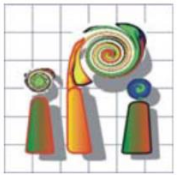
Leibniz University Hannover

Institute of Photogrammetry and Geoinformation



Master thesis

Investigations on Appearance-Based Person Re-Identification for Pedestrian Tracking in Image Sequences

Yanting Liu B.Sc.

Matr.-Nr.: 10007908

Supervisors: apl. Prof. Dr. techn. Franz Rottensteiner
M.Sc. Uyen Nguyen

Hannover, November 2018

Declaration of Authorship

I declare that this thesis and the work presented in it are my own. I confirm that:

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

1 Abstract

With the development of technology, pedestrian tracking has become increasingly popular in the community of computer vision because it is widely used in many applications. To track pedestrians, concatenating detections in adjacent frames of an image sequence by determine defections belong to identical persons is a critical task, which could be treated as a special case of person re-identification (Re-Id).

In this master thesis, we propose Siamese Convolutional Neural Networks (SCNNs) to solve the problem of person re-identification (Re-Id) for pedestrian tracking in image sequence. Our networks are designed as end-to-end learning models which can determine two input images whether belong to the same person or not by comparing their visual similarity. We employ ResNet-50 to extract high-level discriminative representation for pedestrian image. For computation of similarity between feature vectors, we investigate 2 different approaches, named *Baseline*1 and *Baseline*2. In the first approach, *Baseline*1 contains 3 networks, in which the problem of person matching is treated as binary classification task and optimized by crossentropy loss function, while *Baseline*2 is trained by contrastive loss function.

We conduct experiments on two tracking datasets KITTI and MOTChanllenge, and one single-shot dataset Market-1501. Our experiments is divided into two steps. Firstly, we design several experiments to explore training approaches related to fine-tuning, data augmentation and hard samples mining. We conduct an experiment to compare the testing results generated by networks with different number of fine-tuned layers on KITTI. Two groups of contrast experiments are designed for verifying the effect of data augmentation on KITTI and Market-1501. Furthermore, we test two parameters which controls the number of mined samples on Market-1501. Experimental results demonstrate that the best training setting are fine-tuning all layers of ResNet-50 with data augmentation and mining 50% hard samples. We apply this configuration to our proposed networks and evaluate their performances on tracking and singe-shot datasets. In binary classification evaluation model, our best networks achieve at average accuracy of 73.6% on KITTI, 82.4% on MOTChallenge and 94.6% on Market-1501. We employ our networks to person retrieval task on Market-1501, the best result at Rank-1 and at Rank-10 are 24.8% and 53.4%. Experimental results demonstrate that our proposed SCNNs performs well on tracking dataset.

Contents

List of Figures

List of Tables

2 Introduction

2.1 Motivation

Pedestrian tracking over consecutive images of sequences is one of the most active topic in the field of computer vision and used in many applications. For example, in visual surveillance area, it is an efficient method to seek a specific target like lost kid, criminal in crowded place. And in autonomous driving, tracking technology can predict whether pedestrians enter a vehicle path by analysing their trajectories [Klinger, 2016].

There are three major steps in pedestrian tracking. The first step is to detect the location of pedestrians at each frame of sequences, the second step is to determine the dectected pedestrians whether belong to the one person, then pedestrian trajectories can be obtained by connecting detections of the same person across time. In this thesis we concentrate on the first two steps, which can be treated as a special case of re-identification (Re-Id). Specifically, Re-Id is a process to determine detections captured at different frames of image sequences whether belongs to the same person or not based on their appearance properties like shape, color, etc..

Different from traditional Re-Id task in computer vision community, in which images are taken from multiple cameras with various poses, person Re-Id in multi-person tracking task has the following characters: (1) Tracking datasets generally are captured by single or stereo wide-angle camera, dectections in image sequences are not only pedestrains but also other categories. (2) Pedestrian pose is arbitrary and changes gradually over time. (3) External factors like occlusion, large difference of image resolution and unstable lighting condition are the special challenges in tracking datasets as well.

The common framework for person Re-Id is Siamese network [Bromley et al., 1993], in which a pair of images go through two sharing weights sub-networks and the output indicates the similarity between those two input images. Practically, the lacking of data is a problem of person Re-Id. Since in most of person Re-Id datasets, the number of pedestrians is large but the image number for each person is not sufficient. To solve this problem, the input of Siamese network is organized as image pair with a binary label which is decided by two pedestrian images represent the same person or not. So the number of samples for training is increased greatly . With

the development of deep leaning in recent years, Convolutional Neural Networks (CNNs) have achived great successes in solving many typical computer vision problems citephe2015delving. Siamese Convoluational Neural Network (SCNN) [Koch et al., 2015] is developed in consideration of the powerful feature extraction capability of CNNs. Basic idea behind SCNN is using CNNs as feature extractor to produce high-level image representation instead of hand-crafted features. Furthermore, SCNN combines the tasks of feature extraction and similarity computation together by jointly optimizing the representations of the input images through minimization of loss function. With the help of CNN-based Siamese architecture, computed similarity between two images can achieve better performance compared with using hand-crafted features [Zheng et al., 2016].

2.2 Task Description

This master thesis focuses on development and exploration of Siamese Convoluational Neural Network (SCNN) for solving the problem of appearance-based person re-identification in image sequences. Our designed SCNN is a supervised end-to-end learning model which can determine two input images whether belong to the same person or not by calculating the degree of similarity in term of discriminative representation.

In the pre-processing step, pedestrian images need to be organized into pairwise form according to their identities because SCNN architecture requires a pair of image as input. Specifically, if two images belong to the same person, a positive label is assigned to the image pair, otherwise image pair is labeled as negative sample. Furthermore, we also employ online image augmentation strategy to increasing training data.

We propose two different loss functions to investigate two different SCNN architectures named $Baseline1$ and $Baseline2$. Basically, for both baselines, Deep Convolutional Neural Network is used as feature extractor to learn high-level hierarchical features of input images. In the first approach, person Re-Id is simply treated as a binary classification problem and optimized by verification loss function. The similarity between input images is computed by softmax function which conducts the output in binary classes. On the other hand, we train SCNN model using contrastive loss function which contains two terms to minimize distances of positive pairs and maximize distance of negative pairs. The output of in this model is a simple distance computed by similarity metric.

In additional, several approaches like transfer learning, hard samples mining will be investigated. The influence of these approaches on the performance of our person Re-Id method are analyzed in term of confusion matrix results. To evaluate our approaches, we conduct experi-

ments on the publicly available tracking datasets including KITTI [Geiger et al., 2013], MOT15 [Leal-Taixé et al., 2015] and MOT16 [Milan et al., 2016]. Additionally, our model will be evaluated in "query-searching" strategy using single-shot dataset Market-1501 [Zheng et al., 2015], which is a general testing strategy in traditional person Re-ID field.

3 Related Works

Person Re-Id is an independent computer vision task separated from multi-camera tracking system in video surveillance [Gheissari et al., 2006]. Generally, Person Re-Id is a model combining image classification task in training and person retrieval task in testing phase. For training, images are classified according to pedestrian IDs, while in testing query images are retrieved in a large gallery by ranking the similarity which is computed based on learned feature embedding. The research of person Re-Id generally is split into feature representation extraction and similarity metric learning. In this chapter, existing methods with respect to these two aspects are reviewed. Furthermore, the person Re-Id methods used for pedestrian association in multi-person tracking task are presented in the section 3.3.

3.1 Feature Extraction

Extracting high quality features by which different pedestrian can be discriminated is a critical task to influence the final result significantly. In early papers, the researcher prefer to use hand-crafted features [Zheng et al., 2016]. Gheissari et al. [2006] attempted to detect invariant signatures of clothing using novel spatial-temporal segmentation algorithm. For getting edge information, Hessian affine invariant operator and HS histograms were employed. After that, by combining the additional RGB information with the obtained edge information, foreground and background can be separated. In [Gray and Tao, 2008], ensemble of localized features (ELF) schema was designed. Colour histograms in the RGB, HS and YCbCr colour space 8 channels and total 21 texture histograms were computed and fed into discriminative recognition model to person Re-Id. Zhao et al. [2013] proposed 128-dim Scale-Invariant Feature Transform (SIFT) [Lowe, 2004] descriptor to extract feature, which a dense grid of image are sampled. Varior et al. [2016b] employed the local maximal occurrence (LOMO) descriptor, which contains colour histogram and SILTP histogram, and the Colour Names (CN) features. One problem of hand-crafted features is that the domain knowledge about training data is required in advance. The final result heavily depends on selected features.

Recent researches on person Re-Id focus on extracting more robust features using deep learning.

Most of CNN-based Re-Id models are designed in Siamese architecture, in which more training data can be generated through organization of image pairs which are assigned with positive or negative label based on person identity. A novel filter paring neural network (FPNN) based on Siamese architecture was created by Li et al. [2014], which is the first work using deep learning for person Re-Id. Different from hand-crafted features mentioned above, much better feature vectors of image are extracted by learning. Considering of the geometric constraint, the images are divided into several horizontal stripes. Then, these image patches are matched by a patch matching layer for encoding the spatial patterns among the different features. In Yi et al. [2014], input image was divided into three patches along with the horizontal direction with respective to head, body and legs. Each pair of image patches goes through two sub-networks, which include two convolutional layers and a full connected layer. The final similarity is computed by summing the similarity of three image patches pairs using cosine distance metric. In [Ahmed et al., 2015], a deep convolutional architecture was employed for person Re-Id with binary verification loss function. The difference among features from two images were computed in a cross-input neighborhood difference layer.Addtionally, a patch matching layer is added to match the responses of local features in horizontal stripes with a small window for generating high-level local difference representation. [Varior et al., 2016b] introduced LSTMs in Siamese architecture to integrate contextual information for improving discriminative capabilities of local patterns. Specifically, the input image is divided into horizontal stripes which can be treated as spatial sequences like temporal sequences in the case of natural language problem. Varior et al. [2016a] proposed a SCNN model, in which the first four convolutional layers are used to extract global features and a gating function is added in last three convolutional layers to strengthen local features.

With the development of very deep Convolutional Neural Network, the new trend for extracting features is to use deep CNN. Comapred to the above CNN-based feature extractors, deep CNN has more complicated architecture and can extract higher level features for images. For saving training time, this method is implemented generally relying on pre-trained model. Hermans et al. [2017] used the ResNet-50 architecture with leaky ReLU nonlinearitites and the weights provided by He et al. [2016] to generate representation of person. Liu et al. [2017] adopted Alexnet [Krizhevsky et al., 2012] and VGG [Simonyan and Zisserman, 2014] for global discriminative feature learning. The last three fully connected layers of pre-trained models are replaced with comparative attention model, which aims at generating additionally regional features. The whole network is trained by multi-task loss function including triplet loss [Cheng et al., 2016] and identification loss. Benefit from very deep CNN with pre-trianed model, the more abstract and

higher level features can be extracted within less training time.

3.2 Metric learning

The second critical part in person Re-Id is metric learning, which is a method to measure the similarity between two images based on extracted features [Bellet et al., 2013]. There are many different metric learning algorithms with respect to supervised learning and unsupervised learning[Yang and Jin, 2006], the major attention of metric learning used in person Re-Id is only given for supervised distance metric learning.

In the SCNN model designed by Yi et al. [2014], the similarity of two images is computed by cosine distance metric based on image feature vectors. Varior et al. [2016b] proposed a contrastive loss function [Hadsell et al., 2006] based on Euclidean distance to optimize their SCNN model. The contrastive loss function consists of two components which respond to positive pair and negative pair respectively. Through minimization of loss function, the computed Euclidean distance for positive input pairs becomes small gradually whereas it becomes large gradually. So that the feature vectors of same person are close, while the feature vectors of different person are far away. Varior et al. [2016a] improved their SCNN model based on the baseline of [Varior et al., 2016b]. A Gaussian activation function is used to obtain the gate values which indicate the degree of local similarity of each stripe. During training phase, gate function also boosts the back propagation related to local similarities. In contrast to using image pairs, Hermans et al. [2017] introduced a TriNet using triplet loss function [Cheng et al., 2016] that takes three different images as input. In consideration of the absolute distance between positive pair and negative pair, triplet loss is comprised of pull-term and push-term. Furthermore, based on the original triplet loss function in [Cheng et al., 2016], Hermans et al. [2017] designed batch hard triplet loss and batch all triplet loss in consideration of mining hard triplets. The result in Hermans et al. [2017] shows that training using hard samples can improve network performance efficiently compared to using all training samples. In the research of [Liu et al., 2017] and [Bai et al., 2017], a three-branch network is learned by multi-task loss which is consisted of triplet loss and identification loss. Identification loss motivates the network to generate better global features, while triplet loss with Euclidean distance metric is used for similarity measurement. Benefit from Triplet network, the accuracy of Person Re-Id within Market-1501, CUHK dataset reaches a new level in Bai et al. [2017]. However, inputs of Triplet network have to be arranged with three different images. The effectiveness of this method is limited by computational inefficiency particularly in testing phase.

3.3 Person Re-Id for Tracking

Person Re-Id for tracking is also consisted of two steps, feature extraction and metric learning. But different from general person Re-Id task in which image retrieval algorithm is employed in testing phase, person Re-Id for tracking is still treated as classification problem in the phase of testing.

In the work of [Tang et al., 2017], three CNN architectures are investigated. First of all, Tang et al. developed an ID-Net to deal with N-way classification problem using VGG net. Secondly, a SiameseNet is deigned in Siamese Architecture, in which feature extractor uses VGG pre-trained model and similarity computation is based on softmax function, which estimates the probability of binary decision. Thirdly, a pair of images are stacked and fed into StackNet which is a one channel CNN model to model a 2-way classification problem. Additionally, Tang et al. [2017] also proposed a body part fusing method in StackNet, compared with StackNet StackNetPose improved accuracy by 3.1%. Leal-Taixé et al. [2016] designed a Siamese architecture to compare two image patches. In this SCNN the features of stacked images are extracted by the first three convolutional layers, each of them uses PreReLu as activation function. Afterwards, four fully connected layers aim at concatenating two different inputs. A binary softmax function is employed in the last fully connected layer like SiameseNet in [Tang et al., 2017].

3.4 Discussion

The investigations of our thesis focus on dealing with person matching problem in multi-person tracking. Inspired by the works of [Tang et al., 2017] and [Leal-Taixé et al., 2016], the architecture of our model is designed in Siamese structure. There are two reasons for this: (1) in our dataset the number of pedestrian is large but the number of images for each person is small, the traditional one-channel network architecture where training samples are labelled depending on person identity such as ID-Net cannot achieve expected result [Tang et al., 2017]. But in Siamese architecture, inputs are assigned to binary labels only considering of pedestrians belongs to one person or not. Therefore, the number of training data is greatly increased. (2) in Siamese network, feature extraction and similarity computation can be combined together. Inputs are fed into CNNs to generate feature vectors, then, similarity is computed by comparing the extracted representations. Afterwards the produced similarity matrx is applied to loss computation. Finally, the features can be optimized since the parameters in CNNs are updated via minimization of loss function. In consequence, our model can be designed as an end-to-end decision network to estimate person matching problem between two images efficiently.

For feature extraction, we employ deep learning, in which a pre-trained ResNet-50 is adopted as feature extractor. The number of features extracted by deep Convolutional Neural Network are controlled by the number of filters in convolutional layers. The hand-crafted features extracted in [Gheissari et al., 2006], [Gray and Tao, 2008] and [Varior et al., 2016b] are generally the low-level features, while the CNN-based features are high-level features, which are more robust than low-level features. Although ResNet-50 is a remarkable architecture to extract features, it takes a long time to train due to its complex architecture. To improve training efficiency, we use the pre-trained ResNet-50 whose weights are provided by He et al. [2016].

Our similarity measurement methods are designed in consideration of Siamese network architecture. The first method is to use softmax function in the last fully connected layers, and define the classification result as a binary decision similar to SiameseNet in [Tang et al., 2017]. In this case, our SCNN is trained by binary cross-entropy function. The second method in our model is to learn a similarity metric by computing Euclidean distance between two images. Then the model is trained by contrastive loss function like in [Varior et al., 2016b] and [Varior et al., 2016a].

4 Theoretical Background

4.1 Neural Network

4.1.1 Model of Neuron

Neural network is a machine learning algorithm used in many computer vision tasks related to pattern recognition, image classification, etc. This concept is created for the first time in 1943 by the neurophysiologist Warren S. McCulloch [McCulloch and Pitts, 1943]. The basic idea behind neural network is to mimic interactive way including receiving, transmitting and processing information in human brain based on neuron [Hagan et al., 1996].

The most basic unit of a neural network is the artificial neuron. A simply model of a single neuron is shown in Figure 1. The neuron receives several inputs $x_1, x_2, x_3, ..., x_n$, each input variable is multiplied with the corresponding weights $w_1, w_2, w_3, ..., w_n$. Additionally, input data is summed with a bias term b. The summation Σ is activated by an activation function f, so that the non-linear relationship between inputs $x_1, x_2, x_3, ..., x_n$ and output y can be established. The computational processing in mathematical form can be shown as:

$$\sigma = \sum_{i=1}^{n} w_i x_i + b \tag{4.1}$$

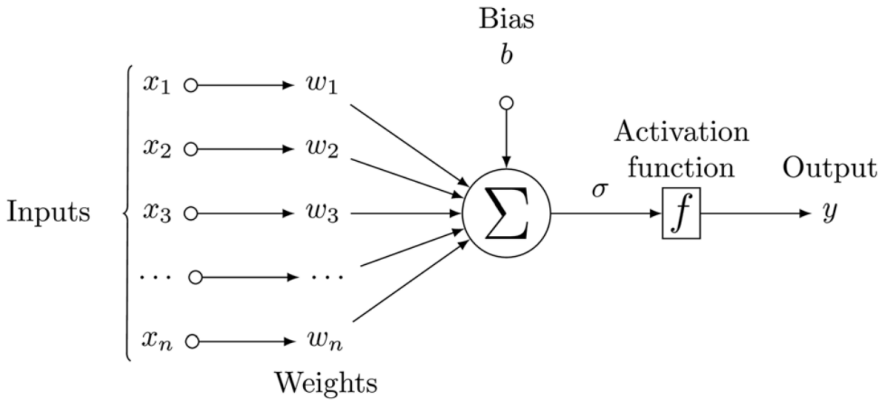


Figure 1: Model of neuron [Dering and Tucker, 2017]

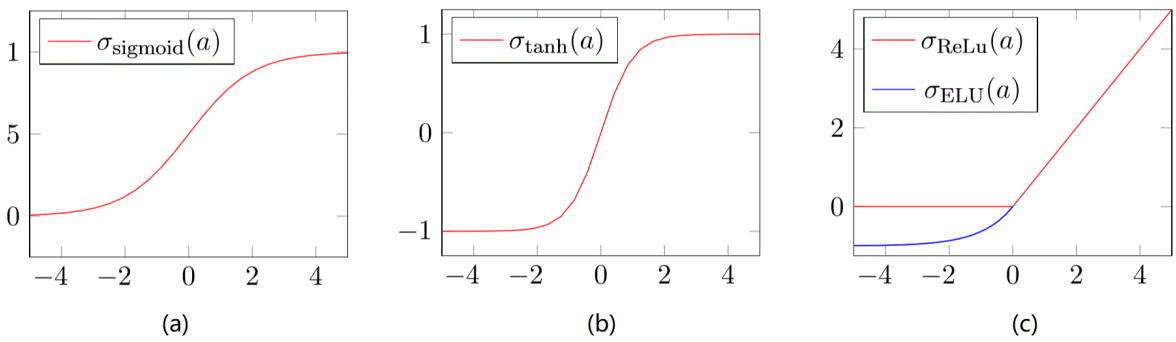


Figure 2: Activation Functions [Skog, 2017]

$$y = f(\sigma) = f(\sum_{i=1}^{n} w_i x_i + b) \tag{4.2}$$

where x_i is one element of input vector, wi is its corresponding weight of x_i, b is bias term of this neuron, σ is the summation of weighted inputs and bias, f is activation, y is predicted output of this neuron.

4.1.2 Activation Function

The role of Activation functions is to perform a non-linear transformation on input. Figure 2 illustrates several common activation functions, each of them has different properties.

Sigmoid The sigmoid activation function is expressed in mathematical form as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

Figure 2(a) shown, the outputs of sigmoid is limited in the range between 0 and 1. When the number approaches to negative infinity, output is 0, conversely as the number approaches to positive infinity the output is close to 1. The advantage of sigmoid is that its output can be treated as normalized probabilistic interpretation in classification task intuitively. Derivative of sigmoid is also easy to be calculated in computational processing. However, since the derivative in region where activation saturates at 0 or 1 is almost zero, gradient vanishing in training can occur easily. Furthermore, convergence result is relatively poor due to sigmoid is not zero-centered function.

The Tangens Hyperbolicus The Tangens Hyperbolicus (tanh) activation function is derived from sigmoid. The shape of tanh presented in Figure 2(b) is very similar to sigmoid. However, its outputs are compressed into [-1,1], zero-centered property is helpful to improve convergence

processing. But gradient vanishing is still a problem cannot be refrained. The mathematical formula is shown below:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.4}$$

Rectified Linear Unit The Rectified Linear Unit is one of the most popular activation functions in the last year, which can solve the problem of gradient vanishing completely. It is simply expressed as:

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases} \tag{4.5}$$

As Figure 2(c) shown, ReLU is divided into two parts with the threshold 0. Obviously, when input signal from previous layer is negative number, output is 0, whereas output is equal to the input. Compared with the functions mentioned previously, ReLU not only can greatly accelerate the convergence, but also reduces computational complexity due to exponential operation is not necessary. However, ReLU probably leads to neurons becoming dead. That means if a large gradient flows through a ReLU neuron, this neuron will never activate and its weights will be frozen in entire training. Setting a small learning rate is a way to avoid this issue.

Exponential Linear Units Figure 2(c) also shows Exponential Linear Units activation function which takes the advantage of ReLU while overcomes dead neuron problem. The expression of ELU is formalized as:

$$ELU(x) = \begin{cases} e^x - 1 & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases} \tag{4.6}$$

Output of ELU exists the possibility of negative number, as the input is smaller than 0. Therefore, the average of outputs from ELU is near to zero. From this perspective, ELU can greatly decrease the risk of neuron dead.

4.1.3 Training of Neural Network

In practice, neural network (NN) is organized of multiple layers shown in Figure 3 and in each layer contains several neurons. Benefit from the multi-layered architecture, handling more complex classification problem become possible for NN. Training is a process to learn the parameters in each layers, so that the network can generate expected result in testing data.

There exist three main steps in training. The first step is called forward propagation. The information provided by inputs is processed layer by layer in the forward direction. More specifically, outputs of neurons from previous layer are treated as inputs for the next layer. The

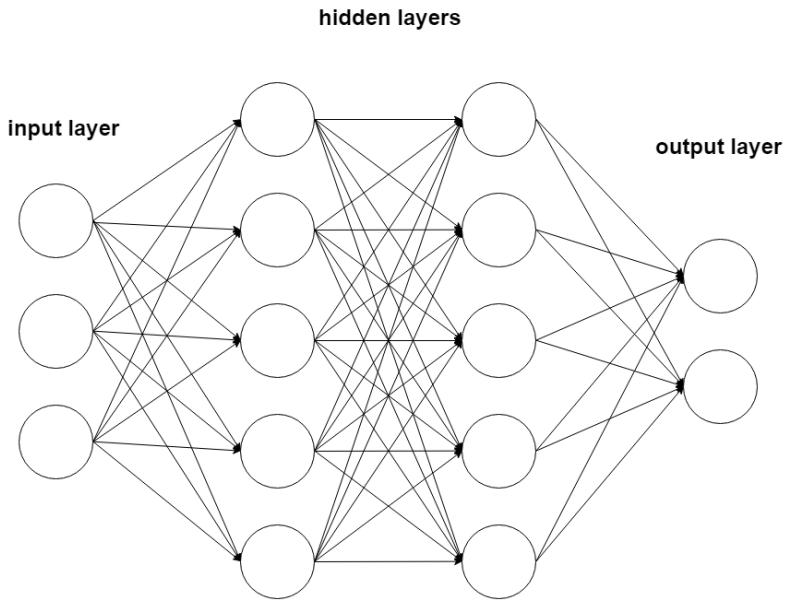


Figure 3: Architecture of Neural Network

number of neurons in output layer refers to the number of classes, outputs predict the degree of membership to the corresponding classes. Additionally, the loss function that denotes the inconsistency between predicted outputs and provided labels is also computed. The second step in training is back propagation [Rumelhart et al., 1986] where the information related to gradient of loss function with respect to the parameters is computed and flows back to hidden layers. In the last step, gradient descent algorithm is applied to update the parameters of network based on computed gradient from the last step. Gradient descent is used to update the parameters with the purpose of minimizing the loss. Finally, the best configuration is obtained as the loss of network is minimum. The details of gradient descent are discussed in next section.

4.1.4 Gradient Descent

A crucial requirement of training is to minimize loss function. The general way to find the optimal solution is gradient descent. In geometrical, gradient points out the changing direction of a function with respect to the value of a parameters. For example, at the point (x_0, y_0) of function $f(x, y)$, along the direction of its gradient $\left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial y_0}\right)$ is the fastest way to reach one of its local minimas or maxima. Similarly, to train neural network, we compute the loss function at first, then update the parameters in the opposite direction of their corresponding derivatives of loss function, in this way the loss is reduced after each updating iteration. Gradient descent algorithm can be carried out by the following four steps [Goodfellow et al., 2016]:

First of all, a function is used for modelling the mapping between a single training sample x^i

and its output y^i in neural network.

$$\widehat{y^i} = f(x^i, \theta) \tag{4.7}$$

where θ represents the parameters of the neural network, $\widehat{y^i}$ is predicted output of network and x^i is its corresponding input. Secondly, the loss function related to predict $\widehat{y^i}$ and target y^i is represented as equation (4.8).

$$J_\theta = \frac{1}{n}\sum_{i=1}^{n} L\big(\widehat{y^i}, y^i, \theta\big) \tag{4.8}$$

where n is the number of all training samples. Weight initialization is an important processing for ensuring the network can be optimized by gradient descent. Because the precondition of gradient descent is that the gradient could not be zero.

Then, we use equation (4.9) to compute the gradient of loss function. It is worth to mention that weights must be updated in the opposite direction of gradient.

$$\bigtriangledown_\theta J_\theta = -\frac{1}{n}\sum_{i=1}^{n} \bigtriangledown_\theta L\big(\widehat{y^i}, y^i, \theta\big) \tag{4.9}$$

In the end, we apply computed gradient estimate to update parameters θ as follow:

$$\widehat{\theta} = \theta - \alpha\frac{1}{n}\sum_{i=1}^{n} \bigtriangledown_\theta L\big(\widehat{y^i}, y^i, \theta\big) \tag{4.10}$$

where α is learning rate, which determines the degree of decay. Setting up a suitable learning rate is a crucial task in training. If using large learning rate, the optimal solution could be missed, whereas it will cost too much time for the convergence. It is better to use a large learning rate at the beginning of training and decrease it over time.

The method mentioned above is called Batch Gradient Descent (BGD). The advantage of BGD is that global optimal solution of loss function can be found due to all training samples are used for gradient computation. However, if the size of training samples is large, training would be very expensive in term of computational cost.

Stochastic Gradient Descent (SGD) is an alternative method to optimize network. It mathematical expression is as:

$$\widehat{\theta} = \theta - \alpha\frac{1}{m}\sum_{i=m}^{n} \bigtriangledown_\theta L\big(\widehat{y^i}, y^i, \theta\big) \tag{4.11}$$

Compared to BGD, the basic idea if SGD is to sample m examples from the whole training samples with their corresponding targets. The gradient is computed by the average gradient of m samples instead of all training samples. On the one hand, it could reduce training time

and converge fast. On the other hand, considering of only a few training samples used for optimization, the convergence result maybe not global minima rather local minima.

Additionally, there are two parameters in SGD used for automatically adjusting learning rate over time. Weight decay λ is a parameter to gradually decrease learning rate after each parameter update. Momentum μ is a parameter to optimize SGD. The basic idea is to simulate the inertia of objects moving. If the direction of the current gradient is consistent with the previous direction, then we enhance the gradient in current direction, whereas the gradient will decay.

4.1.5 Underfitting and Overfitting

Unterfitting occurs when the network is unsuitable for data. If a network produces poor performance in training and testing set, it could be underfitting. Generally, the network with underfitting can be detected by cross validation. In fact, the reason of underfiiting is network is too simple to fit data. From the perspective of network, underfitting could be so solved is by adding the number of parameters, as the complexity of network increases, the prediction capability is also raised.

Overfitting is also a common problem in machine learning. That means a network learns the noise of training data leading a negative performance for new data. Specifically, overfitting is a phenomenon that trained network works well on training set, but the performance on testing set is very poor. In other words, the model is short of generalization.

Dropout is an efficient method to prevent network from overfitting [Srivastava et al., 2014]. The basic idea is that some of neurons in hidden layers are selected randomly with probability m to stop working during forward propagation. And m essentially is a hyperparameter in the interval between 0 and 1 which need to be defined before training. Once a neuron is selected, it will not be able to connect with any other neuron in that iteration. Because selected neurons are changed in each iteration, configuration is also changed. The final output is conducted by averaging over all possible configurations. As a result, overfitting could be solved due to less co-adaptation between neurons. A standard NN without dropout is shown on the left side of Figure 4, while a NN applied dropout with $m = 0.5$ is shown on the other side.

4.2 Convolutional Neural Network

In general, NN is organized in fully connected network structure, i.e., the neurons in the network are connected to each adjacent neuron in hidden layers. When using neural networks to handle image recognition tasks, the network becomes very difficult to train since the number of parameter is too enormous. Convolutional Neural Network (CNN) is very similar to NN which is also

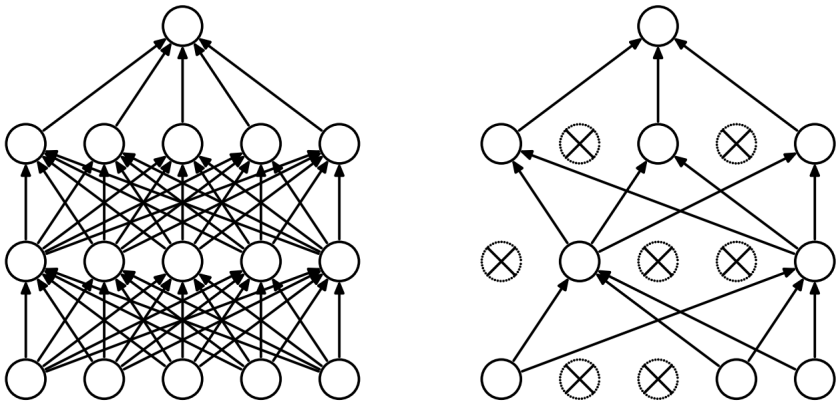


Figure 4: Illumination of Dropout [Srivastava et al., 2014]

constructed based on neurons. But different from all-connected structure of NN, each neuron in CNN only receives some inputs.

4.2.1 Convolutional Neural Network Architecture

Basically, CNN is consists of four main components, namely convolutional layer, nonlinearity, pooling layer and fully connected layers. The role of convolutional layer is to extract various feature maps of input. The output of convolutional layer is in 3D form related to height, width and depth. The height and width imply the size of feature map, while depth is the number of kernel filters. Nonlinertity is similar to activation function, it is applied before features come to pooling layer. The aim of pooling layer is to reduce data volume by abstracting feature maps while maintaining invariance of features. Generally, the last layers in CNN are fully connected layers. Finally, output layer is treated as classifier to predict probability with the help of softmax function. Neurons of intermediate layer in CNN are not connected to all neurons rather linked to a local region of its input layer, which is called receptive field. Weights sharing is realized by convolution operation on neurons located in the receptive field.

4.2.2 Convolutional Layer

Convolutional layer plays a significant role in feature extraction. The major idea is to operate input image using convolution kernel which is also called filter. Each parameter of filter is equivalent to the weight in the traditional NN. In practical, these parameters need to be initialized, then updated through training.

As Figure 5 shown, there are a 5×5 input image and a 3×3 filter. Filter is multiplied with each pixel in a local 3×3 matrix of input image. Finally, the summation of all products is the convolution output for corresponding region of input image. As the filter slides over the input

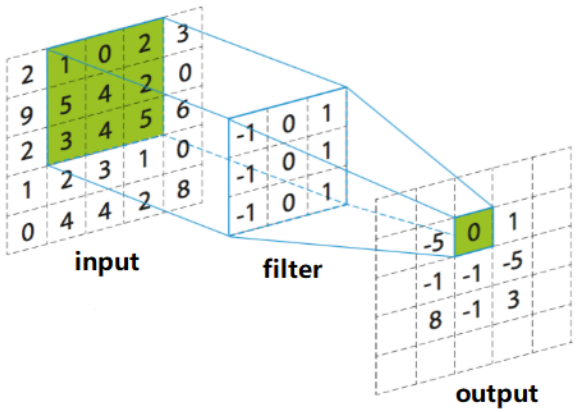


Figure 5: Illumination of operation in convolutional layer[Lin, 2016]

image, the original image is mapped as a new 3×3 feature map.

The size of the feature map is controlled by the following three parameters:

- **Depth** Depth denotes the number of filters used in convolutional layer. Each filter produces one feature map. The final output of this layer is arranged in spatial structure related to the depth.

- **Stride** Stride is the neighbouring distance of the filter moving once on the input image. When the stride is 1, the filter moves only 1 pixel at a time. When the stride is 2, the filter moves 2 pixels at a time. Hence the larger stride is, the smaller the generated feature map is.

- **Zero Padding** As the increasing number of convolutional layer, the size of feature map becomes smaller and smaller since pixels of image boundaries are lost. Padding is a useful method to fill the image boundaries with zeros. In this way, the convolved image has the same size as input image.

In additional, between convolutional layer and pooling layer is nonlinearity layer, which corresponds to the activation function of NN described in Section 4.1. The role of nonlinearity is to integrate the obtained feature map and conduct non-linear transformation.

4.2.3 Pooling Layer

Pooling layer refers to progressively reduce the spatial size of the representation. Through removing the unimportant elements in feature map, the number of trainable parameters is further decreased. For a complex network pooling layer play a role on reducing computational cost and controlling overfitting [Li and Andrej, 2015]. Additionally, pooling is a method to mitigate distortions in the input, thereby introducing translations invariance of features. The common

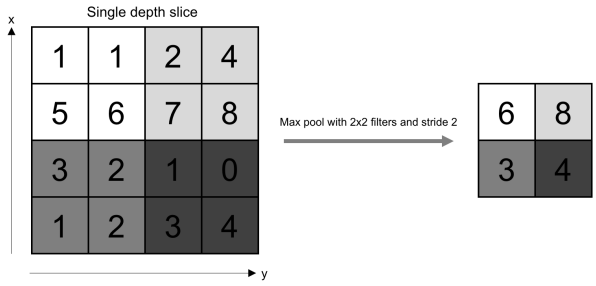


Figure 6: Downsampling in max pooling [Shanmugamani, 2018]

sampling method is max pooling or mean pooling. In the case of max pooling, the largest element in a defined window (e.g. a 2×2 matrix as shown in the Figure 6) of feature map is sampled. Alternatively, mean pooling is conducted by taking the average of all the elements in this window.

4.2.4 Batch Normalization

Batch normalization designed by Ioffe and Szegedy [2015] is an algorithm that makes normalizing outputs of layer over each training mini-batch. Generally batch normalization is carried out between convolutional layers and nonlinearity activation functions. Basically, batch normalization can reduce the impact of codependency of adjacent layers by controlling distribution of outputs using its mean and variance. In detail, after normalizing the outputs have a mean of 0 and its variance is kept as 1. The mathematical processing of batch normalization is shown bellow:

$$y_i = BN_{\gamma,\beta}(x_i) \tag{4.12}$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.13}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu)^2 \tag{4.14}$$

$$\widehat{x_i} = \frac{x_i - \mu}{\sqrt{\sigma^2}} \tag{4.15}$$

$$y_i = \gamma\widehat{x_i} + \beta \tag{4.16}$$

where x_i is the output of convolutional layer and then it need to be normalized before going through activation function, γ and β are two learnable hyperparameters which called as scale and shift, μ is mean and σ is variance of mini-batch. The normalized $\widehat{x_i}$ is generated by μ, σ and the final resulting y_i is calculated also with the help of γ and β .

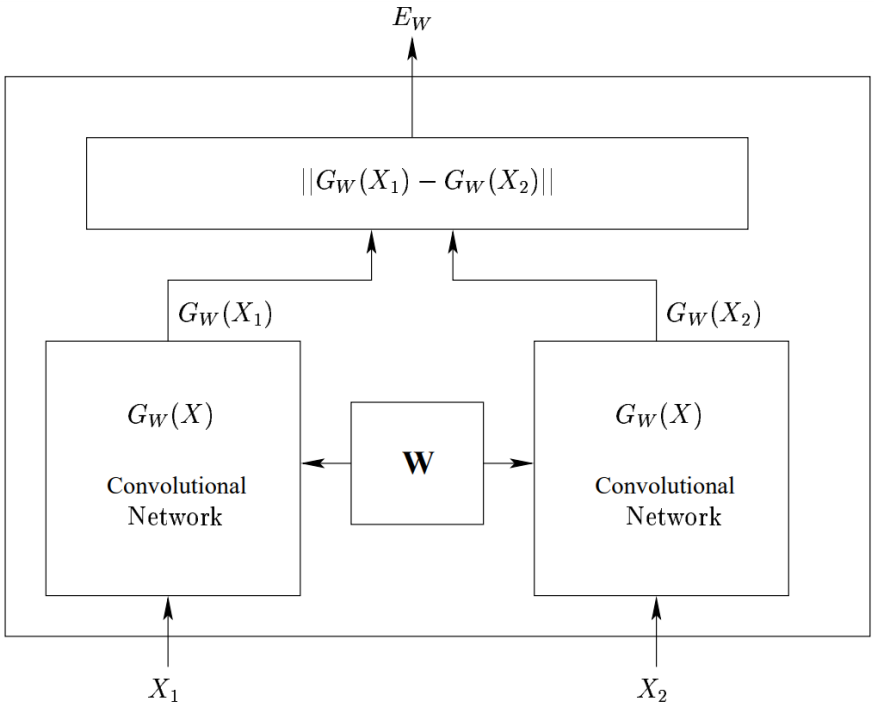


Figure 7: Siamese Neural Network Architecture [Chopra et al., 2005]

Benefit from batch normalization, we can pay less attention of initialization and learning rate. Additionally, the outputs in each layer are compressed into the same scale, thereby batch normalization solves the vanishing gradient problem during back propagation so that makes network converge faster.

4.3 Siamese Convolutional Neural Network

When using Convolutional Neural Network to handle classification task, one requirement is the categories of data have to been known in advance. However, this traditional NN architecture is not suitable for all cases such as person Re-Id application, in which the number of pedestrian is very large and the samples of each pedestrian is small. For solving this problem Chopra et al. [2005] advocated Siamese Convolutional Neural Network (SCNN). The main idea of SCNN is to process two images with two identical sub-networks and learn a similarity matrix which indicates the similarity between two targets computed by their feature vectors. As the similarity score between two input images is smaller, the probability that two images have the same class is larger.

4.3.1 Siamese Neural Network Architecture

X_1 and X_2 are a pair of images with a binary label. If the images X_1 and X_2 with the same person, the label Y of pair is 1 (positive pair). Otherwise, the label Y is 0 (negative pair). Therefore, our input sample is arranged as (X_1, X_2, Y). X_1 and X_2 are fed into two identical Convolutional Networks where the parameters W are shared. $G_W(X)$ denotes extracted feature representation of the responding input image which is generated by Convolutional Network. $G_W(X_1)$and $G_W(X_2)$ can be treated as positive pair with high similarity, while they are classified to negative pair with small similarity.

Then a similarity matrix is added to score the similarity between $G_W(X_1)$ and $G_W(X_2)$, which can be expressed as:

$$D_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\| \tag{4.17}$$

The SCNN is trained by minimizing $D_W(X_1, X_2)$ for positive pairs and maximizing $D_W(X_1, X_2)$ for negative pairs. As consequence, the feature vectors which belongs to the same target would be pulled closer, while the feature vectors of negative pairs are pushed away. The output of SCNN is a simple distance which implicitly indicates the similarity of inputs in target space, and the model generally is trained by minimization of contrastive loss function. Essentially, the output of SCNN can take alternative form when sigmoid activation function is used to combine feature vectors $G_W(X_1)$ and $G_W(X_2)$. In this case, the output is a binary label, i.e., 0 or 1 like training set, and cross entropy loss function is employed for training model. Using different similarity measurement methods would result in different SCNN architectures, the further details will be discussed in Chapter 5.

4.3.2 Loss function

SCNN is a model that can combine the tasks of feature extraction and similarity measurement together. In SCNN, the outputs of the last layers from two sub-networks are the learned feature vectors. Then an additional layer follows Siamese twin for scoring the similarity between feature vectors. In back propagation, the weights in sub-networks are updated so that the feature vectors are adapted and optimized until the loss function achieves at minimum value. There are two common loss functions used for SCNN training, namely contrastive loss function and cross entropy loss function.

Contrastive Loss Function The similarity between inputs X_1 and X_2 is computed using similarity matrix and represented as D_W in Equation 4.17. The resulted similarity is a significant

component of Contrastive loss function:

$$L(W) = \sum_{i=1}^{N}(1-Y)L_p(D_W) + YL_n(D_W) \tag{4.18}$$

where $L(W)$ is the summed loss for all samples, N is the number of training samples. The loss is consisted of two terms, L_p is the partial loss function for positive pairs and L_n denotes the partial loss function for negative pairs. In this case, minimizing contrastive loss function is a processing to minimize similarity matrix for positive pairs and simultaneously maximize similarity matrix for negative pairs.

Hadsell et al. [2006] designed a improved contrastive loss function by adding a margin parameter m. The mathematical form of this improved loss function is:

$$L(W) = \frac{1}{2N}\sum_{i=1}^{N} Y\frac{1}{2}(D_W(X_1, X_2))^2 + (1-Y)\frac{1}{2}\big\{max(0, m - D_W(X_1, X_2))\big\}^2 \tag{4.19}$$

the negative pair has contribution to the loss function only if the computed similarity matrix D_W of this pairs smaller than predefined margin m. Otherwise, this term equals to 0. Specifically, as images from different identities but the computed similarity matrix is larger than m, then this image pair promotes loss decreasing, while the negative pair with small value in term of similarity matrix motivates loss function to increase. Compared with the original contrastive loss function, the improved loss function not only considers normal negative pairs, but also focuses on training hardest negative pairs. As the training continues, loss is decrease gradually, the the feature vectors of positive pairs become closer and feature vectors belong to negative pairs are pushed away.

Cross entropy Loss Function Cross entropy loss function is an alternative choice for training SCNN in the application of image classification. For example, in Siamese-Classification network [Koch et al., 2015], two feature vectors are combined with softmax activation function based on L_1 distance such that the output is produced as a probability value between 0 and 1. For a sample with actual label 1, the cross entropy loss increases as the predicted probability approaches to 0, while it decreases as the output closes to 1. The cross entropy loss function can be expressed as:

$$L(W) = \frac{1}{N}\sum_{i=1}^{N}\big\{(1-Y)\log(1 - P(X_1, X_2)) + Y\log P(X_1, X_2)\big\} \tag{4.20}$$

$$P(X_1, X_2) = f(\|G_W(X_1) - G_W(X_2)\|) \tag{4.21}$$

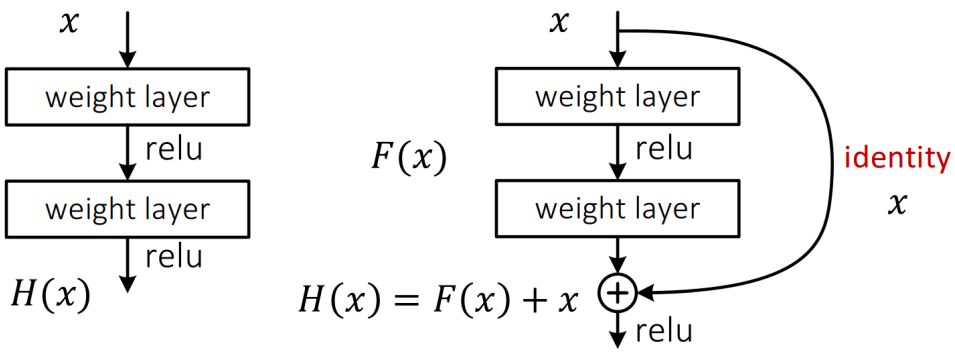


Figure 8: Illumination of building blocks. **Left:** a building block in plain network. **Right:** a building block using residual learning in ResNet architecture[He, 2016]

where $P(X_1, X_2)$ is the predicted result given by softmax activation function. Because training data only contains positive pair and negative pair, binary cross entropy loss function is natural selection which contains two terms with respect to similar pair and dissimilar pair like contrastive loss function. The final decision of SCNN depends on probability. prediction is negative pair when its probability is larger than 50% , otherwise prediction is regarded as positive pair.

4.4 Deep Residual Network

In general, degradation problem is a big challenge as the depth of the neural network continuously increase, it indicates that the accuracy of the model will rise first and then decrease. *Heet al.* [2016] apply deep residual learning to their ResNet, in which the number of layers in a deep CNN is increased to hundreds layers.

The difference between conventional network and residual network is shown in the Figure 8. In the block of a plain network (Figure 8, left), input x is processed by two stacked weight layers and output $H(x)$ is an underlying mapping generated by weight layers. In the block of residual network (Figure 8, right), original mapping $H(x)$ is transferred as a residual mapping $F(x)$ with respect to identity mapping x, whose formulation is: $H(x) = F(x) + x$. Instead of learning underlying mapping $H(x)$ in plain network, residual network learn the residual mapping $F(x)$.

ResNet-50 is one common pre-trained model applying for many different tasks, which consists of total 50 convolutional layers that are formed into a set of blocks (Table 1). The input image must be 224×224. Each block in ResNet-50 is consisted of 3 stacked convolutional layers whose structure is fixed by 1×1, 3×3 and 1×1 convolutions. Additionally, every certain number of blocks are formed as a component. Specifically, $Conv2\_x$ is consisted of one max pooling layer

and 3 blocks, $Conv3\_x$ has 4 blocks, $Conv4\_x$ contains 6 blocks and $Conv5\_x$ has 3 blocks. The last two layers of ResNet-50 are a global average pooling layer and a 1000-way fully connected layer with softmax.

Table 1: Architecture of ResNet-50 [He et al., 2016]

Layer	Output Size	Filter Size
Conv1	112×112	7×7, 64 stride 2
Conv2_x	56×56	3×3 max pooling, stride 2
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
Conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
Conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pooling, 1000 dimensions

4.5 Transfer Learning

Although deep Convolutional Neural Networks have achieved remarkable results in the field of computer vision, it is an expensive method in aspect of data and time. Specifically, a deep CNN requires an extreme volume of data for approximating millions of parameters accurately and the training process generally takes several days even several weeks.

Transfer learning is an idea to apply knowledge learned in one task to another different but related task [LeCun et al., 2015]. For example, in the first task a set of categories like cats and dogs are learned while a different set of categories like ants and wasps are need to be classified. If a model is trained using sufficient number of data in the first task, then the model can easily extract discriminative features to distinguish different targets only use a few samples for retraining the previous model. The CNN model trained for the first task is called pre-trained model. The role

of pre-trained model usually is a feature extractor in which all parameters are fixed, and a new classifier is involved on the top of the features learned by pre-trained model. In practice, learned knowledge is stored in pre-trained model through the approximated parameters, compared to training a network from the scratch using the parameters of pre-trained model as start point can reduce training time and improve performance of model [Li and Andrej, 2015].

There are many public pre-trained models which are provided by researchers with respect to different applications. Such as VGG network introduced in [Simonyan and Zisserman, 2014], Inception architecture proposed by Szegedy et al. [2015] and ResNet-50 designed by He et al. [2016] are the common pre-trained models which are widely used as feature extractors in many tasks related to image classification, segmentation and object detection etc,.

However, pre-trained model based on fixed parameters is a good way only in the condition that new dataset is limited and very similar to original dataset. In other cases, these approximated parameters particularly the parameters of the later layers maybe not suitable for actual task. Because the features learned by CNNs is hierarchical features, that means features extracted in early layer are more generic, the later layers learn features more specific to the details containing in original datasets [Li and Andrej, 2015]. In order to obtain more compliant features for actual task, the weights in some convolutional layers need to be fine-tuned through back propagation. Fine-tuning strategy could be employed in some top layers while freezing the rest layers, and it is also possible to fine-tune all layers of network.

The significant factors which determine the number of fine-tuned layers are the size of new dataset and its similarity to original dataset. Fine-tuning the layers from the top of network or training a linear classifier without fine-tuning is a selection as the dataset is small and similar to the original dataset. Along with the increasing size of dataset and the decreasing similarity, the number of fine-tuned layers needs to be increased, indeed fine-tuning the whole network is also necessary. Additionally, when the dataset is so large that network can be trained without the help of transfer learning, initialization with the weights from a pre-trained model is still a method to obtain better results [Li and Andrej, 2015].

5 Methodologies

In this chapter, we will discuss methods used in this thesis in detail. Firstly, in Section 5.1, the architecture of our designed Siamese Neural Network is introduced in aspects of feature extraction and similarity computation. Then, the approaches employed during training for improving performance of SCNN are given in Section 5.2.

5.1 Our Proposed Model

The aim of this thesis is to design an end-to-end learn model in Siamese architecture to determine two captured pedestrians whether belong to the some person or not. Specifically, in Figure 9, detections from different frames are extracted and formed in image pairs. Then our SCNN model can directly judge the person shown in two images matching or not matching. A SCNN model is consisted of two main components, feature extraction and similarity computation. We select pre-trained ResNet-50 as our feature extractor and investigate fine-tuning approach to improve the performances of our models. We propose two baselines which are optimized by binary crossentropy loss and contrastive loss function repressively.

5.1.1 Pre-Trained Model

Generally, millions of parameters in deep Convolutional Neural Networks are learned during training. For reducing training time and improving the effectiveness of convergence, we use pre-trianed ResNet-50 architecture as feature extractor. One important reason why we select ResNet50 as pre-trained model is that its architecture is deeper than previous deep CNNs like AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2014] etc,. So that it could extract more abstract feature vectors and obtain more abundant information from input images. Additionally, ResNet architecture took part in Large Scale Visual Recognition Challenge 2015 (ILSVRC2015) and COCO 2015 competitions and won the first place related on the tasks of image detection, image classification and image segmentation.

We use pre-trained ResNet-50 without its top fully connected layer to extract representation of input images. In order to prevent the existence of zero variances in forward propagation and

Figure 9: Person Re-Id for Tracking based on Siamese CNN

reduce the risk of gradient vanishing in back propagation, batch normalization layer is added between convolutional layer and activation layer. The weights of ResNet-50 is provided by He [2016] which was trained on ImageNet [He, 2016] for object classification task.

5.1.2 Classification Models

Designing SCNN for person Re-Id should take into account not only feature extractor, but also the methods applying to measure similarity between extracted feature vectors. Our proposed SCNNs use ResNet-50 architecture as feature extractor with the help of transfer learning. Next step is to design the classifier in which feature vectors of image pair are further processed and the finial decision is estimated by computed similarity.

Baseline1

The basic idea of *Baseline*1 is to design a binary classifier. As Figure 10 shown, two images are parallelly processed by two weights-shared ResNet-50 networks to extract corresponding feature vectors. Before entering the fully connected layer, multi-dimensional feature maps is converted to a single vector with 2048 dimension. In order to explore the performance of SCNNs with different similarity measurement methods, we design three variants of *Baseline*1 with respect

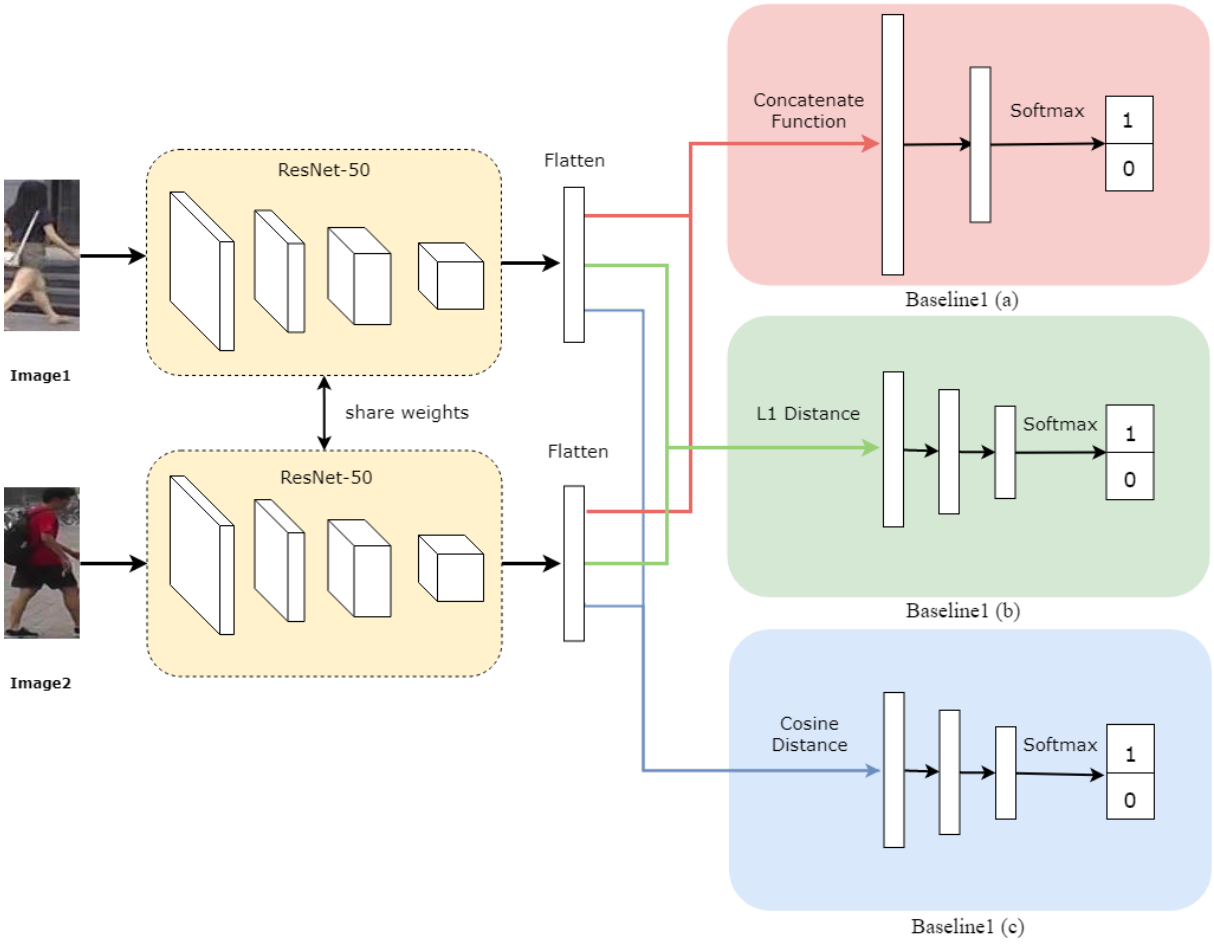


Figure 10: Baseline1 for Person Re-Id with 3 different similarity measurement methods

to three similarity measurement methods, namely concatenate function, consine distance matrix and L_1 distance matrix.

The first variant of $Baseline1(a)$ is a model where the similarity between images is modeled using fully connected layers representing in the first branch of Figure 10. Specifically, the features from sub-networks $Flatten1$ and $Flatten2$ are linked by concatenate layer in which multiple input tensors concatenate to a single tensor along the certain axis. Then, this concatenated vector is processed by two fully connected layers FC_1 and FC_2. FC_1 uses ReLu non-linearity with 1024 dimension and FC_2 uses softmax function to approximate the probability prediction of person matching. This SCNN model is trained by the binary cross entropy loss. The architecture of $Baseline1(a)$ is shown in the Table 2.

As the second branch in Figure 10 shown, we use element-wise absolute distance (L_1) to merge two feature vectors. The Equation 5.1 expresses the mathematical form of L_1 distance between feature vectors X_1 and X_2 related to element i.

$$D_{L1}(X_1^i, X_2^i) = \|X_1^i - X_2^i\| \tag{5.1}$$

Table 2: Architecture of Baseline1(a)

Layer	Output Size	Activation	Connected to
Input_1	224×224	-	-
Input_2	224×224	-	-
ResNet-50_1	multiple	ReLU	Input_1
ResNet-50_2	multiple	ReLU	Input_2
Flatten_1	2048	-	ResNet-50_1
Flatten_2	2048	-	ResNet-50_2
Concatenate	4096	-	Flatten_1 and Flatten_2
FC_1	1024	ReLU	Concatenate
FC_2	2	Softmax	FC_1

where X_1^i and X_2^i are the ith element of extracted feature vectors of image1 and image2. The architecture of $Baseline1(b)$ is shown in Table 3. The last layer of ResNet-50 is replaced by a 3-layer fully connected network. Specifically, ReLU activation function is applied to the fully connected layers FC_1, FC_2 and a softmax function to the last layer FC_3.

Table 3: Architecture of Baseline1(b)

Layer	Output Size	Activation	Connected to
Input_1	224×224	-	-
Input_2	224×224	-	-
ResNet-50_1	multiple	ReLU	Input_1
ResNet-50_2	multiple	ReLU	Input_2
Flatten_1	2048	-	ResNet-50_1
Flatten_2	2048	-	ResNet-50_2
Lambda_L_1	2048	-	Flatten_1 and Flatten_2
FC_1	512	ReLU	Lambda_L_1
FC_2	128	ReLU	FC_1
FC_3	2	Softmax	FC_2

The third branch in the Figure 10 demonstrates $Baseline1(c)$, which applies element-wise cosine distance to indicate the similarity between two feature vectors. The cosine distance of i

th element is computed using the follow equations:

$$D_{cos}(X_1^i, X_2^i) = \frac{X_1^i X_2^i}{\sqrt{X_1^i X_1^i}\sqrt{X_2^i X_2^i}} \tag{5.2}$$

The architecture of $Baseline1(c)$ is the same as SCNN model with L_1 distance representing in Table 3.

Table 4: Architecture of Baseline1(c)

Layer	Output Size	Activation	Connected to
Input_1	224×224	-	-
Input_2	224×224	-	-
ResNet-50_1	multiple	ReLU	Input_1
ResNet-50_2	multiple	ReLU	Input_2
Flatten_1	2048	-	ResNet-50_1
Flatten_2	2048	-	ResNet-50_2
Lambda_*cos*	2048	-	Flatten_1 and Flatten_2
FC_1	512	ReLU	Lambda_*cos*
FC_2	128	ReLU	FC_1
FC_3	2	Softmax	FC_2

Finally, no matter which variant of $Baseline1$ is used, the output of network is generated by a softmax function indicating the probability estimation of positive and negative label mapping into the interval [0, 1]. All the networks are trained by binary cross entropy loss (Equation 4.20 in Section 4.3).

Baseline2

The workflow of $Baseline2$ is shown in the Figure 11 and its architecture is shown in the Table 5. Similar to $Baseline1$, we use ResNet-50 to extract discriminatieve representations for input images, then it is flatten into a vector with 2048 dimension. Different from $Baseline1$, we employ Euclidean distance matrix learning to $Baseline2$. The similarity between two flatten vectors is measured by Euclidean distance using the following equation:

$$D_{L1}(X_1, X_2) = \sqrt{\sum_{i=1}^{n} \left(X_1^i - X_2^i\right)^2} \tag{5.3}$$

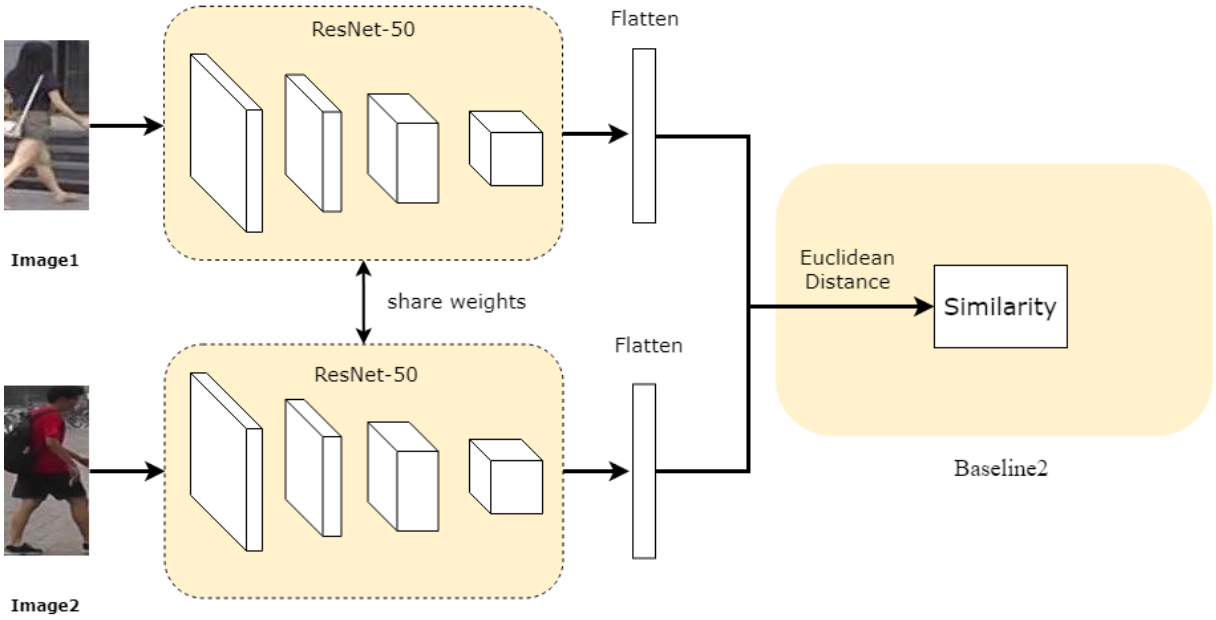


Figure 11: Baseline2 for Person Re-Id with Euclidean matrix learning

Table 5: Architecture of Baseline2

Layer	Output Size	Activation	Connected to
Input_1	224×224	-	-
Input_2	224×224	-	-
ResNet-50_1	multiple	ReLU	Input_1
ResNet-50_2	multiple	ReLU	Input_2
Flatten_1	2048	-	ResNet-50_1
Flatten_2	2048	-	ResNet-50_2
Lambda_*Euc*	1	-	Flatten_1 and Flatten_2

The output of this SCNN model is a single value of Euclidean distance, which explicitly figures out the relationship of images. So that binary decision of person matching relies on the help of a predefined threshold. More precisely, if the result of Euclidean distance matrix for one image pair is larger than the predefined threshold, this image pair can be regarded as negative pair because the distance between two images are far away. The small Euclidean distance means two images are close and this image pair could be treated as positive pair. *Baseline*2 is trained by contrastive loss function (Equation 4.19 in Section 4.3) which contains two terms with respect to positive sample and negative sample. In controlling of minimization of contrastive loss, Euclidean distance of negative pairs is increased, while for positive pair Euclidean distance is decreased.

5.2 Training

5.2.1 Fine-Tuning

For reducing training time, we employ transfer learning approach to our experiments. First of all, we initialize all weights in ResNet-50 from [He et al., 2016]. Then, the initial weights in the fully connected layers are conducted by Xavier normal initializer [Glorot and Bengio, 2010], where value is sampled from a truncated normal distribution with zero-mean and standard deviation $\sigma = \sqrt{\frac{2}{N_{in}+N_{out}}}$, N_{in} and N_{out} denote the number of inputs and outputs. Additionally, the weights in the fine-tuned layers are updated by stochastic gradient descent. We propose an experiment to compare the performances of networks with different number of fine-tuned layers.

5.2.2 Data Augmentation

Our SCNN is designed based on deep learning architecture, in which a large number of available input data is required for training. Indeed, the number of generable positive image pairs are not as many as negative image pairs since the positive pair needs two images corresponding to one person and the number of images for one person is limited. To solve data imbalance problem, we apply data augmentation to positive pairs. Instead of generating augmented images in prepossessing step, data augmentation is directly conducted on GPU in batch generation step while training SCNN. Benefit from this online augmentation strategy, we can generate more kinds of augmented version since there is no requirement of memory space to store augmented pairs. Considering of pedestrian characters, we adopt six geometric distortions adding to positive image pairs, they are rotation, width shift, height shift, shear, zoom and horizontal flip. The random number of transformations are independently applied to each image of positive pair. With the help of data augmentation, the size of training data is increased.

5.2.3 Hard Samples Mining

We adopt hard samples mining to possibly improve the predictive ability of trained SCNN. First of all, we train SCNN with the current training samples. Then the trained SCNN is used to predict training set, namely compute the similarity between each image pair and determine they belong to an identical person or not. We collect the hard negatives in which images are similar-look but belong to different people, and the hard positives which present one identity but with wildly different optical features. Finally, we retrain the initial SCNN using those negative samples. We expect the performance of retrained SCNN using hard samples mining will be improved. Because the samples that are classified incorrectly are generally with large

loss, the network could be optimized by reducing the loss of these samples after retraining. In this thesis, two different method are investigated to mine hard samples, they are hardest mining and moderate mining.

Hardest Samples Mining The aim of hardest samples mining is to find the hardest samples in all training set by ranking the result of similarity matrix learning. Specifically, for mining hardest negative pairs, all negative pairs are sorted in descending order according to computed similarity score, a certain percentage of top-ranked pairs are selected as hardest negatives. Conversely, the top-ranked pairs in the list, in which positive pairs are resorted in ascending order in terms of their similarity, are hardest positive pairs. The number of percentage is a parameter influence on the size of hardest samples. In Chapter 7, an experiment is designed to verify the effect of this parameter on retrained network.

Moderate Samples Mining Because only retraining hardest samples maybe reduce the capability of network to deal with normal pairs, it is common to mine moderate samples which is inspired by Hermans et al. [2017]. The basic idea of moderate samples mining is to find the hardest negative and the hardest positive samples within a batch. For mining moderate negatives, we collect all negative pairs of training set and generate mini batches. In each mini batch, the negative pair with the smallest distance is selected as moderate negatives and added into retraining set. Similarly, all positive pairs are randomly shuffled and create batches, the positive pair with largest distance in batch is picked out and added into retraining set. The number of mining samples is controlled by batch size. The performance of retrained SCNN with different batch size will be discussed in Chapter 7.

6 Data Overview

In this chapter, two common tracking datasets KITTI and MOTChallenge, and one typical person Re-Id dataset Market-1501, are introduces in the first section. Prepossessing steps that extracted bounding boxes with the help of provided ground truth and the geomatrix distortions of data augmentation are given in the second section.

6.1 Dataset Description

6.1.1 KITTI

KITTI is an advanced computer vision benchmark in the field of computer vision related to object detection and tracking [Geiger et al., 2013]. KITTI is captured by high-resolution stereo camera fixed on a vehicle driving around the city of Karlsruhe. The major objects are classified in 8 classes: car, van, truck, pedestrian, person (sitting), cyclist, tram and misc 8 classes. In our case, only training sequences are used for our SCNN experiments because testing sequences have no provided ground truth.

Because camera is fixed on a moving vehicle, image resolution of pedestrians is difficult to maintain high quality as the distance between camera and pedestrian is large. In order to ensure the optical discriminative information can be generated, pedestrian image will only be extracted if its height is larger than 40 pixels. In total 163 identities are collected from 21 training sequences. In our experiments, we select KITTI-16 as testing set and the other sequences to train and validate our networks. Specifically, 80% identities are used for generating training samples, while the validation samples are generated from the rest 20% percent identities. It is worth mentioning that the identities for training, validation and testing have no overlapping, to ensure that the images of a pedestrian used for training set will not appear in testing set.

6.1.2 MOTChallenge

MOT15 [Leal-Taixé et al., 2015] and MOT16 [Milan et al., 2016] datasets are designed as common benchmarks for addressing multiple object tracking problem. Compared with KITTI dataset,

mainly annotated targets in MOT datasets are moving pedestrian and vehicle, which are captured from three different view points, low, medium and high, using static or moving cameras. In total of 755 identities are collected over 5 training sequences from MOT15 and 7 training sequences from MOT16. We choose MOT16-2 and MOT16-11 with 123 identities as testing set for evaluating the performance of our designed SCNNs. Among of the remaining identities, randomly 30% identities are selected as validation set, while the rest 70% identities are selected as training set.

6.1.3 Market-1501

Market-1501 is one of the most common single-shot person Re-Id dataset, in which a single image is captured by multiple cameras with various point views[Zheng et al., 2015]. Different form tracking datasets, the captured target in Market-1501 is only pedestrians. Market-1501 consists of 32668 images for a total of 1501 identities taken by six cameras from different point views at Tsinghua University. Each identity has multiple images with various poses over all cameras.

There are 12936 available images of 751 identities in training set, while testing set has 19732 images of 750 identities. A standard evaluation method of Market-1501 is "query-search", in which a probe image is retrieved within a set of gallery images related to ranking of similarity. Therefore the testing set is split into gallery and query set respectively. At least one image and at most six images for one pedestrian are selected as query images. In total, the number of resulting query images is 3368.

However, Market-1501 dataset is also used for dealing with binary person matching problem in which evaluation method is based on confusion matrix. We split original training data into two parts, 80% identities applied to train SCNNs and the rest 20% identities used for validation. In addition, all pedestrians in testing data also can be found in query set. Therefore, in order to guarantee data independence we only employ images from query set to generate testing samples when evaluation method is carried out using confusion matrix.

6.2 Preprocessing

6.2.1 Bounding Box Extraction

The used datasets in this thesis are not only traditional person Re-Id images like Market-1501 dataset but also the image sequences in which each frame contains many objects of different categories. Hence the bounding boxes for pedestrian class in those benchmarks have to be

Figure 12: A group of pedestrian images sampling from KITTI-02 sequence

Figure 13: A group of pedestrian images sampling in different lighting condition from MOT15-01 sequence

extracted according to ground truth.

Both ground truth files of KITTI and MOT provide information about in which frames the pedestrian appears and each pedestrian is assigned with a unique identity (ID). Furthermore, the position of bounding box in KITTI dataset is provided related to left-top, right-bottom corner coordinates, while in MOT dataset the 2D image coordinates are provided in form of width and height. Summarily, 10902 pedestrian images of 163 identities are extracted from KITTI dataset, in the same way MOT dataset conducts 123671 pedestrian images of 633 identities.

Figure 12 shows an example of extracted pedestrian images from KITTI dataset. Figure 13 illustrates a trajectory of one pedestrian captured by one moving cameras from MOTChallenge dataset. In Figure 14, one student is detected by six different cameras in different environments.

Figure 14: Images of one pedestrian with various pose in Market-1501 dataset

6.2.2 Image Pair Generation

For training our SCNN, the input sample is a pair of images. According to whether pedestrians belong to one identity, a binary label is assigned to each image pair. Specifically, if two images represent the same person then it is a positive pair, otherwise is a negative pair. Positive samples (I_t^i, I_{t+k}^i) are generated by selection of two pedestrian images which have the same ID i from the frames t and $t+k$ respectively, negative samples (I_t^i, I_{t+k}^j) are randomly matched using two pedestrian images with different IDs i and j from the frames at time t and $t+k$ respectively.

Parameter k denotes distance between two selected pedestrian images in term of time series. For Market-1501 dataset, k could be set as a small value because benefit from multiple view points the correlation among the adjacent images is small. But in multiple object tracking datasets, distance parameter k plays a significant role while generating positive samples (I_t^i, I_{t+k}^i). Because the pose of pedestrian changes continuously in sequences. If k is set as small value like Market-1501 dataset, the visual representations between paired images will be very similar. Therefore using different values of distance step not only increases the variety of positive samples, but generalize capability of network can be improved as well.

Inspired from [Leal-Taixé et al., 2016] and considering of the maximum bounding box number for each pedestrian, we generate the image pairs with parameter k equals 1, 3, 5, 8 and 10 in KITTI dataset. The total number of generated image pairs of training set is 73882, validation set is 7816 and testing set is 9347. The setup of parameter K in MOTChallenge dataset is the same as KITTI dataset, there are 168682 training image pairs, 18462 validation image pairs and 52788 testing image pairs. For Market-1501 dataset, the parameter k is only set as 1. The number of image pairs used for training parameters is 20104, for validating model of each epoch is 4266 and for testing network is 5236, and we evaluate the performance of our networks based on confusion matrix.

6.2.3 Batch Generator

Since there is no enough memory space to process our image pairs at once, training samples fed into the network have to be in a batch. Our batch generator is assembled multi processes related to image pairs shuffling, batch generation, image loading, image resize and data augmentation. The first step in batch generator is to shuffle all image pairs for each epoch. Then, shuffled image pairs are split into small batches according to batch size. In our case, batch size is 5, because it is the maximum executable number on our 3GB GPU. That means every 5 image pairs are stacked as a batch. After that, all images of a batch are loaded into the memory and resized from initial size to 224×224 using nearest neighbor interpolation. Finally, after the previous batch fed into

Figure 15: Examples of data augmentation applying random transformation

our SCNNs, the next batch is produced by generator immediately.

6.2.4 Data Augmentation

Our online data augmentation is a strategy conducting augmented version of images from positive pairs during batch generation processing. At first, we defined an augmentation dictionary which contains types of geometric distortions and their parameters. More precisely, we set 5 degree for random rotation, 5% of total width and height for horizontal and vertical random shifting, 0.1 radian for shearing mapping , zooming factor in the range [0.9, 1.1] and turn on horizontal flip. Secondly, a random number of transformations from this augmentation dictionary are independently applied to each resized input images.

Our data augmentation is only used for positive pairs. Because the possible combinations of positive pairs is greatly less than the number of producible negative pairs. So that the number of training data is significantly limited by positive pairs. In image-pairs generation step, additional augmented pair is generated for each positive pair. Figure 15 represents two pedestrians from Market-1501 dataset with comparison of original images and their augmented images.

7 Experiments and Results

First of all, the implementation details are introduced in the Section 7.1. Secondly, we present confusion matrix which will be used for evaluating all experiments in Section 7.2. Then, we design several experiments to explore the performance of SCNN using different training approaches. Section 7.3 demonstrates the results of our network with different experiment settings. The effect of data augmentation is discussed in Section 7.4. Furthermore, we design an experiment shown in Section 7.5 to verify whether retraining of model using mined hard samples mining is useful to improve the performance or not.

By analyzing the results of previous experiments, we apply the best configuration to all proposed Siamese networks on KITTI, MOTChallenge and Market dataset and evaluate the performance of all proposed Siamese networks. The experiments on KITTI is given in Section 7.6 with a comprehensive evaluation. And the performances of *Baseline*1 and *Baseline*2 on MOTChallenge dataset are discussed in Section 7.7 by comparing with the other existing methods. In Section 7.8, the results of experiments on Market-1501 are analyzed not only by confusion matrix, but also by standard "query-searching" evaluation method.

7.1 Implementation Details

All the proposed SCNN models are implemented in Keras framework with the help of Python. Before training our SCNNs, we extract all pedestrian bounding boxes of each image frame in sequences using provided ground truth files. Our CNN-based feature extractor is ResNet-50 [He et al., 2016] which is pre-trained on ImageNet. The number of dimension of flatten layers shown in Figure 10 and Figure 11 is set to 2048. The original fully connected layers in ResNet-50 are substituted by our designed classifiers with respect to different baselines.

For *Baseline*1, the ReLu activation function is used in fully connected layers except the last layer. In the last layer we apply a softmax function to generate binary classification result. In our binary classification case, we regard prediction as negative pair if its probability is larger than 50% , otherwise we treat this sample is positive pair. The optimizer used for updating weights is Stochastic gradient descent (SGD) with base learning rate α of 0.01 and weight decay

$\lambda = 10^{-6}$. Additionally, to optimize SGD we set momentum μ of 0.9. The networks in *Baseline*1 are trained by binary cross entropy loss function.

For *Baseline*2, there is no additional fully connected layer linking flatten layers from two identical ResNet-50. The output of *Baseline*2 is the Euclidean distance between two flatten layers. We introduce a threshold to distinguish positive pairs and negative pairs. If the computed Euclidean distance smaller than 0.5, the sample is treated as positive pair, otherwise it is negative pair. Similar to *Baseline*1, the optimizer of *Baseline*2 is also SGD containing the same parameters. *Baseline*2 is trained by contrastive loss function. Furthermore, we set margin parameter m in contrastive loss as 1 according to previous work [Qi et al., 2018]. Different from *Baseline*1, the evaluation matrix on validation set is a symbolic function by which the accuracy is computed with defined threshold.

7.2 Confusion Matrix Evaluation Method

Our evaluation method for all experiments is based on the results represented in confusion matrix. The idea of confusion matrix is to explicitly illustrate the number of correct predictions and incorrect predictions for each class. In our binary classification case, the confusion matrix is a 2×2 table with four outcomes, namely true positives (TP), true negatives (TN), false positives (FP) and false negative (FN). TP and TN denote that the samples with positive (negative) label are correctly classified as positive (negative) predictions, while FP and FN imply the incorrect positive and negative predictions. Furthermore, we adopt precision, recall and F1 score which are 3 accuracy measures. The equations for 3 measures are shown bellow:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7.2}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.3}$$

For all experiments, we enumerate positive accuracy, negative accuracy, average accuracy, precision, recall, F1 score and training time for one epoch, total 7 indexes to evaluate testing result.

7.3 Fine-Tuning Experiment

A crucial requirement of our SCNN is to generate disseminative representation for each pedestrians. In order to generate discriminative representation and reduce training time simultaneously, we apply ResNet-50 as our basic feature extractor which is pre-trained on ImgNet by He et al. [2016]. The hierarchical features generated by CNN has a property that the early-layer features are applicable to many tasks and the later-layer features are more specific to a particular task. In this case, seeking a strategy to fine-tune ResNet-50 so that it could extract features which are more suitable for our person Re-Id application is our goal. Therefore, we design an experiment to evaluate the performances of networks with different number of fine-tuned layers. In addition, in order to show the effect of transfer learning, we also train a network from scratch.

We implement fine-tuning experiment using a randomly selected network $Baseline1(b)$ on KITTI dataset. Because ResNet-50 feature extractor is applied to all of our proposed networks, the network used for experiment is not a significant factor impacting the final result. The aim of this experiment is investigating transfer learning, so that data augmentation and hard samples mining approaches are not considered in this part. The number of generated image pairs on KITTI dataset is 100392 including around 73% training pairs, 7% validation pairs and 20% testing pairs. The weights in the fully connected layers are initialized by Xavier normal distribution [Glorot and Bengio, 2010]. There are total 173 layers in ResNet-50. We start with the original pre-trained ResNet-50 without fine-tuning. Secondly, we fine-tune the last component ($Conv5\_x$ in Table 1) with 3 residual building blocks, namely the last 32 layers are trainable. Then, we increase the number of fine-tuned layers to 94 which contains all blocks in $Conv4\_x$ and $Conv5\_x$. Finally, we carry out an experiment with all layers retrained using given training data and an experiment of training from scratch.

Table 6: Results of Fine-Tuning using $Baseline1(b)$ on KITTI

FT Layer	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
0	88.2	32.6	60.4	88.2	56.7	69.0	00:30:42
32	**93.3**	19.1	56.2	**93.3**	54.0	63.5	00:39:08
94	90.1	41.6	66.2	91.8	59.6	72.2	00:57:28
all	73.6	**71.4**	**72.5**	73.6	**72.0**	**72.7**	01:28:53
all_scratch	73.6	66.0	69.8	73.6	72.0	**72.7**	01:30:52

During training, we set number of epoch to 10 and record the performance of the network on training and validation set based on loss and average accuracy. The testing set is applied to one model which has maximum performance on validation set in term of the average classification

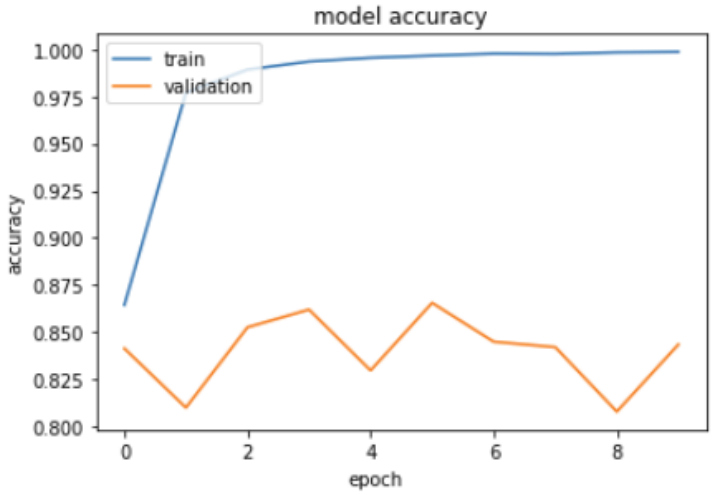


Figure 16: Visualization of Training Process with Loss and Accuracy on Validation Set

accuracy. For instance, in the Figure 16 the model of epoch 5 has maximum accuracy on validation set is selected for testing. The final results of fine-tuning experiment on testing set is shown in Table 6. We observed three phenomena: (1) As the number of fine-tuned layers increases, the average accuracy drops first and then rises. (2) When the weights of later-layer are fine-tuned, the most of testing samples are classified as positive. (3) Transfer learning makes a positive effect, particularly in fine-tuning the whole ResNet-50.

When the pre-trained ResNet directly applies to $Baseline1(b)$ by fixing all parameters, the average accuracy is 60.4% with positive accuracy of 88.2% and negative accuracy 32.6%. The worst result is reported when fine-tuning last 32 layers of ResNet-50, this network performs very poor because the difference between positive and negative accuracy is 74.2%. Additionally, network trained from scratch reaches an average accuracy of 69.8%. Indeed, the best result is achieved when fine-tuning all layers of ResNet-50. Compared with training from scratch, transfer learning improves the average accuracy by 2.7%. Furthermore, network of all-layer fine-tuned network not only has maximum performance but also has similar accuracy for postive and negative.

In conclusion, on the one hand transfer learning based on ResNet-50 has a certain positive effect on our networks. On the other hand, the weights provided by He et al. [2016] is not completely applicable to our person Re-Id task. The original ResNet-50 is trained on ImgNet dataset for object classification, while our task refers to re-identification of pedestrians on tracking dataset.

7.4 Data Augmentation Experiment

Data augmentation is a way to generate more data for training. Our data augmentation strategy mainly focus on increase the number of positive samples by adding some geometric transformations. While the producible positive samples is greatly restricted by the number of image for one identities, the possible combinations of negative samples are huge, which make training samples become imbalanced. For each positive pair in training set, we generate an additional augmented pair and add it to training set. The ratio between positive and negative samples is 1 to 1 in training set.

We design two contrast experiments on the two datasets, KITTI dataset and Market-1501 dataset. There are 73882 training pairs, 7816 validation pairs and 18694 testing pairs in KITTI dataset, while in the case of data augmentation the training pairs is increased twice. For Market-1501 dataset the number of generated training pair is 20104, validation pair is 4266 and testing pair is 5236, the training set contains 40208 image pairs when we use data augmentation. Our used network is $Baseline1(b)$, because this experiment only explores the impact of data augmentation without consideration of which network is selected. All parameters in ResNet-50 architecture are trainable and the parameters in fully connected layers are initialized by Xavier normal distribution. The number of epoch is fixed to 10 for all networks.

Similar to fine-tuning experiment, we select the best model of each network to carry out evaluation. The results of data augmentation experiment is shown in Figure 7. In contrast experiment conducted on KITTI dataset, the average accuracy is enhanced from 72.5% to 73.8%. The results of Market-1501 dataset illustrates that data augmentation improves the performance of network substantially. Not only the average accuracy is increased from 91.4% to 94.6%, but also the gap between positive and negative accuracy is smaller. Obviously, the networks trained by either KITTI dataset or Market-1501 dataset with data augmentation deliver better results. However, as the size of training set becomes larger, computational complexity and the training time also grows.

Table 7: Results of Data Augmentation Generated by $Baseline1(b)$ on Market-1501 and KITTI

Dataset	Augmentation	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
KITTI	on	**75.0**	**72.7**	**73.8**	**75**	**73.3**	**74.1**	2:50:30
KITTI	off	73.6	71.4	72.5	73.5	72.0	72.7	1:28:53
Market-1501	on	92.7	**96.5**	**94.6**	**92.7**	**96.3**	**95.2**	0:42:06
Market-1501	off	**97.6**	85.3	91.4	92.5	86.9	89.6	0:21:14

7.5 Hard Samples Mining Experiment

Generally, the weakness of network is also explicitly represented in poor classification accuracy on hard samples. Therefore, we expect that the performance of our network could be improved by retraining on the hard samples which in general leads to large loss. In person Re-Id task, the hard samples are defined that the image pair is look-similar but belong to two different pedestrians, and the image pair represent the same person but look widely different.

We apply *Baseline*2 architecture for hard samples mining experiment. Because in the previous work, the performance of *Baseline*2 is poorer than *Baseline*1, in particular, negative samples are difficult to be classified correctly. We expect that hard samples mining could improve its performance. In piratical, hard samples is selected by comparing result between computed Euclidean distance and a defined threshold. Because Euclidean distance generated by *Baseline*2 has direct relationship of loss. Our experiment is implemented on Market-1501 dataset in which pedestrian images are captured by 6 cameras with different view points, hence it can generate more hard samples compared to tracking dataset. The total training samples are 20,104, validation samples are 4,266 and testing samples are 5236. We investigate two mining methods hardest mining and moderate mining, and compare their results.

The basic idea of hardest mining sampling method is to rank computed Euclidean distance and collect the top number of ranked samples. First of all, we train a base *Baseline*2 architecture with 10 epochs and select a model with the maximum average accuracy on validation set. Afterwards we apply this best model to predict all training samples. Then all predictions are ranked according to computed Euclidean distance in two groups separately. Specifically, positive image pairs is ranked in descending order, while negative image pairs is ranked in ascending order. The top samples in positive group are image pairs with largest Euclidean distance, in negative group are image pairs with smallest value. A proportion parameter p is introduced to control how many percentage of ranked training samples are selected as hard samples. Furthermore, the best model in the first phase is retrained on hard samples for 10 epochs.

The results of hardest mining experiment with parameter p of 15%, 30% and 50% are represented in Table 8. Figure 17 depicts the distribution of positive predictions and negative predictions related to networks in Table 8, in which a histogram and a kernel density estimate (KDE) are illustrated based on Euclidean distance. The left distribution in the left side with blue color is for positive results, while the distribution with orange color is for negative results. It can be seen that the overlapping area denotes incorrectly classified samples. In the view of average accuracy, the original network reaches at 83.5%, while the other networks which adopted hardest mining are 67.1%, 77.5% and 81.4% respectively. By comparing these results to the retrained

networks, we observe that the average accuracy drops significantly as retraining top 15% and 30% samples and reduces a little as retraining with top 50% samples. From the perspective of class accuracy, the original network performs positive accuracy of 89.6% and negative accuracy of 77.4%, the difference between them is 12.2%. This difference in retrained networks is only 2.4%. But F1 score of retrained networks is not improved. In consequence, hardest mining is not an efficient method to improve performances of networks.

Table 8: Results of Hardest Samples Mining using *Baseline*2 with p 15, 30 and 50 on Market-15010

proportion p	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
Original	**89.6**	77.4	**83.5**	**89.6**	79.9	**84.5**	00:20:33
15	66.4	67.3	67.1	67.4	67.3	67.3	00:04:28
30	76.1	78.9	77.5	76.1	78.3	77.2	00:05:38
50	80.2	**82.6**	81.4	80.1	**82.1**	81.1	00:09:41

This happens because the outliers in the data are often selected in hardest mining, so that the network is weakened to handle "normal" case. Therefore, the improved mining method is to select the positive image pair with maximum Euclidean distance and the negative image pair with minimum Euclidean distance in mini batch. Parameter to determine the number of collected moderate samples is the size of batch. Different from hardest mining, we shuffle the predictions and generate mini batches, the top-ranked sample in mini batch is selected for retraining.

The results of moderate mining experiment with batch size of 2, 3 and 7 are shown in the Table 9 and their distributions are displayed in Figure 18. Similar to hardest mining, the gap between positive and negative accuracy is roughly eliminated. Furthermore, the average accuracy also reduces at the first and then achieve at a result that is very close the original accuracy. Summarily, hard samples mining has a positive effect on improvement of our SCNNs on Market-1501 dataset. Particularly, moderate mining performs better than hardest mining, since the average accuracy of network retrained on moderate data increases by 2.6% compared to the network retrained on hardest data.

7.6 Experiments and Results on KITTI

According to the conclusions from Section 7.3, Section 7.4 and Section 7.5, our training strategy on KITTI dataset contains:

- fine-tuning all layers of pre-trained ReNet-50
- data augmentation

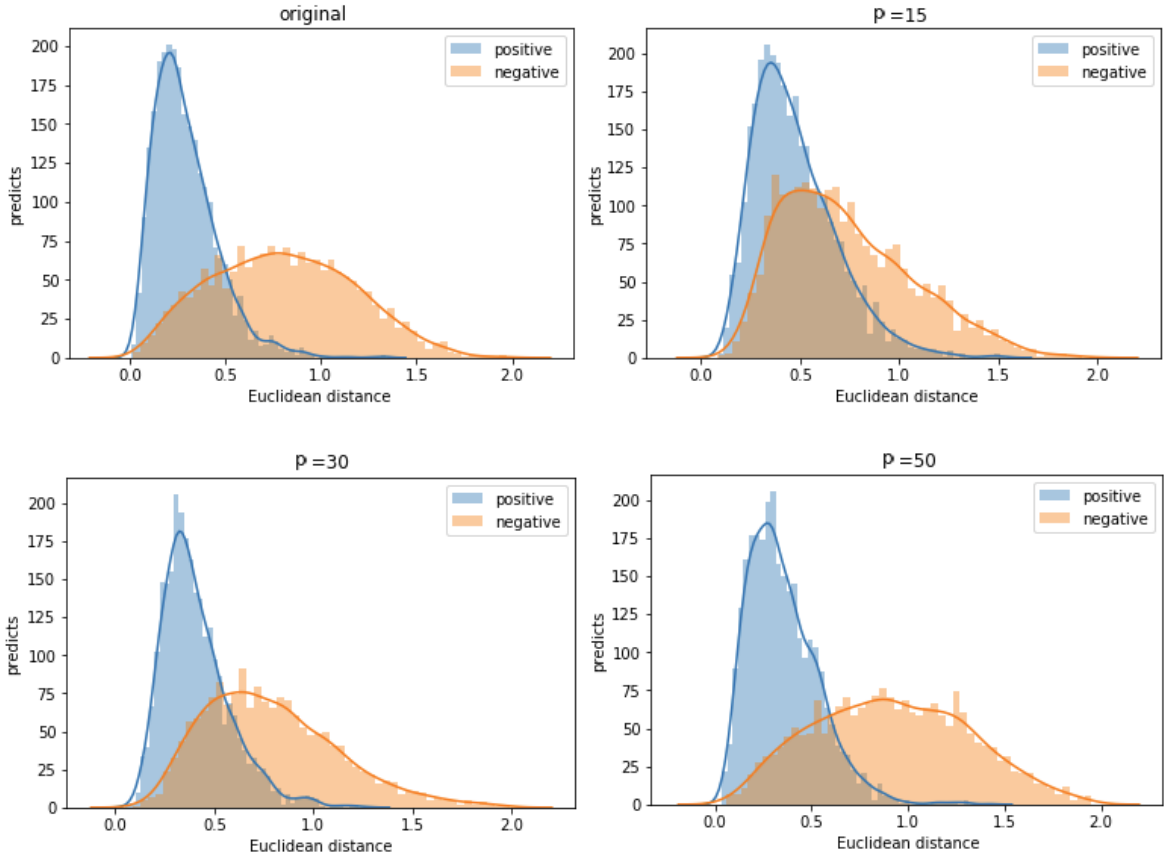


Figure 17: Prediction Distributions for Hardest Mining Experiments using *Baseline*2 with P 15, 30 and 50 on Market-1501

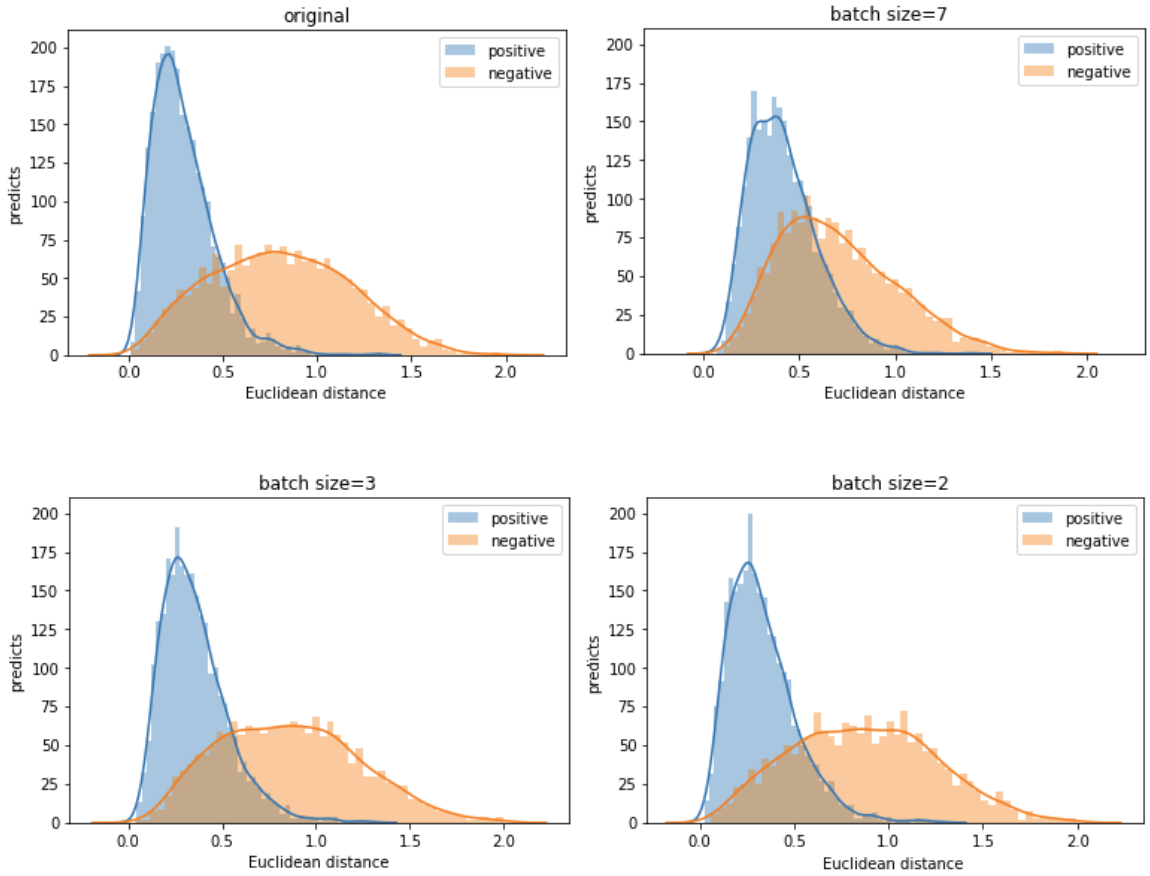


Figure 18: Prediction Distributions for Moderate Mining Experiments using *Baseline*2 with batch size 2, 3, 7 on Market-1501

Table 9: Results of Moderate Samples Mining using *Baseline*2 on Market-1501 with Batch Size 2, 3, 7

Batch Size	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
Original	**89.6**	77.4	**83.5**	**89.6**	79.9	**84.5**	00:20:33
7	64.2	69.5	66.9	66.9	67.8	67.3	00:04:28
3	74.3	**78.7**	76.5	74.3	77.8	77.2	00:06:24
2	84.5	81.6	83.0	84.5	**82.1**	83.3	00:09:48

- hardest mining with percentage 50 on *Baseline*2
- moderate mining with batch size 2 on *Baseline*2

With the help of data augmentation, the number of available image pairs in training set is increased from 73882 to 147764. Training data are in balanced on positive and negative samples. The validation set has 7816 image pairs and testing set has 18694 image pairs. The identities used for matching image pairs are in non-coincidence among training, validation and testing set. As the results of hard samples mining experiment shown, we can not obtain expected result with hard samples mining approach, considerinf of time cost for one experiment, we decide to only apply this approach for *Baseline*2. There are total 6 networks trained in our experiment, they are 3 variants of *Baseline*1, *Baseline*2 and two retrained networks based on *Baseline*2 using hardest mining and moderate mining approaches. From a practical point of view, we control training time for one network in roughly 24 hours on our 3GB GPU by setting epoch number as 10. The results derived from one model with the best performance on validation set are shown in Table 10.

It costs around 2 hours and 50 minutes for one epoch when training *Baseline*1 and *Baseline*2, while retraining *Baseline*2 only needs 1 hour and 20 minutes due to only 50% samples mined as hard samples. In comparison of two baselines, the performance of *Baseline*1 is better than *Baseline*2. Especially *Baseline*1(b) which is a network computing similarity with element-wise L_1 distance matrix achieves at average accuracy of 73.8% and F1 score of 74.1%.

7.7 Experiments and Results on MOTChallenge

The performance of networks training on MOTChallenge is also evaluated based confusion matrix. The training strategy on MOTChallenge follows:

- fine-tuning all layers of pre-trained ReNet-50
- without data augmentation

Table 10: Results on KITTI

Network	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
Baseline1(a)	68.0	72.3	70.2	68.0	71.1	69.5	02:49:51
Baseline1(b)	**75.0**	**72.7**	**73.6**	**75.0**	**73.3**	**74.1**	02:50:30
Baseline1(c)	73.8	67.5	70,7	73.8	69.4	71.5	02:52:07
Baseline2	63.1	75.0	69.0	63.1	71.6	67.1	02:47:58
Baseline2_hardest	62.5	65.9	64.1	62.5	64.7	63.6	01:24:04
Baseline2_moderate	64.0	70.1	67.4	64.0	68.7	66.2	01:23:41

- hardest mining with percentage 50 on $Baseline2$
- moderate mining with batch size 2 on $Baseline2$

The strategy is similar to the experiment on KITTI except for data augmentation approach. The reason is that even without the help of augmentation the number of training image pairs is also sufficient to approximate parameters in SCNNs. As a result, for reducing computational cost data augmentation is not considered. In order to generate comparable results, we select MOT16-2 and MOT16-11 to generate testing image pairs since they are also used in [Tang et al., 2017]. In this experiment, the amount of training samples achieves at 168,682, validation set contains 18,462 image pairs and testing set has 52,788 image pairs. The number of epoch for all 6 networks is set as 6 and the convergence time is approximate 20 hours. The evaluation results based on confusion matrix are shown in Table 11.

For each epoch, it costs about 3 hours and 20 minutes. When retraining $Baseline2$ using hard samples, it takes about one and half hours. The networks in $Baseline1$ architecture have better results compared to $Baseline2$. The best results are generated by $Baseline1(c)$ and $Baseline1(b)$, the accuracy achieve at 82.4%. hard samples mining still only acts on reducing the gap between positive accuracy and negative accuracy in $Baseline2$ without significant improvement on average accurancy and F1 score.

Table 12 shows the results obtained from networks designed by Tang et al. [2017] for person Re-Id. Our networks designed in $Baseline1$ architecture performs better than ID-Net. Particularly, the accuracy of the best networks $Baseline1(b)$ and $Baseline1(c)$ achieve at 82.4% and increase by 2% comparing with $ID-Net$. The results show that our SCNNs can be applied to solve the problem of person Re-Id for tracking.

Table 11: Results on MOTChallenge

Network	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
Baseline1(a)	82.3	80.1	81.6	82.3	**81.2**	81.7	03:18:54
Baseline1(b)	84.2	**80.5**	**82.4**	84.2	**81.2**	82.7	03:17:21
Baseline1(c)	**85.7**	79.2	**82.4**	**85.7**	80.5	**83.0**	03:19:05
Baseline2	64.9	76.5	70.7	64.9	73.4	68.9	03:16:52
Baseline2_hardest	67.1	71.5	69.3	67.1	70.2	68.6	01:35:22
Baseline2_moderate	68.1	70.3	69.2	68.1	69.6	68.8	01:36:08

Table 12: Comparison with Existing Person Re-Id Models on MOTChallenge

Network	Accuracy[%]
Baseline1(b)	82.4
Baseline1(c)	82.4
ID-Net [Tang et al., 2017]	80.4
SiameseNet [Tang et al., 2017]	84.7
StackNet [Tang et al., 2017]	86.9
StackNetPose [Tang et al., 2017]	**90.0**

7.8 Experiments and Results on Market-1501

7.8.1 Evaluation Method

For generating comparable results, we introduce Market-1501 dataset which is a dataset widely used in traditional person Re-Id task. Different from person Re-Id for tracking, person Re-Id using single-shot dataset is treated as a person retrieval task which is generally evaluated in "query-searching" standard evaluation method. Each query image has to be matched with all the gallery images according to similarity score between two images. In the experiment, we employ Euclidean distance computed between feature vectors as similarity score. We assume that there are total M query images and P gallery images. Then we compute corresponding Euclidean distance for all possible combinations of image pairs and sort all MP image pairs in ascending order for finding the top matches. That is the basic idea about rank-K-accuracy measurement. In our case, we report rank-1 and rank-10 for Market-1501 dataset and compare with state-of-the-art models. Rank-1 indicates according to ranked Euclidean distance the query image and its top match represent one person, while rank-10 means there exists at least on positive pair in top 10 matches.

Figure 19 shows an example of rank-1 result, where Euclidean distance between query image and returned images is represented on above. It is obvious that the top match is a positive pair.

Figure 19: Visualization of Rank-1 Results based on Euclidean Distance with $Baseline1(b)$ on Market-1501

Figure 20: Visualization of Rank-10 Results based on Euclidean Distance with $Baseline1(b)$ on Market-1501

Figure 20 depicts that the image with minimum Euclidean distance is not matched query image, but it is satisfied with the condition of rank-10.

Additionally, confusion matrix evaluation method is also applied on Marker-1501 dataset for comparing the performance of networks trained on image sequences and single-shot.

7.8.2 Results

The training strategy on Market-1501 dataset has following steps:

- fine-tuning all layers of pre-trained ResNet-50
- data augmentation
- hardest mining with percentage 50
- moderate mining with batch size 2

For evaluating intermediate stage of training, we split original training data into two parts, 80% identities apply to training SCNN models and the rest 20% identities for validation set. With the help of data augmentation, there are 40208 image pairs used for training SCNN. Besides 4266 image pairs are used for validating network performance after each epoch. Because pedestrians appear in testing set are also contained in query set, for confusion matrix evaluation, our SCNN models are only tested on 5,236 image pairs generated from query set. For alternative evaluation

strategy, we generate feature vectors for all query and gallery images using our trained ResNet-50 and compute their similarity applying Euclidean distance. Because the size of training set is smaller than the other dataset, we set 15 epoches for each network, completion of training needs roughly 10 hours.

Table 13 illustrates that the performances of all networks training on Market-1501 are better than KITTI and MOTChallenge datasets. Indeed, $Baseline1(b)$ which merges flatten layers with L_1 distance results in the best performance with 94.6% accuracy and 94.5% F1 score. The performances of networks constructed in $Baseline2$ architecture are still worse than $Baseline1$, but compared with KITTI and MOTChallenge dataset, the average accuracy increases to 86.9%. Furthermore, different from above, the accuracy of retrained $Baseline2$ applying moderate mining approach is boosted from 86.9% to 89.2%, and the F1 score also increases by 2.1%. One possible reason is that the image quality is better than tracking datasets. Thus, it is with less difficulty to generate distriminative feature for images.

Table 13: Results on Market-1501 with Confusion Matrix Evaluation Method

Network	Pos. Acc.[%]	Neg. Acc.[%]	Ave. Acc.[%]	Precision[%]	Recall[%]	F1[%]	Time per Epoch
Baseline1(a)	91.5	94.9	93.2	91.5	94.7	93.1	00:42:17
Baseline1(b)	92.7	**96.5**	**94.6**	92.7	**96.3**	**94.5**	00:46:47
Baseline1(c)	**96.6**	91.3	93.9	**96.6**	91.8	94.1	00:47:06
Baseline2	88.0	85.8	86.9	88.0	86.1	87.0	00:46:39
Baseline2_hardest	80.1	83.3	86.3	80.1	84.3	82.0	00:22:24
Baseline2_moderate	87.9	90.5	89.2	87.9	90.2	89.1	00:20:10

Table 14 shows the comparison with state-of-the-art algorithms. By observation of Rank-10, $Baseline1(c)$ achieves the best result at Rank-10 of 53.4%, the second place is $Baseline1(b)$ with Rank-10 of 52.9%. When a specific pedestrian is retrieved from gallery set, our networks has ability to find at least one positive sample in the first 10 return images. However, as matching precision requirement raises to Rank-1, namely the query image and its first retrieved result from the whole gallery set must positive pair, our networks performs not well. Our maximum performance is generated by $Baseline1(b)$ with Rank-1 of 24.8%, the next is $baseline(c)$ with Rank-1 of 24.4%, while *Spindle* [Zhao et al., 2017] trends to 76.9% at Rank-1 and 94.6% at Rank-10. There is a large gap between our proposed network and the state-of-the-art models.

The essential reason rendering this poor result is the shortage of negative samples in training. For each query image, network needs compare it with all gallery images and rank it according to similarity. Specifically, the number of query image in Market-1501 is 3,368 and gallery image is 19,732, thereby the possible combination between query images and gallery images is in the

level of billion. If we expect our network has a strong generalization capability to match each query image, the number of negative image pair used for training must achieve at the same level of possible combination in testing. However, our training set only contains 40,208 image pairs, compared with the required quantity only a few combinations are applied for training. Summarily, the parameters approximated by small size data is not suitable for the retrieval task, because it is difficult to obtain the optimal matching for 750 different identities. In the further work, we would propose an online hard negative mining approach to improve the performance of our networks. The main idea is to mine the samples with high value and ignore most of "normal" samples, thereby training network more efficiently.

Table 14: Comparison with state-of-the-art on Market-1501

Network	Rank-1[%]	Rank-10[%]	Network	Rank-1[%]	Rank-10[%]
Baseline1(a)	22.7	51.7	Triplet-CNN[Liu et al., 2016]	45.1	78.4
Baseline1(b)	**24.8**	52.9	SCSP[Chen et al., 2016]	51.9	-
Baseline1(c)	24.4	**53.4**	Histogram[Ustinova and Lempitsky, 2016]	59.5	86.94
Baseline2	13.9	41.9	LSTM-Siamese[Varior et al., 2016b]	61.6	-
Baseline2_hardest	15.0	40.3	Gate-Siamese [Varior et al., 2016a]	65.9	-
Baseline2_moderate	16.5	45.4	Spindle [Zhao et al., 2017]	**76.9**	**94.6**

7.9 Discussion

Our experiments are consisted of two major parts. The first part focuses on training approaches related to fine-tuning, data augmentation and hard samples mining. The aim of these experiments is to seek the best configuration so that the networks could produce better results. The fine-tuning experiment shows that the best setup is fine-tuning all layers of ResNet-50. Furthermore, network training using pre-trained model has better performance compared with training from scratch. The experiment of data augmentation provides a point that it is a useful method to improve feasibility of networks performing on small size datasets. Our online data augmentation approach is implemented on positive samples, with the increase number of positive pairs more negative pairs are generated at the same time. The rate of positive and negative pairs is 1 to 1 in training set. Hard samples mining is an approach to retrain *Baseline*2 so that the distributions of negative and positive predictions are reconstructed resulting in more balanced accuracy rate. The best performance is conducted with 50% in hardest mining method and with batch size 2 in moderate mining method.

In the second part, according to above conclusions we train 6 networks on tracking dataset KITTI, MOTChallenge and single-shot dataset Market-1501. The training strategy is fine-tuning

Figure 21: Visualization of False Positive Predictions with *Baseline*1(*a*) on KITTI-16

all layers in ResNet-50, applying data augmentation except for MOTChallenge and retraining *Baseline*2 with hard samples mining.

From the prospective of datasets, our networks outperform on Market-1501, the best result achieves at average accuracy of 94.6% and F1 score of 94.5%. For tracking datasets, the best accuracy achieves at 82.4% and 73.8% on MOTChallengae and KITTI respectively. These results demonstrates that in the application of person matching our networks on single-shot dataset have better performance than on tracking datasets. By observation of incorrect predictions in tracking dataset, we found the networks is hard to make a correct decision in the case of overlapping, which is also the big difference between Market-1501 and tracking datasets. Figure 21 gives some typical examples of overlapping, as ground truth provided, the pedestrians in two images represent an identical person, but our network predicts it as negative sample. Because similarity between two images is computed based on optical features, occlusion is an extremely negative factor impacting result. It is worth mentioning that our SCNNs have a powerful capability to extract features in low resolution. As Figure 22 shown, the maximum image size in two pairs is 39×78 and the minimum size is 17×41. In this case our network still can provide a good performance.

From the perspective of our proposed SCNNs, by comparing of 3 variants of *Baseline*1, *Baseline*1(*b*) and *Baseline*1(*c*) can generate better results. *Baseline*1(*a*) is a simply binary classification network by concatenating two flatten layers, *Baseline*1 and *Baseline*1(*b*) contain a difference layer computed by element-wise L_1 and element-wise cosine distance respectively. The experiment results figure out for *Baseline*1 the networks with the help of distance computation are more suitable for our person Re-Id application.

In comparison of *Baseline*1 and *Baseline*2, the results show that the networks designed in *Baseline*1 architecture have better performances than *Baseline*2. The classifiers in *Baseline*1 are designed with fully connected layers, in contrast, samples is classified in *Baseline*2 only

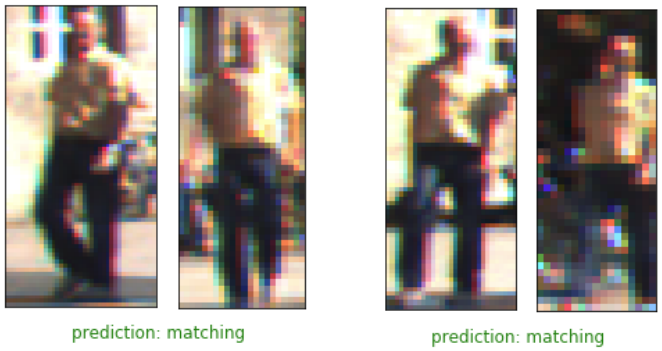


Figure 22: Visualization of True Positive Predictions with $Baseline1(a)$ on KITTI-16

relying on the relationship between a certain Euclidean distance and predefined threshold. We assume that fully connected layers following distance layer take positive effect on improving the classification accuracy, because the added parameters in these layers can learn more information about difference.

8 Conclusion and Outlook

In this thesis, we design two different SCNN networks to solve the problem of appearance-based person re-identification (Re-Id) for tracking. Our SCNN-based networks are implemented by combination ResNet-50 feature extractor with different similarity measurement methods. $Baseline1$ contains 3 networks, in which the problem of person matching is treated as binary classification task. Concretely, $Baseline1(a)$ concatenates flatten layers from two identical ResNet-50 and estimates probability in the last fully connected layer. $Baseline1(b)$ and $Baseline1(c)$ combine feature vectors by element-wise L_1 and element-wise cosine distance respectively. On the other hand, in the $Baseline2$, we predict the person Re-Id using Euclidean distance between extracted feature vectors.

We conduct experiments on two tracking datasets KITTI and MOTChanllenge. In addition, for generating more comparable results we also experiment our SCNNs on the single-shot dataset Market-1501. Our experiments consisted of two major parts. While the experiments in the first parts focus on training approaches related to fine-tuning, data augmentation and hard samples mining to improve the performance of networks. Experimental results show that the best transfer learning strategy for our networks is fine-tuning all layers of ResNet-50. Furthermore, with the help of data augmentation performances of networks are improved by 3.2% on Market-1501 dataset and 1.3% on KITTI dataset. But hard samples mining hardly conducts improvement for $Baseline2$. The experiments in second part refer to evaluate the performance of our proposed networks. Our best result produced by $Baseline1(b)$ achieves at overall accuracy of 73.6% and F1 score 74.1% on KITTI dataset. $Baseline1(b)$ and $Baseline1(c)$ have the best performance with overall accuracy of 82.4%. For Market-1501, $Baseline1(b)$ outperforms the other networks in binary classification task with accuracy of 96.5%, when we use our networks for person retrieval task, the best result at Rank-1 and at Rank-10 are 24.8% and 53.4%.

The experiment results show that our SCNNs perform well on person re-identification task on tracking data even pedestrians image captured in the conditions of unstable illumination and low resolution. But overlapping is still a big challenge in our case, because our SCNNs only consider of appearance feature to determine the similarity between two image whether belong to one person. As a person is occluded by the other person or targets, our networks would be hard

to generate discriminative representation. Furthermore, the performance of our $Baseline2$ is poorer than the results conducted by $Baseline1$. A possible reason is the existence of is training sample imbalance in mini batch. Batch size in our experiment is fixed to 5, because we suppose to minimize convergence time using maximum batch size. In the further study, we could set batch size as an even number and keep the same number of positive and negative samples in batches, it could help to improve the performance of $Baseline2$. Additionally, We also observe that our SCNNs can not achieve a satisfactory results in "query-searching" evaluation method. To improve this result, online hard negative samples strategy could be applied to increase training data. It would reduce the work of training "normal" image pairs and focus on learning hard samples. Finally, though hard mining method does not help to significantly enhance our $Baseline1(b)$, we gave up experiment of hard samples mining using $Baseline1$ due to the limitation of time, and this should be carried out in future.

Bibliography

Ejaz Ahmed, Michael Jones, and Tim K Marks. An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3908–3916. IEEE Computer Society, 2015.

Xiang Bai, Mingkun Yang, Tengteng Huang, Zhiyong Dou, Rui Yu, and Yongchao Xu. Deep-person: Learning discriminative deep features for person re-identification. *arXiv preprint arXiv:1711.10658*, abs/1711.10658, 2017.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, abs/1306.6709, 2013. URL `http://arxiv.org/abs/1306.6709`.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *IJPRAI*, 7:737–744, 1993.

Dapeng Chen, Zejian Yuan, Badong Chen, and Nanning Zheng. Similarity learning with spatial constraints for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1268–1277. IEEE Computer Society, 2016.

De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, pages 1335–1344. IEEE Computer Society, 2016.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005.

Matthew L Dering and Conrad S Tucker. A convolutional neural network model for predicting a product's function, given its form. *Journal of Mechanical Design*, 139(11):111408, 2017.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *I. J. Robotics Res.*, 32:1231–1237, 2013.

Niloofar Gheissari, Thomas B Sebastian, and Richard Hartley. Person reidentification using spatiotemporal appearance. In *null*, pages 1528–1535. IEEE, 2006.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATSâ10)*, pages 249–256, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *ECCV*, volume 5302, pages 262–275. Springer, 2008.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR (2)*, volume 2, pages 1735–1742, 2006.

Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural Network Design*, volume 20. Pws Pub. Boston, 1996.

Kaiming He. Deep residual networks, 2016. URL `https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf`. Proceedings of ICML conference.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pages 448–456, 2015.

Tobias Klinger. *Probabilistic Multi-person Localisation and Tracking*. PhD thesis, Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover, 2016.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015.

Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *CVPR Workshops*, pages 33–40. IEEE Computer Society, 2016.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Feifei Li and Karpathy Andrej. Transfer learning and fine-tuning convolutional neural network, 2015. URL `http://cs231n.github.io/transfer-learning/`.

Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, pages 152–159. IEEE Computer Society, 2014.

Lijing Lin. Towards better analysis of deep convolutional neural networks, 2016. URL `http://vis.pku.edu.cn/blog`.

Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. End-to-end comparative attention networks for person re-identification. *IEEE Trans. Image Processing*, 26(7):3492–3506, 2017.

Jiawei Liu, Zheng-Jun Zha, QI Tian, Dong Liu, Ting Yao, Qiang Ling, and Tao Mei. Multi-scale triplet cnn for person re-identification. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 192–196. ACM, 2016.

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

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

Lei Qi, Jing Huo, Lei Wang, Yinghuan Shi, and Yang Gao. Maskreid: A mask based deep ranking neural network for person re-identification. *CoRR*, abs/1804.03864, 2018. URL `http://arxiv.org/abs/1804.03864`.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

R. Shanmugamani. *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*, volume 20. Packt Publishing, 2018. ISBN 9781788295628.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, abs/1409.1556, 2014.

Johannes Skog. Re-identification with recurrent neural networks. Master's thesis, Lund University, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE*, pages 1–9, 2015.

Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person reidentification. In *IEEE*, pages 3539–3548. IEEE Computer Society, 2017.

Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016.

Rahul Rama Varior, Mrinal Haloi, and Gang Wang. Gated siamese convolutional neural network architecture for human re-identification. In *ECCV*, volume 9912, pages 791–808. Springer, 2016a.

Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, and Gang Wang. A siamese long short-term memory architecture for human re-identification. In *ECCV*, pages 135–153. Springer, 2016b.

Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006.

Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *ICPR*, pages 34–39. IEEE Computer Society, 2014.

Haiyu Zhao, Maoqing Tian, Shuyang Sun, Jing Shao, Junjie Yan, Shuai Yi, Xiaogang Wang, and Xiaoou Tang. Spindle net: Person re-identification with human body region guided feature decomposition and fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1077–1085. IEEE Computer Society, 2017.

Rui Zhao, Wanli Ouyang, and Xiaogang Wang. Unsupervised salience learning for person re-identification. In *CVPR*, pages 3586–3593, 2013.

Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *IEEE*, pages 1116–1124. IEEE Computer Society, 2015.

Liang Zheng, Yi Yang, and Alexander G Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, abs/1610.02984, 2016.