CNN. These activation functions are described below.

- **Sigmoid function**

Sigmoid function has been widely used, but because of its own shortcomings, it is seldom used now. The Sigmoid function is defined as:

$$f(x)=\frac{1}{1+e^{-x}} \tag{2-2}$$

The image corresponding to the function is:

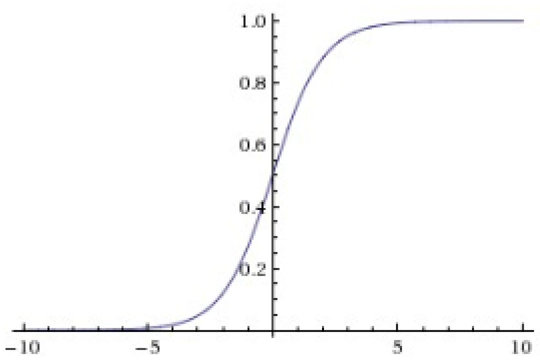


Figure 2-2 Sigmoid function graphics

Advantage:

1. The output of Sigmoid function is between (0,1). It is monotonous and continuous, with limited output range. It can be used as output layer.

2. Differentiation is easy.

Disadvantages:

1. It has saturation defect. When the Sigmoid function is near the ends of 1 and 0, the gradient becomes almost zero. However, the gradient descent method updates the parameters by multiplying the learning rate by the gradient. So if the gradient approaches 0, there will be no information to update the parameters, resulting in non-convergence of the model.

2. Sigmoid output's mean does not 0, which will result in that the output after Sigmoid activation function is non-zero-mean as the input of the latter layer of network. If the input into the next layer of neurons are all positive, it will lead to all gradients be positive. Then it will always be positive gradient when updating parameters.

- **Tanh function**

Tanh activation is a transformation of the above Sigmoid activation function, and its mathematical expression is as follows

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2-3}$$

The image corresponding to the function is:

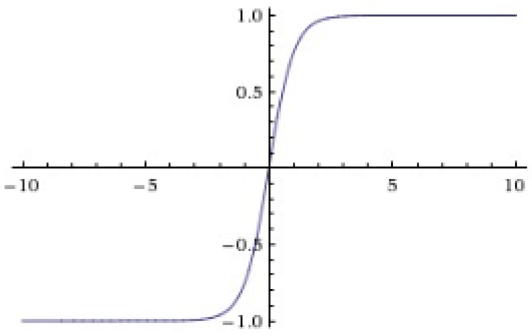


Figure 2-3 Tanh function graphics

It converts the input data to - 1 - 1 and the output is 0 means, but it still has the biggest problem the same as Sigmoid function - the loss of the gradient due to saturation property.

➢ **ReLU**

ReLU represents the Rectified Linear Unit, which is a non-linear operation. It is a very popular activation function in recent years. It is defined as:

$$y = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases} \tag{2-4}$$

The corresponding image is

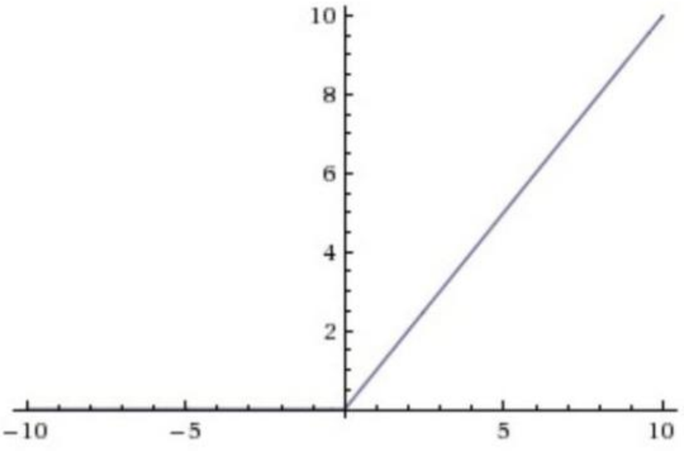


Figure 2-4 ReLU function graphics

Constructing sparse matrices whose most of the elements are 0, that is, sparsity, can remove redundancy in data and preserve the features of data as much as possible. In fact, this can be achieved by Relu, which is Max (0, x). Because of the exist of sparse matrices, the neural network has become faster and better. So we can see that most of the current convolutional neural networks basically use ReLU function.

Advantages of ReLU

1. Compared with the Sigmoid and Tanh activation functions, the ReLU activation function can greatly accelerate the convergence rate of the stochastic gradient descent method, because it is linear and there is gradient disappearance problem.

2. The calculation of ReLU is simpler. Only one threshold filter is needed to get the result.

Disadvantages:

During training process, neurons may die and weights cannot be updated. For example, a large gradient passes through the ReLU function. If this happens, the gradient that passes through the neuron will always be zero from this point on. That is to say, ReLU neurons die irreversibly during training.

➢ **Leaky ReLU**

Leaky ReLU activation function is a variant of ReLU activation function. The main purpose is to repair the weakness of training being fragile in ReLU activation function. Instead of turning the part of x < 0 into 0, Leaky ReLU activation function gives it a small slope, such as 0.01. Its mathematical form can be expressed as

$$f(x)=\begin{cases}\partial x & x\le 0\\ x & x>0\end{cases} \tag{2-5}$$

where alpha is a very small constant. In this way, when the input is less than 0. There is also a small gradient.

The image corresponding to the function is:

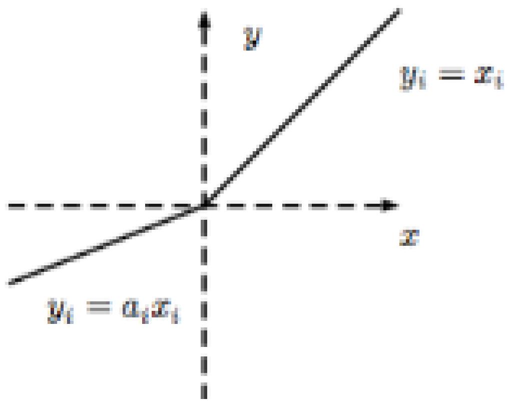


Figure 2-5 Leaky ReLU function graphics

- **Maxout**

The other type of activation function is not f(wx+b) acting on an output form, but is $\max(w_1x+b_1, w_2x+b_2)$, which is a Maxout type. It can be found that the ReLU activation function is a special form of $w_1=0, b_1=0$ in Maxout. So Maxout has the advantage of ReLU and avoids the disadvantage of ReLU. However, it also increases the parameters of the model, resulting in a larger storage of the model.

2.3.5 The development of convolutional neural networks

ImageNet (ImageNet Large Scale Visual Recognition Challenge, ILSVRC)[1] has always been known as the "Olympic" in the field of international computer vision. In this part, I will introduce several classical architecture. The following figure 2-6 shows the Top-5 error rate of ILSVRC over the years.

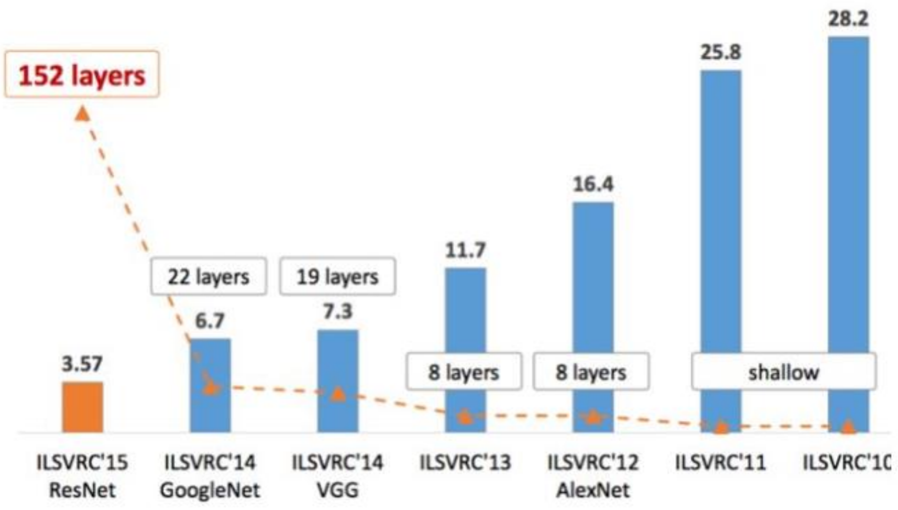


Figure 2-6 Top-5 error rate in ILSVRC Target Classification Task in 2010-2015

- **AlexNet**

AlexNet won the ImageNet Competition in 2012 with an absolute advantage of more than 10.9 percentage points. Since then, deep learning and convolutional neural networks

[1] http://www.image-net.org/challenges/LSVRC/

have become more and more popular. Research on deep learning has sprung up. The AlexNet architecture is shown as figure 2-7:

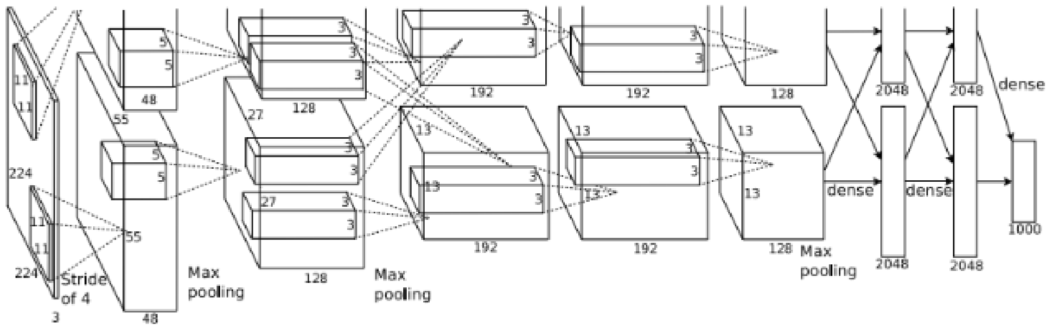


Figure 2-7 AlexNet network mode [30]

The first five layers of Alexnet are convolution layers, the last three layers are fully connected layers. The output of the last fully connected layer is a 1000-dimensional SoftMax classifier. The optimization goal is to maximize the average of multiple logistic regression.

Analysis of AlexNet:

1. Image preprocessing: Scale the image to get an image of 256 in length or width, and then intercept it to get 256×256 samples. The training sample images are subtracted from the mean value.

2. Data augmentation: The input picture is a three-channel color picture of 256×256. In order to enhance the generalization ability of the model and avoid over-fitting, the author uses the idea of random clipping to clip the original 256×256 image randomly, and gets the images of $3 \times 224 \times 224$ size, and then inputs them to the network training.

3. Local response normalization(LRN): adding a local response normalization part after ReLU will make the network achieve better generalization effect. The calculation formula is as follows:

$$b_{x,y}^{i} = \frac{a_{x,y}^{i}}{(k + \partial \sum_{j=\max(0,i-\frac{n}{2})}^{\min(N-1,i+\frac{n}{2})} (a_{x,y}^{i})^{2})^{\beta}} \tag{2-6}$$

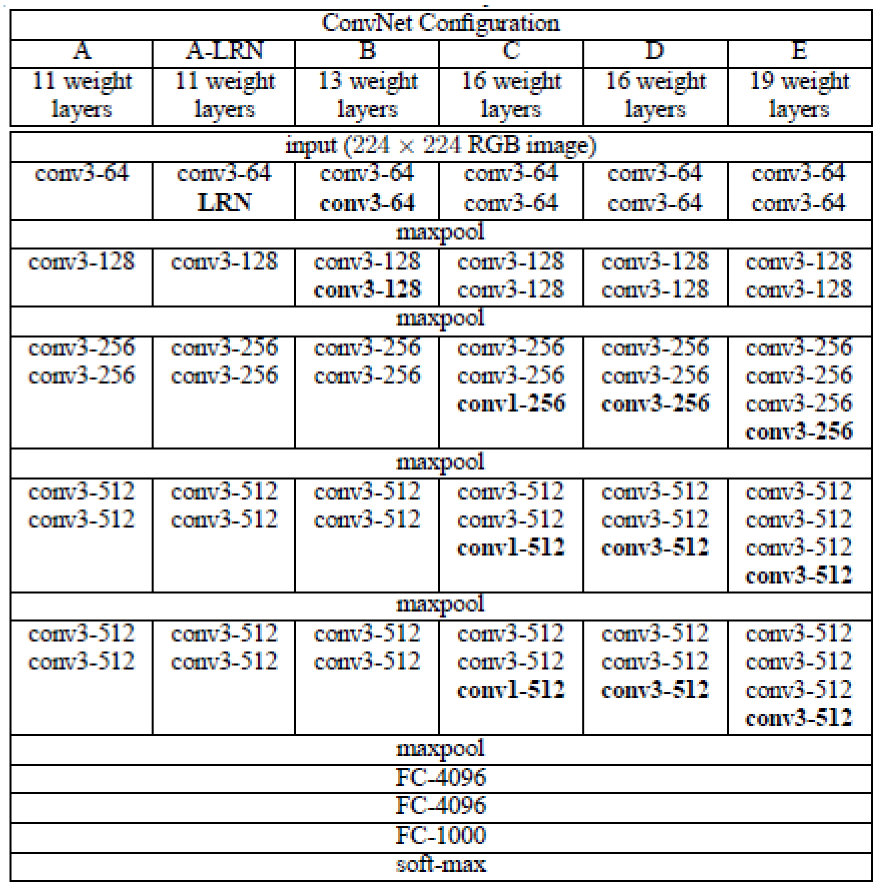
$a_{x,y}^{i}$ denotes the convolution of the input position (x, y) for the first time and the result of ReLU. $b_{x,y}^{i}$ denotes the normalized result. n denotes the nth convolution of the same position around the ith time. N denotes the total convolution times of this layer. k, n, ∂, β are hyperparametric.

4. Dropout: This architecture introduces Dropout into the first two fully connected layers. It adopts the method of inactivating neurons with a certain probability (e.g. 0.5) and those neurons will not join the forward and backward propagation.It is equivalent to about half of the neurons not functioning. Then when testing, multiply the output of all neurons by 0.5. The use of Dropout alleviates the over-fitting of the model effectively.

5. Dual GPU Parallel Architecture: Due to the limitation of GPU hardware at that time (Type: NVIDIA GeForce GTX 580), only 3G memory is available. With this architecture, training can be accelerated.

➢ **VGGNet**

VGG-Nets is proposed by the VGG (Visual Geometry Group) of Oxford University. It is the basic network of the first prize-location task and the second prize-classification task in the ImageNet Competition in 2014. VGG can be seen as a deepened version of AlexNet. At that time, VGGNet seemed to be a very deep network, because the number of layers is as high as 16-19 layers. All the convolution layers in VGGNet use the same convolution core, which is 3×3 in size and 1 in step size. It imitates AlexNet's structure, but by reducing the size of the convolution core, it increases the number of network layers and reduces network parameters. The following table describes the network structure and birth process of VGG-Net:



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

Figure 2-8 VGG network mode[31]

➢ **GooLeNet**

GoogLeNet beats VGG-Nets and won the championship in the ImageNet classification task in 2014. Google LeNet, unlike AlexNet and VGG-Nets, which only rely on deepening the network structure to improve network performance, has opened up a new idea. While deepening the network (22 layers), it has also made innovations in network structure, introducing Inception structure instead of the traditional operation of simple convolution + activation. GoogLeNet has further advanced the study of convolutional neural networks to a new level.

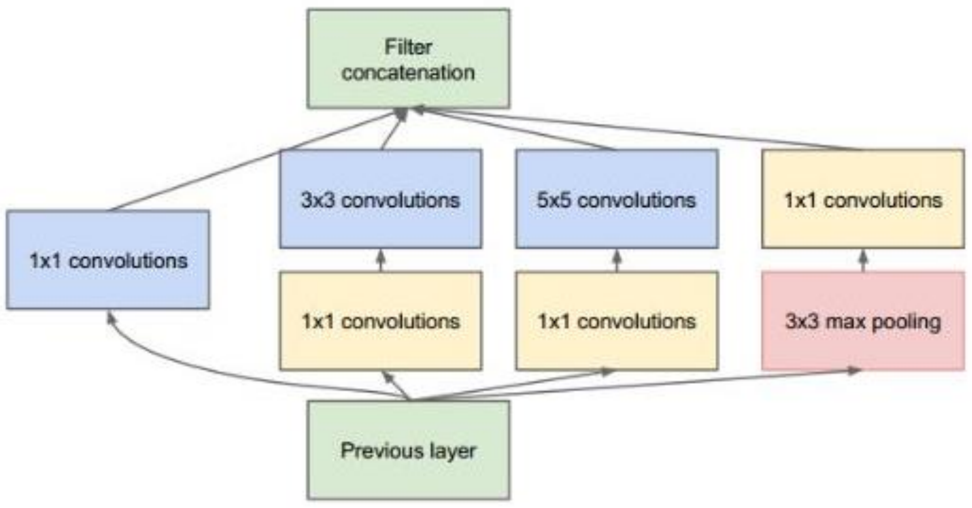


Figure 2-9 Inception network model[32]

1. The above structure is Inception. The convolutional strides in the structure are all 1. In addition, zero filling is used to keep the size of the response graph consistent. Finally, there is a ReLU layer immediately behind each convolution layer. There is a layer called concatenate layer before the output. That is to say, four groups of feature response maps of different types but of the same size are stacked side by side to form a new feature response map. There are two main tasks in Inception structure: 1) Feature extraction of input response charts is done in four ways: 3×3 pooling and 1×1, 3×3 and 5×5 convolution cores. 2) In order to reduce the amount of calculation, at the same time, the information can be transmitted through fewer connections to achieve more sparse characteristics, it uses 1×1, convolution core to achieve the dimension reduction.

2. There are three Loss units in the architecture of the GoogLeNet network, which is designed to help the convergence of the network. The purpose of adding Loss unit in the middle layer is to make the low-level features have good distinguishing ability when calculating the loss, so that the network can be better trained.

3. GoogLeNet also has an advantage which worth mentioning, that is, replacing all the subsequent fully connected layers with simple global average pooling, with fewer parameters. In AlexNet, the parameters of fully connected layer in the last three layers account for almost 90% of the total parameters. Using large networks allows GoogLe Net to remove fully connected layer in width and depth, but it does not affect the accuracy of the results. GoogLeNet achieves 93.3% accuracy in ImageNet which is faster than VGG.

➢ **ResNet**

ResNet is the winner of the ILSVRC2015 competition and the model of my work refers to Residual Neural Network (ResNet) — ResNet was proposed by Kaiming He.etc four Chinese from Microsoft Research Institute. They trained a 152-layer neural network successfully by using ResNet Unit. The error rate on top5 was 3.57%, whose effect was very prominent. So why is ResNet doing so well? In fact, ResNet solves the problem that deep CNN model is difficult to train. VGG network tries to find out how deep the depth of deep learning network can be to continuously improve the accuracy of classification. For the traditional deep learning network, we generally believe that the deeper the network depth (the more parameters) the stronger the non-linear expression ability, the more things the network can learn. On the basis of this rule, the classical CNN network developed from LetNet-5 (layer 5) and AlexNet (layer 8) to VGGNet (16-19) and then to Google Net (layer 22). According to the experimental results of VGGNet, to some extent, the depth of the network is crucial to the performance of the model. When the number of network layers increases, the network can extract more complex feature patterns. So when the model is deeper, better results can be obtained theoretically. But we find that it is not sure that deeper networks perform better. On the contrary, we find that over-deepening the number of layers in conventional CNN networks will lead to a reduction in classification accuracy. The comparison results are shown in Figure 2-10. We call this problem as Degradation Problem.

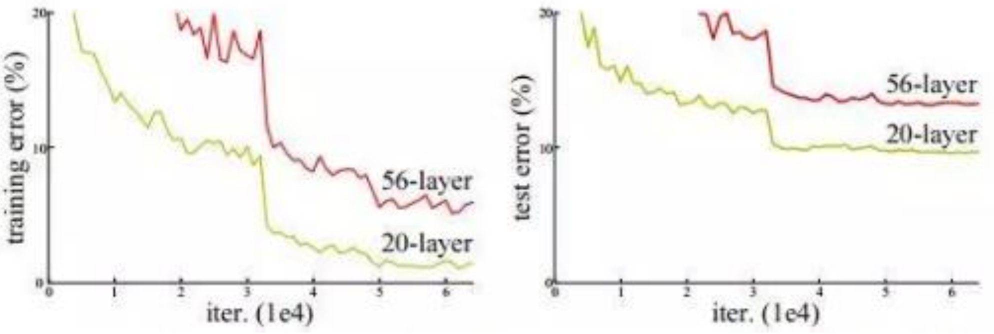


Figure 2-10 Training error(left) and test error(right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error

Kaiming He proposed residual learning to solve the degradation problem. For accumulation layer (a few layers of accumulation), when the input is x, the feature it learns is H (x). Now we hope that it can learn the residual F (x) = H (x) - x, so the original learning feature is H (x). The reason for this is that residual learning is easier than direct learning of original features. When the residuals are F(x)=0, the accumulation layer only do the identity mapping at this time, at least the network performance will not decline. In fact, the residuals will not be 0, which will also enable the accumulation layer to learn new features on the basis of input features, thus having better performance. In figure n, on the left is an ordinary network, and on the right is the structure of ResNet residual learning unit. ResNet change the learning target. Instead of learning a complete output H(x), ResNet learns the difference between output and input which is H(x) –x called 'residual'. This is somewhat similar to the "short circuit" in the circuit, so it is a short cut connection.

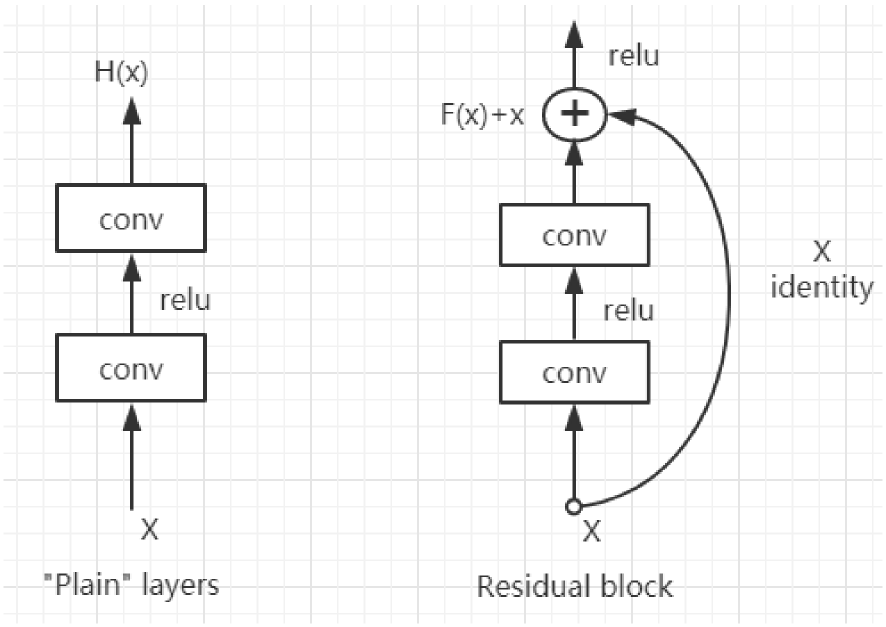


Figure 2.11 Ordinary Network and ResNet Residual Learning Unit

2.4 Deep learning framework

In the initial stage of deep learning, every researcher needs to write a lot of repetitive codes. In order to improve the efficiency, these researchers wrote the codes to a framework and put them on the Internet for all other researchers to use. Then different frameworks emerged. At present, the most popular deep learning frameworks in the world are

TensorFlow, Caffe, Torch, PyTorch and so on. I use PyTorch in this work. PyTorch was developed by the torch7 team, using Python as the development language. Comparing to TensorFlow which is static, PyTorch supports the dynamic neural networks through a reverse automatic derivation technique. PyTorch is more visual and easier to use.

2.5 Some terms in the deep learning

2.5.1. Parameters

Parameters are the variables that the model can learn automatically from the data. It refers the to configuration variables within the model, which can be estimated by data. Parameters are learned or estimated from data and are usually not set artificially. They are often saved as part of the final model. They are the key to deep learning algorithms. We can regard models as hypotheses and parameters as tools to implement hypotheses for a particular data. Generally speaking, model parameters are the optimal solution of the algorithm after considering all possible values.

In statistics, we may assume a distribution for a variable, such as a Gauss distribution. The Gauss distribution has two parameters: mean and standard deviation. This understanding is also valid in deep learning, and these parameters can be estimated by data as a part of the prediction model.

In programming, we may pass parameters to functions. In this case, as a function argument, a parameter can take one of a series of values. In deep learning, we use models to predict functions and parameters in external data.

2.5.2 Hyper-parameters

Hyper-parameters are parameters that set values before starting the learning process, not parameters that are obtained through training. Usually, it is necessary to optimize the hyper-parameters and select a set of optimal hyper-parameters for the learning machine in order to improve the learning performance and effect. Hyper-parameters' selection is independent from training process, and the training process will not affect the hyper-parameters. But after the training, we can consider whether the hyper-parameters can be optimized according to the training results. If they can be optimized, we will adjust the value of the hyper-parameters and then begin to train for next epoch. The following introduces the adjustment rules of different hyper-parameters.

(1) Learning rate

Learning rate (lr) refers to the magnitude of updating the network weight in the optimization algorithm. The learning rate can be constant, decreasing, momentum-based or adaptive. Different optimization algorithms determine different learning rates. When the learning rate is too large, the model may not converge and the loss will continue to oscillate up and down; if the learning rate is too small, the convergence speed of the model will be slow, which requires longer training time. Usually the lr value is [0.01, 0.001, 0.0001]

(2) Batch size

Batch size is the number of samples that each training neural network sends into the model. In convolutional neural networks, large batches usually make the network converge faster. However, due to the limitation of memory resources, large batches may lead to insufficient memory or program core crash. Bath_size is usually taken as [16, 32, 64, 128]

(3) Optimizer

At present, Adam is a fast convergent and often used optimizer. Stochastic gradient descent (SGD) converges slowly, but the momentum can accelerate the convergence. Meanwhile, the stochastic gradient descent algorithm with momentum has better optimal solution, that is, the model will have higher accuracy after convergence. Usually Adam is used more if you are pursuing speed.

(4) Number of iterations

The number of iterations refers to the number of times that the whole training set is input into the neural network for training. When the difference between the test error rate and the training error rate is small, the current iteration number can be considered appropriate. When the test error rate first decreases and then increases, the number of iterations is too large, and the number of iterations needs to be reduced. Otherwise, it is prone to over-fitting.

2.5.3 Training set, Validation set, Test set

Usually, when training supervised deep learning model, the data will be divided into training set, verification set and test set. The original data is divided into three sets in order to select the model with the best effect (which can be understood as accuracy) and the best generalization ability.

- **Training set**

After determining the model, the training set is used to train parameters (instead of hyper-parameters)should be used. The function of training set is to fit the model and train the classification model by setting the parameters of the classifier. When the validation set is used, different values of the same parameter are selected and multiple classifiers are fitted.

- **Validation Set**

After multiple models are trained by the training set, the validation set is used to find the best model. Use each model to predict on the validation set and record the accuracy of the model. The parameters corresponding to the best model are selected. So in this way it can adjust the parameters of the model.

- **Test set**

After the optimal model is obtained by training set and verification set, the test set is used to predict the model. It is used to measure the performance and classification ability of the optimal model. That is to say, the test set can be regarded as a data set that never existed. When the parameters of the model have been determined, the test set can be used to evaluate the performance of the model. Test set doesn't join the training process.

Dividing the original data into three datasets is also to prevent the model from over-fitting. When all the original data are used to train the model, the result may be that the model fits the original data to the greatest extent, that is to say, the model exists to fit all the original data. When a new sample appears and then the model is used to predict, the effect may be worse than using only a part of the data training model.

2.6 Summery

This chapter introduces the definition of the deep learning, CNN, and highlight ResNet network which is used in the model of this work.

3 Global and Local based Person Re-identification

3.1 Introduction

With the popularity of video surveillance technology, the role of ReID technology has become increasingly important. It can help people automatically complete the task of searching for specific person from massive images or video data.

Feature extraction with CNN and feature matching is two important parts of ReID. However, as mentioned in 1.2.3, the different images of pedestrians acquired by different cameras usually have large changes in attitude and perspective, which greatly increases the difficulty of ReID algorithm for pedestrian matching.

In order to solve the technical problems above, it is necessary to propose a new ReID algorithm which can adapt to the change of pedestrian posture better and extract more robust features, so as to improve the final correct recognition rate or reduce the error recognition rate.

In this work, I designed a four-branch network. The first neural network uses the original global body image of the pedestrian as the first input and outputs the first recognition feature. The second, third, fourth neural networks use the different body part of the pedestrian as the input and output the features. The body parts of the pedestrian includes the head, upper body and lower body. The global and local features are combined as the total recognition features and then do the feature matching using them.

The advantages of this model are that it has more robust pedestrian feature matching ability, thereby improving the correct recognition rate and reducing the error recognition rate.

3.2 Model framework

Considering the excellent performance of ResNet-50 convolutional neural network in the field of pedestrian identity re-identification [33], this work uses ResNet-50 as the benchmark model framework. Figure 3-1 is a model framework of this work, which consists of four sub-networks: a global descriptor learning sub-network and three part descriptor learning sub-networks. Each subnetwork consists of a ResNet-50 basic model and an ID classifier.

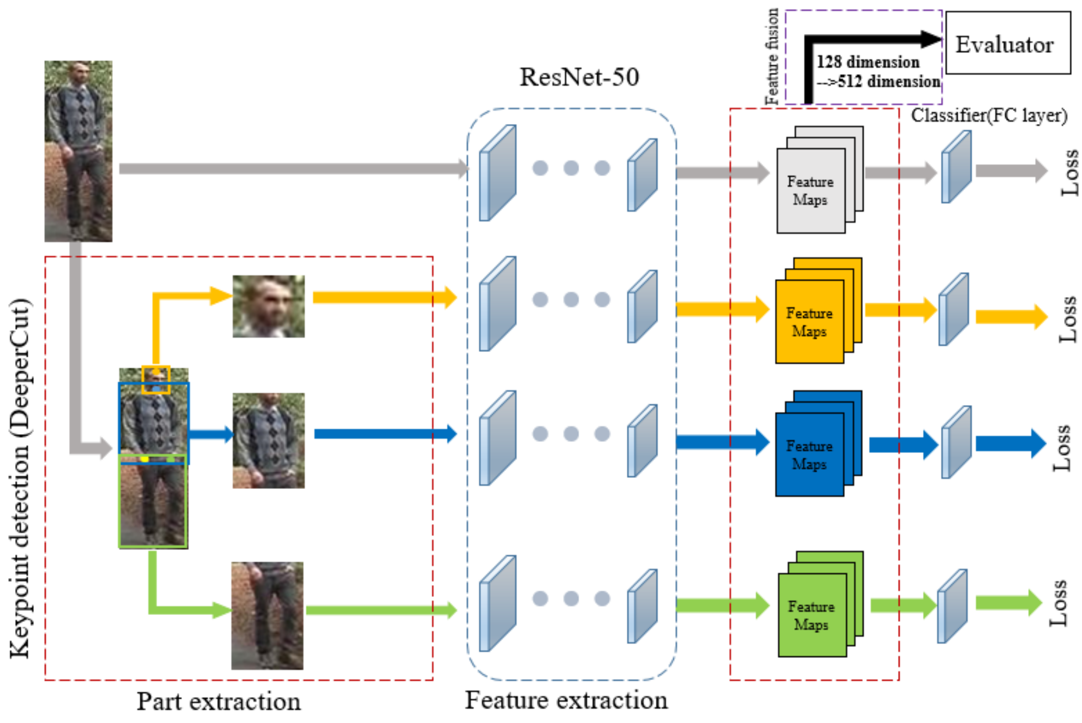


Figure 3-1 The framework of the this work's model

3.2.1 Key point detection

The first step of dividing the parts is to detect the person key point. The key point detection method proposed by [34][35] detected 14 key points of pedestians. In Deepercut algorithm, the CNN which adopts ResNet is used to detect the body parts candidates. This network is trained and tested on the "MPII Human Pose"("Single Person") dataset. And the ResNet-152 gets 87.8% PCK (Percentage of Correct Keypoints) Besides, this model proposes novel "image-conditioned pairwise terms". It can compress enough candidate area nodes to a certain number of nodes. As shown in the figure below, it can determine whether they are different important joints according to the distance between the nodes in the candidate region. Finally, this model can detect 'ankle', 'knee', 'hip', 'wrist', 'elbow', 'shoulder', 'chin', 'forehead' which are 14 key points. Figure 3-2 is an example.

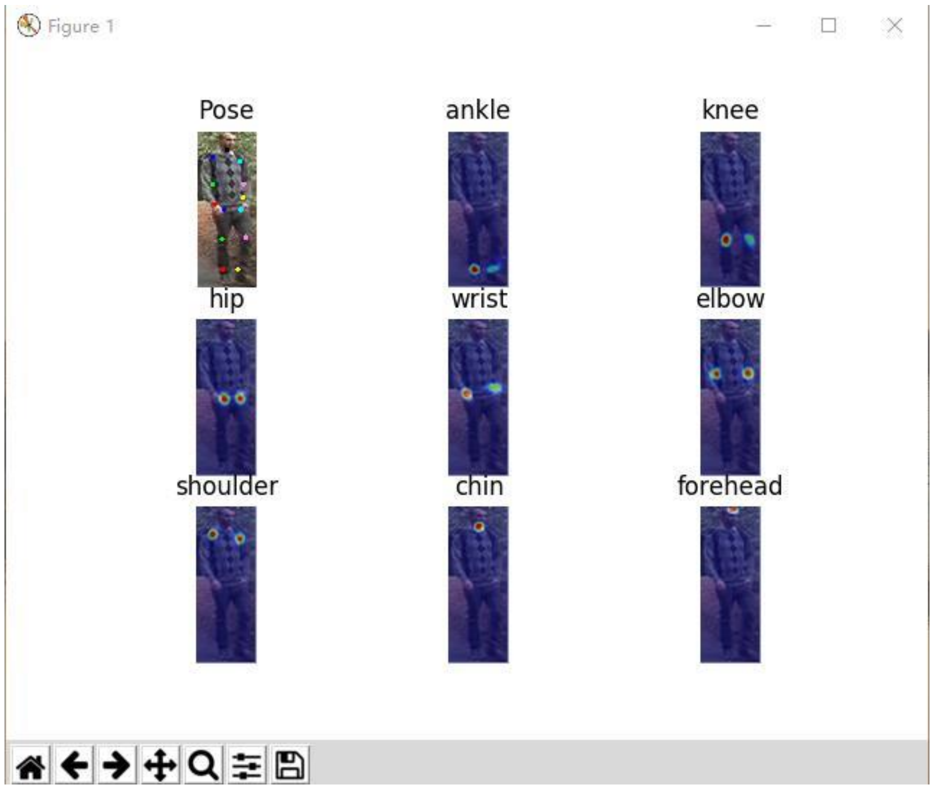


Figure 3-2 The example of the DeeperCut keypoint detection

3.2.2 Divide the parts

Body part extraction has been studied by many researchers. Parts are divided according to the position of key points of human body. For example, [36] is divided into seven parts (five fine-grained parts and two coarse-grained parts) according to the way shown in the left picture in Figure 3-3. [37] is divided into three parts according to the way shown in the middle picture in Figure 3-3. Pedestrian images in personal recognition are captured in an unconstrained environment and are susceptible to occlusion, viewpoint changes, posture changes and so on which make it difficult to detect fine-grained parts. Besides, excessive fine-grained partitioning will lead to inaccurate parts positioning because it will result in noise in the part area, which will affect the performance of ReID experiment. In this work, the coarse-grained partitioning method in reference [37] is adopted as shown in the right picture in figure 3-3.

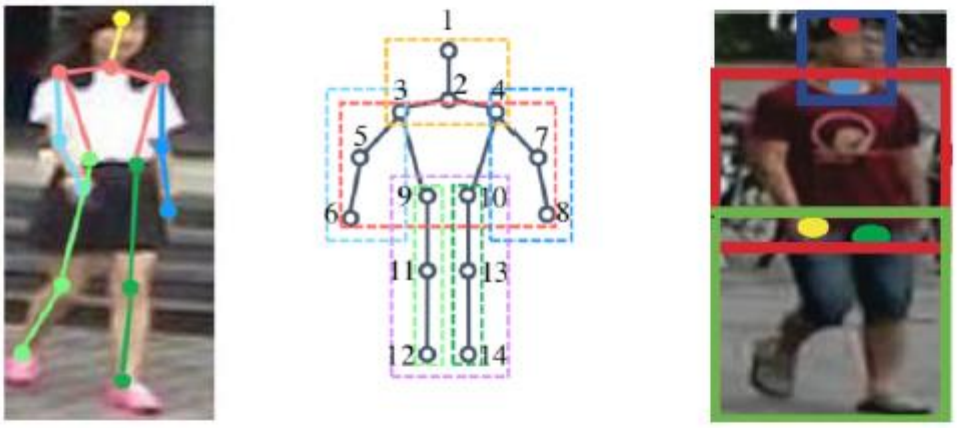


Figure 3-3 The left picture is 14 keypoint locations, the middle is the 7 parts, the right is four keypoints and three parts.

According to the method which 3.2.1 mentioned, this work uses four of fourteen detected key points: forehead, chin, left hip, right hip. Based on the position of these four key points and the division method of [37], I divide the pedestrian global image into three parts images. Assume the size of the input pedestrian global image is $H \times W$. The coordinate of the forehead is (x_1, y_1) The coordinate of the chin is (x_2, y_2) Firstly, the head part region is segmented according to the following formulas(3-1):

$$\begin{aligned} &R^{head} = [(x_m - \frac{b}{2}, y_1 - \alpha), (x_m + \frac{b}{2}, y_2 + \alpha)], \\ &b = y_2 - y_1 + 2 \cdot \alpha, \\ &x_m = \frac{x_1 + x_2}{2} \end{aligned} \tag{3-1}$$

The shape of the region is a rectangular. The first item is the coordinates of the upper left corner, and the second item is the coordinates of the lower right corner. α is a parameter for controlling overlap between adjacent parts. I set α as 15 in this work.

Assume the coordinate of the left hip is (x_3, y_3). The coordinate of the right hip is (x_4, y_4). Then the upper body part region and lower body part region are segmented according to the following formulas (3-2):

$$
\begin{aligned}
R^{upper} &= [(0, y_2 - 2\cdot\alpha), (W-1, y_m + 2\cdot\alpha)],\\
R^{lower} &= [(0, y_m - 2\cdot\alpha), (W-1, H-1)], \qquad (3\text{-}2)\\
y_m &= \frac{y_3 + y_4}{2}
\end{aligned}
$$

The segmented part images and the whole body are shown in the left-most column of figure 3-1. After resize their sizes to 256×128, I input them into the corresponding sub-network.

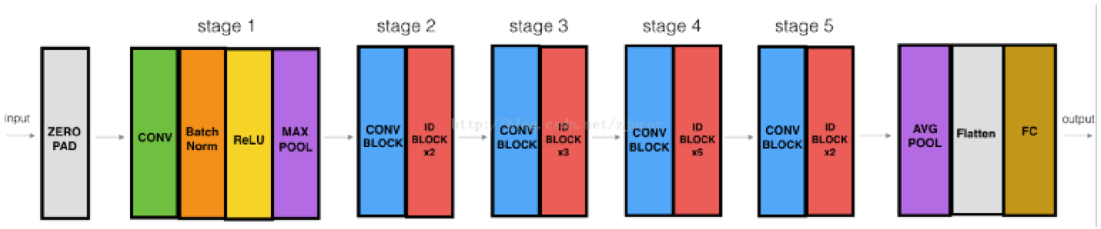
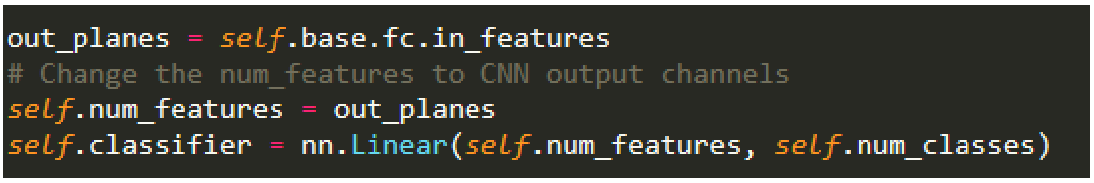


Figure 3.4 The basic framework of ResNet-50

3.2.3 Feature extraction

In this work, the four branches train independently and similarly. So I describe the training process of one branch in detail. Firstly, the image is processed by ResNet-50 convolution neural network. After fully connected layer, I extract the features after the "avgpool" layer and a 128-dimensional feature map is obtained. As mentioned in 2.3, the fully connected layer plays the role of classifier in CNN. This layer can map the distributed feature representation learned to the sample labeled space. So I append another fully connected layer after the Resnet50 in order to make the dimension of the output feature the same as the number of training samples classes. In this way, the feature identity classifier training is realized.



```
out_planes = self.base.fc.in_features
# Change the num_features to CNN output channels
self.num_features = out_planes
self.classifier = nn.Linear(self.num_features, self.num_classes)
```

Figure 3-5 The code example of the added fully connected layer

3.2.4 Loss Function

➢ **Softmax loss**

Unlike double classification function sigmoid, Softmax is suitable for multi-qclassification problems. The aim is to present the results of multi-classification in the form of probability. Softmax function, also known as normalized exponential function, can "compress" a K-dimensional vector containing any real number into another K-dimensional real vector. And it will let the range of each element be between 0 to 1, let the sum of all elements be 1. In this work which is a muti-classificatino problem, I use the Softmax loss.

Given a training image x, the label of the image is d. Take the global body part for example. The output vector after the classifier of this model is $y = [y_1, ..., y_m] \in R^m$. The prediction probabilities that this training image belongs to identity classes $m \in 1, ..., k$:

$$p(m \mid x) = \frac{e^{ym}}{\sum_{i=1}^{k} y_i} \tag{3-3}$$

Identity classification loss function defines

$$L_{global} = -\sum_{m=1}^{M} \log(p(m \mid x)) q(m) \tag{3-4}$$

q(m) is defined as

$$q(m) = \begin{cases} 1, & m = d \\ 0, & m \neq d \end{cases} \tag{3-5}$$

To minimize the loss function of identity classification, it is equivalent to maximize the prediction probability of training image x belonging to identity label d. The loss functions of head, upper body and lower body are similar, which are defined as $L_{head}, L_{upper}, L_{lower}$ respectively.

➢ **Triptlet Loss**

The figure 3-6 is the network structure used Triptlet Loss. The first part is a common convolutional neural network. But unlike the general deep learning architecture, it adds an embedding layer. Embedding means a mapping relationship, that is, the output features after the end of the convolutional neural network are mapped from the original feature space to a new feature space. The new features can be called an embedding of the original feature, and then Triplet loss is used as a supervisory function to update the gradient of the network. Triptlet loss is different from software max. The final number of classes of Softmax is determined. However, the aim of metric embedding learning is to learn a function $f_\theta(x): R^F \rightarrow R^D$ and then it can learn a good embedding. Similar images are similar in embedding space.

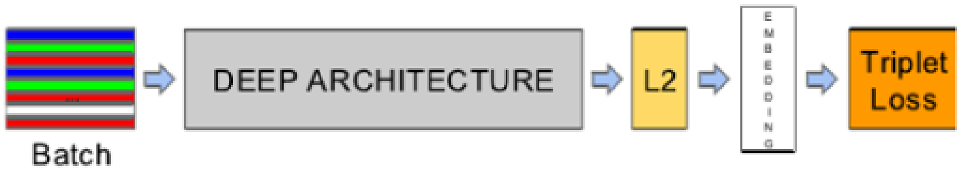


Figure 3-6 The network structure used Triptlet Loss

The triple consists of Anchor (A), Negative (N), Positive (P), any image can be used as a base point (A), and the image belonging to the same person is its P, and the image belonging to the different person is its N.

The learning objectives of Triplet Loss can be visualized as follows:

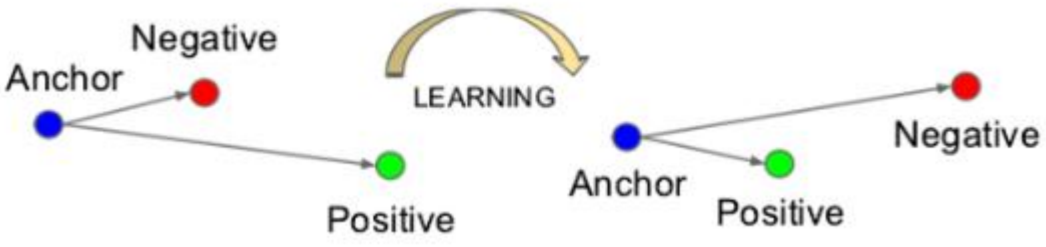


Figure 3.7 The visualization of learning objectives of Triplet Loss

Before learning the network, the European distance between A and P may be very large, and that between A and N may be very small as shown on the left. Then during the process of learning, the distance between A and P will gradually decrease, while the distance between A and N will gradually increase. In other words, the network will directly learn the separability between features: the distance between features of the same classes should be as small as possible, while the distance between features of different classes should be as large as possible. This means that by learning, the distance between classes is greater than that within classes.

This work uses the batch hard Triplet loss [refer to In Defense of the Triplet Loss for Person Re-Identification] Batch hard Triplet Loss, "hard" means that when forming the triplets, it select the hardest negative and the hardest positive in the batch.

The formula is defined as:

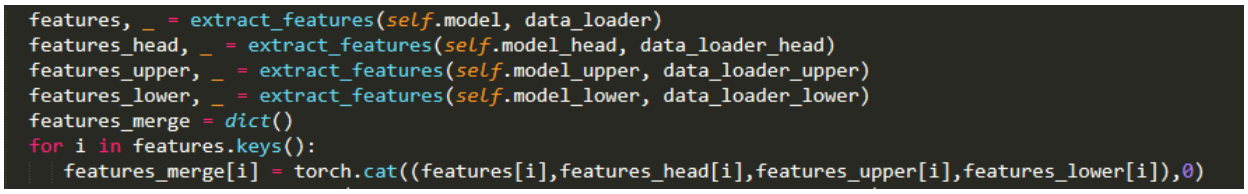
$$L_{BH}(\theta; X) = \sum_{l=1}^{P}\sum_{\alpha=1}^{K}[m + \max_{p=1...K} D(f_\theta(x_\alpha^i), f_\theta(x_p^i) - \min_{\substack{j=1...P\\ n=1...K\\ j\neq i}} D(f_\theta(x_\alpha^i), f_\theta(x_n^j))] \qquad (3\text{-}6)$$

And it will get PK terms for the loss.

3.2.5 Feature matching

➢ **Feature fusion**

There are four branches totally. For each branch, the output is the feature which is a tensor. I concatenate four tensors together and then do the feature matching.



```
features, _ = extract_features(self.model, data_loader)
features_head, _ = extract_features(self.model_head, data_loader_head)
features_upper, _ = extract_features(self.model_upper, data_loader_upper)
features_lower, _ = extract_features(self.model_lower, data_loader_lower)
features_merge = dict()
for i in features.keys():
    features_merge[i] = torch.cat((features[i],features_head[i],features_upper[i],features_lower[i]),0)
```

Figure 3-8 The code example of feature fusion

➢ **Euclidean Distance**

Euclidean distance is the easiest distance measure to understand intuitively.

As shown in figure 3-9, it is the Euclidean distance between point A (x_1, y_1) and point B (x_2, y_2) on two-dimensional plane:

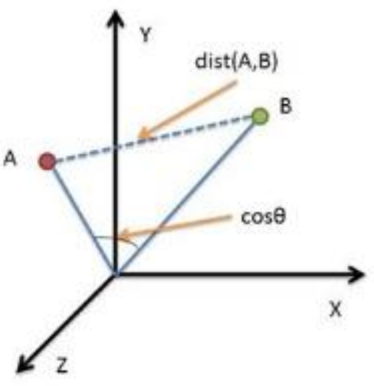


Figure 3.9 The Euclidean distance between point A and point B on two-dimensional plane

And the Euclidean distance between two n-dimensional vectors point $A(x_1, y_1)$ and point $B(x_2, y_2)$ on two-dimensional plane is:

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.7}$$

So we can derive the Euclidean distance between two n-dimensional vectors $A(x_{11}, x_{12}, ..., x_{1n})$ and $B(y_{11}, y_{12}, ..., y_{1n})$ on n-dimensional plane:

$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2} \tag{3-8}$$

3.3 Summery

This chapter introduces this work's model framework, including key point detection, parts division, feature matching. The next chapter will introduce the experimental process and the results of the model.

4 Experimental results and analysis

4.1 Datasets

In order to better illustrate the effectiveness of the proposed algorithm, I train and test the algorithm on two datasets. And compare them with other existing methods. This part mainly introduces the main ReID datasets that will be involved in the experiment.

➢ **Market-1501 dataset**

Market-1501 dataset is one of the largest person ReID datasets at present. It was collected in front of a supermarket in Tsinghua University in summer. It was constructed and released in 2015. It consists of 1501 pedestrians and 32668 pedestrian rectangular frames captured by six cameras, including five high-resolution cameras and one low-resolution camera. Each pedestrian is captured by at least two cameras and may have multiple images in one camera. There are 751 people in the training set, including 12,936 images, with an average of 17.2 training data per person. There are 750 people in the test set, including 19,732 images, with an average of 26.3 test data per person. The pedestrian detection rectangular frame of 3368 query images was drawn manually, while the pedestrian detection rectangular frame in gallery was detected by DPM detector. The following figure 4-1 is the directory structure of the dataset:

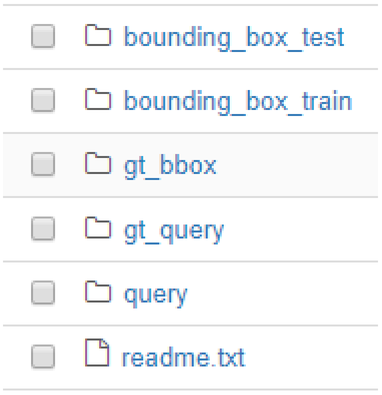

bounding_box_test
bounding_box_train
gt_bbox
gt_query
query
readme.txt

Figure 4-1 The directory structure of the raw Market1501 dataset

1) "bounding_box_test" - 750 people for testing set, containing 19,732 images

2) "bounding_box_train" - 751 people for training set, including 12,936 images

3) "Query" - Random selection of an image for 750 people in each camera as query, so a person's query has a maximum number of 6, including 3,368 images.

4) "gt_query" --.m format. This file contains the image index of the good and junk images. It is used in performance evaluation.

5) "gt_bbox" – there are 25259 hand-labeled bounding boxes, which is consistent with the pedestrian test set and training set to distinguish good, junk and distractors.

➢ **CUHK03 Dataset**

CUHK03 is the first large-scale person ReID dataset for deep learning. The images of the dataset are collected on the campus of the Chinese University of Hong Kong (CUHK). The data is stored in the format of "cuhk-03.mat" MAT file, which contains 1467 different pedestrians and is collected by 5 pairs of cameras. Each individual image is captured from two different cameras. There are an average of 4.8 images per camera. Sample's labeling in the dataset is detected by manual or existing pedestrian detectors automatically.

"cuhk-03.mat" MAT file":

Detected: 5x1 cell, the bounding boxes which are detected by pedestrian detector

Labeled: 5x1 cell, the bounding boxes which are labeled by human

Testsets: 20x1 cell test protocol

4.2 Evaluation method

4.2.1 CMC curve

The accumulative matching characteristic (CMC) curve reflects the retrieval accuracy and is generally replaced by Rank-1, Rank-5, Rank-20 score. Rank-1 recognition rate means that after matching according to a certain similarity matching rule, the ratio of the number of correct labels for the first time to the total number of test samples. Rank-5 recognition rate means that there are five opportunities to judge whether there is a correct match (selecting the five items with the greatest matching degree).

CMC curve is the most popular performance evaluation methods in the field of pedestrian recognition. Considering a simple single-gallery-shot case, each gallery ID) in dataset has only one instance. For each re-identify, the algorithm ranks the distance between the query and al gallery samples from small to large. CMC Top-k accuracy is calculated as follows:

$$Acc_k \begin{cases} 1 & if\ top-k\ ranked\ gallery\ samples\ contain\ the\ query\ identity \\ 0 & otherwise \end{cases} \tag{4-1}$$

This is a shifted step function. The final CMC curve is obtained by averaging the shifted step functions of all queries.

Although CMC is well defined in the single-gallery-shot case, it is not clearly defined in the multi-gallery-shot case because there may be multiple instances for each gallery identity. For example, the methods for computing CMC curves and CMC Top-k accuracy on CUHK-03 and Market-1501 datasets are quite different.

CUHK03: In this dataset, query and gallery images come from different camera perspectives. For each query, sample an instance from each gallery identity randomly, and then the CMC curve is calculated in the single-gallery-shot method. Random sampling is repeated N times, and eventually output the CMC curve.

Market-1501: Query and gallery sets may come from the same camera perspective, but for each query identity, its gallery samples from the same camera will be excluded. For each gallery identity, they do not randomly sample only one instance. This means that query will always match the "simplest" positive sample in gallery while calculating CMC, instead of paying attention to other more difficult positive samples.

4.2.2 mAP

Because in this work, ReID is muti-shot, that is to say, there is more than one matching pedestrian images in the gallery set. It is biased to use only a few topk as the measurement criteria. All the ranking results should be taken into account. Therefore, mAP is an important evaluation index of image retrieval problem. It reflects the average performance of the algorithm on the whole test set.

For the ReID task, we often care about two issues:

How many proportions of the query results are the same IDs?

How many proportions of images with the same ID have been retrieved?

These two questions correspond to “precision” and “recall” respectively.

Precision is the proportion of the images with the same ID as query to the query results. The calculation formula is as follows:

$$precision = \frac{|\{images\ with\ the\ same\ ID\ as\ query\} \cap \{query\ results\}|}{|\{query\ results\}|} \tag{4-2}$$

Recall is the proportion of the number of images with the same ID as query in the query results to the total number. The calculation formula is as follows:

$$recall = \frac{|\{images\ with\ the\ same\ ID\ as\ query\} \cap \{query\ results\}|}{|\{images\ with\ the\ same\ ID\ as\ query\}|} \tag{4-3}$$

Average precision:

$$AveP = \sum_{k=1}^{n} P(k)\Delta r(k) \tag{4-4}$$

$$\Delta r(k) = \begin{cases} \frac{1}{N} & ID(result_k) = ID(query) \\ 0 & ID(query) \neq ID(result_k) \end{cases} \tag{4-5}$$

Finally, calculate the average of AP values for each query yields mAP.

$$mAP = \frac{\sum_{i=1}^{N_q} AveP_i}{N_q} \tag{4-6}$$

The following figure is an example:

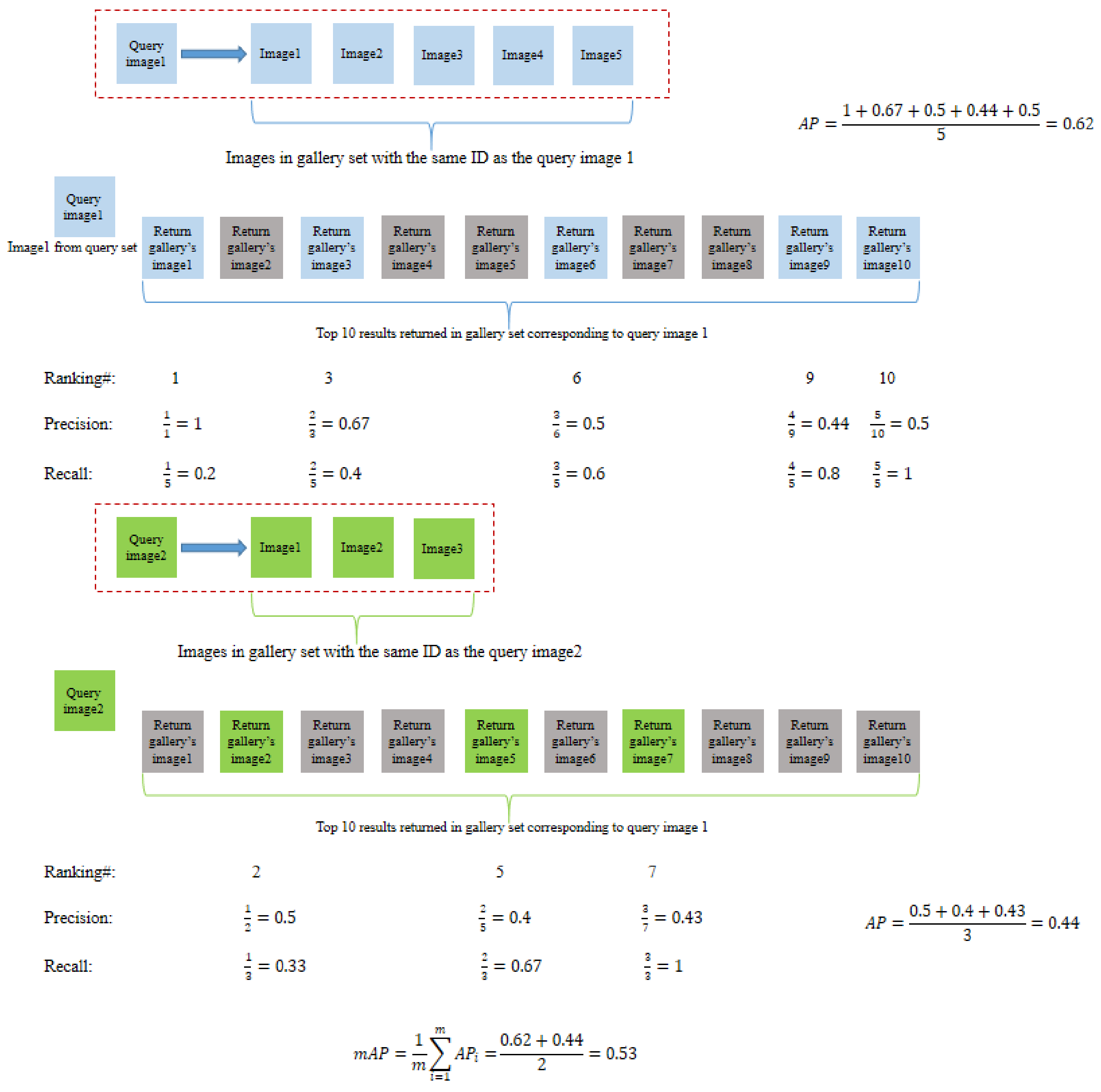

Query image1
Image1
Image2
Image3
Image4
Image5
Images in gallery set with the same ID as the query image 1
$AP = \frac{1+0.67+0.5+0.44+0.5}{5} = 0.62$
Query image1
Image1 from query set
Return gallery's image1
Return gallery's image2
Return gallery's image3
Return gallery's image4
Return gallery's image5
Return gallery's image6
Return gallery's image7
Return gallery's image8
Return gallery's image9
Return gallery's image10
Top 10 results returned in gallery set corresponding to query image 1
Ranking#: 1 3 6 9 10
Precision: $\frac{1}{1}=1$ $\frac{2}{3}=0.67$ $\frac{3}{6}=0.5$ $\frac{4}{9}=0.44$ $\frac{5}{10}=0.5$
Recall: $\frac{1}{5}=0.2$ $\frac{2}{5}=0.4$ $\frac{3}{5}=0.6$ $\frac{4}{5}=0.8$ $\frac{5}{5}=1$
Query image2
Image1
Image2
Image3
Images in gallery set with the same ID as the query image2
Query image2
Return gallery's image1
Return gallery's image2
Return gallery's image3
Return gallery's image4
Return gallery's image5
Return gallery's image6
Return gallery's image7
Return gallery's image8
Return gallery's image9
Return gallery's image10
Top 10 results returned in gallery set corresponding to query image 1
Ranking#: 2 5 7
Precision: $\frac{1}{2}=0.5$ $\frac{2}{5}=0.4$ $\frac{3}{7}=0.43$
Recall: $\frac{1}{3}=0.33$ $\frac{2}{3}=0.67$ $\frac{3}{3}=1$
$AP = \frac{0.5+0.4+0.43}{3} = 0.44$
$mAP = \frac{1}{m}\sum_{i=1}^{m} AP_i = \frac{0.62+0.44}{2} = 0.53$

Figure 4-2 The example of calculating mAP for two query images

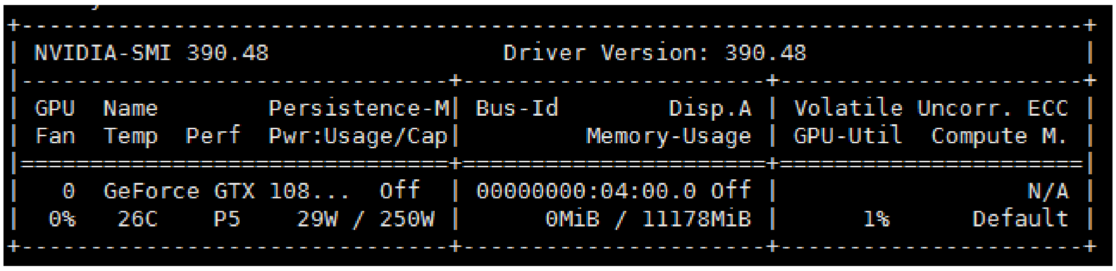
4.3 Experimental environment and conditions

I do the experiment on the server of Information and Countermeasure Laboratory.

On the server physical machine, the container belonging to each user is established. When the container is built, it is configured with the container driven by graphics card as the template. The static IP address is set for each container, and the port mapping is configured on the physical machine, so that the port 30XX of the physical machine corresponds to the port 22 of the container, thus SSH forwarding can be realized. On the personal computer, use ssh username@IP -p 30XX to remote access the container.

Configured graphics card driver is NVIDIA-Linux-x86_64-390.48 in container

GPU Type: GTX 1080 Ti



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:04:00.0 Off |                  N/A |
|  0%   26C    P5    29W / 250W |      0MiB / 11178MiB |      1%      Default |
+-------------------------------+----------------------+----------------------+
```

Figure 4-3 GPU status monitoring

CUDA (Computer Unified Device Architecture) is a computing platform launched by NVIDIA, a graphics card manufacturer. CUDA is a general parallel computing architecture developed by NVIDIA, which enables GPU to solve complex computing problems.

NVIDIA cuDNN is a GPU acceleration library for deep neural networks. It emphasizes performance, ease of use and low memory overhead. NVIDIA cuDNN can be integrated into higher-level machine learning frameworks, such as Google's TensorFlow and the popular Caffe by the University of California, Berkeley. Simple plug-in design allows developers to concentrate on designing and implementing neural network models rather than simply tuning performance, while also achieving high performance modern parallel computing on GPU.

I first installed CUDA Version 9.1, but had some problems with running tensorflow, and then reconfigured the 9.0 environment. So I use two containers.

For container1, the environment configurations are as follows:

Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-124-generic x86_64); Parallel Computing Library CUDA Version 9.1.85; CUDNN Version 7.1.3; Python: 3.6.2; PyTorch: 1.0.1.post2

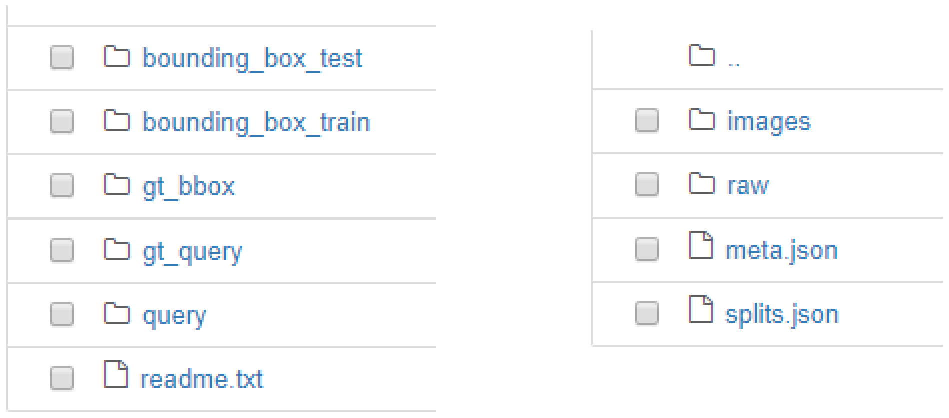
For container2, the environment configurations are as follows: Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-124-generic x86_64); CUDA Version 9.0.176; CUDNN Version 7.0.5; Python: 3.6.2; TensorFlow-gpu: 1.7.0

4.4 Experiment process

4.4.1 Training process

Take the Market1501 dataset for example

➢ **Preprocessing the data**

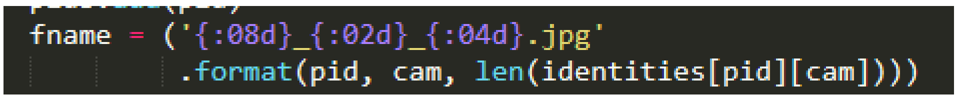


bounding_box_test
bounding_box_train
gt_bbox
gt_query
query
readme.txt

..
images
raw
meta.json
splits.json

Figure 4-4 The left picture is the raw Market1501 dataset and the right picture is the processed dataset.

As the above figure 4-4 show, on the left is the raw Market1501 dataset. On the right is the processed dataset. I put all the images in the "images" folder and use json file to classifier the images. And I rename the images to the format as figure 4-5.



```
fname = ('{:08d}_{:02d}_{:04d}.jpg'
         .format(pid, cam, len(identities[pid][cam])))
```

Figure 4-5 The code example of rename the images

pid is the identity of the person. cam is the camera number of the image. len(identities[pid][cam]))) is the number of the images for this person's this camera.

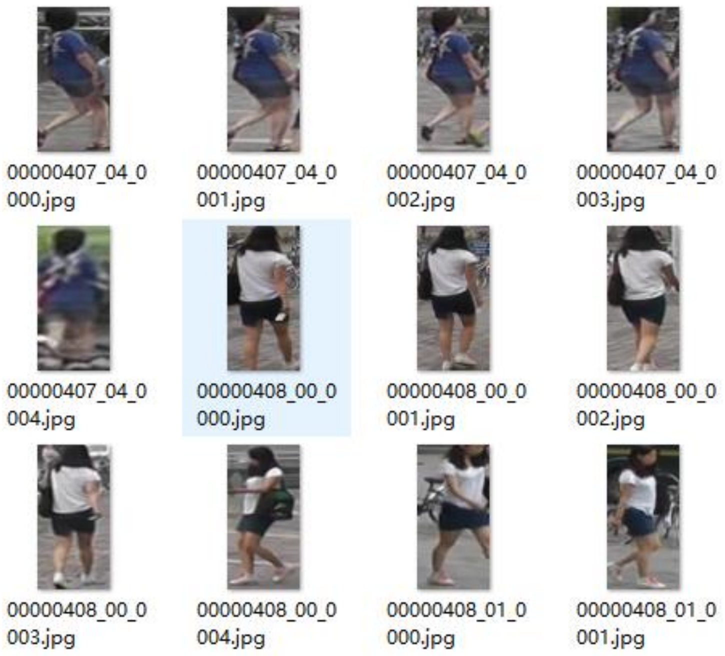
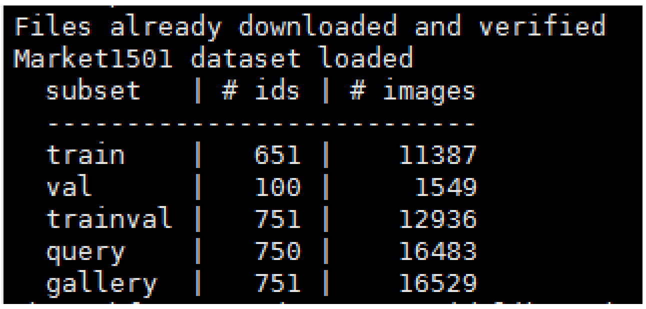
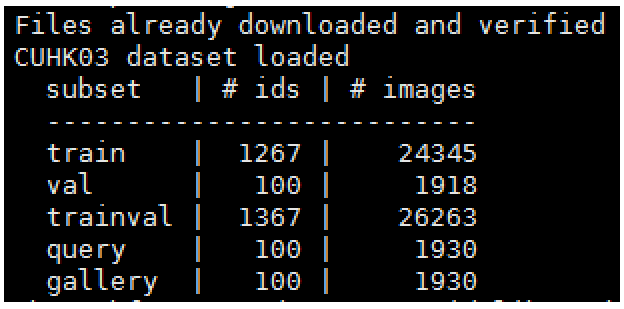


Figure 4-6 The renamed images example

I classify the images in "bounding_box_train" images to the "trainval" class, "bounding_box_test" images to the "gallery" class, "query" images to the "query" class. For the images in "trainval", I divide them into two parts. The top 70% images are for training, and the latter 30% images are for validation.



```
Files already downloaded and verified
Market1501 dataset loaded
  subset   | # ids | # images
  ---------------------------
  train    |   651 |    11387
  val      |   100 |     1549
  trainval |   751 |    12936
  query    |   750 |    16483
  gallery  |   751 |    16529
```



```
Files already downloaded and verified
CUHK03 dataset loaded
  subset   | # ids | # images
  ---------------------------
  train    |  1267 |    24345
  val      |   100 |     1918
  trainval |  1367 |    26263
  query    |   100 |     1930
  gallery  |   100 |     1930
```

Figure 4-7 The left is the classification of Market1501 dataset, the right is the classification of CUHK03 dataset

Then I input the images into the network to detect the 14 key points[2]. And I extract 4 of them: forehead, chin, left hip, right hip and divide the image into three parts: head, upper

[2] https://github.com/eldar/pose-tensorflow

body, lower body as described in 3.2.2. Then put them in three different folders which are the same path with “images” folder and the images in the folds have the same names as the images in “images” folder: images_head, images_lower and images_upper. Final dataset folders are shown in figure 4-8.

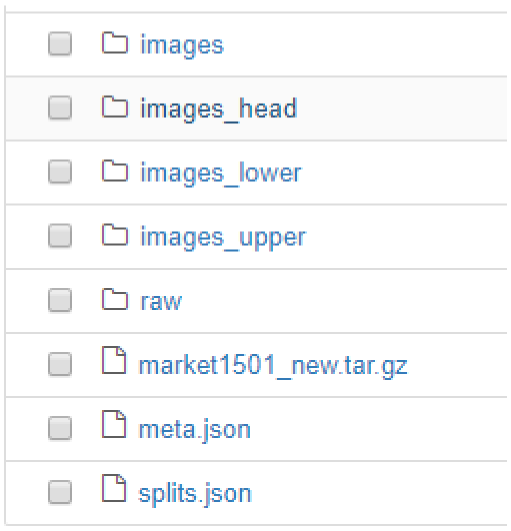

images
images_head
images_lower
images_upper
raw
market1501_new.tar.gz
meta.json
splits.json

Figure 4-8 Final dataset folders

➢ **Training**

The training process of each four sub-network is similar. Firstly, the image passes by ResNet-50 convolution neural network to obtain a 128-dimensional feature (extracting the feature of "avgpool" layer before the last fully connected layer). Secondly the feature passes the identity (ID) classifier. Then calculate the loss function of the ID classification and optimize. I use the Softmax loss function and Triplet loss function About the parameter setting: The batch size is 512, epoch is 40, the initial learning rate is 0.1, the weight decay is 5e-4, momentum is 0.9.

The training time is about 5 hours.

4.4.3 Test process

For the test images input, use the convolution neural network to extract the features. I extract the feature of 128 dimension before the last fully connected layer. And concatenate the four features together to get the final feature of 512 dimension and then calculate the distance. And finally use CMC (Cumulated Matching Characteristics) and mAP as mentioned in 4.2 to evaluate. The experiment result is shown in 4.5.

4.5 Experimental results

4.5.1 Comparison with the baseline model

The baseline model is a branch of ResNet-50 convolutional neural network. The loss function is the pedestrian identity classification based on Softmax Loss. The network structure is as follows:

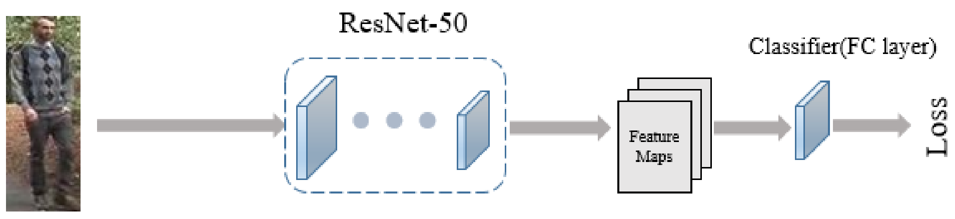


Figure 4-9 The framework of the baseline

I train the baseline network and this work's model based on Softmax Loss function respectively. Following figure 4-10 is the experiment result on the Market1501 dataset.

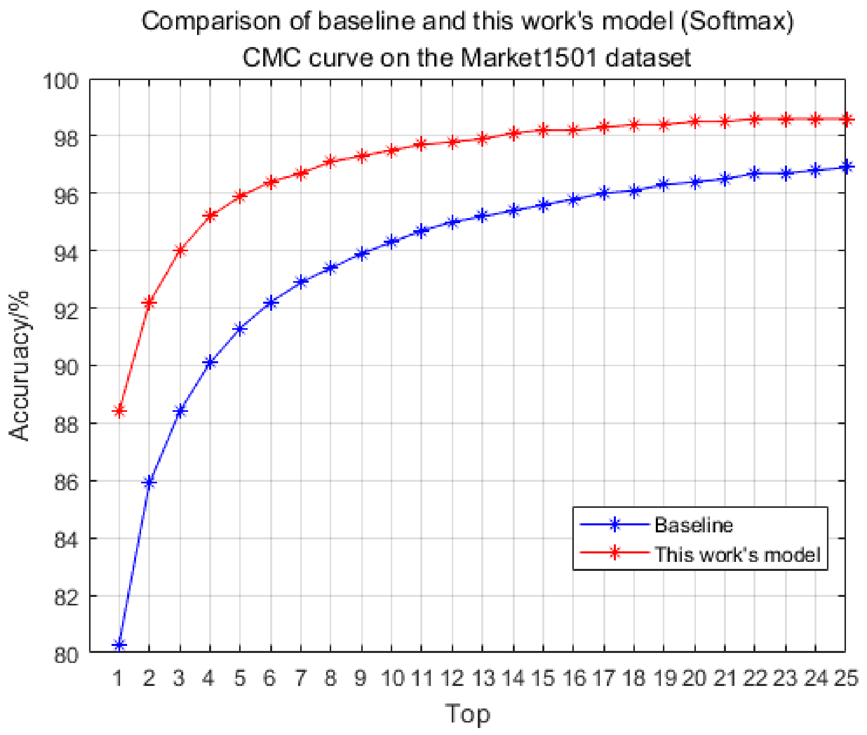


Figure 4.10 CMC curve's comparison of baseline and this work's model on Market1501

Table 4.1 mAP's comparison of baseline and this work's model on Market1501

Model	mAP
Baseline	59.3
This model	69.9

Following figure 4-11 is the experiment result on the CUHK03 dataset.

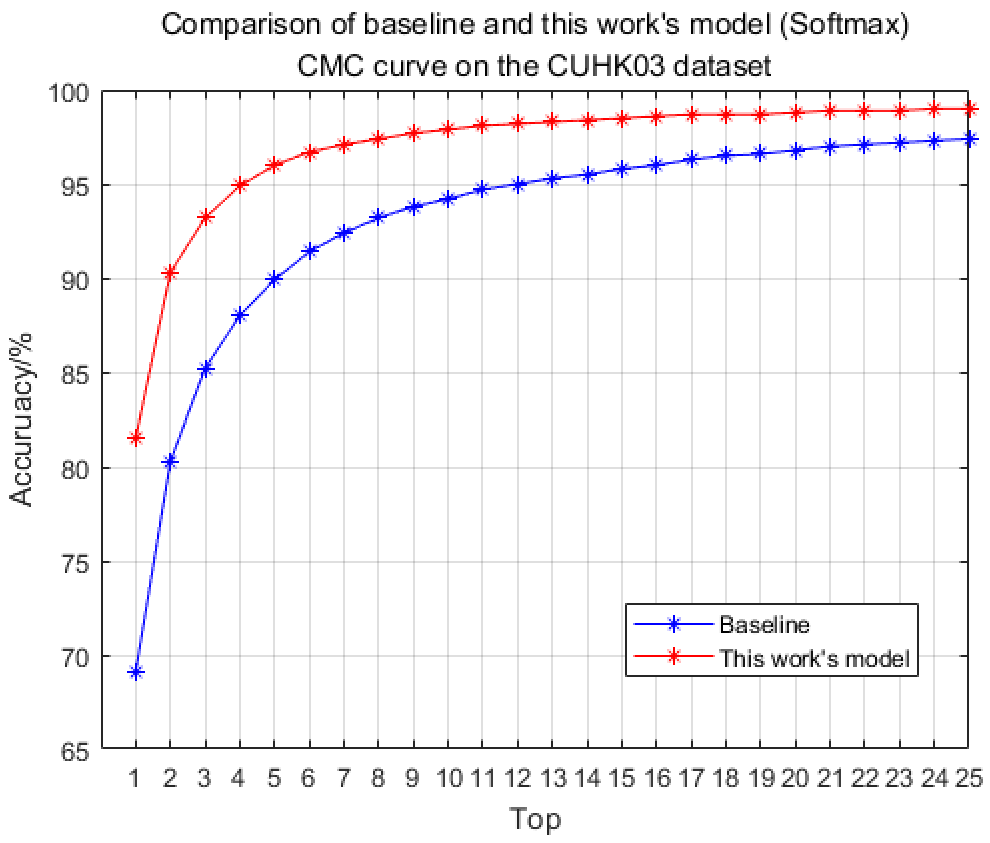


Figure 4-11 Comparison of baseline and this work's model on the CUHK03

Table 4-2 mAP's comparison of baseline and this work's model on CUHK03

Model	mAP
Baseline	63.2
This model	76.0

Based on the baseline model, this model adds three branches of parts. Figures 4-10 and

4-11 show that on Market 1501 dataset and CUHK03 dataset, the accuracy of this model has been greatly improved compared to the baseline. For the Market1501 data set, the accuracy of Rank1 was improved by 8 percentage points and that of mAP by 10 percentage points. For CUHK03 data set, Rank1's accuracy increased by 12 percentage points and mAP's by 12.8 percentage points.

4.5.2 Comparison of Softmax Loss and Triplet Loss

I train the model with Softmax and Triplet loss respectively. Figure 4-12 and figure 4-13 are experiment results.

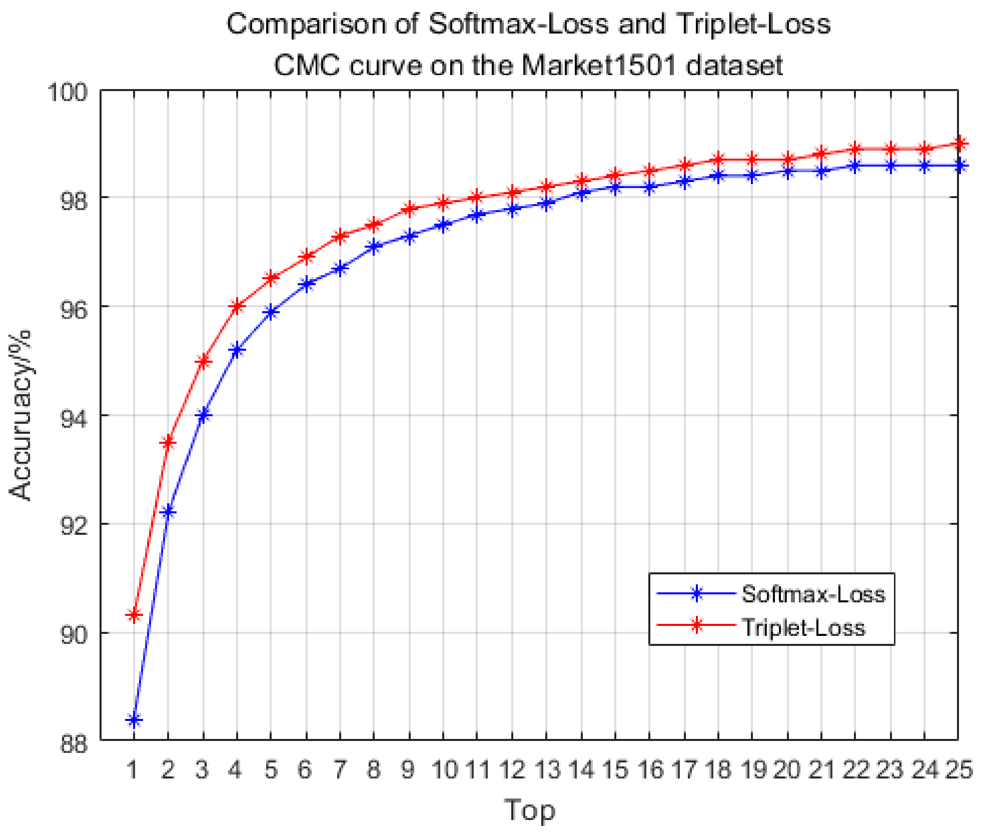


Figure 4-12 CMC curve’s comparison of Softmax and Triplet on Market1501 of this work’s model

Table 4-3 mAP’s comparison of Softmax loss and Triplet loss of this work’s model on Market1501

Loss function	mAP
Softmax Loss	69.9
Triplet Loss	76.8

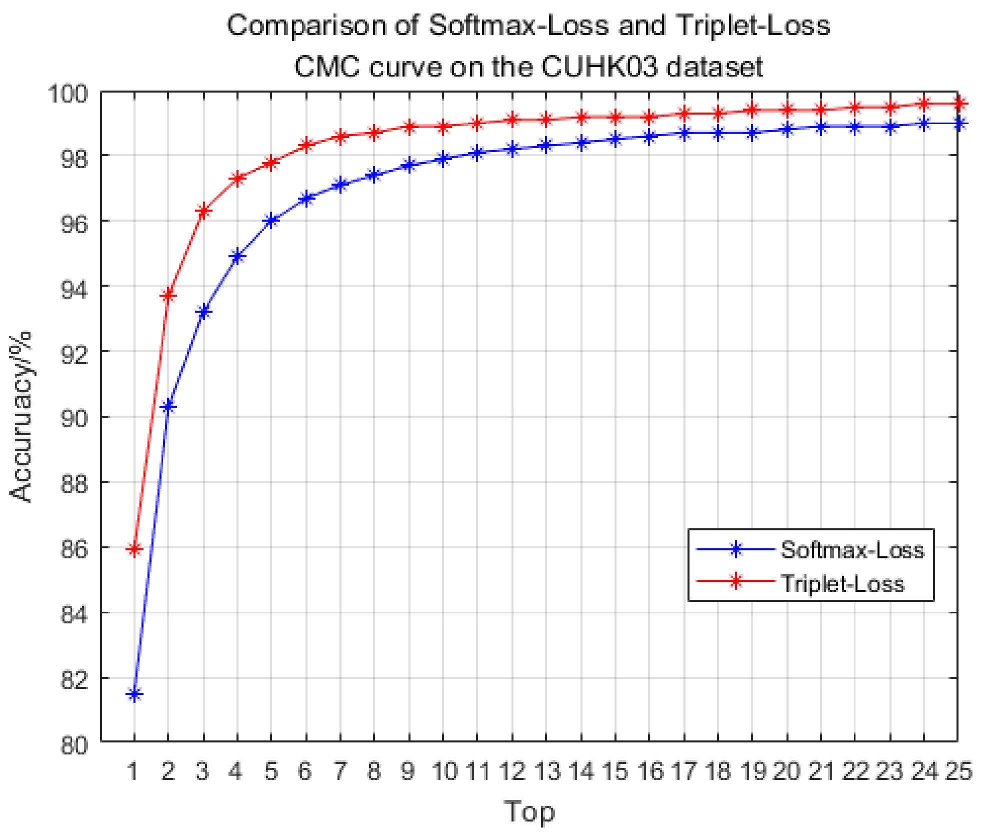


Figure 4-13 CMC curve's comparison of Softmax and Triplet on CUHK03 of this work's model

Table 4-4 mAP's comparison of Softmax loss and Triplet loss of this work's model on CUHK03

Loss function	mAP
Softmax Loss	76.0
Triplet Loss	82.9

This experiment compares the effects of two loss functions, Softmax and Triplet, on this work's model. Looking at Figures 4-12 and 4-13, I find that Triplet Loss improve the accuracy on two datasets, Market 1501 and CUHK03. For the Market1501 dataset, the accuracy of Rank1 was improved by 2 percentage points and that of mAP by 7 percentage points on Triplet Loss. For CUHK03 data set, Rank1's accuracy increased by 12 percentage points and mAP's by 5 percentage points on Triplet Loss.

4.5.3 Comparison with the existing methods

Table 4-5 Comparison on Market1501 dataset

Methods	Rank-1	Rank-5	Rank-10	mAP
HydraPlus(ICCV2017)[38]	76.9	91.3	94.5	-
PAR(ICCV2017)[39]	81.0	92.0	94.7	63.4
SVDNet(ICCV2017)[40]	82.3	92.3	95.2	62.1
PDC(ICCV2017)[41]	84.4	92.7	94.9	63.4
GLAD(ACM MM2017)[37]	89.9	-	-	73.9
AACN(CVPR2018)[42]	85.9	-	-	66.9
MLFN(CVPR2018)[43]	90.0	-	-	74.3
Baseline	80.3	91.3	94.3	59.3
This work's model	90.3	96.5	97.9	76.8

Table 4.6 Comparison on CUHK03 dataset

Methods	Rank-1	Rank-5	Rank-10	mAP
In[44]	75.5	95.2	99.2	-
Deep[45]	84.1	-	-	-
Unlabeled(ICCV2017)[46]	84.6	97.6	98.9	-
A[47]	83.4	97.1	98.7	-
DGD(CVPR2016)[48]	72.6	91.6	95.2	-
Baseline	69.1	89.9	86.2	63.2
This work's model	85.9	97.8	98.9	82.9

Table 4-5 and table 4-6 compare the performance of this model with that of published models on Market 1501 and CUHK03 datasets, respectively. On Market 1501 dataset, the results of this model are 90.3% for Rank1, 96.5% for Rank5, 97.9% for Rank10 and 76.8% for mAP, which are better than those of other models listed in the table. On CUHK03 dataset, this work's model achieves 85.9% Rank1, 97.8% Rank5, 98.9% Rank10 and 82.9% mAP, which is also better than other models listed in the table.4.6

4.6 Summary

This chapter makes three comparisons with the control variable method. Firtly, compare the model with the baseline model, and concludes that the performance of this model is better. By comparing the triplet loss with the Softmax loss function, it is concluded that the performance of triplet loss is better, so triplet loss is used in this work’s model. Finally, compared with other published models, the performance of this model is the best among them.

5 Conclusion

5.1 Full text summary

In recent years, with the improvement of people's public safety awareness, ReID research has attracted more and more attention. This work designs an end-to-end Person ReID Network structure based on ResNet-50 model, and achieves good results. The main work and research results are as follows:

1. At the beginning of this work, the background, definition, development and research status of ReID are introduced in detail. Two key technologies in ReID are pedestrian feature extraction and distance measurement learning. Traditional methods focus on designing better artificial features and learning better distance measurement. Recent studies focus on deep learning.

2. A ReID method based on pedestrian parts is proposed. The existing DeeperCut network is used to detect the key points of pedestrians and segment the pedestrians. The segmented images are input into four parallel ResNet-50 networks for training, and the dimension of feature extraction is increased to improve the accuracy of pedestrian recognition.

3. This work analyses in detail the landmark model structure in the research of convolutional neural network, and the advantages of each model structure. The last layer of convolutional neural network is usually the loss layer. It is an important aspect of model design to select the appropriate loss function. In this work, two loss functions, Softmax and Tripletloss, are used to train to find a more suitable loss function for pedestrian recognition.

5.2 Outlook

This work focuses on the issue of person identity re-identification, but there are still many areas worthy of in-depth study, mainly the following points:

1. This work only uses the features of the parts, and the pedestrian attributes are also used in the literature [49]. It can combine the attributes with the features of the parts, which will increase the feature dimension and improve the accuracy of ReID.

2. In recent years, the resolution of surveillance cameras is getting higher and higher.

Some cameras can even photograph the faces of pedestrians clearly. Face information is an important factor to distinguish pedestrian identity. How to combine face recognition with ReID is a topic worthy of study.

3. In the actual video surveillance system, pedestrians have time domain information. It is necessary to optimize the proposed model by utilizing pedestrian gait, posture and some behavioral characteristics, and extend the image-based pedestrian identity re-identification problem to video-based pedestrian identity re-identification problem.

4. Pedestrian detection and identity re-identification are generally two independent issues. It will bring great convenience to intelligent surveillance technology to merge the two to form an integrated framework of pedestrian detection and recognition, and do pedestrian recognition based on the pedestrian boundary frame detected by the detector.

5. This work only integrates, improves and fine-tunes the existing network, and does not large-scale adjust network. With sufficient time and equipment, more reasonable attempts can be made and new network structures can be proposed.

Reference

[1] Wanchao Yao. Pedestrian Recognition Algorithms Based on Convolutional Neural Network [D]. Zhejiang: Zhejiang University, 2017

[2] Timothy Huang, Stuart Russel. Object identification in a Bayesian context[C]. In IJCAI. 1997, volume 97, 1276-1282

[3] Wojciech Zajdel, Zoran Zivkovic, BJA Krose. Keeping track of humans: Have I seen this person before? [C]. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, 2005, 2081-2086

[4] Niloofar Gheissari, Thomas B Sebastian, Richard Hartley. Person re-identification using spatioterm-poral appearance[C]. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR2006).IEEE, 2006, volume2, 1528-1535

[5] Michela Farenzena, Loris Bazzani, Alessandro Perina, Vittoria Murino, Marco Cristani. Person re-identification by symmetry-driven accumulation of local features[C]. In Computer Vision and Pattern Recognition(CVPR), 2010 IEEE Conference on. IEEE, 2010, 2360-2367

[6] Loris Bazzani, Marco Cristani, Alessandro Perina, Michela Farenzena, Vittorio Murino. Multiple-shot person re-identification by hpe signature[C]. In Pattern Recognition(ICPR), 2010 20th International Conference on. IEEE, 2010, 1413-1416

[7] Dong Yi, Zhen Lei, Shengcai Liao, Stan Z Li, et al. Deep metric learning for person re-identification.[C]. In ICPR. 2014, volume 2014, 34-39

[8] Yuanlu Xu, Bingpeng Ma, Rui Huang, Liang Lin. Person search in a scene by jointly modeling people commonness and person uniqueness[C]. In Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014, 937-940

[9] Chen D, Yuan Z, Hua G et al. Similarity learning on an explicit polynomial kernel feature map for person re-identification[C]. In CVPR, 2015:1565-1573

[10] Liao S, Hu Y, Zhu X et al. Person re-identification by local maximal occurrence representation and metric learning[C]. In CVPR, 2015:2197-2206

[11] Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, Gang Wang. A siamese long short-term memory architecture for human re-identification[C]//European Conference on Computer Vision. Springer, 2016:135–153

[12] Haiyu Zhao, Maoqing Tian, Shuyang Sun, Jing Shao, Junjie Yan, Shuai Yi, Xiaogang Wang, Xiaoou Tang. Spindle net: Person re-identification with human body region guided feature decomposition and fusion[C]. CVPR, 2017

[13] Longhui Wei, Shiliang Zhang, Hantao Yao, Wen Gao, Qi Tian. Glad: Global-local-alignment descriptor for pedestrian retrieval[J]. arXiv preprint arXiv:1709.04329, 2017

[14] Florian Schroff, Dmitry Kalenichenko, James Philbin. Facenet: A unified embedding for face recognition and clustering[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.2015:815-823

[15] Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, Shuicheng Yan. End-to-end comparative attention networks for person re-identification[J]. IEEE Transactions on Image Processing, 2017

[16] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, Nanning Zheng. Person re-identification by

multichannel parts-based cnn with improved triplet loss function[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016:1335-1344

[17] Rahul Rama Varior, Mrinal Haloi, Gang Wang. Gated siamese convolutional neural network architecture for human re-identification[C]//European Conference on Computer Vision. Springer, 2016:791-808

[18] Alexander Hermans, Lucas Beyer, Bastian Leibe. In defense of the triplet loss for person reidentification[J]. arXiv preprint arXiv:1703.07737, 2017

[19] Xiao Q, Luo H, Zhang C. Margin Sample Mining Loss: A Deep Learning Based Method for Person Re-identification[J]. 2017

[20] Taiqing Wang, Shaogang Gong, Xiatian Zhu, Shengjin Wang. Person re-identification by discriminative selection in video ranking[J]. IEEE transactions on pattern analysis and machine intelligence, 2016.38(12):2501–2514

[21] Dongyu Zhang, Wenxi Wu, Hui Cheng, Ruimao Zhang, Zhenjiang Dong, Zhaoquan Cai. Image-to-video person re-identification with temporally memorized similarity learning[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2017

[22] Jinjie You, Ancong Wu, Xiang Li, Wei-Shi Zheng. Top-push video-based person reidentification[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.2016:1345–1353

[23] Xiaolong Ma, Xiatian Zhu, Shaogang Gong, Xudong Xie, Jianming Hu, Kin-Man Lam, Yisheng Zhong. Person re-identification by unsupervised video matching[J]. Pattern Recognition, 2017. 65:197–210

[24] Niall McLaughlin, Jesus Martinez del Rincon, Paul Miller. Recurrent convolutional network for videobased person re-identification[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016:1325–1334

[25] Rui Zhao, Wanli Oyang, Xiaogang Wang. Person re-identification by saliency learning[J]. IEEE transactions on pattern analysis and machine intelligence, 2017. 39(2):356–370

[26] Hao Liu, Zequn Jie, Karlekar Jayashree, Meibin Qi, Jianguo Jiang, Shuicheng Yan, Jiashi Feng. Video based person re-identification with accumulative motion context[J]. arXiv preprint arXiv:1701.00193,2017

[27] Zheng Z, Zheng L, Yang Y. Unlabeled samples generated by gan improve the person re-identification baseline in vitro[J]. arXiv preprint arXiv:1701.07717, 2017

[28] Zhong Z, Zheng L, Zheng Z, et al. Camera Style Adaptation for Person Re-identification [J]. In CVPR, 2018: 5157-5166

[29] Song G, Leng B, Liu Y, et al. Region-based Quality Estimation Network for Large-scale Person Re-identification[J]. arXiv preprint arXiv:1711.08766, 2017

[30] Alex Krizhevsky, Nya Sutskever, Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks[C]. In Advances in neural information processing systems. 2012, 1097-1105

[31] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556,2014

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going deeper with convolutions[C]. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recoginition. 2015, 1-9
[33] Zheng L, Yang Y, Hauptmann A G. Person re-identification: Past, present and future[Z]. arXiv:1610.02984,2016
[34] Insafutdinov E, Pishchulin L, Andres B et al. DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model[C]. In ECCV, 2016:34-50
[35] Insafutdinov E, Andriluka M, Pishchulin L et al. Articulated Muti-person Tracking in the Wild[C]. In CVPR, 2017:6457-6465
[36] Zhao H, Tian M, Sun S et al. Spindle net: Person re-identification with human body region guided feature decomposition and fusion[C]. In CVPR, 2017:1077-1085
[37] Wei L, Zhang S, Yao H et al. GLAD: Global-local-alignment descriptor for pedestrian retrieval[C]. In proceedings of ACM on Multimedia Conference, 2017:420-428
[38] X. Liu, H. Zhao, M. Tian, L. Sheng, J. Shao, S. Yi, J. Yan, and X. Wang. Hydraplus-net: Attentive deep features for pedestrian analysis[C]. In ICCV, 2017
[39] Zhao L, Li X, Zhuang Y et al. Deeply-Learned Part-Aligned Representations for Person Re-Identification[C]. In ICCV, 2017:3219-3228
[40] Sun Y, Zheng L, Deng W et al. SVDNet for Pedestrian Retrieval[C]. In ICCV, 2017:3820-3828
[41] Su C, Li J, Zhang S et al. Pose-Driven Deep Convolutional Model for Person Re-identification[C]. In ICCV, 2018:3980-3989
[42] Xu J, Zhao R, Zhu F et al. Attention-Aware Compositional Network for Person Re-identification[Z]. arXiv preprint arXiv:1805.03344, 2018
[43] Chang X, Hospedales T M, Xiang T. Multi-level factorisation net for person re-identification[Z]. arXiv preprint arXiv:1803.09132, 2018
[44] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification[Z]. arXiv preprint arXiv:1703.07737, 2017
[45] M. Geng, Y. Wang, T. Xiang, and Y. Tian. Deep transfer learning for person re-identification[Z]. arXiv preprint arXiv:1611.05244, 2016
[46] Z. Zheng, L. Zheng, and Y. Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro[C]. In The IEEE International Conference on Computer Vision (ICCV), Oct 2017
[47] Z. Zheng, L. Zheng, and Y. Yang. A discriminatively learned cnn embedding for person re-identification[Z]. arXiv preprint arXiv:1611.05666, 2016
[48] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. 2016. Learning deep feature representations with domain guided dropout for person re-identification[Z]. arXiv preprint arXiv:1604.07528, 2016
[49] Lin Y, Zheng L, Zheng Z et al. Improving person re-identification by attribute and identity learning[Z]. arXiv preprint arXiv:1703.07220, 2017