Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Identity tracking in multi-view camera surveillance system

Kateřina Jandová

Supervisor: Ing. Jan Krček
Field of study: Open informatics
Subfield: Artificial intelligence
May 2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jandová** Jméno: **Kateřina** Osobní číslo: **406424**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sledování objektů mezi vícero scénami se zachováním jejich indentity

Název diplomové práce anglicky:

Identity tracking in multi-view camera surveillance system

Pokyny pro vypracování:

Lately there is a trend of growing demand for complex street surveillance camera systems. Such systems attempt to solve problems of detection, tracking and re-identification of objects. (1) Research the real-world problem of single object identity tracking, by finding its most likely match throughout cameras of the system. It is recommended to make use of additional spatio-temporal constrains, as well as the visual image data. (2) Suggest and implement method which finds top-n likeliest matches for a given object throughout the system. Use Python language with OpenCV 3 for the module development. (3) Evaluate and discuss the results on GoodVision real data as well as benchmarks DukeMCMT [4] (used to evaluate state of the art approaches in MOT challenge) and VehicleReId [5]. (4) Finally suggest possible improvements and fields for future work.

Seznam doporučené literatury:

[1] O. Javed, Z. Rasheed, K. Shafique and M. Shah: Tracking Across Multiple Cameras With Disjoint Views. University of Central Florida, 2003 IEEE.
[2] LiangJia Zhu, Jenq-Neng Hwang, Hsu-Yung Cheng: Tracking of multiple objects across multiple cameras with overlapping and non-overlapping views. Circuits and Systems, 2009. ISCAS 2009. IEEE.
[3] Erik Bochinski, Volker Eiselein and Thomas Sikora: High-Speed Tracking-by-DetectionWithout Using Image Information. Communication System Group, Technische Universitat Berlin, 2017 IEEE.
[4] Multi-Target Tracking in Multiple Non-Overlapping Cameras using Constrained Dominant Sets. Yonatan Tariku Tesfaye, Eyasu Zemene, Andrea Prati, Marcello Pelillo, Mubarak Shah. http://vision.cs.duke.edu/DukeMTMC/, arXiv:1706.06196
[5] VehicleReId - Vehicle Re-Identification for Automatic Video Traffic Surveillance [ATS-CVPR 2016], https://medusa.fit.vutbr.cz/traffic/research-topics/detection-of-vehicles-and-datasets/vehicle-re-identification-for-automatic-video-traffic-surveillance-ats-cvpr-2016/

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Krček, GoodVision s.r.o.

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **10.01.2018** Termín odevzdání diplomové práce: ____________

Platnost zadání diplomové práce: **30.09.2019**

Acknowledgements

I wish to express my sincere thanks to my supervisors in the UK, Dr. Hubert P. H. Shum, Edmond S. L. HO, and my supervisor Ing. Jan Krček, for providing all valuable advice and for their guidance. I would also like to thank my family for their support.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic nal thesis.

In Prague, 20. May 2018

Abstract

Person and vehicle re-identification (re-ID) are important challenges for the analysis of the burgeoning collection of urban surveillance videos. As this data is usually populated with both vehicles and pedestrians, it would be preferable to unify the tasks under one framework. Due to the contrasting composition of humans and vehicles, vehicle re-identification typically requires a more advanced network that is robust to drastic shape changes as the viewing angle is altered. Motivated by the success of the triplet loss function in both domains, we propose a re-ID framework which does not require any specialised design for the data stream on which it is applied. We create two functions to minimise the intra-class distance and maximise the inter-class distance respectively and combine them with the triplet loss to create our novel loss function. Moreover, we proposed an extended method with the spatio-temporal constraint for multiple camera tracking. The proposed framework is comprehensively evaluated on the Market-1501, CUHK03, DukeMTMC4ReID, VeRI and VehicleReId data sets and outperforms state-of-the-art methods on both tasks.

Keywords: re-identification, person, vehicle, multiple camera tracking, deep learning, cnn, neural networks, resnet, computer vision

Supervisor: Ing. Jan Krček
GoodVision s.r.o.

Abstrakt

Re-identifikace lidí a vozidel je rychle se rozvíjející oblastí pro analýzu rostoucí sbírky videozáznamů městského sledování z důvodu bezpečnosti. Vzhledem k tomu, že na těchto videozáznamech jsou většinou vozidla i chodci, bylo by vhodné sjednotit re-identifikaci chodců a re-identifikaci vozidel pomocí jednoho softwarového systému. Kvůli rozdílnému vnímání lidí a vozidel vyžaduje re-identifikace vozidel obvykle pokročilejší neuronovou síť než re-identifikace osob. To je způsobeno tím, že při změně úhlu pohledu na vozidlo dochází k obrovským změnám tvaru vozidla, a proto neuronová síť musí být dostatečně robustní. Motivováni úspěchem funkce triplet loss v obou doménách navrhujeme architekturu neuronové sítě, která nevyžaduje specializovaný návrh datového toku, na kterém je aplikována. Vytváříme dvě funkce, abychom minimalizovali vzdálenosti uvnitř třídy a maximalizovali vzdálenosti mezi třídami. Následně tyto vzdálenosti spojíme s tripletovou ztrátovou hodnotou a vytvoříme tak novou ztrátovou funkci. Navíc navrhujeme rozšířenou metodu s prostorově časovým omezením pro sledování více kamer. Navrhovaný rámec je komplexně vyhodnocen na datových sadách Market-1501, CUHK03, DukeMTMC4ReID, VeRI a VehicleReId a překonává nejmodernější metody u obou úkolů.

Klíčová slova: re-identifikace, kamera, počítačové vidění, lidé, vozidla, počítačové učení, neuronové sítě

Překlad názvu: Sledování objektů mezi vícero scénami se zachováním jejich indentity

Contents

Chapter 1

Introduction

Recently, there has been a growing interest in object tracking, which itself has raised the need for recognising and identifying different objects monitored by multiple cameras. This is an enormous step forward for advanced analyses of movements of people or cars in a wider area. A direct real-world application could be in the domain of urban surveillance. More specifically, the vehicle traffic of a city's busy street could be analysed and optimised by tracking the cars' movements. This can be accomplished by capturing pictures of cars with a camera and matching them to pictures taken by another camera in a different location. Thus, it is possible to understand the main flow of the vehicles and take traffic-related actions accordingly. Such actions could include traffic lights installation, road maintenance, and traffic jam prevention.

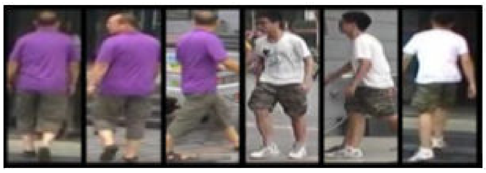
Generally, the observation of objects in multiple cameras requires detecting and, tracking the object, then identifying it again in the next camera. Therefore, assuming the single camera tracking is effective, the main problem of tracking an object across multiple cameras is to estimate the correspondence between tracks of that object when it appears in the new camera. [1] An approach includes establishing the overlapping area and then following the paths, however in many cases the cameras do not have overlapping views. Therefore, it is advantageous to estimate the missing view between cameras or re-identify the object based on visual information. In practice, this is often challenging due to illumination changes, viewpoint variations, and occlusion. This thesis considers all these factors and proposes a powerful solution.

Joned et al. [2] focuse only on the re-identification of people using a convolutional neural network. This approach could potentially be extended with the combination of time-space information of the cameras. This thesis concentrates on the tracking of many objects across multiple cameras. We assume that the input of our framework will consist of single tracks for each camera. Therefore, the task we are dealing with is the re-identification of the objects between the cameras.

The re-identification task (re-ID) is a core challenge for the computer vision community whereby a detection is required to be matched with another detection of the same identity, typically from a different viewpoint. With the increasing volume of large-scale urban surveillance data, re-ID has started to attract a large amount of attention. In the past few years, deep learning techniques have received significantly increased popularity for both pedestrian [3, 4, 5, 6] and vehicle [7, 8, 9] re-ID. However, the two tasks are generally tackled separately.

The challenge of re-identifying a vehicle is severely different from that of re-identifying a person. For wide area video surveillance on humans, the same identity when viewed from a different pose angle will typically look fairly alike. The shape of the detection will remain upright and the colour information, predominantly extracted from articles of clothing, will be of a similar pattern (Figure 1.1(a)). The same condition cannot be satisfied for vehicles. Colour information can become far more distorted in different lighting due to the reflectiveness of the body of a car. The shape information of a car viewed from the front is significantly different than that viewed from a $45°$, or $90°$, angle (Figure 1.1(b)). On the contrary, many high-end vehicle re-ID algorithms attempt to use license plate information [10, 11, 7], which is not applicable in the human domain.

It is customary to split the two tasks and design a network which can specifically target the individual task's respective challenges. However, this is inadequate. In the real world, urban surveillance videos provide a mixed stream of data, consisting of both vehicles and pedestrians, on which analysis is required. Current models are a long way from being able to handle this blended data. Convolutional neural networks (CNNs) with a triplet loss function show some potential in generalizing


(a) : Pedestrian shape and colour information remains somewhat consistent across different viewpoints.

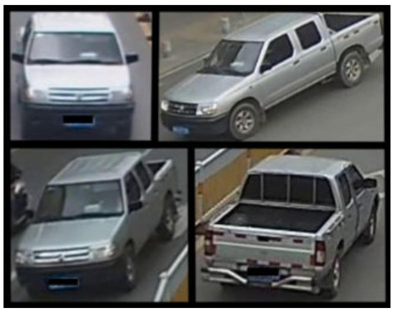

(b) : Vehicle shape and colour information changes drastically across different viewpoints.

Figure 1.1: Person and vehicle re-ID tasks present different challenges.

across the tasks, having been successfully applied to both human [3] and vehicle [12, 13] re-ID.

In this thesis, we investigate an approach unifying the two tasks. We propose a novel loss function which can be trained on either person or vehicle data and achieve state-of-the-art performance on each task. Furthermore, a method with spatio-temporal constraints is proposed. To the best of our knowledge, this is the first proposed architecture which is capable of successfully generalizing across both tasks.

1.1 Goals of the thesis

The goals of the thesis are following:

1. To review state-of-the-art approaches regarding the tracking of an object across multiple cameras. To research the real-world problem of single object identity tracking, through finding its most likely match between footage of different cameras across the system. We focus on researching the real-world problem in the chapter 2.7.2 and on reviewing state-of-the-art approaches in the chapter 3.

2. To suggest methods for solving the task of tracking an arbitrary object across multiple cameras. Usually, the methods could give us a probability estimation of the match. Therefore, the task is to implement the method which finds the top-n likeliest matches for a given object throughout the camera network system. Description of the proposed methods is in the chapter 4.

3. To describe the testing data (GoodVision real data, benchmarks DukeMCMT [14] which are used to evaluate state of the art approaches in MOT challenge and VehicleReId [15]). To evaluate and discuss the results. We illustrate the evaluation in the chapter 5.

4. To suggest possible improvements and fields for future work, which is described in the chapter 6.1.

1.2 Structure of the thesis

This thesis is structured as follows. The technical background will be presented in the second chapter 2.7.2. We explain the tracking and re-identification task as well as all methods which are crucial to our proposed methods. Firstly, we describe the tracking problem and we split it into subproblems: Tracking across multiple cameras with overlapping views, Tracking across multiple cameras with non-overlapping views and re-identifying objects without knowledge about camera views. We explain why re-identifying objects without knowledge about camera views are the most important for us in the section 2.2. Then we focus on different kind of neural networks which we are using as a backbone of our framework. This thesis is focused on loss function and mining the hardest samples for training the neural network, therefore we describe some popular methods for calculating loss function and mining the hardest samples in the next part of technical background. Finally, we describe the evaluation methods which we used in the Evaluation chapter 5.

In the third chapter 4, a literature review of a variety of papers will be given and the research related to our approach will be discussed.

We explain the proposed methods in the fourth chapter 4. Firstly, we define the task with details. Then we describe our proposed methods: loss function with class distance engineering for person and vehicle re-identification, sample mining and evaluation method with taking into account the spatio-temporal constraints. Lastly, we provide implementation details.

The proposed methods are tested and evaluated in the fifth chapter 5. We provide evaluation on many datasets (CUHK03 [16], DukeMTMC4REID [17], Market1501 [18], VeRi [19], and VehicleReId [20]), therefore we demonstrate that our proposed methods are consistent. We evaluate person re-identification as well as vehicle re-identification problem and we also provide evaluation of re-identification based only on visual aspects and also re-identification with spatio-temporal constraints. Our trained network is published to reproduce the results in the link [21].

We summarize our contribution and achievements of the goals in the sixth chapter 6.1.

Chapter 2

Technical background

This chapter offers the overview of methods which were used to solve the tracking across multiple cameras. After describing the methods, we will proceed to analyse how the tracking problem can be solved using a combination of these. The chapter is divided into two main sections: tracking and object re-identification.

2.1 Tracking

Tracking object in videos is the process of locating one or multiple moving objects over time. More precisely, "tracking" in our case refers to detecting the object in each frame and connecting these detections for each object. [22] The final result is called the track of the object. Figure 2.1 depicts one frame from the video with calculated tracks of the cars. Tracking in multiple cameras is the task of connecting tracks from each camera. Videos from more cameras at the same time can be available. Usually, the cameras are close to each other, therefore we can assume if the object is in the first camera, it would also be on another camera. The goal of multiple camera tracking is not only to track the object in every single camera but also to join the tracks taken by different cameras into one track. The joining should only assign the tracks which have the same identification sign. Accordingly, the main part of the tracking in multiple cameras consists of the re-identification of the object. It is worth noting that the multiple cameras trackings with overlapping views indicate the cameras are pointed to the same place. Part of each camera's view contains the same background and the same objects on the foreground, which is called overlapping area or overlapping view. The cameras can be pointed to the same area from different point of views.

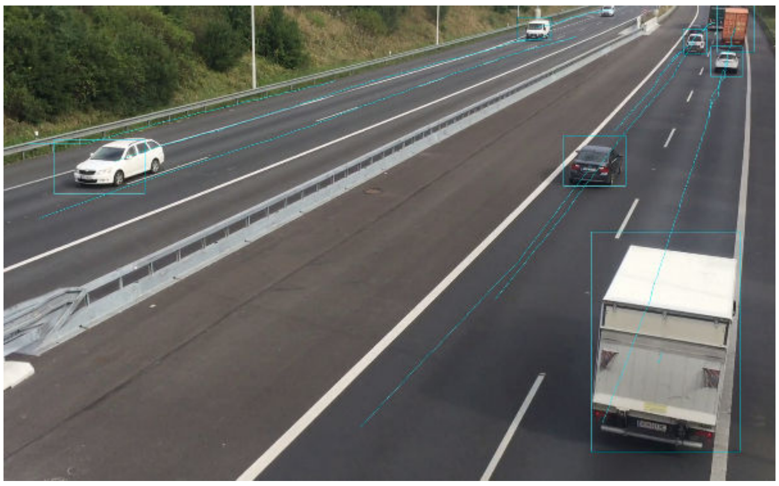


Figure 2.1: Tracking objects in one camera. The rectangles are detection of the objects. The lines are the identified tracks of the objects across multiple frames.

2.1.1 Tracking across multiple cameras with overlapping views

There are quite different approaches towards tracking and identifying objects across multiple cameras. Javed et al. use deep learning to estimate the overlapping view according to the tracks of the re-identified object. [23] When the overlapping view is known, the problem is simplified. The accuracy of the algorithm in the estimation of the overlapping view directly affects the accuracy of tracking people. Naturally, the problem arises when two persons cross each other in the overlapping area: in this case, the algorithm would likely fail and misclassify.

Another approach is to calculate the overlapping area using computer vision techniques and then utilise the appearance and movements of the object to get the final prediction. Three main schemes are usually used to solve tracking across multiple cameras with overlapping views problem: the feature matching scheme, the 3D information based scheme, and the alignment scheme. [24] Feature points are selected in both cameras and they are compared for example by Bayesian Network for better similarity measure. Afterwards, the algorithm estimates the lines which separate the overlapping view of the two cameras. The 3D information based scheme creates a 3D model of the scene. However, the camera needs to be calibrated which makes this approach computationally demanding. The alignment scheme aligns different camera views through geometrical transformation and achieves consistent labeling. After establishing the field of view, a calibration of the brightness of neighboring cameras is established according to the histogram matching method. [25] Subsequently consistent labeling across cameras is computed.

2.1.2 Tracking across multiple cameras with non-overlapping views

Learning methods for inter-camera space-time and appearance probabilities are used in this paper [23]. One approach is to perform training using a single person in the environment, however, this is not always possible. Therefore, only appearance matching can be used for the learning part and it is necessary to label all persons across cameras during training (only the best matches could be used for training).

2.1.3 Re-identification objects without knowledge about camera views

The paper [15] propose an approach on vehicle re-identification without any knowledge about localization or movements of the cars. Real-time traffic information is obtained by this approach. The method is based on linear regression with SVMs according to feature vectors which consist of colour histogram and oriented gradients. First, the vehicles are detected in the video by a detection system that creates 3D bounding boxes around the vehicles. Only the side and front (or back) faces are extracted. It should be less time consuming and the roof is mostly not important for identification. The extracted picture is then fitted into a grid and colour histograms and histograms of oriented gradients are computed for each field of the grid. Finally, the feature vectors are produced and clustered by the linear SVM.

2.2 Object re-identification

As mentioned in the introduction, this study firstly focuses on re-identifying objects. Then, the space-time factors of the cameras will be analyzed and tested. This chapter is focused on the technical background. Different approaches to the analysis problem, searching suitable solutions and finally, the solution of the problem will be described in this section. This part of the Technical Background is focused on the methods for object re-identification. We describe our spatio-temporal method for re-identification task in the section 4.2.4.

Deep learning approach is nowadays the most frequent technique for the task of object re-identification. More precisely, convolutional neural networks are used. In the following, the problem of object re-identification will be analysed and a discussion about neural networks will be given.

The main idea in the task of object re-identification is to track an object in one camera view and then identify the same object in the next camera view. Its most common real-life application is for security purposes. The most widely used approach for this task includes the use of feature vectors or feature maps.

The object (a person in the case of Figure 2.2) on the Probe Image refers to the person which has to be re-identified. Therefore, given a Gallery Set, the program needs to identify an image of the same person.

There are two main approaches to the object re-identification problem: hand-crafted system and deep learning. The hand-crafted system, for example, uses colour histograms, local binary patterns or Gabor filters. The disadvantage of traditional hand-crafted systems is that they do not produce consistently accurate results in scenarios when the cameras are in different light conditions. Modern neural network based methods are more robust and have shown greater accuracy.

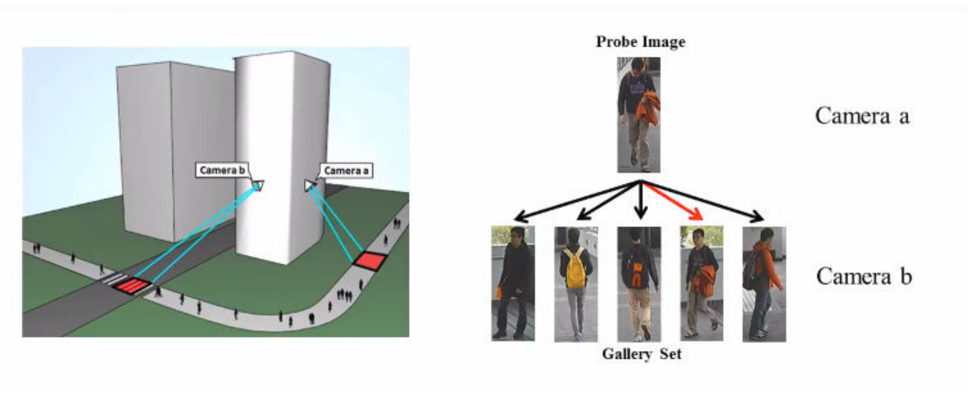


Figure 2.2: Visualisation the object re-identification problem. [26]

2.3 Neural networks

The neural networks are inspired by the way the human brain works. A neural network is composed of layers of neurons. Each neuron has an activation function. In mathematical terms, the neural network is set of algorithms which are designed to recognise patterns. The patterns are a mathematical representation of the real world data, for example, images, text or sound. They are expressed by mathematical vectors and matrices. Neural networks can be used for clustering and classification, among others. [27]

Firstly, we detail a single neuron in a neural network. It receives numerical input from other neurons (or from an input vector) x_1, x_2 and computes an output y. Each input has an associated weight (w_1, w_2), which is assigned on the basis of its relative importance to other inputs. The neuron applies a non-linear activation function f to the weighted sum of its inputs and bias b as shown in Figure 2.3. [28]

The purpose of the activation function f is to introduce non-linearity into the output of a neuron. This is important because most real-world data is non-linear and we want neurons to learn these non-linear representations.

Every activation function takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions you may encounter in

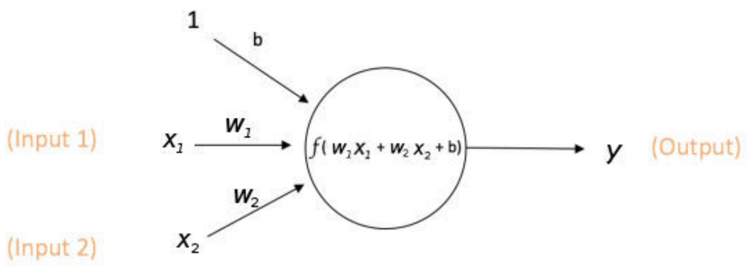


Figure 2.3: A single neuron [28]

practice: **sigmoid** (output is between 0 and 1) $\sigma(x) = \frac{1}{1+\exp(-x)}$, **ReLU** (replace negative values with zero) $f(x) = \max(0, x)$.

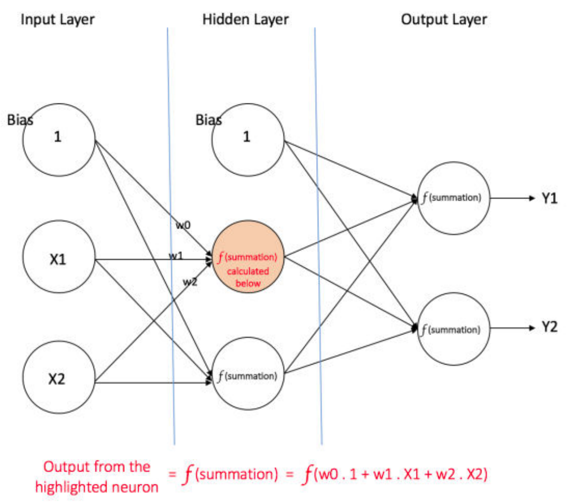
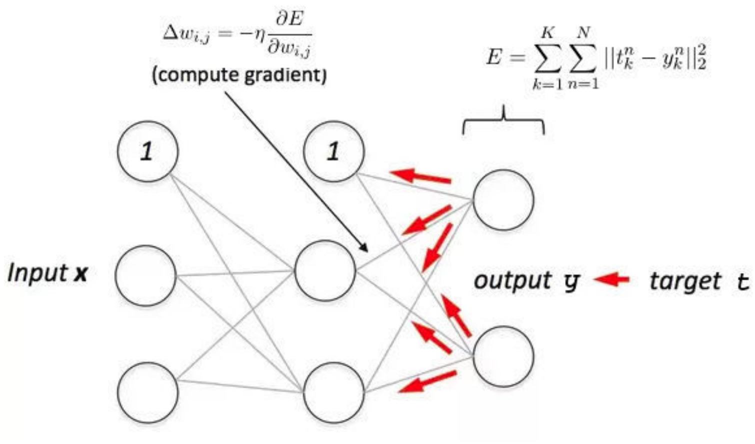

Input Layer
Hidden Layer
Output Layer
Bias
1
Bias
1
X1
X2
w0
w1
w2
f(summation) calculated below
f(summation)
f(summation)
f(summation)
Y1
Y2
Output from the highlighted neuron = f(summation) = f(w0 . 1 + w1 . X1 + w2 . X2)

Figure 2.4: A neural network [28]

A whole neural network is composed of an input layer, hidden layers, and output layer, where each layer has many neurons, see Figure 2.4. Firstly, we randomly initialise weights and biases and feed the neural network with training data. Then we calculate error by loss function. The loss function gives us value how far is each result of the neural network from the corresponding ground truth result. The loss function could be designed for each task separately, but we can use general loss function which gives us the Euclidean distance between the neural network result y and required output t.



$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}$
(compute gradient)
$E = \sum_{k=1}^{K} \sum_{n=1}^{N} ||t_k^n - y_k^n||_2^2$
1
1
Input **x**
output y
target t

Figure 2.5: Backpropagation and weight updation. [28]

Then we need to recalculate weights and biases according to calculated error (loss value). We propagate these loss values back through the network using **Backpropagation** to calculate gradients. Then we use an optimization method called **GradientDescent** to modify all weights and biases in the network with an aim of reducing the loss value at the output layer. This is shown in Figure 2.5. We calculate error E, where y_k^n is output value from a k-th neuron from the output layer for n-th input vector (sample). Then the gradient is computed and the weights (and bias) are updated by the gradient $\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}$, where η is learning rate (the step which dictates how fast the neural network is learning) and E is the error. This approach continues until the input layer is reached.

Usually, this process (feeding the network, calculating the loss function and backpropagation) is calculated on a mini-batch (a subset of the training set) because of the training complexity. We test the neural network on a test set. We want to achieve good results on both the training set and test set, therefore we validate the neural network during training. When the validation error stops decreasing and the training error still decreases, it is time to interrupt the training because the neural network starts to overfit (learn specifically only the train set).

The neural networks are explained more precisely in one of the following resources [29], [27], [30]. The technical background will focus on different kinds of neural networks which this thesis uses in the implementation.

2.3.1 Convolution neural networks

For image classification, the most widely used type of neural networks is called convolutional neural networks. They consider the neighborhood of pixels instead of one vector. They differ from other types of neural networks by sharing the weights between the filters and they use pooling (subsampling) to condense information from the previous layer.

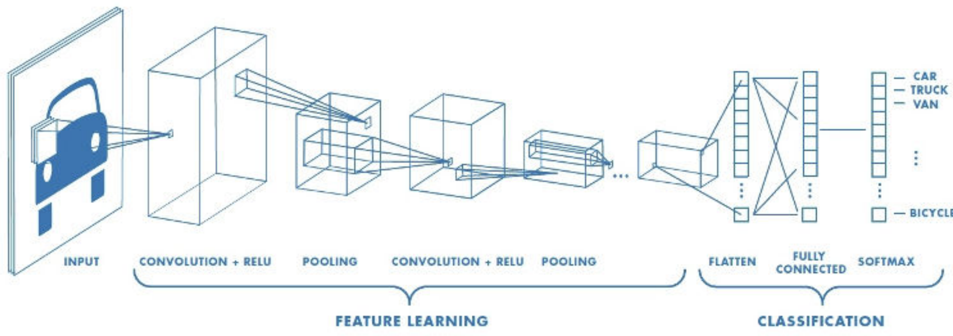


Figure 2.6: The example of a convolutional neural network consider the neigbourhood of pixels and convolute it. [31]

It is beyond the scope of this thesis to discuss details about a convolutional neural network. J. Johnson focuses on CNN [32]. This paper focuses on the specific architecture of convolutional neural network named Resnet.

Latest research leads to developing deeper and deeper convolutional neural networks. [33] [34] [35] With a deeper network, the algorithm can produce more precise results. But extremely deep networks come with a high computational cost. It is also hard to train a deep neural network because of vanishing and exploding gradients. The deep neural networks can also lead to degradation problem, which means the accuracy gets saturated with increasing network depth and then the accuracy degrades rapidly. Unfortunately, the degradation is not caused by overfitting. If more layers are added to a suitably deep model, it will lead to higher training error. [36] Kaiming He et al proposed new neural network architecture named Residual network.

2.3.2 Residual network

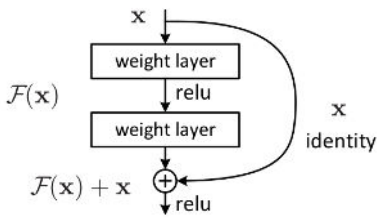
A ResNet (Residual network) [37] offers better training in deep neural networks. Therefore it provides progress in the degradation problem too. ResNet creates skip connections which make it possible to use the output from previous layers combined with the output of the pure unchanged input of the previous layers. The result is fed into the next layer. This method enables very deep networks to be trained effectively.

The residual network is built using residual blocks. A residual block is shown in the Figure 2.7a. The input vector (or matrix) $\mathbf{x_i}$ is fed into the first hidden layer. The equation is the same as the equation used in a layer of a plain neural network. The equation for the hidden layer is $\mathbf{z_i} = \mathbf{W}\mathbf{x_i} + \mathbf{b}$, where $\mathbf{W}$ is the matrix of weights and $\mathbf{b}$ is vector of bias parameters. The ReLu activation function g is then used as follows $\mathbf{x_{i+1}} = g(\mathbf{z_i})$. The same equations are valid for the next layers. Until this point, everything is the same as a plain neural network. The difference between a plain neural network and a residual network is in the shortcut. The shortcut is the copy of $\mathbf{x}$ pushing forward without change, as depicted in figure 2.7a. However, the equation for the last (in our case for the second) ReLu activation function g in the block is different. The equation has to consider the $\mathbf{x}$ from the shortcut: $\mathbf{x_{i+2}} = g(\mathbf{z_{i+1}} + \mathbf{x_i})$ It should be noted that residual networks do not add any extra parameters. The shortcut only propagates $\mathbf{x}$ without using any extra parameters. For this reason, the method has similar complexity as a plain neural network.

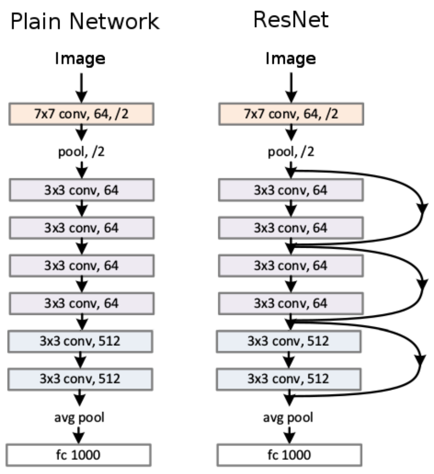
A Residual network is composed of many residual blocks. Therefore the difference between a plain network and a residual network is only in the shortcuts. The residual network is created from plain network adding shortcuts every two layers, as in Figure 2.7b.

When adding more layers to neural networks, the training error will first decrease but subsequently, create spikes due to the increased depth. This is the degradation problem which was described above. Nevertheless, this is not the case in residual networks, as the error only decreases regardless of the depth of the network.

One of the possible reason behind the efficiency of residual networks is the following. From the equation $\mathbf{x_{i+2}} = g(\mathbf{W}\mathbf{x_i} + \mathbf{b} + \mathbf{x_i})$ we can see that $\mathbf{W}$ and $\mathbf{b}$ will be zero because of the ReLu activation function. However, with the addition of


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

(a) : The example of a convolutional neural network consider the neigbourhood of pixels and convolute it.



(b) : On the left image is the plain neural network and on the right side is the ResNet. There is only one difference - the "shortcuts".

Figure 2.7: Residual network [37].

the shortcuts, we will still have $\mathbf{x_i}$ as the output of the layer, making it non-zero. This makes the network easy to train. Clearly, obtaining reliable results when training is a very important factor for getting good testing accuracy.

2.4 Loss functions

Every neural network is trained with a pre-defined loss function. This loss function is computed during training and it indicates how far away learned samples are from the ground truth data. Hence, the larger its value, the worse the accuracy of the neural network. During training, the cost function is minimised by a procedure called backpropagation. Essentially, the neural network recalculates weights and biases so that the cost function is minimum. This is usually done with the gradient descent method.

2.4.1 Triplet loss function

As mentioned above, one way to learn the parameters of a neural network is to define and apply the gradient descent method [38] on a triplet loss function. The triplet loss function compares two pairs of images (in total three images) at the same time. As seen in Figure 2.8, the first image is called "probe image" or "query image" (the algorithm is looking for the same person according to this image). The second is a positive match image, which means the detected person has the same identity as the probe image. Finally, the last one is a negative match image, this means it has different identity than the person in the probe image. Every image is fed into the neural network separately and the triplet loss method compares the feature vectors generated by a convolutional neural network for each image.

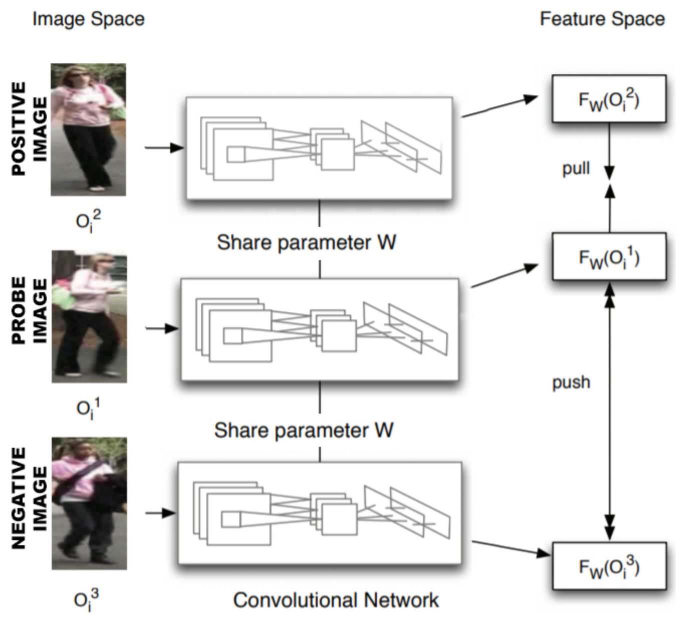


Figure 2.8: Convolution neural network with triplet loss function. [39]

This method pulls the feature vectors of the probe image and the positive image close together, while simultaneously doing the opposite for the probe image and negative image. That is, it pushes the probe image and negative image as far from each other as possible. During this procedure, the neural network creates a feature vector for each image. This feature vector is basically a representation of the coordinates in a new multidimensional space, called the embedding space.

Consequently, the triplet loss function compares the distances of the feature vectors (coordinates) between each image in the embedding space. It should be noted that from this point, "distance between images" refers to the distance between feature vectors (coordinates in embedding space) of the images.

$$L_e = \sum_{pr,p,n}^{N} \max((||f(x_{pr}) - f(x_p)||_2^2 - ||f(x_{pr}) - f(x_n)||_2^2 + \alpha), 0), \quad s_{pr} = s_p; s_{pr} \neq s_n. \tag{2.1}$$

Equation 2.1 is graphically visualised in Figure 2.9. The green term is the calculation of the normalised Euclidean distance between the probe image and positive image. The red term of the equation is the calculation of the normalised Euclidean distance between the probe image and the negative image. Accordingly, the loss function should make the difference of the green part d_i and red part d_j as big as is possible. The parameter α represents the margin. The margin α ensures the minimal margin between d_i and d_j will be met. If the margin α is skipped in the equation, the CNN could learn both distances d_i and d_j as zeros and the loss function will not penalise it.

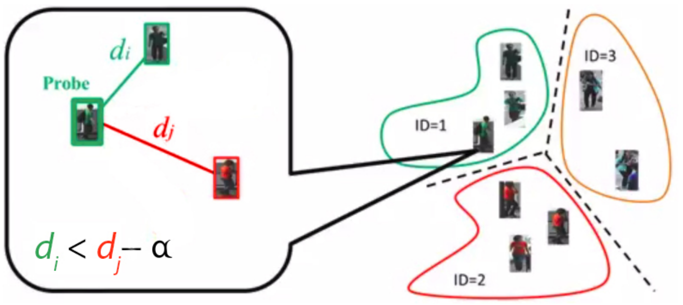


Figure 2.9: Visualisation of the triplet loss equation. [5]

The problem can be seen as manifold learning. The triplet loss function classifies all images from one person into a class. In other words, each class belongs to a single person. The goal of the triplet loss function is to make bigger distances between negative images and smaller distances between positive images.

2.4.2 Quadruplet loss function

The quadruplet loss function is a different type of loss function. The output of any loss function is always a float number called loss value, however, the difference remains as to how to calculate this loss value. The quadruplet loss function is based on the triplet loss function explained in the previous subsection 2.4.1. The main difference is that the quadruplet loss function takes four images as input instead of the three images of the triplet loss function input. The difference between the triplet loss function and the quadruplet loss function is highlighted by red colour in Figure 2.10.

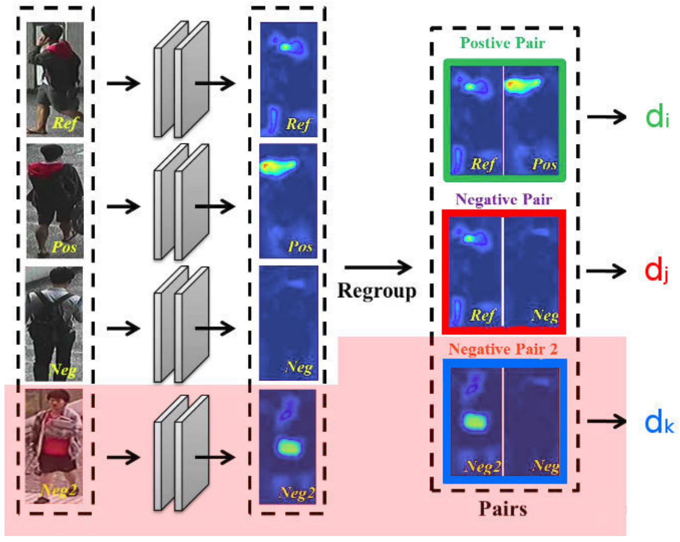


Figure 2.10: Neural network with quadruplet loss function. The difference between triplet loss function and quadruplet loss function is the red part. [5]

The four images are seen in Figure 2.10 represent the input of the quadruplet loss function. The first three images are the same as in the triplet loss function, i.e. probe image, positive image - the same person as in the probe image, negative image - a different person than the person in the probe image. The fourth image is another negative image. The algorithm chooses four images from the batch and regroups them as into pairs as follows: The first pair is composed of probe image (also called reference image) and the positive image. The second pair contains the probe image and a negative image, while the third pair is composed of two negative images regardless of whether the pair contains the same probe image as previous pairs. The loss function calculates the following distances between images for each pair: signed d_i - the distance between the probe image and the positive image, d_j - the distance between the probe image and the negative image, d_k - the distance between two negatives images, which are chosen arbitrarily.

The principle is the same as in the triplet loss function: the algorithm keeps a distance between positive images as low as is possible and it pushes a distance between negative images as high as is possible. The quadruplet loss contains two main parts. The first part of the equation (the first summand) is from the triplet loss function and the second part (the second summand) is the new one. 2.2

$$
\begin{aligned}
L_q = &\sum_{pr,p,n1}^{N} \max((||f(x_{pr}) - f(x_p)||_2^2 - ||f(x_{pr}) - f(x_{n1})||_2^2 + \alpha_1), 0) \\
&+ \sum_{pr,p,n1,n2}^{N} \max((||f(x_{pr}) - f(x_p)||_2^2 - ||f(x_{n1}) - f(x_{n2})||_2^2 + \alpha_2), 0), \\
&s_{pr} = s_p; s_{pr} \neq s_{n1}; s_{pr} \neq s_{n2}; s_{n1} \neq s_{n2}.
\end{aligned}
\tag{2.2}
$$

The neural network transforms all images into the embedding space as mentioned before. Clusters are formed in the new high-dimensional space according to the personal identity. The images with the same personal identity are close to each other, essentially forming a cluster. The class contains all images with the same personal identity. Therefore, "class" can be used interchangeably as "cluster" in the optimal solution. Moreover, the intra-class variance should be low and inter-class variance should be high in the optimal solution. If this condition is fulfilled, the neural network is more accurate not only during training but more importantly when testing. The intra-class variance indicates the distances between positive images (between images inside one class) and the inter-class variance expresses the distances between negative images (between images from different classes).

The distance $d_k = ||f(x_{n1}) - f(x_{n2})||_2^2$ represents the distance between two negative images. The triplet loss function minimises mainly the intra-class variance. The quadruplet loss also helps maximise inter-class variance. Therefore, the negative images should be as far away from each other as possible in the embedding space.

As this distance increases, the neural network will improve the performance of the testing data, because it will be able to separate the classes more effectively.

The quadruplet loss function has two margins, α_1 and α_2. These margins serve the same purpose as the margin in the triplet loss function. The first term (first part of the equation) aims to obtain the correct orders with the same probe in training data. The second term (the second part of the equation) provides a help from the perspective of orders with different probe images, therefore the algorithm has to prioritise the first term. The function ensures this by using different margins, specifically α_2 being lower than α_1.

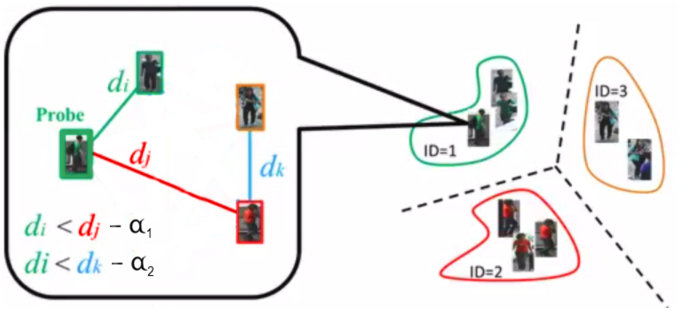


Figure 2.11: Visualisation of the quadruplet loss function. [5]

2.5 Mining methods for hard samples selection in neural network training

Choosing appropriate data samples when computing the loss function is a very significant factor for successfully training a neural network. When a hard sample mining method selects the suitable number of the hardest samples, the neural network training produces optimal results and also effectively approaches convergence. Unfortunately, designing a proper hard sample mining method is not a trivial task. When the hard sample mining method chooses only a few hard samples, the training converges to a suboptimal solution. On the other hand, if it selects too many hard training samples, the training can end up overfitting. This section describes different kinds of hard sample mining methods.

The method *"Margin-based online hard negative mining"* [5] applies margin α

as a threshold. The margin α is used to confine the distance between positive and negative pairs in a quadruplet sample. The method should predefine a suitable margin threshold. However, it is a challenging problem. The algorithm with a small threshold returns only a few hard samples. In contrast, a large threshold would produce too many hard training samples. The following algorithm overcomes this dilemma by calculating adaptively the margin threshold accordingly when training. More precisely, It utilises the margin threshold to find the suitable number of hard samples. The main idea behind the adaptive margin is to express the average distance of the two distributions: The first being the positive pair distance distribution and the second, the negative pair distance distribution. Therefore the adaptive margin threshold will be the average distance between the two distinct distributions 2.3.

$$\begin{aligned}
\alpha &= \omega(\mu_n - \mu_p) \\
&= \omega\Bigg(\Big(\frac{1}{N_n}\sum_{pr,n}^{N} ||f(x_{pr}) - f(x_n)||_2^2\Big) - \Big(\frac{1}{N_p}\sum_{pr,p}^{N} ||f(x_{pr}) - f(x_p)||_2^2\Big)\Bigg), \\
s_{pr} &= s_p; s_{pr} \neq s_n,
\end{aligned} \tag{2.3}$$

where ω is the correlation coefficient, μ_p and μ_n are mean values of the two distributions. N_p and N_n are the numbers of positive and negative pairs respectively. Although, it should be noted that this method cannot be used from the start of the training, as the distributions are unpredictable and will change drastically during the first few iterations. For this reason, the training starts with a constant alpha margin and is only updated when the distributions become stable. Specifically, the method then starts to calculate the adaptive margin by selecting samples that surpass it for the calculation of the triplet loss function. The algorithm calculates the two margin thresholds using the same approach for the quadruplet loss function.

Another method for hard sample mining is called *"Batch All"* [39]. The training dataset contains images from different classes P (person identities) and it has K images for each class. The Batch All method is sampling all possible $PK(PK - K)(K-1)$ combinations of triplets 2.4.

$$
\begin{aligned}
L_{BA}(\Theta; X) &= \sum_{i=1}^{P} \sum_{pr=1}^{K} \sum_{p=1, p \neq pr}^{K} \sum_{j=1, j \neq i}^{P} \sum_{n=1}^{K} \max(0, m + d_{pr,p,n}^{i,j}), \\
d_{pr,p,n}^{i,j} &= D(f_\Theta(x_{pr}^i), f_\Theta(x_p^i) - D(f_\Theta(x_{pr}^i), f_\Theta(x_n^j),
\end{aligned}
\tag{2.4}
$$

which is performed on a mini-batch X and where Θ is the distance function. Usually, the Euclidean distance is used. P refers to the set of classes in the batch and K is the set of images in the batch. x_{pr}^i corresponds to the pr-th image of the i-th person in the batch. The first two sums $\sum_{i=1}^{P} \sum_{pr=1}^{K}$ iterate through all probe images, while the sum $\sum_{p=1, p \neq pr}^{K}$ covers all positive images with the selected probe image. Finally the sums $\sum_{j=1, j \neq i}^{P} \sum_{n=1}^{K}$ determine to sample through all negative images with the probe image.

Improvements of the Batch All method for hard data mining are provided by a method called *"Batch Hard"* [3]. The algorithm chooses images from the batch by randomly sampling P classes, then randomly sampling K images of each class. In the end, we end up with a batch of PK images. The algorithm trains the network on these batches. The Batch Hard method selects the hardest positive and the hardest negative samples within the batch for each sample in the batch and forms the triplets for computing the loss function 2.5.

$$
\begin{aligned}
L_{BH}(\Theta; X) = \sum_{i=1}^{P} \sum_{pr=1}^{K} \max(0, \max_{p=1..K} & D(f_\Theta(x_{pr}^i), f_\Theta(x_p^i)) \\
& - \min_{j=1..P, n=1..K, j \neq i} D(f_\Theta(x_{pr}^i), f_\Theta(x_n^j)) \\
& + \alpha).
\end{aligned}
\tag{2.5}
$$

The labeling in equation 2.5 is the same as in the Batch All method. pr, p and n represent the probe image, positive image, and negative image respectively.

2.6 Re-ranking method with a k-reciprocal set

The re-ranking method is used to re-order the most similar images with the probe image according to their nearest neighbours. We will now explain the flow of the algorithm. Firstly, the embedding feature vectors are produced by evaluating the trained neural network with the probe image and all test images as inputs. Then, the distances between the embedding vector of the probe image and the embeddings vectors of the test images are calculated. The test images are consequently sorted according to their distances with the probe image. In other words, the ranking is calculated. A re-ranking method takes this calculated ranking and re-orders the test images according to the common nearest neighbours of the probe image and the test image. In this section, the k-reciprocal encoding method to re-rank the re-identification task results are described. The hypothesis of Zhong et al. [40] is that if a gallery image is similar to the probe in the k-reciprocal nearest neighbors, it is more likely to be a true match. The Jaccard distance is used for calculating the score according to the k-reciprocal nearest neighbours. This method does not require any human interaction or any labeled data, thus making it suitable for large-scale datasets.

The k-reciprocal nearest neighbours basically belong to the top-k rank for each other when the other image is taken as the probe. The method consists of the following three steps.

1. The k-reciprocal feature vector is created from the weighted k-reciprocal neighbor set and the Jaccard distance between two images is computed by their k-reciprocal features.

2. A local query expansion approach further improves the re-ID performance; firstly proposed by Zhong et al. [40].

3. The final distance is calculated as the weighted aggregation of the original distance (original ranking score) and the Jaccard distance.

The k-nearest neighbors (the top-k samples of the ranking list) of the probe image

p are defined as $N(p,k)$.

$$N(p,k) = g_1, g_2, ..., g_k, |N(p,k)| = k,$$

where p, g correspond to the probe image and gallery image (test image) respectively. Then, the k-reciprocal neighbors is defined as $R(p,k)$.

$$R(p,k) = g_i|(g_i \in N(p,k)) \wedge (p \in N(g_i,k)).$$

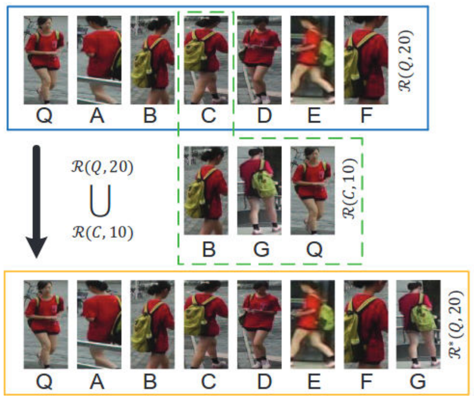


Figure 2.12: Creating the k-reciprocal set. [40]

Unfortunately, the positive images may be excluded from the k-nearest neighbors because of the variations in illumination, pose, view and occlusions. Therefore the set is expanded into a more robust set $R^\star(p,k)$. In Figure 2.12 we provide visualisation of the example for the C image which is in the top-k for probe image Q. The method finds $\frac{1}{2}k$-reciprocal images for the gallery image C and if the number of the $\frac{1}{2}k$-reciprocal images is higher than $\frac{2}{3}|R(q,\frac{1}{2}k)|$, all non-common images from the $\frac{1}{2}k$-reciprocal images are added into the final k-reciprocal set $R^\star(p,k)$. In the mathematical way:

$$\begin{aligned}
&R^{\star}(p,k) \leftarrow R(p,k) \cup R(q, \frac{1}{2}k),\\
&s.t. |R(p,k) \cap R(q, \frac{1}{2}k)| \geq \frac{2}{3}|R(q, \frac{1}{2}k)|,\\
&\forall q \in R(p,k),
\end{aligned}$$

where k-reciprocal neighbors of the probe image p are used along with $\frac{1}{2}k$-reciprocal neighbors of each top-k nearest neighbor g_i of the probe image p. More positives samples are added to the final set $R^{\star}(p,k)$ by this operation.

The Jaccard distance is calculated for each pair composed from the probe image p and every gallery image g_i by comparing their k-reciprocal nearest neighbor set $R^{\star}(p,k)$. The idea proposed by Zhong et al. [40] is that if two images p, g_i are similar, their k-reciprocal nearest neighbor sets overlap $|R^{\star}(p,k) \cap R^{\star}(g_i,k)| \geq 0$. The two images are even more similar when they have more common samples. The Jaccard distance is based on this approach and it is calculated as:

$$d_J(p, g_i) = 1 - \frac{|R^{\star}(p,k) \cap R^{\star}(g_i,k)|}{|R^{\star}(p,k) \cup R^{\star}(g_i,k)|}$$

2.7 Evaluation of re-identification task

Usually, the mean average precision score, ranking, and CMC curve are all used for evaluating neural networks which deal with a re-identification problem. This section explains the methods used in the evaluation section of this thesis.

The re-identification problem is evaluated using a set of probe images (also called query images) and a gallery set. The algorithm compares each probe image with all images from the gallery set. It returns the gallery images, ordered according to their similarity with the probe image. Thus, the result for each probe image is the whole gallery set in a correct order. The ranking labels refer to the number of images of the sorted gallery set we need to iterate until we find the ground truth image (the image with the same object identity as the probe image). In the other words, the correct rank for the probe image is the order of the first ground truth image in the sorted gallery set.

The following terms are used in the Evaluation part 5 of the thesis: image-to-image, image-to-track, a single query, and a multi-query. **Image-to-image** re-identification means that we compare a query image with each image from the gallery set. **Image-to-track** re-identification indicates a comparison between a query image and each track, where the track consists of gallery images. **Single query:** in this scenario the query set contains only one image per object id per camera. **Multi-query:** here the query set consists of many images per object id per camera.

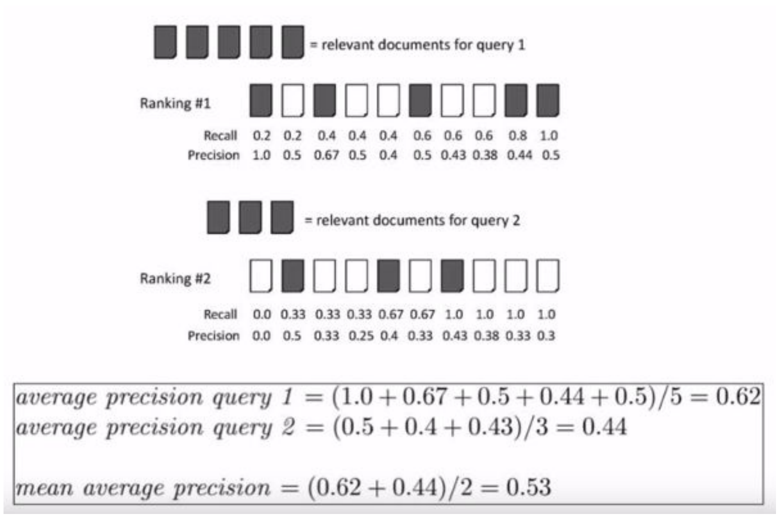
2.7.1 Mean Average Precision (mAP)

The mean Average Precision (mAP) score represents the performance of the neural network through all query images and ranks. Basically, mAP calculates the average precision in the rank for each query image. Consequently, it calculates the mean through all average precisions of query images:

Grey rectangles in Figure 2.13 indicate all ground truth images ("relevant documents") for the query image. Two query images are depicted in the figure, therefore two rankings and two different groups of ground truth images are also in the figure. Firstly, the mAP calculates the precision for each image in the rank. For example, the third image in Ranking #1 has a precision score of 0.67:

$$precision = \frac{true\_positives}{true\_positives + false\_positives} = \frac{2}{2+1} = 0.67$$

The precision score of the image considers only true_positives and false_positives in the rank of this image alone. The algorithm essentially computes the average



= relevant documents for query 1

Ranking #1

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

= relevant documents for query 2

Ranking #2

Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

$$\textit{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$

$$\textit{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\textit{mean average precision} = (0.62 + 0.44)/2 = 0.53$$

Figure 2.13: Example of calculation of the Mean Average Precision (mAP) [41]

precision for all ground truth images and for each query image. Then, it calculates the mean of the computed average precisions.

2.7.2 Cumulated Matching Characteristics (CMC)

The Cumulated Matching Characteristics (CMC) curve is a graph visualisation of the rank performance. For example, rank number 4 in Figure 2.14 has accuracy 95% which means that 95% query images have their ground truth images in rank 4.

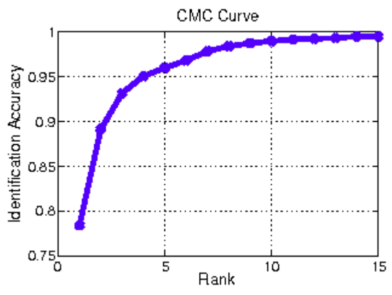


Figure 2.14: Example of CMC curve

Chapter 3

State of the art

Historically, popular methods for person re-ID were typically comprised of two components: designing hand-crafted features and learning distance metrics [42]. Most works focused on developing features invariant to variations in light, pose and viewpoint while using conventional distance metrics like the Mahalanobis distance [43], Bhattacharyya distance, and the l_1- and l_2-norms. Research has also been performed on a post-processing technique called re-ranking [40, 44]. In more recent years, the focus has shifted towards deep learning methods, which are routinely used to obtain state-of-the-art results over a wide variety of challenges in computer vision and machine learning. [33] [34] [35] This chapter focuses on the recent research especially the papers which inspired this thesis.

Typically, two types of CNN model have been employed to solve the person re-ID task: the classification model that is used across a broad spectrum of computer vision problems and, more commonly for re-ID, the Siamese model which takes multiple images as input, such as pairs [16, 45], triplets [46], and quadruplets [5].

Triplets have been used extensively in the field of person re-ID. Generally, each query image is matched with a positive sample and a negative sample, forming a triplet as input to the CNN. Wang et al. [47] proposed to use the triplet loss function to learn image similarity. Cheng et al. [48] introduced an improved triplet loss function which decreases the distance of similar IDs and increases the distance of dissimilar IDs. Hermans et al. [3] proposed *Batch Hard* mining in order to find harder triplets to improve the efficacy of training. Explained in Section 2.5. Chen et al. [5] added a second negative image to train with an additional negative pair and form a quadruplet loss. This further enlarges the inter-class variations by pushing negative pairs away from positive pairs with respect to different probe images.

Our implementation is based on the neural network proposed by Hermans, Beyer, and Leibe [3]. They proposed optimal settings and methods to train CNN with triplet loss function. The idea is to create embedding system. The neural network should work as map function from real space (real images) to embedding space (explanation in 2.4.1). The triplet loss function optimises the embedding space. Therefore data points with the same identity are closer to each other than those with different identities. Some state-of-the-art approaches are focused on cross-image classification meaning only comparing how similar are two images. This approach is prohibitively expensive, each probe image has to go through the network paired up with every gallery image.

Triplet loss became unpopular but Hermans, Beyer, and Leibe [3] refers following: when the triplet loss is implemented naively, it produces disappointing results. The mining of hard triplets is an essential part of learning neural network using the triplet loss, as otherwise, training will quickly stagnate. On the other hand, mining the hardest triplets is time-consuming and it is not defined exactly what are appropriate hard triplets. Besides that, selecting only a few hard triplets makes the training overfitted, however selecting too many hard triplets leads the training to slow convergence and suboptimal solution. Hermans, Beyer, and Leibe [3] analyzed the design space of triplet losses and evaluated which one works best for person re-identification, resulting in both faster training and better performance.

The first important approach from the paper inspiring this thesis is mapping the images into embedding space and comparison distances between images in the embedding space because it is less time-consuming than cross-image comparison. The second important approach is mining hardest triplet images. Firstly randomly sampling P classes (person identities) from the batch and then randomly sampling K images of each class (person), thus resulting in a batch of PK images. Therefore for each sample in the batch, the algorithm can select the hardest positive and the hardest negative samples within the batch. Hermans, Beyer, and Leibe [3] suggested hard data mining method called Batch Hard which according to their experiments returned the best result in comparison with Batch All [39] and with Lifted Embedding loss [49].

The quadruplet loss function explained in section 2.4.2 is an improvement of the triplet loss function proposed by Chen et al. [5]. The disadvantage of the triplet loss is following, the training set usually returns good results but on the test set, it is not so accurate, which means that the triplet loss function has a low generalization.

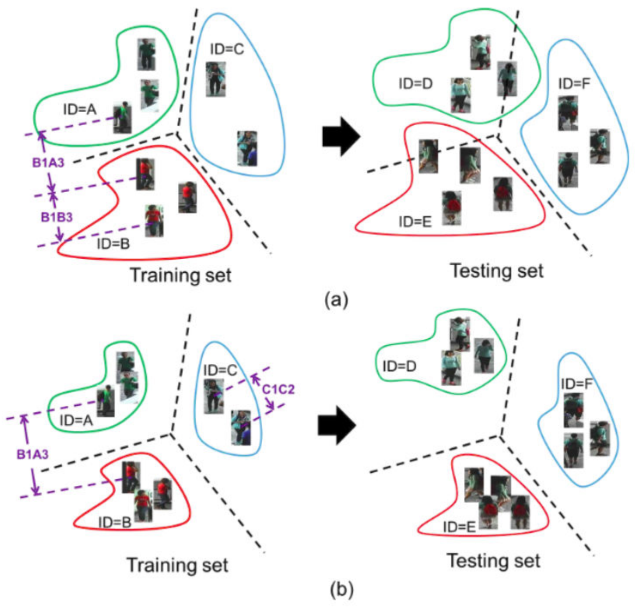


Figure 3.1: (a) Visualization of the low generalization with triplet loss function (b) Visualization of the better generalization with quadruplet loss function [5]

The dashed lines in Figure 3.1 indicate ground truth clustering. The triplet loss is in figure (a) and the quadruplet loss is depicted in figure (b). The individual classes are highlighted by different colours. One class means all images of a single person. Both loss functions provide an accurate result in training set, but the triplet loss is worse in the testing set. Large intra-class variance and small inter-class variance is in the results of the triplet loss function performed on the training set. The intra-class variance means the distance between images inside one class. On the other hand, the inter-class variance is distances between images from different classes. The small inter-class variance leads to the worse accuracy in the test set. Therefore the quadruplet loss function maximises also inter-class variance and it contributes to the better accuracy of the test set.

Only methods improving the training phase were introduced so far. Zhong et al. [40] proposed method augmenting precision during an evaluation the probe image. The method is called the re-ranking method. The trained neural network is completed, the embedding coordinates are calculated for the query image and also for gallery images. Suddenly, the algorithm evaluates the query image and returns sorted gallery images according to similarity with probe image, called also ranking. The re-ranking method improves the order of the sorted gallery images without any other knowledge, only the embeddings coordinates are known.

The re-ranking method is based on the idea of the k-nearest neighbor's method. An underlying assumption is that if a returned image ranks within the k- nearest neighbors of the probe, they are probably true images (image of the same person as probe image). However, false images can be included in the k-nearest neighbors of the probe, so the re-ranking method is focused to exclude the false positive images. The k images are considered in the method.

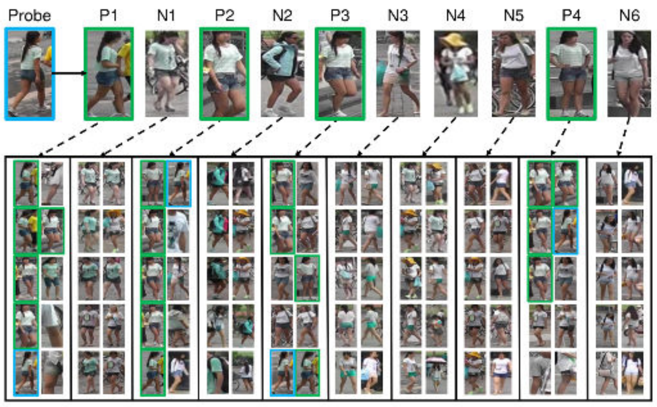
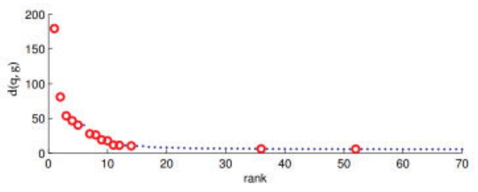


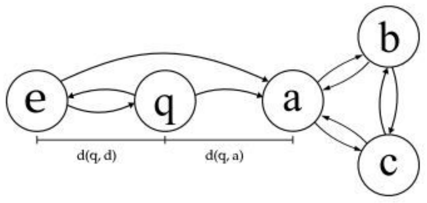
Figure 3.2: Main idea of the re-ranking method [40].

For example, in Figure 3.2, the first image is probe image and the following images in the row are the sorted gallery images. P1, P2, P3, and P4 are the positive images and N1, N2, N4, N5, and N6 are negative images. In the optimal solution, the P1, P2, P3, and P4 should be in the top-4 ranks. However, in our case, there are also negative images among them.

Therefore Qin et al. [50] proposed improvements based on clusters in embedding space. They pointed to the two main observations: Distance functions degrade quickly in high dimensional spaces and Non-reciprocity of near neighbor relationships. Usually high dimensional embedding vectors are used for the object re-identification. It has the negative effect, that most distance or similarity measures quickly degenerate at points far away from the query vector. This disadvantage is illustrated in Figure 3.3a). The distance measure $d(q, g)$ is plotted into the graph. The ground truth images are denoted by a red circle. Most of the ground truth images are correctly in the top ranks, however few of them are between negative images in higher ranks. The similarity curve flattens quickly and the distances between images are almost analogous. Therefore the distance measure is very useful for images close to the query, but it loses its utility far away from the query. Re-Ranking methods push the more relevant images closer to the query image.



(a) : Distance function degrades quickly in high dimensional space. It causes that a few of the ground truth images (denoted by red circles) are between negative images in higher ranks [50].



(b) : Even the distances from q to e and from q to a are similar, e will be in the same class as q with higher probability according to a re-ranking method. Because node a has many neighbors which are not close to node q. [50].

Figure 3.3: Observations during ranking process.

The non-reciprocity of near neighbor relationships to come out from relations between embedding feature vectors. The near neighbor relationships are not symmetric, however, the distance d is a symmetric function. K-nearest neighbors (i.e. the top-k ranking list) of a query image q as the k most similar images in the gallery G are defined in equation 3.1.

$$N(q, k) \subset G, \tag{3.1}$$
$$|N(q, k)| = k, \tag{3.2}$$
$$d(q, a) > d(q, b) \forall a \in N(q, k), b \in G \backslash N(q, k). \tag{3.3}$$

The similarity distance d is a symmetric function, however the nearest neighbor relationships are not symmetric. Therefore $a \in N(b, k)$ does not imply $b \in N(a, k)$

in general. Many papers define the set of k-reciprocal nearest neighbors $R(q, k)$ as $R(q, k) = g \in N(q, k) \wedge q \in N(g, k)$ [40] [50] [51] [52] [53]. The k-reciprocal nearest neighborhood relationship is visualised in the Figure 3.3b. q is the query image and distances $d(q, e)$ and $d(q, a)$ are compared. $N(q, 2)$ set contains images a and e but $R(q, 2)$ set contains only image e because image a has more connections with a lot of images which are not related to the q image. It indicates that the image a is probably an negative image. The re-ranking method used in this thesis is based on the idea of pushing the more relevant images closer to the query image according to k-reciprocal nearest neighbors.The exact re-ranking method with a k-reciprocal set is described in the Technical background 2.6.

Although similar to person re-ID, vehicle re-ID has received comparatively little attention. This is inconsistent with other computer vision tasks in the vehicle domain, like detection and classification, which have received increased attention in recent years. This lack of popularity can be attributed to the inferiority of large-scale vehicle re-ID datasets compared with their human re-ID counterparts.

Given the additional complexity of the vehicle re-ID problem, more creative methods have been proposed to obtain satisfactory results. Liu et al. [19] developed a two branch CNN to learn deep features and the distance metric simultaneously. Liu et al. [7] combined hand-crafted features and high-level attributes learned by a CNN with license-plate recognition and spatio-temporal information. Zhou et al. [9] trained a model on a toy car dataset to in order to infer a multi-view vehicle representation from any input view.

Triplet-wise training has also been effectively applied for vehicle re-ID. Zhang et al. [12] achieved state-of-the-art results by combining the triplet loss with a classification loss and also by ensuring negative samples in one triplet act as positive samples in another triplet. Bai et al. [13] fed groups of images into their triplet network to mitigate inter-class variance and propose a mean-valued triplet loss to enhance learning.

Chapter 4

Proposed solution

This chapter is focused on our mathematical solution and subsequent implementation of the proposed solution.

4.1 Problem definition

Our task could be divided into two phases. First, the object re-identification problem explained in ‘Technical Background’ 2.2 is investigated, then tracking in multiple cameras, also described in ’Technical Background 2.1, is looked at. If the re-identification problem can be solved, multiple camera tracking can also be fully determined because we only join tracks of the re-identified objects. However, achieving 100% accuracy at re-identification is very difficult. This section concentrates on both phases of the problem.

Videos from at least two different cameras are given as input. The objects are detected and signed via image coordinates ($(x, y) \in \mathbb{Z}^2$) in each camera. Tracking algorithms are applied to each camera to return tracks of each object. The tracks have to be re-identified, joined and assigned with the same id in the multi-camera tracking problem. Multi-camera re-identification is based on the spatio-temporal information of each camera. The re-identification based on features extracted from each image is solved by deep learning whilst spatio-temporal connections between cameras are solved via hand-crafted methods. As output, we obtain tracks of each object across all given cameras.

We focus primarily on vision-based re-identification because it is a more important task than the space-time factors. The positions and distances between cameras are usually not provided in real tasks. Moreover, the person can move at a non-constant velocity between cameras (in no camera's view) resulting in an appearance in the second camera at a different time than expected.

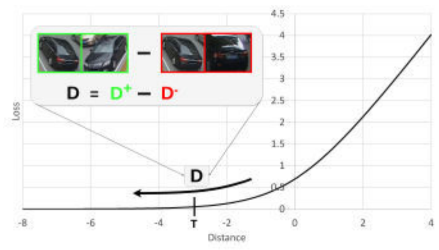
4.2 Proposed method

4.2.1 Re-identification based on visual aspects

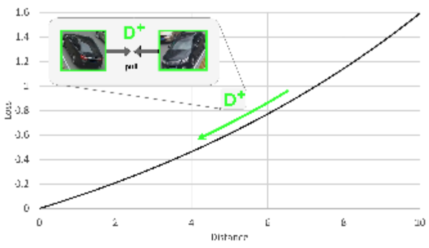
We proposed a new loss function with class distance engineering to solve the re-identification task. Our loss function is based on the triplet loss function [3] explained in section 2.4.1. We use the triplet loss function as the main part of the new proposed loss function.

One limitation of the triplet loss is that although it broadly categorises images of the same cluster together, the distance between images of the same identity remains fairly large whilst the distance between images of different identities is relatively small.

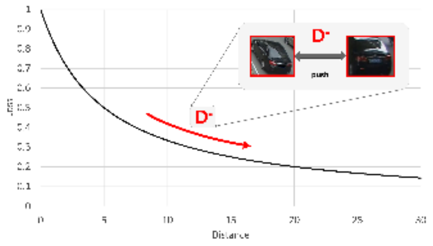
Motivated by Chen et al. [5], we propose an improved triplet loss to minimise the intra-class distance and maximise the inter-class distance. We design separate terms in our loss function for each respective purpose. Rather than re-using images from the triplets, as in [5] we mine the hardest positive and hardest negative pairs in order to severely challenge the framework and learn from the toughest examples.



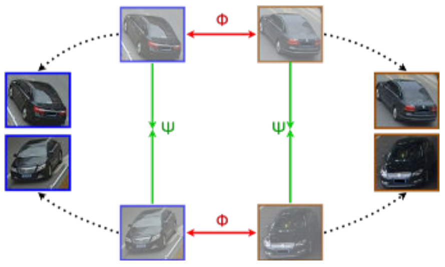
(a) : $\text{softplus}(D)$



(b) : $\Psi(D^{+}) = \Psi^{D^{+}} - 1$



(c) : $\Phi(D^{-}) = \frac{\Phi}{\Phi + D^{-}}$



(d) : Combined effect of the loss function.

Figure 4.1: Visualisations of the softplus, Ψ and Φ functions used to calculate our overall loss.

Let $\mathcal{T}$ be the set of triplets, where $t = (x_0,\, x_0^{+},\, x_0^{-}) \in \mathcal{T}$ is a triplet comprising of a query image x_0, a positive image x_0^{+} from the same identity as x_0 and a negative image x_0^{-} from a different identity. $\mathcal{H}^{+}$ is the set of positive pairs $h^{+} = (x_1, x_1^{+})$ with lowest similarity (i.e. the hardest positive pairs) in the batch and $\mathcal{H}^{-}$ is the set of negative pairs $h^{-} = (x_2, x_2^{+})$ with highest similarity (likewise, the hardest negative pairs) in the batch. We set $|\mathcal{H}^{+}| = |\mathcal{H}^{-}| = |\mathcal{T}| = 4P$ where P is the number of identities in each batch. Our loss function is formulated as follows:

$$\begin{aligned}\mathcal{L}_{ours} = {} & \alpha_1 \sum_{t \in \mathcal{T}} \text{softplus}\left(||f(x_0) - f(x_0^{+})||_2^2 - ||f(x_0) - f(x_0^{-})||_2^2 \right) \\ & + \alpha_2 \sum_{h^{+} \in \mathcal{H}^{+}} \Psi\left(||f(x_1) - f(x_1^{+})||_2^2 \right) + \alpha_3 \sum_{h^{-} \in \mathcal{H}^{-}} \Phi\left(||f(x_2) - f(x_2^{-})||_2^2 \right),\end{aligned} \tag{4.1}$$

where $f(x)$ is the feature vector of image x, $\Psi(D) = \psi^D - 1$, and $\Phi(D) = \frac{\phi}{\phi+D}$. The weights α_1, α_2 and α_3 are hand-tuned depending on the relative importance of the intra-class and inter-class distances, respectively. Therefore, in our experiments, we set $\alpha_1 = 1$, $\alpha_2 = 0.75$ and $\alpha_3 = 0.3$. We also have $\psi = 1.1$ and $\phi = 5$. In the rest of this section, we provide additional discussion on the motivation behind our design choices.

The first term we use is a modified triplet loss function presented in [3]. Their analysis shows that replacing the traditional hard margin α from (2.1) with the softplus function $\mathrm{softplus}(D) = \log(1 + \mathrm{e}^D)$ is beneficial. The softplus function is shown in Figure 4.1(a).

As the difference between the positive and negative distances crosses a threshold, T, the softplus loss starts to experience diminishing returns. The framework does not gain many rewards for further separating these distances. The functions that we incorporate in the second and third terms are specifically constructed in order to complement the softplus term, so the overall loss function continues to be rewarded after the threshold, T, is reached. Unconventionally, we add these terms separately. This allows us to tune weights depending on how important intra- and inter-class distances are to the task at hand. By tuning these parameters, we can obtain a trade-off which provides the optimal loss function for the unification of person and vehicle re-ID.

The second term focuses on pulling together the positive pairs. We design the function $\Psi(D) = \psi^D - 1$, where $\psi > 1$ is a constant, in order to heavily punish large distances between positive pairs as seen in Figure 4.1(b). This forces the network to pull images from the same class together during training in order to keep the loss minimal. Note that this is especially important in the vehicle domain and one of the reasons why our network is robust to the shape changes that are incurred when the same vehicle is viewed from different pose angles.

The third term works similarly but aims to push negative images away from each other. We adopt $\Phi(D) = \frac{\phi}{\phi+D}$ to punish negative pairs with small distances and reward pairs with large distances as seen in Figure 4.1(c).

It is intuitive to see that as the intra-class distance decreases and the inter-class distance increases through the second and third terms, the term inside the softplus function will become a larger negative number which results in a smaller loss (Figure 4.1 (a)).

4.2.2 Sample Mining

One of the most important elements of building a framework which utilises a triplet loss function is the effective mining of triplets. We require it to effectively match vehicles with significant distortions in shape, thus it is imperative that the model is trained on the most difficult samples available.

Let p be the identity of the image $x_{p,i}$ in the batch, B, and let $f(x_{p,i})$ be its feature vector, where $p = 1, \dots, P$ and $i = 1, \dots, 4$. Each query image $x_{p,i}$ is paired with its hardest positive image x^+ and hardest negative image x^-, where:

$$x^+ = \max_{x_{q,j} \in B} \left(||(f(x_{p,i}) - f(x_{q,j})||_2^2 \right), \qquad \text{such that } p = q, \tag{4.2}$$

$$x^- = \min_{x_{q,j} \in B} \left(||(f(x_{p,i}) - f(x_{q,j})||_2^2 \right), \qquad \text{such that } p \neq q. \tag{4.3}$$

Together, we obtain the triplet $t_{p,i} = (x_{p,i}, x^+, x^-)$ and these form the set of triplets, $\mathcal{T}$, where $|\mathcal{T}| = 4P$.

In a similar manner, we scan across the entire distance matrix to find the set of hardest positive pairs, $\mathcal{H}^+$, and the set of hardest negative pairs, $\mathcal{H}^-$, with $|\mathcal{H}^+| = |\mathcal{H}^-| = 4P$.

$$x_h^+ = \max_{x_{t,l} \in B} \left(||(f(x_{s,k}) - f(x_{t,l})||_2^2 \right), \qquad \text{such that } s = t, \tag{4.4}$$

$$x_h^- = \min_{x_{v,n} \in B} \left(||(f(x_{u,m}) - f(x_{v,n})||_2^2 \right), \qquad \text{such that } u \neq v. \tag{4.5}$$

Finally, we obtain all hardest samples for computation our loss function

$$(x_{p,i}, x^+, x^-, x_{s,k}, x_h^+, x_{u,m}, x_h^-),$$

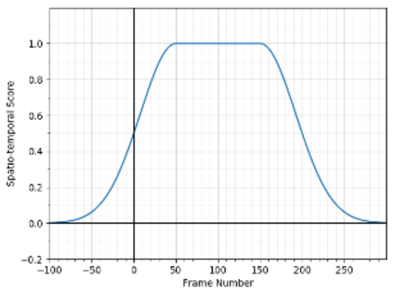
where

$$x_{p,i} = x_0; x^+ = x_0^+; x^- = x_0^-; x_{s,k} = x_1; x_h^+ = x_1^+; x_{u,m} = x_2; x_h^- = x_2^-$$

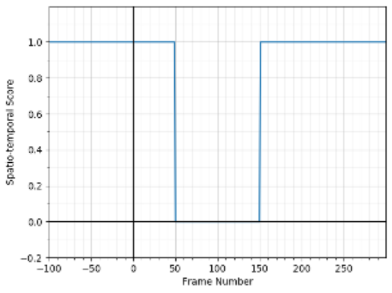
according to Equation 4.1.

4.2.3 Multi-camera Tracking

We solved the re-identification problem by using the residual neural network [54] with our proposed loss function. Our model returns the ten most similar images for each probe image (according to object identity). For multi-camera tracking, we only need to take the most similar image to the probe image and assign the tracks of these two images together. The input to the model could be the whole track and not only an image, therefore, we can use this to our advantage and use information from all images on the track. Multi-camera tracking could also be referred to as image-to-track re-identification [19] because we re-identify a track with the given probe image. Most of the state-of-the-art methods for image-to-track re-identification use the maximum metric. The distances between the probe image and all images from a track are calculated and the image with the maximum similarity (minimum distance) is chosen as the representative image of the track. We propose to use the mean metric instead of the maximum. The distances between the probe image and all images from a track are calculated in the same manner but we instead calculate the mean score which represents the track. The mean metric allows us to consider information from the entire track, whilst taking the maximum considers only one image from the track, so we lose a lot of useful information. Moreover, taking the maximum could lead to an incorrect result because the maximum image could be a false positive, therefore not an optimal representation of the track. Using the mean metric, we managed to obtain promising results further discussed in Evaluation chapter 5.



(a) : Function for calculation spatio-temporal score for close cameras.



(b) : Function for calculation spatio-temporal score for cameras far away.

Figure 4.2: Functions to calculate spatio-temporal score.

4.2.4 Re-identification with spatio-temporal constraints

We also implemented the spatio-temporal constraints into the image-to-track evaluation. The scores based on a visual aspect are calculated the same way as for image-to-image case and then the spatio-temporal scores are calculated separately.

It is a hard problem to focus on spatio-temporal constraints because of many reasons. Firstly, in real life, we usually do not know the distances between the cameras. Even if we know the coordinates of cameras, we have to consider road distances and not air distances. This is a solvable problem, for example by using separate deep learning system, dedicated to this task.

the deep learning could be used to learning the distances between the cameras. But the cameras could have unsynchronised time and also different fps (frames per second). Therefore even if we know the distances between cameras, it is a problematic task to synchronise the camera settings. When we solve all these problems we are dealing with a next problem. The spatio-temporal constraints depend on a complexity of the camera system.

We divided the calculation of the spatio-temporal score into two parts according to their overlapping views and how close they are to each other. The threshold t

indicates the number of frames when the cameras are not close to each other anymore. Cameras are close when there is no place for changing direction between their views or there is a low probability that an object stops between the camera views. It leads to the expectation that the frame delay d will be stable and consistent. The shortest time when the object disappears in the first camera and appear in the second camera is recalculated into the frame delay d. Therefore the frame delay d expresses the number of frames when an object is in a non-monitoring area between the cameras. We hold frame delays for each pair of the cameras. The cameras are close to each other when $d \leq t$.

The first part of calculation focuses on close cameras. Objects, which are in the camera view in the "correct" time, are given a higher score than the others. The visualization is in Figure 4.2a. The query frame $q_f = 100$ and delay $d = 50$. Therefore the function consists of two Gaussian functions and the part in the middle of the Gaussians. One Gaussian has the center in $c_1 = q_f - d = 50$ and the second in the center $c_2 = q_f + d = 150$. Frame numbers in the figure indicate a frame number of an image in gallery g_f. If the frame number of the gallery image is between 50 and 150, it means that the image has a higher probability to be from the same object as the query image because the cameras have overlapping views (or they are really close to each other). Therefore the object should be at the similar time in the view of both cameras.

The second part of calculation focuses on cameras which are farther away from each other. Then we cannot rely on estimated time when an object will appear in the next camera because the object can stop between the cameras, for example, for one hour and then continue. The goal is the opposite of the first case. We penalise images which are at the same time in the second camera because these images, for sure, cannot be from the same object (when the cameras are far away from each other). A visualisation of the concept is in Figure 4.2b. Query frame is $q_f = 100$ and the delay is $d = 50$. Frame numbers indicate a frame number of gallery image g_f.

We design our final function from previous concepts to calculate spatio-temporal score as follows:

$$\mathrm{f}_{\mathrm{ts}}(g_f, q_f, d) = \begin{cases} h \exp\left(-\frac{4\ln(2)(g_f-(q_f-d))^2}{\sigma^2}\right), & (d \leq t) \wedge (g_f \leq (q_f - d)), \\ h \exp\left(-\frac{4\ln(2)(g_f-(q_f+d))^2}{\sigma^2}\right), & (d \leq t) \wedge ((q_f + d) \leq g_f), \\ 0, & (t \leq t) \wedge ((q_f - d) \leq g_f) \wedge (g_f \leq (q_f + d)), \\ h, & \text{otherwise}, \end{cases} \tag{4.6}$$

$$\sigma = \frac{\max_{g_{f,i} \in B}(g_{f,i}) - \min_{g_f \in B}(g_f)}{\delta}, \tag{4.7}$$

where $g_f, q_f, t,$ and d are frame number of a query image, frame number of a gallery image, threshold of the close cameras and frame delay respectively. h indicates an upper bound of the spatio-temporal score values. Finally, δ is ratio how wide the Gaussian should be in relation to a total number of frames in a video. $\delta = 10$ is used in our experiments. The final score is calculated from the original score based only on visual aspect and from the spatio-temporal score.

$$\text{score} = \text{visual\_score} + \gamma\, \text{spatio\_temp\_score}, \tag{4.8}$$

where γ is constant giving ratio value to the spatio-temporal constraint.

4.3 Implementation

The proposed solution is implemented in Python 3.5 with OpenCV 3.4 and TensorFlow 1.6. OpenCV (Open Source Computer Vision Library) was created for computational efficiency. The library focuses on real-time applications. It can take advantage of multi-core processing because it is written in optimised C/C++. [55]

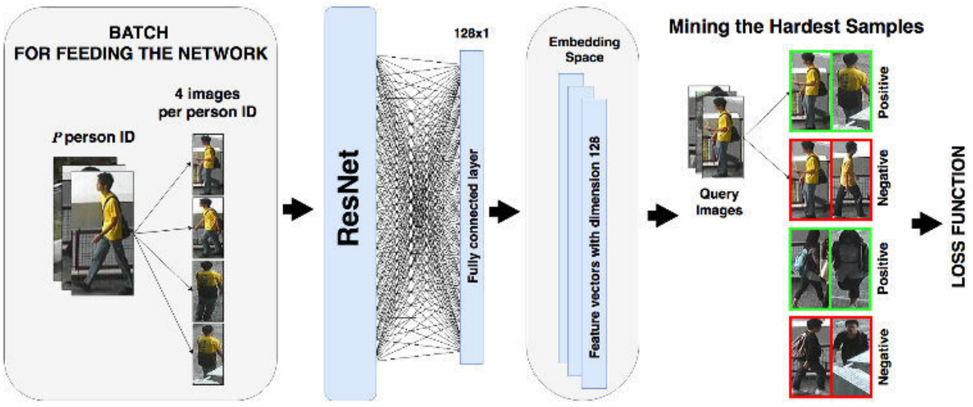


Figure 4.3: An overview of the architecture with class distance engineering. Each batch of images is processed with ResNet [54] to obtain feature vectors. We then mine the hardest triplet for each query image and the hardest $4P$ positive and negative pairs. Next, we feed these hardest samples into our novel loss function to force the network to maximise the inter-class distances and minimise the intra-class distances.

TensorFlow is an open source software library for high-performance numerical computation, mainly developed for machine learning and deep learning. [56]

We present our architecture in Figure 4.3. The input batch is generated by taking four images of P identities. In our implementation, we use ResNet50 (residual network) [54] from TensorFlow library slim 2.3.2 as the backbone of the model that maps the input batch into the embedding space. The distance matrix between all feature vectors is then calculated and the hardest samples are mined. These samples are then fed into our novel loss function.

The available data sets are smaller than ideal for deep learning and the images in the data sets are of low resolution. This could potentially lead to the model overfitting. Across our experiments, we found that removing two units of the `conv5_x` block of ResNet produced optimal results.

Chapter 5

Evaluation

This chapter describes the testing datasets, which were used in this thesis. Afterward, it illustrates the different approaches and their results, which will be compared with state of the art. The evaluation was run in two phases. The first evaluation is result of the implementation for object re-identification (image-to-image re-identification, explained in Section 2.7) and the second evaluation is calculated on multiple cameras tracking (image-to-track re-identification, explained in Section 2.7).

5.1 Datasets

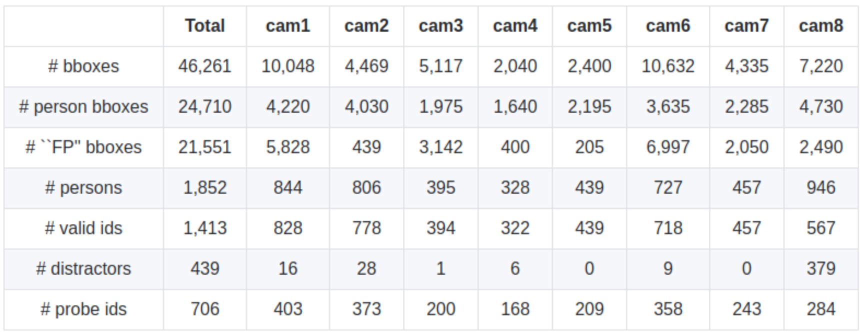
There is more datasets available for object re-identification than for track re-identification. Therefore this implementation was tested and tuned firstly for image-to-image re-identification and then for image-to-track re-identification. Moreover, there are not so many datasets for vehicle re-identification neither. We evaluated image-to-image re-identification on CUHK03 [16], DukeMTMC4REID [17], Market1501 [18], VeRi [19], and VehicleReId [20]. The image-to-track is evaluated on VeRi and VehicleReId because rest of the datasets do not contain tracks. Both, VeRi and VehicleReId provide also the frame number of each image and distances between cameras. Therefore we can apply our spatio-temporal method described in 4.2.4.

5.1.1 Person re-identification datasets

The implementation was trained and tested on **Market1501** dataset [18], which is collected in front of a supermarket in Tsinghua University. The dataset contains 32,668 annotated bounding boxes of 1,501 identities from six cameras. Overlap view exists among different cameras. Each person identity is present in at least two cameras, therefore the cross-camera search can be performed. The dataset provides already separated data into train set, test set, and query set. Each person identity may have multiple images under each camera and also there are multiple queries and multiple ground truths for each identity. The dataset contains three types of the bounding boxes, called "good", "junk", and "distractor". The ratio of the overlapping area to the union area (of the object and rest of the bounding box) is larger than 50%, the bounding box is marked as "good". If the ratio is $20 - 50\%$, the bounding box is marked as "junk", otherwise, it is marked as "distractor". The "junk" image should not have an influence on the re-identification accuracy. For testing, we use the single query (explained in Section 2.7) setting as this is a more difficult task and a more likely challenge to occur in the real world scenario.

The **CUHK03** [16] dataset contains 13164 bounding boxes of 1360 persons. It contains two settings: one with manually annotated bounding boxes and one with automatically detected bounding boxes. We evaluate our method in both settings using the most popular split of 1260 identities for training and the remaining 100 for testing.

Evaluation of our re-identification implementation was tested also on **DukeMTMC4REID** 2017 dataset (Duke Multi-Target, Multi-Camera Tracking Project) [17]. The dataset provides videos from 8 synchronised cameras in the front of Duke University campus. The statistic of the dataset is summarised in Figure 5.1. We are using predefined separation between train and test set.



	Total	cam1	cam2	cam3	cam4	cam5	cam6	cam7	cam8
# bboxes	46,261	10,048	4,469	5,117	2,040	2,400	10,632	4,335	7,220
# person bboxes	24,710	4,220	4,030	1,975	1,640	2,195	3,635	2,285	4,730
# ``FP" bboxes	21,551	5,828	439	3,142	400	205	6,997	2,050	2,490
# persons	1,852	844	806	395	328	439	727	457	946
# valid ids	1,413	828	778	394	322	439	718	457	567
# distractors	439	16	28	1	6	0	9	0	379
# probe ids	706	403	373	200	168	209	358	243	284

Figure 5.1: Statistics of DukeMTMC4ReID 2017 dataset [17].

5.1.2 Vehicle re-identification datasets

The **VeRi** [19] dataset has 37,781 images of 576 vehicles for training and 11,579 images of 200 vehicles for testing. The dataset provides training and test sets separation as well as track test set. It contains information about camera distances and frames for each image. Therefore this dataset is suitable for image-to-image re-identification but also for image-to-track re-identification. Moreover, the camera system is robust and challenging for a re-identification task, see Figure 5.2. The standard procedure for computing the similarity between a query image and a track is to calculate the similarity between the query image and all images on the track and then to simply take the maximum. We improve this procedure and return the mean similarity of the entire track. We argue that the standard procedure is prone to outliers of similarities on the track which may provide an unrealistic representation of the track as a whole. In the real world scenario, our method is much more robust, particularly with respect to potential occlusion as cars pass one another. We provide results for the mean and maximum similarities for comparison.

Vehicle Re-Identification [15] dataset is created from two different cameras turned to the same point of a highway. The dataset contains five pairs of videos, where each pair of videos is from two cameras extracted at the same time. Each video has responding annotation file, where are coordinates of 3D boxes around cars and their ID.

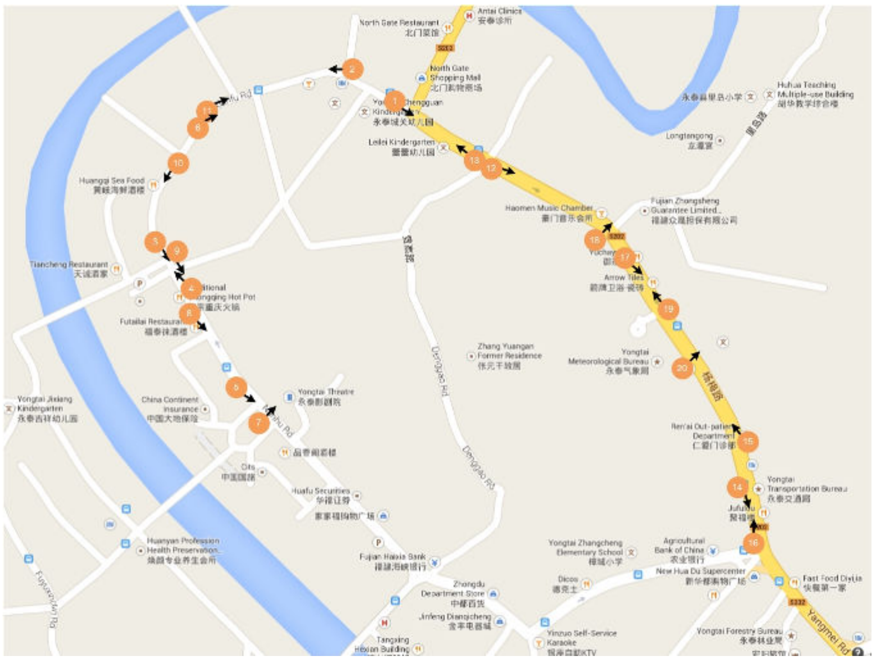


Figure 5.2: Camera system in VeRi data set. [19].

5.2 Comparison with State of the Art

CUHK03: Our results on both the labelled and detected CUHK03 data set are presented in Table 5.1. All methods consistently show stronger performance on the labelled CUHK03 data set than on the detected one. It is intuitive to see that the detected data set is a more complex challenge as the automated detections are more likely to include misalignment of bounding boxes, missing body parts, and occlusion. Nevertheless, our results on the detected data set emphatically exceed the other state-of-the-art methods on the labelled data set. This categorically demonstrates the effectiveness of our method.

In particular, our method outperforms the mAP of the popular LSRO [61] by a comfortable 6.0% on the detected data and we exceed the second best rank-1 score by 2.6%.

CUHK03 Labelled			**CUHK03 Detected**		
Method	mAP	rank-1	Method	mAP	rank-1
JLML [57]	-	83.2	JLML [57]	-	80.6
JAL [58]	-	89.5	JAL [58]	-	88.4
FT-JSTL+DGD[59]	-	75.3	CRAFT [60]	72.4	84.3
DPFL [6]	82.8	86.7	DPFL [6]	78.1	82.0
			LSRO [61]	87.4	84.6
Ours	88.0	90.5	**Ours**	87.2	88.5
Ours + re-ranking	**92.2**	**91.5**	**Ours + re-ranking**	**93.4**	**91.0**

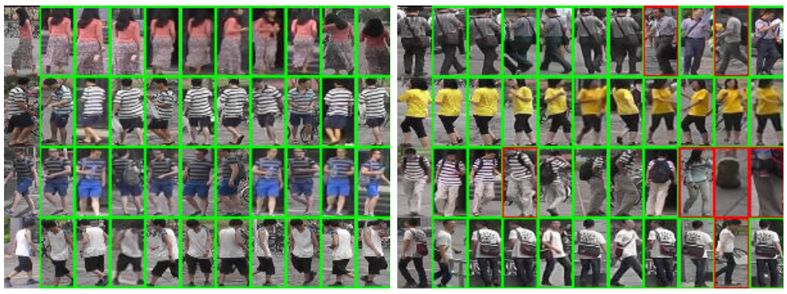
Table 5.1: Comparison with state-of-the-art methods on the CUHK03 labelled and detected data sets.

Market-1501 (Single Query)					
Method	mAP	rank-1	Method	mAP	rank-1
JLML [57]	65.5	85.1	JAL [58]	80.3	**92.1**
DeepTransfer [62]	65.5	83.7	TriNet [3]	81.1	86.7
DaF [63]	72.42	82.30	RE [64]	83.9	89.1
DPFL [6]	73.1	88.9	**Ours**	72.4	87.5
HA-CNN [65]	75.7	91.2	**Ours + re-ranking**	**84.4**	89.8

Table 5.2: Comparison with state-of-the-art methods on the Market-1501 data set with the single query setting.

It is worth noting that our performance on the detected data set after re-ranking exceeds even our performance on the labelled data set. This proves our model is robust and can be applied successfully to real-world scenarios.

Market-1501: Our results on the Market-1501 dataset can be found in Table 5.2. We attain state-of-the-art results, outperforming Random Erasing by 0.5%. We present our qualitative results in Figure 5.3; the highest scoring query images are shown in Figure 5.3(a) and the median query images are shown in Figure 5.3(b). In particular, it can be seen that the framework is successful at distinguishing people even when wearing similar clothing (Figure 5.3(a) row 2 vs. 5.3(b) row 3). This can be attributed to the hard negative pairs that are mined during training and is further evidence of the effectiveness of our loss function.



(a) : Best rank-10 results on Market-1501.

(b) : Median rank-10 results on Market-1501.

Figure 5.3: Qualitative results of our framework on the Market-1501 data set. The left column of each image is the query and the remaining ten images represent the highest ranked images returned by our framework. A green border indicates a ground truth while a red border indicates a different identity from the query.

DukeMTMC4REID				
Method	mAP	rank-1	rank-5	rank-10
LOMO+XQDA [66]	13.5	27.9	-	-
GOG+kLFDA$_{\text{exp}}$ [17]	27.0	49.6	-	-
Ours	49.05	65.83	80.66	85.90
Ours + re-ranking	**67.93**	**71.25**	**81.16**	**84.80**

Table 5.3: Comparison with image-to-image methods on the DukeMTMC4REID data set

DukeMTMC4REID: We do not compare our results with state-of-the-art methods because we used the newest version of the Duke dataset and there are no state-of-the-art results yet. We can compare our results with the results provided by Gou et al. [17] whose created the DukeMTMC4REID dataset, Table 5.3. We outperformed the used methods by 20.93% in mAP and 21.65% in rank-1.

VeRi: We present our results image-to-image and also image-to-track on the VeRi dataset in Table 5.4 and 5.5 respectively. Our method clearly outperforms the state of the art at the vehicle re-ID task. When we follow standard procedure for image-to-track re-identification and take the maximum similarity of the track images with the query, we obtain a mAP of 8.23% higher than the next best result. When we take the mean similarity of the track, we increase our mAP by a further 3.36% and achieve better scores at each rank. When we apply our spatio-temporal

VeRi (image-to-image)				
Method	mAP	rank-1	rank-5	rank-10
VGG+C+T+S [12]	57.4	86.59	92.85	-
VGG+C+T [12]	58.78	86.41	92.91	-
Ours	66.05	88.62	95.05	96.78
Ours + re-ranking	**70.99**	**90.29**	**93.92**	**96.25**

Table 5.4: Comparison with state-of-the-art image-to-image methods on the VeRi data set for the multiple query setting.

VeRi (image-to-track)				
Method	mAP	rank-1	rank-5	rank-10
XVGAN [9]	24.65	60.20	77.03	-
NuFACT [7]	48.47	76.76	92.79	-
PROVID [7]	53.42	81.56	95.11	-
GSTE* [13]	59.47	53.51	-	-
Ours (max)	67.70	86.23	95.77	97.79
Ours (mean)	71.06	88.86	96.19	98.09
Ours (spatio-temporal)	**77.09**	**92.79**	**97.97**	**99.05**

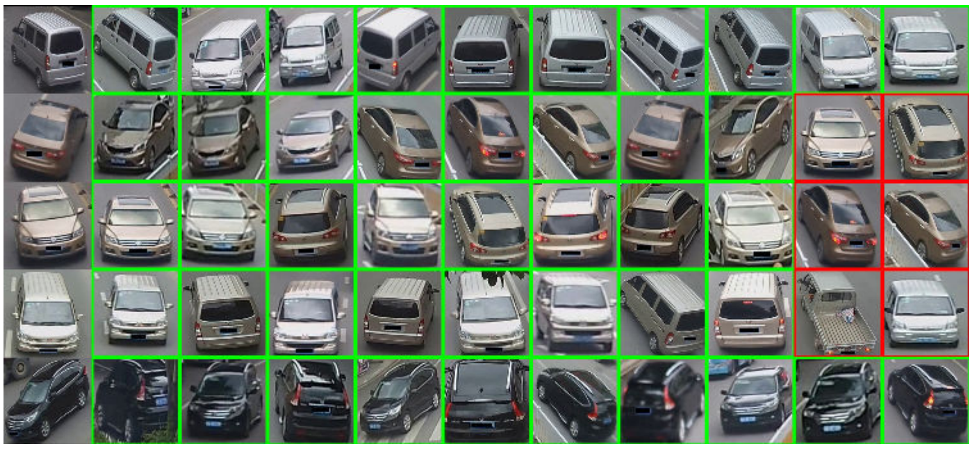
Table 5.5: Comparison with state-of-the-art image-to-track methods on the VeRi data set for the multiple query setting. *We compare with their CMC(1) result.

method the results are even better, more precisely, the mAP increases by another 6.03%. The threshold $t = 500$ is used for the spatio-temporal method to calculate re-identification on the VeRi dataset. This demonstrates both the consistency of our framework and the advantage of using the mean similarity of the track over the maximum. We present our qualitative results in Figure 5.4; the highest scoring query images are shown in Figure 5.4(a) and the median query images are shown in Figure 5.4(b).

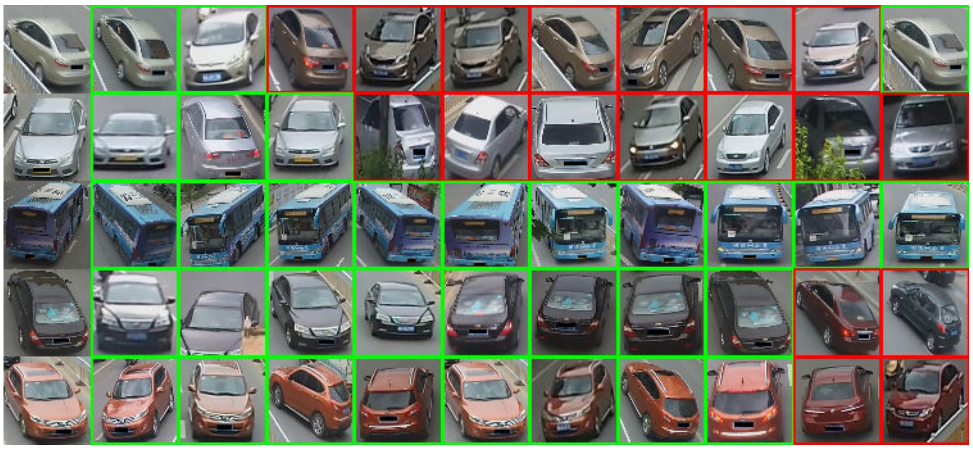
VehicleReId: We did not find any state-of-the-art methods testing on the VehicleReId dataset and also there is not exact approach how to split the train and test set from the dataset, therefore, we only provide results of our method in Table 5.6. The threshold $t = 1500$ is used for the spatio-temporal method to calculate re-identification on the VehicleReId dataset.

VehicleReId				
Method	mAP	rank-1	rank-5	rank-10
Ours image-to-image	82.95	90.93	95.03	96.45
Ours image-to-image + re-ranking	84.71	86.29	94.3	95.03
Ours image-to-track (max)	77.68	67.84	89.51	93.36
Ours image-to-track (mean)	92.51	88.75	96.3	97.77
Ours image-to-track (spatio-temporal)	**93.72**	**90.50**	**97.47**	**98.56**

Table 5.6: Results of our methods on the VehicleReId dataset.



(a):



(b):

Figure 5.4: Qualitative results of our performance on the VeRi data set using the mean similarity of the track with the query image. The left column of each image is the query with a unique identity and the remaining ten images are the image representations of the highest ranked tracks. A green border indicates a ground truth while a red border indicates a different identity from the query.

Chapter 6

Conclusion

In this thesis, the real-world problem of single object identity tracking has been summarised, by finding its most likely match throughout cameras of the system by visual matching and also with spatio-temporal constraints. Furthermore, state-of-the-art methods in this field have been discussed. We have developed a framework which can generalise across both person and vehicle re-identification tasks. We have created a novel loss function, which incorporates specifically designed terms to maximise the inter-class distance and minimise the intra-class distance. Our sample mining techniques, which are tailored to the design of the loss function, are implemented as well. Also, a new spatio-temporal method to improve the final ranking has been presented. We have suggested and implemented a method, which finds top-n likeliest matches for a given object throughout the system, in Python with OpenCV 3.

We demonstrate the efficacy of the network by testing it on popular data sets for both person and vehicle re-identification. Satisfactory performance across both tasks was not previously considered feasible. Not only does our framework perform adequately, but it obtains state-of-the-art results on *both* tasks with no change to the network architecture. Therefore we have completed all assignments of the thesis.

6.1 Future work

As we have established that this architecture can generalise over the two tasks, a natural potential research direction is amalgamating the tasks, training the same framework on data sets from both domains and performing vehicle and person re-ID simultaneously. Additional future work could be done based on the spatio-temporal method. For example, the frame delay between cameras can be automatically established by deep learning. This could be achieved using an end-to-end architecture of neural networks, created to train images with information about frame number and camera id. Then the neural network should be able to learn spatio-temporal information by itself. Research how to feed the network with this information and how to design a loss function has to be done.

Appendix A

Bibliography

[1] M.Shah S. Khan. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *PAMI*, 2003.

[2] M. Joned E. Ahmed and T. K. Marks. An improved deep learning architecture for person re-identification. *IEEE*, 2015.

[3] Lucas Beyer Alexander Hermans and Bastian Leibe. In defense of the triplet loss for person re-identification. *CVPR*, 2017.

[4] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. A Multi-task Deep Network for Person Re-identification. pages 3988–3994, 2016.

[5] Jianguo Zhang Kaiqi Huang Weihua Chen, Xiaotang Chen. Beyond triplet loss: a deep quadruplet network for person re-identification. *CVPR*, 2017.

[6] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Person Re-identification by Deep Learning Multi-scale Representations. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, (iii):2590–2600, 2017.

[7] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. PROVID: Progressive and Multimodal Vehicle Reidentification for Large-Scale Urban Surveillance. *IEEE Transactions on Multimedia*, 20(3):645–658, 2018.

[8] Yi Zhou and Ling Shao. Cross-View GAN Based Vehicle Generation for Re-identification. *BMVC*, 1:1–12, 2017.

[9] Yi Zhou, Li Liu, and Ling Shao. Vehicle Re-Identification by Deep Hidden Multi-View Inference. 27(7):3275–3287, 2018.

[10] Mahmood Ashoori Lalimi, Sedigheh Ghofrani, and Des McLernon. A vehicle license plate detection method using region and edge based methods. *Computers & Electrical Engineering*, 39(3):834 – 845, 2013. Special issue on Image and Video Processing Special issue on Recent Trends in Communications and Signal Processing.

[11] Wu Liu, Xinchen Liu, Huadomg Ma, and Peng Cheng. Beyond Human-level License Plate Super-resolution with Progressive Vehicle Search and Domain Priori GAN. *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, pages 1618–1626, 2017.

[12] Yiheng Zhang, Dong Liu, and Zheng Jun Zha. Improving triplet-wise training of convolutional neural network for vehicle re-identification. *Proceedings - IEEE International Conference on Multimedia and Expo*, (July):1386–1391, 2017.

[13] Yan Bai, Yihang Lou, Feng Gao, Shiqi Wang, Yuwei Wu, and Lingyu Duan. Group Sensitive Triplet Embedding for Vehicle Re-identification. *IEEE Transactions on Multimedia*, 9210(c):1–14, 2018.

[14] Andrea Prati Marcello Pelillo Yonatan Tariku Tesfaye, Eyasu Zemene and Mubarak Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. `http://vision.cs.duke.edu/DukeMTMC/`. [Online; accessed 22-January-2018].

[15] Vehiclereid - vehicle re-identification for automatic video traffic surveillance. `https://medusa.fit.vutbr.cz/traffic/research-topics/detection-of-vehicles-and-datasets/vehicle-re-identification-for-automatic-video-traffic-surveillance-ats-cvpr-2016/`, 2016. [Online; accessed 22-January-2018].

[16] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[17] Mengran Gou, Srikrishna Karanam, Wenqian Liu, Octavia Camps, and Richard J. Radke. Dukemtmc4reid: A large-scale multi-camera person re-identification dataset. July 2017.

[18] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015.

[19] Ma H. Fu H Liu X., Liu W. Large-scale vehicle re-identification in urban surveillance videos. *IEEE*, 2016.

[20] Dominik Zapletal, Adam Herout, and Adam Herout Graph@fit. Vehicle Re-Identification for Automatic Video Traffic Surveillance. *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1568–1574, 2016.

[21] Daniel Organisciak Edmond S. L. Ho Nauman Aslam Katerina Jandova, Dimitris Sakkos and Hubert P. H. Shum. Triplet loss with class distance engineering for person and vehicle re-id. `https://github.com/bmvc-reid/re-id`, 2018. [Online; accessed 10-May-2018].

[22] Volker Eiselein Erik Bochinski and Thomas Sikora. High-speed tracking-by-detectionwithout using image information. *Communication System Group, Technische Universitat Berlin, IEEE*, 2017.

[23] K. Shafique O. Javed, Z. Rasheed and M. Shah. Tracking across multiple cameras with disjoint views. *University of Central Florida, IEEE*, 2003.

[24] Hsu-Yung Cheng LiangJia Zhu, Jenq-Neng Hwang. Tracking of multiple objects across multiple cameras with overlapping and non-overlapping views. *Circuits and Systems, ISCAS, IEEE*, 2009.

[25] M. Barkowsky U. Fecker and A. Kaup. Improvingthe prediction efficiency for multi-view video coding usinghistogram matching. *Picture Coding Symposium (PCS 2006), Beijing, China*, 2006.

[26] ComputerVisionFoundation Videos. `https://www.youtube.com/watch?v=_o2SLgjejAE`. [Online; accessed 15-February-2018].

[27] `https://deeplearning4j.org/neuralnet-overview`. [Online; accessed 15-February-2018].

[28] `https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/`. [Online; accessed 10-May-2018].

[29] University of Edinburgh. http://www.inf.ed.ac.uk/teaching/courses/mlp/lectures.html. [Online; accessed 22-January-2018].

[30] Francois Duval. *Artificial Neural Networks: Concepts, Tools and Techniques explained for Absolute Beginners (Data Sciences)*. 2017.

[31] MathWorks. https://uk.mathworks.com/discovery/convolutional-neural-network.html. [Online; accessed 1-February-2018].

[32] Justin Johnson. https://uk.mathworks.com/discovery/convolutional-neural-network.html. [Online; accessed 29-January-2018].

[33] Vincent Vanhoucke-Alexander A. Alemi Christian Szegedy, Sergey Ioffe. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI*, 2017.

[34] John Langford-Robert Schapire Furong Huang, Jordan Ash. Learning deep resnet blocks sequentially using boosting theory. 2017.

[35] Alex Kot Ze Lu, Xudong Jiang. Deep coupled resnet for low-resolution face recognition. *IEEE*, 2017.

[36] Jian Sun Kaiming He. Convolutional neural networks at constrained time cost. *CVPR*, 2014.

[37] Shaoqing Ren-Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *CVPR*, 2015.

[38] Sebastian Ruder. http://ruder.io/optimizing-gradient-descent/. [Online; accessed 10-February-2018].

[39] Guangrun Wang-Hongyang Chao Shengyong Ding, Liang Lin. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 2015.

[40] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:3652–3661, 2017.

[41] Victor Lavrenko. https://www.youtube.com/watch?v=pM6DJ0ZZee. [Online; accessed 2-April-2018].

[42] Wei-Shi Zheng, Gong Shaogang, and Xiang Tao. Person re-identification by probabilistic relative distance comparison. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 649–656, 2011.

[43] Peter M. Roth, Martin Hirzer, Martin Köstinger, Csaba Beleznai, and Horst Bischof. *Mahalanobis Distance Learning for Person Re-identification*, pages 247–267. Springer London, London, 2014.

[44] Song Bai and Xiang Bai. Sparse contextual activation for efficient visual re-ranking. *IEEE Transactions on Image Processing*, 2016.

[45] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS:3–20, 2016.

[46] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:815–823, 2015.

[47] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[48] 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. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1335–1344, 2016.

[49] S. Jegelka H. O. Song, Y. Xiang and S. Savarese. Deep metric learning via lifted structured feature embedding. *CVPR*, 2016.

[50] L. Bossard T. Quack D. Qin, S. Gammeter and L. Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. *CVPR*, 2011.

[51] Timothee Cour Kai Yu Dimitris N. Metaxas Shaoting Zhang, Ming Yang. Query specific rank fusion for image retrieval. *CVPR*, 2014.

[52] Luc van Gool Danfeng Qin, Christian Wengert. Query adaptive similarity for large scale object retrieval. *CVPR*, 2013.

[53] Jonathan Brandt Shai Avidan Ying Wu Xiaohui Shen, Zhe Lin. Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. *CVPR*, 2012.

[54] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, pages 1–17, 2017.

[55] `https://opencv.org/`. [Online; accessed 22-March-2018].

[56] `https://www.tensorflow.org/`. [Online; accessed 22-March-2018].

[57] Wei Li, Xiatian Zhu, and Shaogang Gong. Person re-identification by deep joint learning of multi-loss classification. *IJCAI International Joint Conference on Artificial Intelligence*, pages 2194–2200, 2017.

[58] Wangmeng Xiang, Jianqiang Huang, Xianbiao Qi, Xiansheng Hua, and Lei Zhang. Homocentric Hypersphere Feature Embedding for Person Re-identification. pages 1–11, 2018.

[59] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning Deep Feature Representations with Domain Guided Dropout for Person Re-identification. 2016.

[60] Ying Cong Chen, Xiatian Zhu, Wei Shi Zheng, and Jian Huang Lai. Person Re-Identification by Camera Correlation Aware Feature Augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):392–408, 2018.

[61] Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in Vitro. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3774–3782, 2017.

[62] Mengyue Geng, Yaowei Wang, Tao Xiang, and Yonghong Tian. Deep Transfer Learning for Person Re-identification. 2016.

[63] Rui Yu, Zhichao Zhou, Song Bai, and Xiang Bai. Divide and Fuse: A Re-ranking Approach for Person Re-identification. pages 1–13, 2017.

[64] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random Erasing Data Augmentation. 2017.

[65] Wei Li, Xiatian Zhu, and Shaogang Gong. Harmonious Attention Network for Person Re-Identification. (I), 2018.

[66] F. Liang T. S. Huang L. Cao Z. Li, S. Chang and J. R. Smith. Learning locally-adaptive decision functions for person verification. 2013.