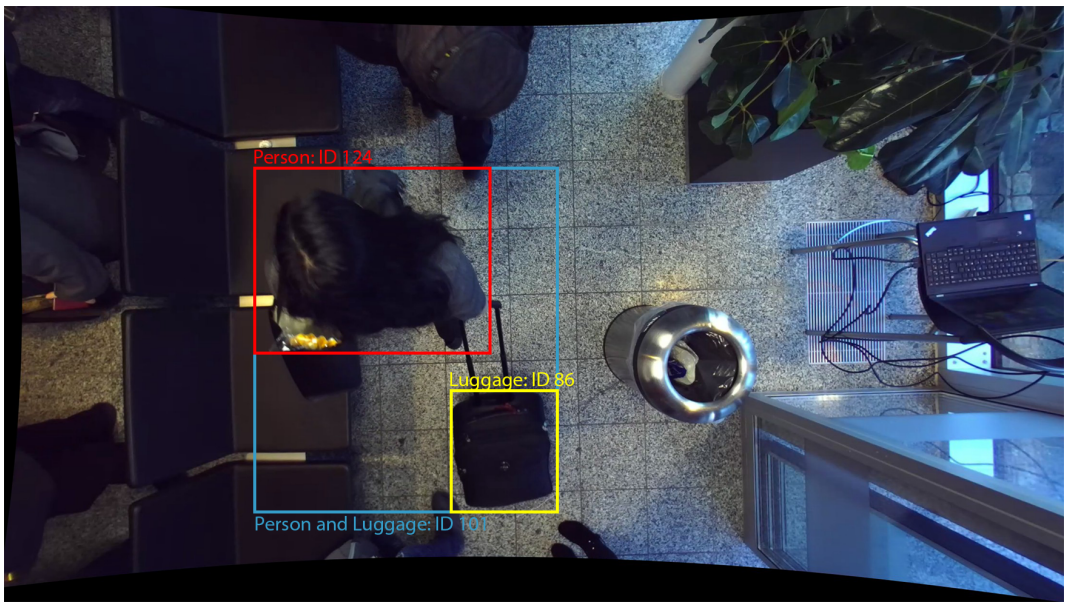
Person re-identification of people and their luggage



VGIS 10

Simon Gørtz Flou Nielsen

Vision, Graphics and Interactive Systems
Aalborg University

AALBORG UNIVERSITY

STUDENT REPORT

Vision, Graphics and Interactive Systems
Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
https://vgis.create.aau.dk/

Title:
Person re-identification of people and their luggage

Theme:
Computer Vision

Project Period:
Spring 2022

Project Group:
VGIS 1047

Participant(s):
Simon G. F. Nielsen

Supervisor(s):
Kamal Nasrollahi

Copies: 0

Page Count: 53

Date of Completion:
June 1, 2022

Abstract:

As travel becomes more and more common, the use of Re-identification systems become more common. For surveillance or queue estimation, Re-identification is a versatile and useful tool. By using Re-identification methods e.g. airports can keep track of passengers and their queue times for boarding and unboarding planes. For a network to distinguish a passengers from each other, a system specifically trained and developed for that purpose needs to be implemented. That is the purpose of this paper, to research person Re-identification within this context. This paper explores a dataset structure containing two bounding boxes, one for the person and another for the luggage they are carrying. This dataset was used throughout the paper, for testing multiple ways of utilising it for Re-identification. One such method tested was a Dual-TransReID network, which reported an accuracy of *95%*. This is an improvement to the baseline test of a single TransReID network, which scored an accuracy of *90%* with a single bounding box containing both person and luggage.
Code base can be found at: https://github.com/MrFlou/Multi-Net-ReID

Preface

This report was written for the fourth semester of the AAU masters program Vision, Graphics and Interactive systems, in the months of February to the 2nd of June of 2022. The focus of this report is to research into Re-identification using machine learning and different structures of datasets. This is programmed using Python 3.7.6 with a selection of libraries, e.g. TensorFlow, PyTorch, Numpy, Pandas, etc.

A big thanks to Aske Lejbølle and Kamal Nasrollahi for their feedback and supervision during the writing of this report.

Group VGIS 1047, Aalborg University, June 1, 2022

Simon Gørtz Flou Nielsen
<sgfn17@student.aau.dk>

Acronyms and abbreviations

Acronym	Definition
AP	Average Precision
BBOX	Bounding Box
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DNN	Deep Neural Network
GD	Gradient Descent
GMM	Gaussian mixture mode
GPU	Graphics Processing Unit
IoU	Intersection-over-Union
mAP	Mean Average Precision
MLP	Multi-Layer Perceptron
R1	Rank-1
Re-ID	Re-identification
RGB	Red Green Blue
RNN	Recurrent Neural Network
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SotA	State of the Art
STD	Standard deviation
YOLO	You Only Look Once

Contents

Part I

Problem analysis

1 - Introduction

Person Re-ID easily gets very difficult in large traveling hubs, e.g. an airport, where thousands of people pass through on a daily basis. Even though, some people may be distinguishable by their clothing, other may wear similar clothing, e.g. a football team in their national colours. In person Identification and Re-ID is gaining more and more attention, as Deep Neural Networks (DNN) prove more and more capable of such tasks. With the increased demand for automated systems that can detect and identify people, many researchers have taken their swing at this topic. In those scenarios Re-ID can prove difficult, with many of the current solutions for Re-ID. After years of research in the field of Re-ID, many improvements have been made, but the problems have not been solved completely [1][2][3][4][5]. Therefore, additional information or a different approach is required to successfully provide a correct Re-ID in those cases. There are multiple avenues to take to provide this additional information. As a majority of people in airports, typically, carries one or two additional carry-ons, using features from those may be useful to create a more descriptive representation of a person. The big question is, how exactly this information should be combined with features from the person. This paper aims to introduce a method that combines a person's identificatory features with additional information gathered from the luggage they carry.

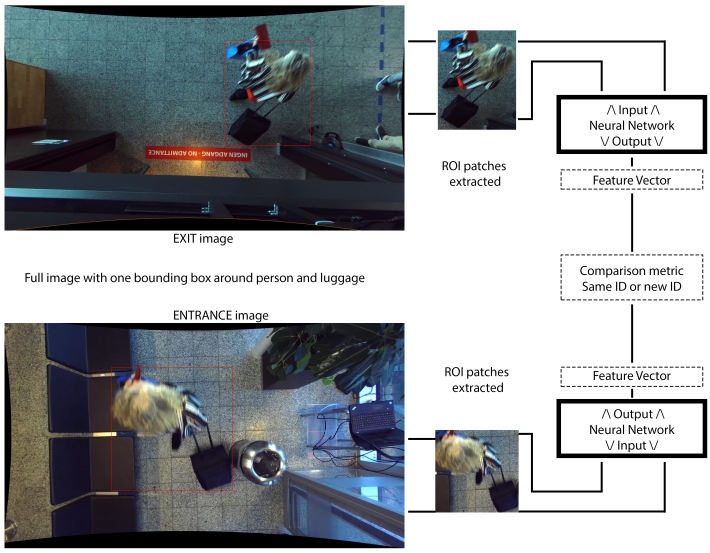


Figure 1.1: This shows the pipeline for the average feature extraction with bounding boxes defining the ROI. To capture the whole person and luggage a lot of area surrounding is also captured within the bounding box.

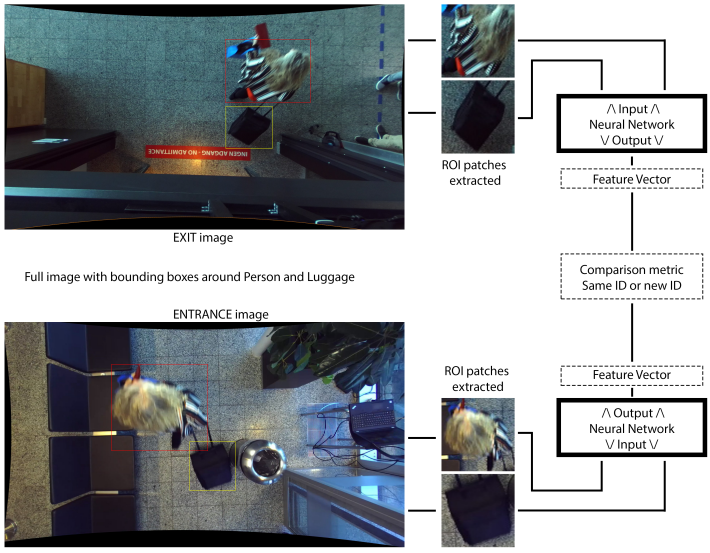


Figure 1.2: This figure shows the bounding boxes for both person and luggage, individually extracted patches. condensing the data given to the networks, to the most important data

1.1 Initial Problem Description

How can a Computer Vision system be used to Re-identify individual people with the use of the luggage they carried?

2 - Previous Solutions

Within this chapter, previous solutions will be analysed with a focus on the methods papers such as TransReID and SimCLR used for improving on problems similar to the ones found for this project. Additionally, the similar problems faced in the previous solutions, will provide an understanding to similar problems faced in this project.

2.1 Person Identification and re-identification

Identifying and understanding the State of the Art solutions (SotA) can help to provide an understanding of similar methods useful for using a person's luggage as an identifying descriptor.
The task of re-identification is to identify a person across two or more cameras, where that person can appear at different times and locations. With this in mind, an ideal system needs to be developed that should in theory be able to handle a wide variety of identification scenarios. Scenarios include but are not limited to: Time of day, illumination conditions, camera angles, resolution and more. To aid in this Re-IDing problem, Region of Interests (ROIs) or bounding boxes are often made for an image containing multiple objects of interest. These ROIs are made by another system, purposely built for finding and drawing bounding boxes around objects of interest. These bounding boxes help by narrowing the available data to only what is perceived relevant for the task at hand.

2.1.1 Methods

There are quite a few methods of identifying people in an image and comparing them to another detection of a person. These methods vary from use case to case. Many of these methods include forms of dimensionality reduction. This section of the paper will explain some of the methods used for Re-ID.

Feature Vector Extraction

The method focused on in this paper is feature vector descriptors, which describes an image within a 1-dimensional array of values. There are multiple methods for extracting these feature vectors from images or other forms of data. This is also

seen as dimensionality reduction, where as the complexity of the data is reduced to hopefully identify similarities between data points. The most common way of extracting feature vectors is through DNNs, such as a Convolutional Neural Network (CNN) for images. These features are usually represented as a n length array, comprised of values which provides a dimensionally reduced representation of the full data.

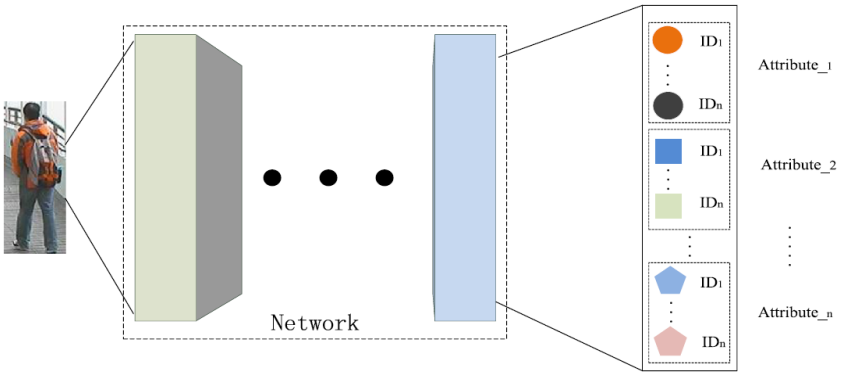


Figure 2.1: A overview to feature extraction from an image though a network developed to extract and classify images to IDs [6].

When extracting feature vectors it is common to use systems/networks tailored to the weight and extract tailored feature vectors for specific proposes. These tailored networks and systems can be trained with methods like supervised triplet-loss training (further explained in section 5.2.1).

MLP: Multilayer Perceptron

Multilayer Perceptron (MLP) is a section of fully connected neurons called a feed-forward neural network. An MLP often simply consists of input, hidden and output layers. MLPs are often the very simple neural networks, that contain the most basic functions almost all more advanced neural networks contain[6]. These elements are Neurons, Weights, Biases and activation functions. A neuron in a network has one bias and a weight pr neuron from the layer it is receiving input from. These weights and biases are trainable variables, that a neural network adjust through training with the optimization formula and loss function. An activation function is often also called a activation layer, this function could be a Rectified Linear Unit Activation function. This function takes an input and if the value is below 0 it gives the output of 0. In this activation function, anything above 0 will have the same output as the input.

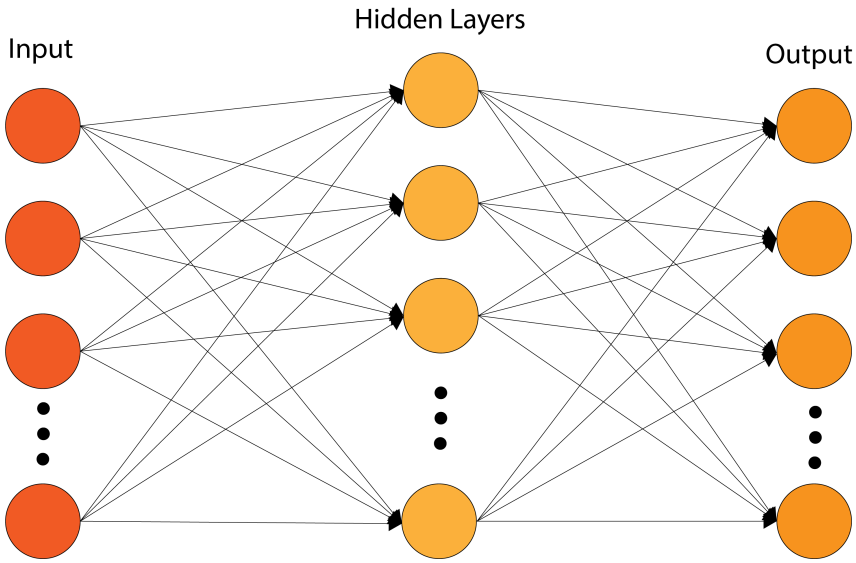


Figure 2.2: An image showing the basics of a Multilayer Perceptron[6]. Illustrated with one hidden layer, though in many cases, there are multiple hidden layers.

The number of hidden layers in MLP networks varies depending on the task the network is given. MLP networks are used for a wide variety of simpler machine learning tasks.

CNN: Convolutional Neural Network

CNN is a method that maintains the spatial information in an image, using convolutional layers, pooling layers and fully-connected layers in the simplest of cases. With the Convolutional layers, it is able to keep spatial information, which is often lost with just fully connected layers. With the help of pooling layers, it is able to learn about larger features as it shrinks the image down in size. There are many architectures out there for CNNs such as: AlexNet[7], VGG-nets[8], GoogLeNet/InceptionNet[9], ResNet[10] and many others[11]. Most of these networks have been used for a wide range of computer vision tasks including Object detection and Re-ID. For most of these networks, there are available pre-trained weights, commonly trained on datasets such as ImageNet[12] or other publicly available datasets. This allows for the practice called Transfer Learning which can improve performance and training time. Transfer learning is when a neural network was trained for one task and uses what it learned to perform another task, with or without additional training[13]. This is typically used if a given dataset is of limited size, as it can improve the effectiveness of the neural network[11].
These extractions of feature vectors can come from full image extractions or ROI extraction. ROI extractions is the method of placing a bounding box around the

object of interest in a larger image, cutting out data to focus on what is of importance to the task at hand.

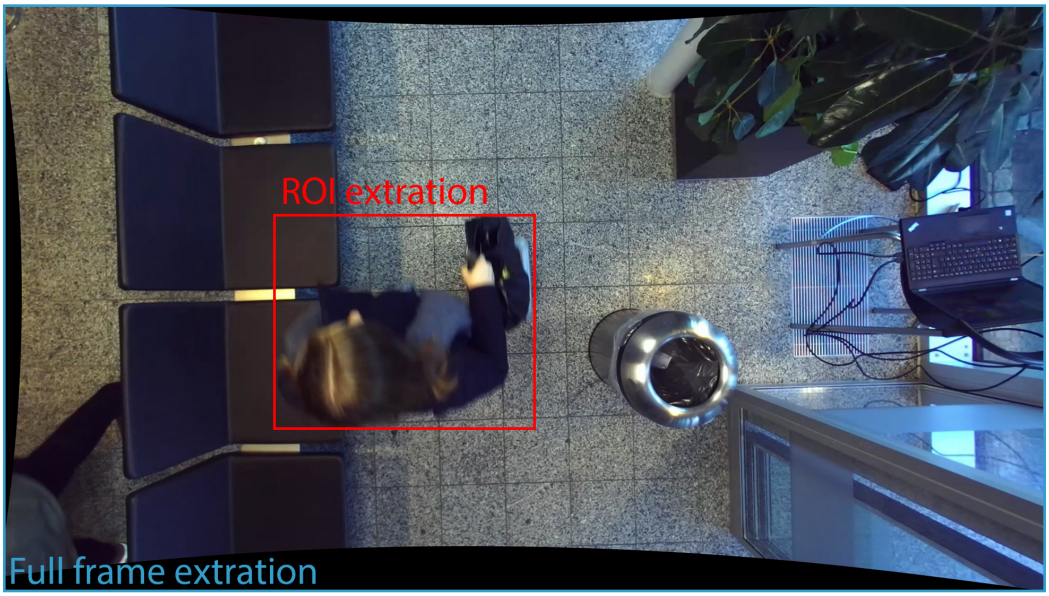


Figure 2.3: Full Frame Extraction in blue and ROI extraction in red

These methods are the backbone of this paper's tests, as they are used to test the systems in the following chapters. Using these methods of extraction for Re-Identification, the following section will look into more SotA methods of Re-identification.

2.2 Baselines

The methods described in the previous section, are very general ways of extracting features. Nowadays, in Re-ID many different proposals exist for this task. Like the "Visibility-Aware" Approach for person Re-Identification by *Yang Et al.*[14], where the system proposed learns to extract information from partly occluded people. Another paper with relevancy for this paper, is the "Partial Person Re-identification" written by *Tianyu Et al.*[15]. This paper propose a system that distinguish people by cropped small parts. With many more proposals out there. With this in mind and the scope of this project, papers with suitable access to code base and relevancy to the topic would be good baselines. Finally two papers where selected; SimCLR[2] and TransReID[4] published in the year 2020 and 2021 respectively. These two papers have an available code base and proper relevancy, further explained in the sections below. These baseline papers would be used for guidance and evaluation of the proposed systems for this paper.

2.2.1 TransReID: Transformer-Based Object Re-Identification

The first baseline paper is one tackling the extraction of features with the use of a transformer-based network "TransReID: Transformer-based Object Re-Identification"[4]. Compared to many previous attempts using CNN networks, this new approach utilises transformer architecture, a newer approach proposed way back in the “Attention Is All You Need” paper[16]. The novel idea at the time was to make a network architecture fully built on attention mechanisms. With this use of attention mechanisms, it was theorised that a network would have a better time generalizing to the data it would be trained on.

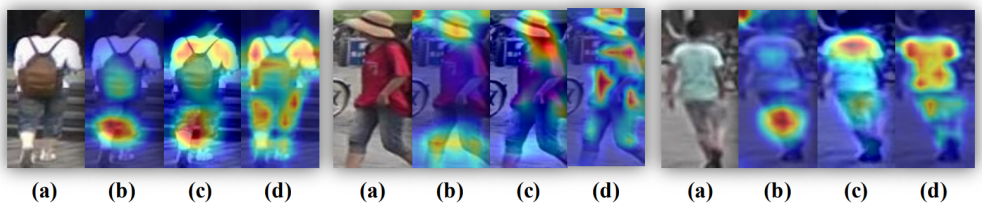


Figure 2.4: Grad-CAM[17] visualization of attention maps, image from TransReID paper[4]: "(a) Original images, (b) CNN-based methods, (c) CNN+attention methods, (d) Transformer-based methods which captures global context information and more discriminative parts."

Framework

The proposed framework consists of a linear projection of flattened patches, this allows the system to handle any given input resolution. This linear projection also includes a learnable position embedding. Included in this side information embedding which includes non-visual information from the camera.

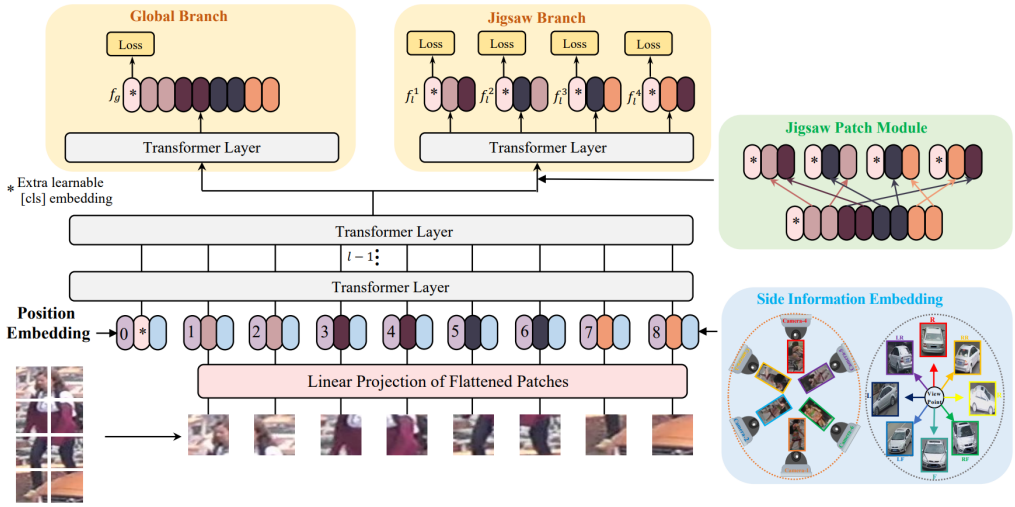


Figure 2.5: Image from TransReID paper[4]: "Framework of proposed TransReID. Side Information Embedding (light blue) encodes non-visual information such as camera or viewpoint into embedding representations. It is input into transformer encoder together with patch embedding and position embedding. Last layer includes two independent transformer layers. One is standard to encode global feature. The other contains the Jigsaw Patch Module (JPM) which shuffles all patches and regroups them into several groups. All these groups are input into a shared transformer layer to learn local features. Both global feature and local features contribute to ReID loss. "

Following these position embeddings comes two transformation layers, where the second one contains a jigsaw patch module. This module tries to adapt and overcome a challenge with transformer networks. Transformer layers is a newer method, which adopts a self-attention mechanism[18]. This self-attention mechanism deferentially weighs parts of its input to guide its attention. The attention mechanism provides context to what order the input data should be interpreted as, which is different from an RNN network that processes data in order[19].

Performance

The TransReID was able to score a mAP of *89.0%* on the market-1501 with a Rank-1 of *95.1%*. TransReID showing great promise for descriptive feature vector extraction to aid in Re-ID. It is shown in the paper for TransReID that the self-attention transform layers, does effectively guide the network to extracting a useful feature vector[4] (see figure 2.5). The TransReID outperformed all other SotA methods, mentioned in the accompanying paper. It outperformed multiple popular person ReID datasets.

Datasets	MSMT17	Market	Duke	OCC_Duke	VeRi
Model	mAP | R1	mAP | R1	mAP | R1	mAP | R1	mAP | R1
Baseline(ViT)	61.8 | 81.8	87.1 | 94.6	79.6 | 89.0	53.8 | 61.1	79.0 | 96.6
TransReID*(ViT)	67.8 | 85.3	89.0 | 95.1	82.2 | 90.7	59.5 | 67.4	82.1 | 97.4
TransReID*(DeiT)	66.3 | 84.0	88.5 | 95.1	81.9 | 90.7	57.7 | 65.2	82.4 | 97.1

Table 2.1: mAP and R1 will be elaborated more on in section 2.5

2.2.2 SimCLR: A Simple Framework for Contrastive Learning of Visual Representations

The second baseline is a take on unlabeled training method, previously demonstrated by Exemplar-CNN[20], Instance Discrimination[21], CPC[22], AMDIM[23], CMC[24], MoCo[25] and many others. Out from this came the paper "A Simple Framework for Contrastive Learning of Visual Representations"[2], which we will be looking at as a baseline for the system in this paper. SimCLR is a classification approach which provided advances in self- and semi-supervised training compared to previous approaches. This approach to image classification, SimCLR was able to achieves an *85.8%* top-5 accuracy for the ImageNet dataset with *1%* of labeled images[2]. SimCLR was written by Ting Chen, Simon Kornblith, Mohammad Norouzi and Geoffrey Hinton, as a research example into Self- and Semi-Supervised learning and Re-ID.

Framework

The framework for SimCLR is much simpler, than many common techniques for self-supervised learning. Which in past techniques often required significant modification to the network architecture and training procedure.[2] Here SimCLR takes a more simplified approach to the problem of image classification. It starts off by learning generic representation of the images, using an unlabeled dataset. Then it will fine-tune its weights by training on a small amount of labeled images to achieve its accuracy.

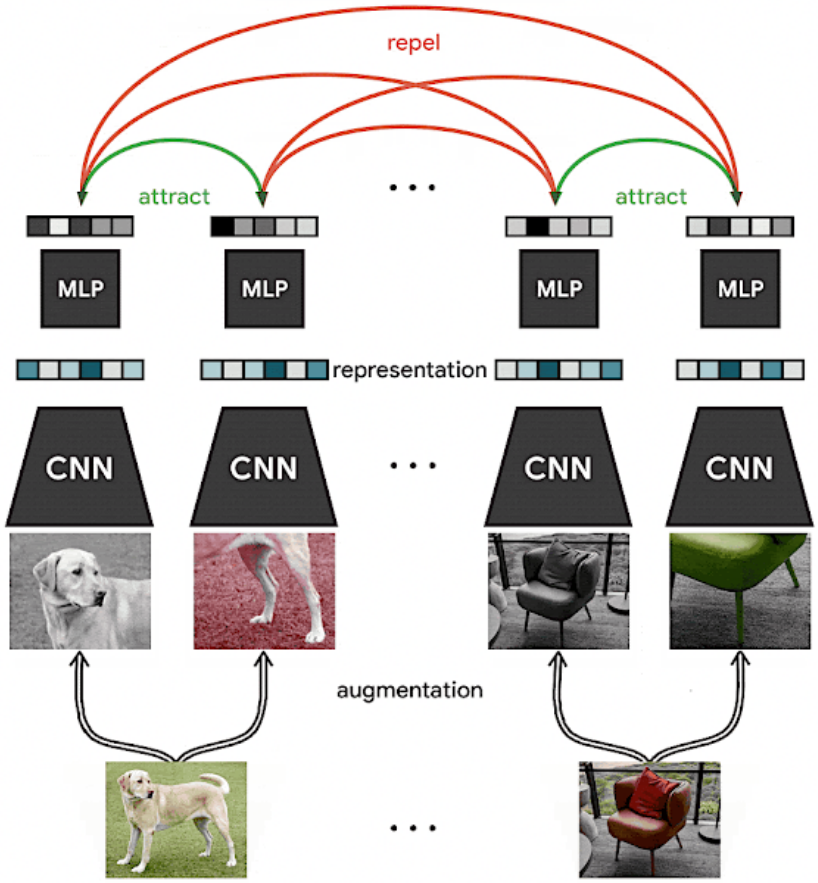


Figure 2.6: A overview of how SimCLR framework is built and how it trains on datasets given [2]

In the beginning SimCLR draws random images from the original dataset, then augments them with simple transforms twice (Cropping, Color Distortion, Gaussian blur). They discovered in their research, that giving different views of the same image is important. This works to prevent trivial forms of agreements, such as agreements of the color histograms. To this end, they tested a range of different transforms to be applied to a single image. These transforms includes: Crop and Resize, Crop, Resize and Flip, Color Distort(drop), Color Distort(jitter), Rotate(90,180,270), Cutout, Gaussian Noise, Gaussian Blur and Sobel Filtering.

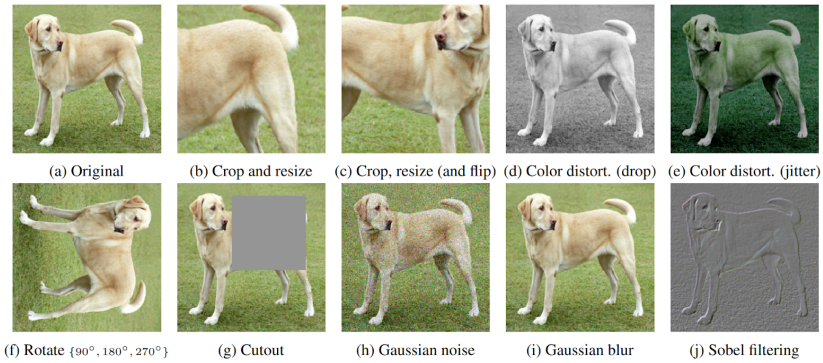


Figure 2.7: Examples of the transformation in action when applied to an image.[2]

Following their testing, it was discovered that the combination of two transformations worked the best to improve the performance of the model. The two most important transformations to the model were the combination of random cropping and random color distortion. By independently distorting the colors of each crop, the model can learn to focus on more generalizable features than by matching color histograms. As without this type of transformation of colors and crops, the model could begin to focus solely on color and ignore other more generalizable features. By doing so with the two transformations, the model is only able to maximize agreement between more useful and generalized representations.

Performance

After training SimCLR is able to achieve an accuracy of *76.5%* Rank-1. This is compared to the previous best at that time, the CPC v2[26], which scored a Rank-1 accuracy of *71.5%*. This performance matches supervised learning in a smaller ResNet-50[2].

2.3 Verification and Metric Method

Another method of verification two images of being the same ID, is using a binary-class model such as a Siamese network[6]. The Siamese network has been purposely built and trained for comparing two images and is a Multi-Stream network. A Siamese network consist of two identical but independent networks, working side by side to compare two images. This network receives input from two images, and outputs a two similarity scores. The higher the positive similarity score is, the likelier it is that the two images are of the same ID. The opposite is also true; the lower the positive similarity score and higher the negative similarity score is the less likely they belong to same ID. Though a weakness of such a verification method is that often it can be shallow and not produce the best results[6].
Another method is the metric method, which aims to maximise the distance between IDs and minimise the distance between the same IDs. This can be done with a single network using the triplet based method. This method produces feature vectors that can be compared up against other feature vectors extracted from different images. The metric Triplet method consists of using 3 anchors; Test anchor, Positive anchor and Negative anchor. It aims to pull the Test anchor and Positive anchor together and the Negative anchor and Test anchor away from each other. This is to minimize the inter distance within IDs feature vectors and maximize the distance between different IDs feature Vectors[6]. This can be seen in figure 2.8 below.

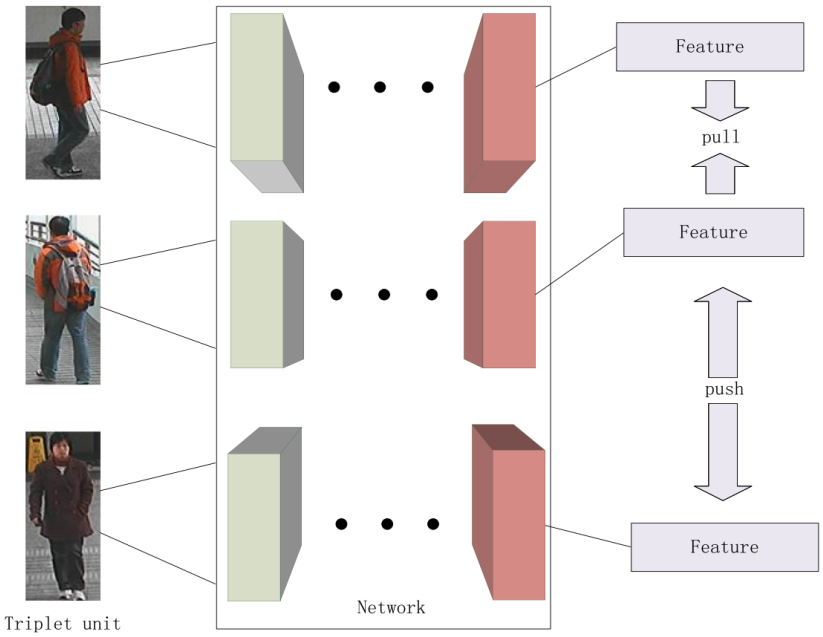


Figure 2.8: The first and second images are the test anchor and the positive anchor. The last image is the negative anchor. The test and positive anchor is being pulled together and the negative is being pushed apart from the rest.[6]

2.4 Dataset

To train and test the methods in question and evaluate their effectiveness. A dataset is required to train and evaluate Neural Networks e.g. some form of data representative of the task set for the network to solve.
The dataset used is a modified version of the one used in the One-To-One paper[27], with additional annotations for luggage carried by the people in the data. In the dataset used for this modification, there were 7529 bounding boxes containing 116 identities. These bounding boxes contained both person and their luggage in one bounding box. For this project, the person and the luggage needed to be separated into two bounding boxes.

After separating the dataset ended up containing 8163 bounding boxes for luggage, while still containing 7529 bounding boxes for the people. The reason there are more luggage bounding boxes is because there are multiple people carrying multiple pieces of luggage with them. With this, there are 6080 person bounding boxes with luggage attached to them. In the portion of bounding boxes with luggage, there are 96 IDs left from the original 116 IDs in the full dataset. Meaning that there were 20 people in the One-To-One dataset, that did not carry luggage

while they where being recorded during collection.

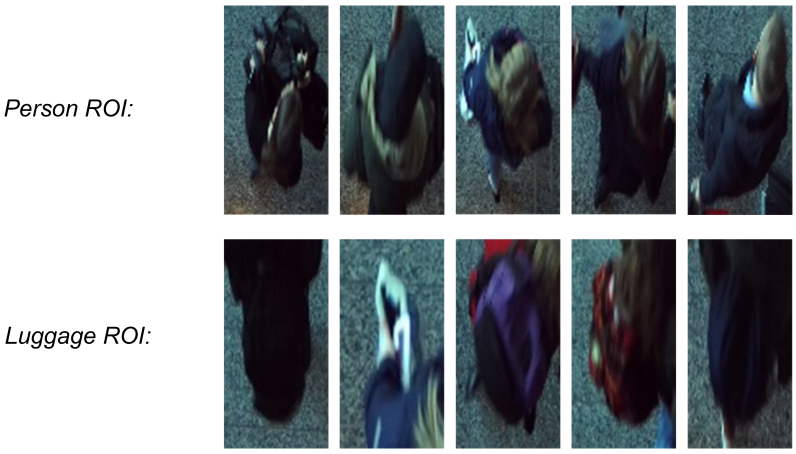


Figure 2.9: Collage of the ROI cutouts for the luggage and person bounding boxes.

2.5 Evaluation

When researching new Neural networks it is important to be able to evaluate the system through relevant metrics. For these, it is important to make sure the metrics for each of the tests are measured on also exist or can be generated for the baselines, in order to compare their performance against each other. These metrics can also help determine the quality of the system, such as an interference time metric that can show the efficiency of the Neural network. Knowing the interference time of the networks helps to determine if the network can be used in a real-time product. For this possible use case, a less accurate network could be desirable if it meant a faster interference time. Fast interference time is desirable for a network, to run real-time or near real-time[4]. For this paper, interference time has not been measured, as it was not the main concern at the time. The main concern of the paper is the accuracy of the system with the addition of luggage information.

As in the baseline papers, there are multiple metrics that can provide an overview of the performance of the networks[4, 2]. When evaluating the effectiveness of the network, it is important to have ground truths to compare against. With these ground truths, a test of the networks can be performed and their output can be

compared with the ground truths to measure accuracy. These ground truths come from the dataset explained in the section above 4.2. One such metric that can be calculated from ground truths and network output, is Mean Average Precision (mAP). The mAP is calculated as presented in the Market-1501 dataset[28]. mAP is the mean of all Average Precision (AP), the AP is calculated as an area under the Precision-Recall curve. Another metric that can be extrapolated from comparing with ground truths is a Rank-K score. Rank-K is where K is the top number of best predictions given from the network. The ground truth must be within the number of K top guesses from the network to have a successful guess. Rank-1 will be the primary measurement mentioned throughout this paper, this will often be referred to as accuracy.

Additional valuable information is the characteristics of the feature vectors extracted from the image patches. These characteristics are metric values such as; the inter and intra ID distances. With these, it can be seen how much two IDs overlap each other and how varied feature vectors are within IDs. For these metrics, having a low intra distance and large inter distance, aids in classification and distinction between classes[29].

2.6 Development

For developing a system handling Re-ID'ing, it is important to have a specific approach in mind. How should the system be trained and what parts should it consist of. For this project the focus will be on simply handling the dataset, with a combination of feature vectors from two different sources (see section 4.2). With this approach it was determined to focus research on Dual Neural Networks and Multi-Stream Neural Networks. These research areas require multiple evaluations, for both areas should be compared to the baseline Neural Networks in section 2.2. The methods of verification described in section 4.1, helps to determine use cases and possible weaknesses of a network. With the verification methods it also provides information on how to handle a scenario where training data might have limitations. A verification system that can compare two feature vectors from two images to determine their likeness and thereby determine if they are the same, is especially preferred.

After decisions on how the Networks and dataset should be used and validated, delimitation can now be established for giving focused guidelines for the rest of the project.

3 - Delimitations

With the limited time and resource some delimitations needs to be established to narrow in the scope of the project. To determine the scope of the project, it was important to consider what resources were available. One main concern was the dataset needed to train and test a system of this idea and structure. Due to the relative niche case of the dataset needed, described in section 4.2, it was decided to create one dataset from another. The changes and additions to the previous dataset are described in 4.2.

Due to these delimitations, a focus was put on the feature extraction for Re-ID, with available data already being processed by a previous step in the pipeline. The previous step in the pipeline is assumed to give a dataset containing bounding boxes for luggage and people. With a list that addresses each person to one or more luggage that they might be carrying.

Another delimitation to be made is the architecture of the neural network, as the computing resources can vary greatly between architectures. The computational resources allocated to the project were: 1 GPU: NVIDIA RTX 3080 10GB, System RAM: 16GB DDR4 and 1 CPU: i7-8400x 6 core 4GHz. The GPU is the one handling most of the training, in this case is a CUDA-enabled device. Meaning that the GPU can effectively utilize the highly parallel capability of a GPU for training a network[30]. With the memory allotted for training, networks can only reach a certain size before training speed and errors will befall training. Therefore, a moderate size network with good performance is preferable considering the scope of this project.

The number of methods for Re-ID, are more than can be explored in this paper alone. Because of this, a delimitation of what methods to explore and how to approach them had to be made. In the end, the idea of a Dual-stream network and a Dual-Net network were chosen. With these two networks it was also decided to use the Triplet Loss for the loss function.

3.1 Final problem formulation

"How can a network use a bounding box set of person and luggage to Re-ID them across multiple instances and what architecture works well for generating descriptive feature vectors."

Part II

Problem Solution

4 - Design

When designing a Machine Learning system for Re-Identification it is important to understand previous solutions and have well-determined inputs and outputs for the expected system. With the newly compiled dataset made for this paper, a two image input and 1 feature vector output will be the initial testing base.

4.1 Methods

In the following section, an elaboration of topics and methods that are used thoughout the paper. These parts consist of; the architecture, which is the way the Neural Network operates with the information given, the comparison of feature vectors and how they are compared and validated.

4.1.1 Neural Network Architecture

For the Neural Networks made specifically for this paper, a ResNet-50 was used as the base architecture. ResNet is a residual network architecture, built with multiple layer depths up to 152 layers[31]. The ResNet-50 is a medium/compromise between the size of the resources for this project and the accuracy of the architecture. In TensorFlow there are three depths of layers, the architecture can be built as: 50, 101 and 152. In a ResNet architecture, there are blocks with a "shortcut" that bypasses some convolution layers making the architecture a residual network[31]. With this, the last prediction layer is replaced with a 128 dense layer to output a feature vector for Re-Identification.

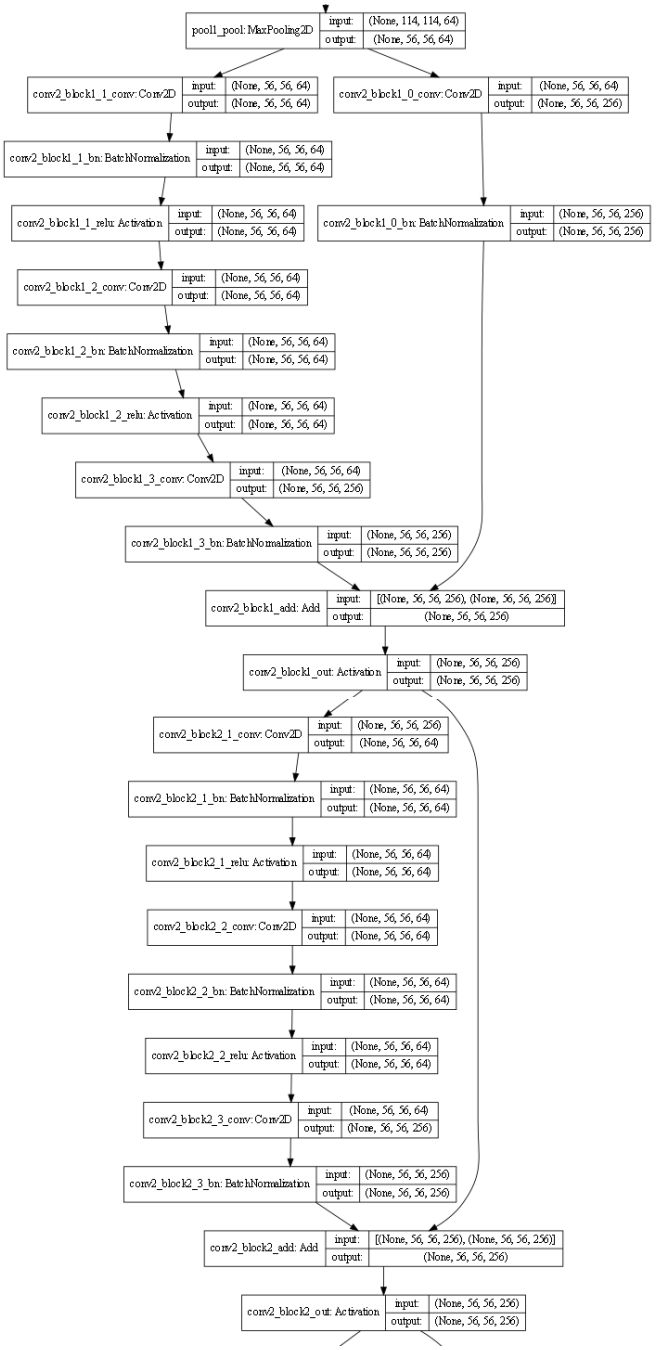


Figure 4.1: Here are the two main block types in the ResNet. These show the "shortcut" that is common in residual network architecture. These layers can change in neurons and filters pr layer but generally a ResNet keeps the sequence of layers.

This network functions well with the limited resources of the computer system used for this paper.

Types of layers

In Neural Network there are many types of layers, that give the Network its functionality. For CNN's the layer it gets its name from is the Convolution layer, another common layer for CNNs is a pooling layer. The Convolution layer is a type of layer designed to consider spatial information in data like images. The layer does this by utilizing a kernel, which it uses to understand relations between pixels and their neighbors. Another layer that often accompanies Convolution layers, is the Pooling layer. This layer down samples the input, to aid the following layers generalise larger features in the dataset. A common form of pooling layer is a MAX pooling layer. This form of pooling layer, picks the highest value within its kernel as the representative for that cluster of pixels.

4.1.2 Feature Comparison

Features vectors can describe what they were extracted from, this can be validated through comparison with other feature vectors. To compare feature vectors there are multiple methods, one such method is distance measurement between two feature vectors. This distance measurement can show how similar two feature vectors are to each other. With a distance measurement, one can also distinguish between two highly different feature vectors. When finding the distance between two, one can also use this to help identify feature vectors to clusters of other IDs feature vectors. There are many formulas for calculating distance between feature vectors, the Euclidean distance method is among one of the most used for tasks like this[32]. This distance measurement is also used in the loss function Triplet Loss Function, to give a loss value to train the Neural Network. For the output of a Network like this, a distance measurement is also used again to test and validate the performance of the Network.

4.2 Dataset

In this project, a custom-made dataset was used, as one with the specific requirements was not available. The dataset currently used consists of two sets of bounding boxes and one set of images. The two sets of bounding boxes are one for people and the other one for the luggage they are carrying at that time. This dataset also have each person and luggage, given an unique identification number.
Thoughout the paper there are multiple datasets mentioned; Person Solo, Luggage Solo, Person/Luggage 2 BBOX and Person/Luggage 1 BBOX dataset. Person solo dataset contains only the persons bbox and images with their respective IDs. The same goes for the Luggage Solo dataset, which contains only luggage bboxes and their IDs. The person/luggage 2 BBOX contains both the images and BBOX for

Persons and luggage separately with an additional file. This additional file links a person to the luggage they are carrying in the same frame. Person/luggage 1 BBOX is as close to the original dataset as it can be. This dataset contains the full frames and the bounding box containing both the person and luggage in that particular frame.

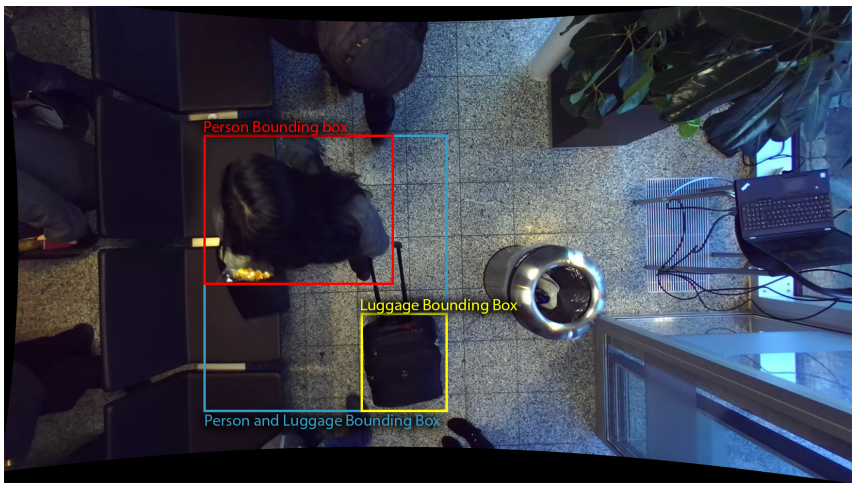


Figure 4.2: The Blue box is the Person/Luggage 1 BBOX dataset, the Person/Luggage 2 BBOX is the Red and Yellow boxes. Where the solo datasets contain only the Red or Yellow boxes.

5 - Optimisation and Loss functions

When training a network there are a range of different optimisation and loss functions to consider. The idea is to find the optimal solution for a non-convex function, for which a theoretical global optimal solution is not guaranteed. Therefore an optimisation algorithm is designed to explore the given function and find an optimal solution[33]. The two parts for optimising consists of a function to calculate the error or loss of the network from the desired output and another algorithm to traverse the solution space to find an optimal solution for the problem. The two parts that will be explained are selected from the baseline system TransREID2.2.1, as these are some of the most commonly used methods for machine learning problems like these.

5.1 Optimisation Algorithm

The purpose of the optimisation function is to effectively search for the best solution to the given problem. It does this during training through back propagation, by changing the weights, biases and it sets other variables in the neural network throughout training. The optimisation algorithm used in this project is stochastic gradient descent or SGD for short. SGD is one of the more popular optimisation algorithms used for neural networks training[34]. It is an alternative to the standard gradient descent (GD), as it does its changes in a single update and SGD uses a subset of samples to update the parameters in the neural network. With this approach SGD uses the variance between samples and potentially let the system avoid local minimas[34]. In testing and training, SGD is used as a comparison with the baseline papers[4, 2]. The formula for SGD is written as such:

$$w := w - n \nabla Q_i (w)$$

In the formula above, $Q_i (w)$ is the value of the loss function at i in the dataset it is being trained on. n in this case is the step size or learning rate.

There are two types of minimas when exploring a function space, local minimum and global minimum. What a optimiser is trying to do is to find the global minima, but often it will fall into a local minima. There can be multiple local minimas

in a given problem space, but these local minimas often catch an optimiser algorithm if the learning rate is too small. The TensorFlow SGD implementation has two hyper parameters that can be set according to the networks specifications. For this algorithm in TensorFlow, the SGD has Learning rate and momentum as hyper parameters. The first hyper parameter is the learning rate, this can be set for how far steps the optimiser will take pr step. The other hyper parameter is momentum, this can accelerate the gradient decent in relevance to direction. This hyper parameter was not used for testing as previous baselines did not make use of it.
For optimisation there can often be specified a learning rate, this is how much the optimiser algorithm changes the variables each time it updates them. Having a high learning rate means it changes the variables to a greater degree than it would do with a lower learning rate.

5.2 Loss functions

While training a neural network, a loss function informs the optimisation algorithm how the system preforms during training. The purpose of this function is to find the current state of the neural network, it does this by calculating loss for the network. The objective of the loss function is to minimise its score on the given training set with criteria set for the task at hand.

Loss functions can be classified into two categories; Regression problems and classification problems. Facial recognition and Re-identification often falls into the regression problem category, as they often deal with embeddings end/or feature vectors describing what is being seen. Where a lot of research today focuses on classification problems, the use of cross-entropy and mean square error loss functions won't be as effective[35]. For problems such as these loss functions such as Triplet loss, would perform with greater accuracy for the task at hand[36, 37]. This loss function will be explained further in the following section.

One such loss function is the Softmax cross entropy loss function. This function is suited to well for classification networks, with the task to differentiate between subjects in a dataset. But a function such as that one is not well suited for tasks such as facial recognition[38]. For things like facial recognition and re-identification, loss functions like Triplet loss performs well[38].

5.2.1 Triplet loss

The triplet loss function is an approach for training network with outputs like feature vectors and the like. This function calculates its loss by measuring the distance between three examples. The three examples in this case will be called

anchors, as it uses a test anchor, positive anchor and a negative anchor. The positive anchor is an example with the same identity as the test anchor, to evaluate how close the test anchor is to other examples with the same identity. The negative anchor can be any other example as long as it has a different identity. This is to evaluate how close it is to the test anchor, this helps to differentiate and generalize important features for the examples. With this method, it is possible to create a large set of combinations for the three different anchors. This can guide the network to achieve better performance in creating more representing feature vectors, moving the different identities further apart from each other.[36]
The triplet loss function is calculated with[39]:

$$L_{tri} = \sum_{\substack{a,p,n \\ y_a = y_p \neq y_n}} max(d_{a,p} - d_{a,n} + m, 0) \tag{5.1}$$

This formula aims to guide the networks to find a low intra-identity distance while also giving a high inter-identity distance between all identities in the training dataset. In the function above, m is the margin for the negative anchor. This margin serves to push the negative anchor further away, to guide clusters of anchors away from each other.

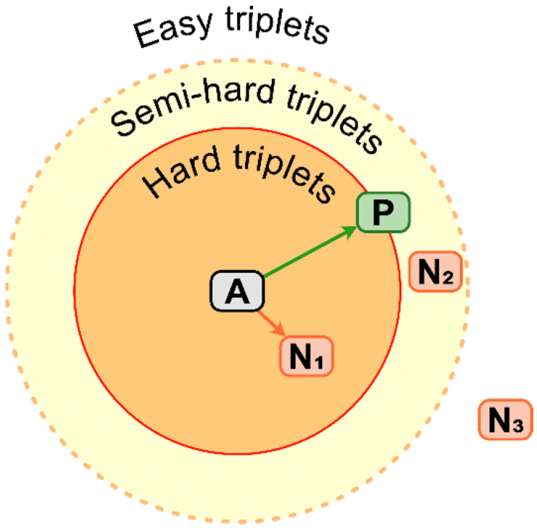


Figure 5.1: The illustration shows three different negative anchors, *N1, N2, N3*, the test anchor is shown as *A* and the positive anchor as *P* [40]

In the formula 5.1 the a is the test anchor, p is the positive anchor and n is the negative anchor for the triplet. The formula finds the euclidean distance between the anchors with the denotation $d_{a,p}$ and $d_{a,n}$, this is the distance between the test and positive anchor and the test and negative anchor. The last item in the formula for the loss function is m, which is the margin that is forced between positive and negative anchors[39].

Triplet Hard Loss and Triplet Semi-Hard Loss

There are two alterations to triplet loss, which can improve its performance. These two alterations is not the focus of this paper, to keep the baseline and experiments test in line with each other. A short description of both alterations, as future tests could be performed with them in mind.

Triplet Hard Loss:
This alteration to the Triplet Loss function is relatively simple as it mostly follows the normal Triplet Loss to a large degree. The method of Triplet Hard Loss can be considered an optimization of the Triplet Loss function. With Triplet Hard Loss the idea is choosing the hardest negative sample, this can in some cases give it advantages over other Metric Loss Functions[41]

Triplet Semi-Hard Loss:
This optimisation works much like Triplet Hard loss, except it creates an average from the negative samples as a negative anchor. It does this while still creating a negative anchor that is further away from the test anchor than the positive anchor[42].

5.3 Neural Network fusion

When handling multiple networks and combining their outputs, it is important to understand the many methods of doing so. For this paper the two networks handle two images; one image of a person and an image of the luggage they are carrying. These are either fed into each of their own networks or separate streams in a multi stream network. In both of these cases, there are a connection point/fusion between two networks or streams[43]. To facilitate an output that is effective, in this case a feature vector, handling of this fusion is important. In many simple cases, a concatenation of feature vectors can provide, some benefits to the accuracy and effectiveness[44]. One of the optimisations that can be tested from here, is a follow-up network. This follow-up network should take the concatenated feature vectors as input and produce a new more robust and descriptive feature vector.

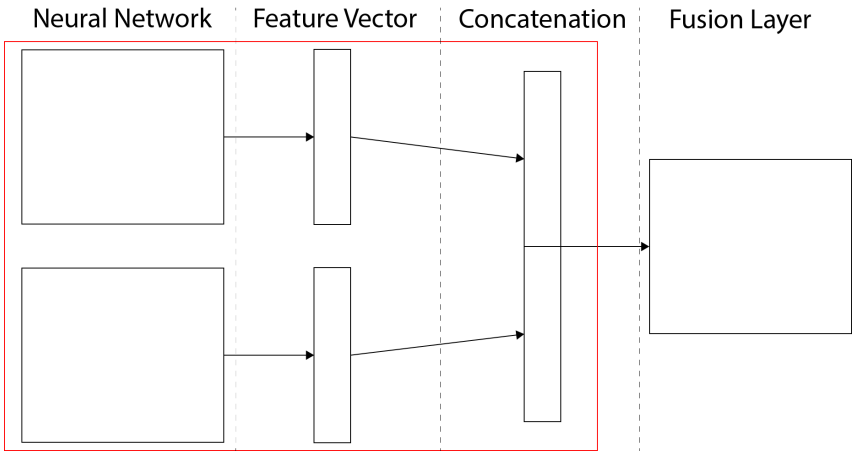


Figure 5.2: In this figure are a simple fusion of two networks shown. In the red box is a simple Concatenation fusion, the last layer outside is another network
layer made to improve the fusion.

5.4 Optimization of Dataset

For training and testing a Neural Network, a large and sufficiently varied dataset is needed. If the dataset is not large enough or there is a lack of variation between data points, there are data augmentation methods that can improve such cases. Data augmentation means to alter the existing data in meaningful ways, to improve the amount of data the Neural Network has to train on. Such augmentations for an image dataset can include but is not limited to: Cropping, Rotation, Hue shift, Saturation shift, Contrast Shift, Black and White and skewing the image.
Another optimisation of the dataset, is the splitting for Training and testing. When training a neural network there should always be a smaller section of the dataset never seen during training. This smaller section is then used to validate the network for over-fitting.

6 - Implementation

Throughout this chapter, we will explain and document the implementation of the system. An explanation of the different frameworks for Neural networks used and statistics for the dataset created for this paper. The systems tested and the system proposed in this paper is written in the programming language Python. Most systems developed for research in Machine learning use python for extermination and rapid development. For python different frameworks have been made for Machine learning, the two used for this paper are TensorFlow and PyTorch.

6.1 Programming language Python

The chosen programming language for this paper is Python. It is chosen for its versatility and ease of experimentation. Python solved these two aspects with a large collection of libraries and ease of readability. Python is a general-purpose programming language developed by "Python Software Foundation"[45] as of writing the current version is 3.10.4.

6.1.1 Additional Libraries

For python, additional libraries were used to speed up the system. Many of the libraries used are data management and manipulation tools. These libraries include: Pandas, Numpy and pathlib. Pandas allow for ease of importing data from csv files, that store data about the classes and the number of bounding boxes for a given ID. NumPy allows fast manipulation of arrays as the library is written in C, which allows for faster data manipulation though its built in functions. PathLib allows for quick navigation of folder structures, used to get paths for checkpoints during training.

6.2 Neural Network Frameworks

For in python itself, it would be very difficult and highly inefficient to write a machine learning system in pure python. Therefore, there are libraries written in other programming languages such as C and utilizing CUDA that increases performance and extend the capabilities of python. The two Machine Learning libraries used for this paper is TensorFlow and PyTorch, both developed to train

and run neural networks in python. Both library frameworks have support for CUDA wich enables them to utilize CUDA ready GPUs to speed up training and interference time of machine learning.

6.2.1 TensorFlow

TensorFlow is developed by the Google Brain Team[46], as a toolkit to help research into machine learning. The toolkit consists of Neural Networks templates, data augmentation tools, dataset manipulation tools and a host of other pre-built pipelines. TensorFlow allows the usage of CUDA enabled GPUs to speed up the training and interference time of neural networks. CUDA takes use of the computational parallel ability of GPUs, allowing it to run the networks much faster than on CPUs.[30]

With this library follows a large selection of additional tools, such as: TensorFlow-Datasets, TensorBoard and TensorFlow I/O. Each of these have a more specified use case, compared to the general library of TensorFlow. The most important of these libraries is TensorBoard, this extension to TensorFlow allows us to see multiple datapoints during and after training a network to better understand it's effectiveness. TensorBoard can show the weights and biases during training and is able to visualise its progress and visualise the loss value for each step in training allowing us to catch errors faster in days long training sessions.

A benefit that was utilized from TensorFlow library is the call-back features. These features allows the program to execute commands during training, there by extending and solving edge cases that can occur during training. One such edge case would be a collapse of the loss function giving or an error in the loss results mid training giving NaN as a result. Cases such as these can be reacted upon with call functions.

```
1  log_dir = os.path.dirname(__file__) + '\\logs\\' + startTime
2  file_writer = tf.summary.create_file_writer(log_dir)
3  tensorBoard_callback = tf.keras.callbacks.TensorBoard(
4      log_dir=log_dir,
5      histogram_freq=1,
6      profile_batch = '10,15'
7      )
8
9  checkpoint_filepath = log_dir
   ↪  +'\\saves\\'+'Epoch{epoch:02d}-Loss{loss:.5f}.hdf5'
10 model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
11     filepath=checkpoint_filepath,
```

```
    save_weights_only=False,
    verbose=1,
    save_freq=95,
    monitor='loss',
    save_best_only=True)

_ = model.fit(
    data_generator(batch_size),
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    verbose=True,
    callbacks=[tensorboard_callback,model_checkpoint_callback]
)
```

In the previous section of code are two callback functions and the fit training function for TensorFlow. The first callback function is the function that logs information for TensorBoard, this tracks the stats of training. The second callback function is a checkpoint save function, this saves snapshots of the networks weights at multiple points during training.

Neural Network Architecture

For the implementation, the TensorFlow model zoo is used. The model zoo has multiple reassembled neural network architectures, ready to be imported and compiled for multiple tasks. For this project the Resnet-50 (see design section 4.1.1), is a framework architecture for the dual-net and multi-stream test.

The Resnet architecture comes reconfigured with a prediction output layer of 1000, this is to match the imagenet training set requirements. This last layer is a softmax prediction layer, that tries to classify the input to 1000 predefined classes. To keep the pre-trained weights for the whole of Resnet-50, this layer has to be initialized with the network. The softmax output layer will subsequently be removed for a new output layer to function as a feature extractor. The implementation of the Neural Network Architecture:

```
# Loading the Resnet-50 from the TensorFlow Model Zoo, with imagenet pre
↪  trained weights.
resnet50 = tf.keras.applications.resnet50.ResNet50(
    include_top=True,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
```

```
    pooling=None,
    classes=1000,
)

# Creating a new temporary model, to remove the last prediction layer of the Resnet-50 and replacing it with a 128 neuron output.
model = models.Sequential()
resnet50Out = resnet50.layers[-2].output
layer128 = tf.keras.layers.Dense(128)(resnet50Out)
resnet50 = models.Model(inputs = resnet50.input, outputs = layer128)

# recompiling the new Neural Network with the Triplet loss function and SGD Optimizer.
resnet50.compile(
    optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False, name="SGD"),
    loss=tfa.losses.TripletHardLoss()
    )
```

With these lines of code, a Resnet-50 with pre-trained weights and a 128 vector output is created and compiled.
Once compiled and ready to be trained, it can to display the number of parameters in the architecture. The Dual-Net network are two identical networks trained individually, therefore it has half the total parameters as the Multi-Stream network. When the Dual-Net networks are combined they have the same total parameters as the Multi-Stream.

```
Model: "Dual-Net (1 of 2)"
_________________________________________________________________
Total params: 23,849,984
Trainable params: 23,796,864
Non-trainable params: 53,120
_________________________________________________________________
Model: "Multi-Stream"
_________________________________________________________________
Total params: 47,699,968
Trainable params: 47,593,728
Non-trainable params: 106,240
_________________________________________________________________
```

For creating the Multi-Stream network, it is implemented as such:

```
# Loading and creation of the left side Resnet-50 with 128 neuron output.
resnet50_Left = tf.keras.applications.resnet50.ResNet50(
    include_top=True,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
)
#Giving the layers a new name to keep the distinction between the two sides of the network.
for layer in resnet50_Left.layers :
    layer._name = layer.name + str('_Left')
resnet50Out_Left = resnet50_Left.layers[-2].output
layer128_Left = tf.keras.layers.Dense(128)(resnet50Out_Left)
layer128_Norm_Left = tf.keras.layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))(layer128_Left)
# Loading and creation of the right side Resnet-50 with 128 neuron output.
resnet50_Right = tf.keras.applications.resnet50.ResNet50(
    include_top=True,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
)
#Giving the layers a new name to keep the distinction between the two sides of the network.
for layer in resnet50_Right.layers :
    layer._name = layer.name + str('_Right')
resnet50Out_Right = resnet50_Right.layers[-2].output
layer128_Right = tf.keras.layers.Dense(128)(resnet50Out_Right)
layer128_Norm_Right = tf.keras.layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))(layer128_Right)
# Concatination of the two networks and defining inputs and outputs of the new network architecture
output_concat = tf.keras.layers.concatenate([layer128,layer128_Norm_Right], name="Output")
resnet50_Multi = models.Model(inputs = [resnet50_Left.input,resnet50_Right.input], outputs = output_concat)
```

```
# Compiling the Neural Network with the Triplet loss function and SGD Optimizer.
resnet50_Multi.compile(
    optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False, name="SGD"),
    loss=tfa.losses.TripletHardLoss()
    )
```

Next are the MLP networks that will be used in the second iteration of tests. These networks are created from the ground up, without using the model Zoo of TensorFlow.

```
# Creating the layer structure with the input size being relative to the network above (256 or 7680)
inputLayer      = Input(shape=(feature_length), name='input')
layer256        = tf.keras.layers.Dense(256)(inputLayer)
layer256_norm   = tf.keras.layers.BatchNormalization()(layer256)
layer256_relu   = tf.keras.layers.Activation('relu')(layer256_norm)
layer128        = tf.keras.layers.Dense(128)(layer256_relu)
layer128_norm   = tf.keras.layers.BatchNormalization()(layer128)
layer128_relu   = tf.keras.layers.Activation('relu')(layer128_norm)
outLayer        = tf.keras.layers.Dense(128)(layer128_relu)
outLayer_Norm   = tf.keras.layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))(outLayer)
MLP_FollowUp = models.Model(inputs = inputLayer, outputs = outLayer_Norm)

# Compiling the Neural Network with the Triplet loss function and SGD Optimizer.
MLP_FollowUp.compile(loss=tfa.losses.TripletHardLoss(), optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False, name="SGD"))
```

The MLP Follow-Up network has in and of itself 116,736 to 2,017,280 parameters, the reason for this disparity is the output of the networks above. The output feature vector varies from 256 length for both Dual-Net and Multi-Stream, and 7680 for the Dual-TransReID.

```
Model: "MLP_FollowUp_TransReID"
_________________________________________________________________
Total params: 2,017,280
Trainable params: 2,016,512
Non-trainable params: 768
```

```
6  _________________________________________________________________
7
8  Model: "MLP_FollowUp_Dual-Net_Multi-Stream"
9  =================================================================
10 Total params: 116,736
11 Trainable params: 115,968
12 Non-trainable params: 768
13 _________________________________________________________________
```

With this all networks are ready to train and be evaluated on.

6.2.2 PyTorch

PyTorch is developed by Facebook's AI Research lab (FAIR)[47]. This library is very similar in its purpose to TensorFlow, to help research and development in neural networks. PyTorch is mainly used for the TransReID, where it is used for training and construction of the model architecture.

6.3 Datasets

When implementing the datasets for the training of the system speed and ease of loading were a priority to help with rapid testing. For this the dataset is split into two parts; entrance and exit images, that is split into further two parts; person and luggage. When it is loaded the first time, it finds all the person images in both the entrance and exit. Then looks up in a CSV file which luggage belongs to which person. In the total dataset there are 6080 person bounding boxes with luggage attached.

```
Dataset Folder
├── PersonToLuggage.csv
├── Entrance
│   ├── Person
│   └── Luggage
└── Exit
    ├── Person
    └── Luggage
```

To aid in giving IDs to all the people a system called Deep-SORT Helped to keep track and quick ID new people and luggage.
Deep-SORT is a method of tracking bounding boxes across multiple frames and keeping them labeled with an ID[48]. It does so by comparing a feature vector extraction between each frame and the position for the bounding box. If it can find no match, it will assign the bounding box a new ID.

Entrance 10631 images 3598 label files
Exit 10918 images 3931 label files
Total label files 7479

7 - Experiments

After gathering and annotating the dataset with people and luggage, it was now possible to start training networks and begin exterminating.

From the chapters 4 and 3 a plan of action was developed. A preliminary test should be done, as to evaluate the baseline TransReID on the dataset made for this paper.

7.1 Preliminary test

To evaluate the systems a baseline would have to be set by the networks described in section 2.2. These baseline tests will be done on the same dataset made for this project, described in section 4.2. It is the same dataset that will be used for the following iterations of tests. TransReID was tested with Person and luggage individually and both in a single bounding box. It scored an accuracy of *78%* on persons solo, an accuracy of *84%* on luggage solo and then *90%* on the test set containing both person and luggage.

7.2 First Iteration

The first iteration of tests includes multiple networks and ways of training them. The first test is a Dual Network test, this is two individually trained networks and the concatenated to one another. The second test is a Multi-Stream network, where the whole network is trained at once. The third and last test is training two Trans-ReID networks and simply concatenate them to one another. In this test, we wanted to see the accuracy, of a system given two image inputs and trained on both of them.

7.2.1 Iteration 1 of Dual-Network Identification

For the first test, there would be made a Dual network system. This is comprised of two neural networks trained individually on either the person data or the luggage data. They are then combined at the end with each of their feature vectors concatenated to one another.

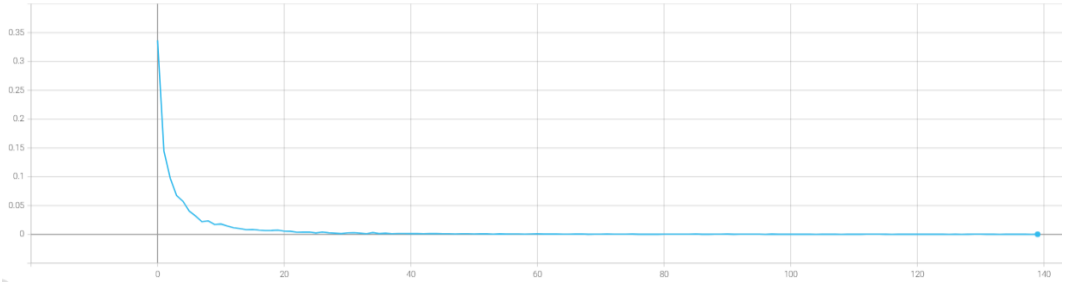


Figure 7.1: Graph showing the loss during training of the Dual-Net Person network. Start Loss: 0.96169, End Loss: 0.12524
Logged with TensorBoard

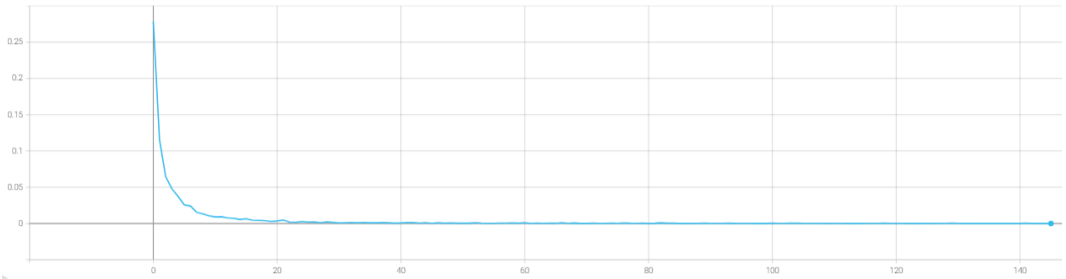


Figure 7.2: Graph showing the loss during training of the Dual-Net Luggage network. Start Loss: 0.95405, End Loss: 0.09841
Logged with TensorBoard

Letting the networks individually train on either the Luggage or Person data, lets them create descriptive feature vectors for each part. Then after training the two networks, they were put through a test subset of the original dataset, described in section 4.2, to evaluate the accuracy of the network as a whole. Three evaluations were done on this iteration of the system, one for each of the solo networks and as a whole. The solo evaluation of the network trained on persons scored an accuracy of *72.3%*. The solo network trained on luggage scored *78.1%* and when both were combined they scored an accuracy of *86.7%*. For this test, there is also an increase to the STD of the inter and intra distances, as the two networks are combined.

Network	Tested on	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Solo-Net	Luggage	78.14	3.8658303e-08 / 1.61	0.15	1.4551915e-11 / 1.48	0.29
Solo-Net	Person	72.31	0.01 / 1.63	0.17	1.4551915e-11 / 1.54	0.36
Dual-Net	Both	86.67	0.19 / 2.10	0.20	1.4551915e-11 / 2.01	0.52

Table 7.1: Stats for Dual-Net and the individual nets individually

The solo nets in table 7.1 above, are the separate networks of the dual-net tested individually. The dataset they are tested on would either be the Person Solo or Luggage Solo dataset, depending on what the network was trained on.

7.2.2 Iteration 1 of Multi-Stream Identification

For the second test in the first iteration, a test of a multi-stream network was done. This network is trained with both Person and Luggage data at the same time. This should help the output of the network to be working together to create a more fully descriptive feature vector. The output for this network is a feature vector with a length of 256 dimensions. As the two internal networks are trained together, compared to the previous tests where they were trained separately. The idea is when the network is trained as a whole, the output is more coherent together.

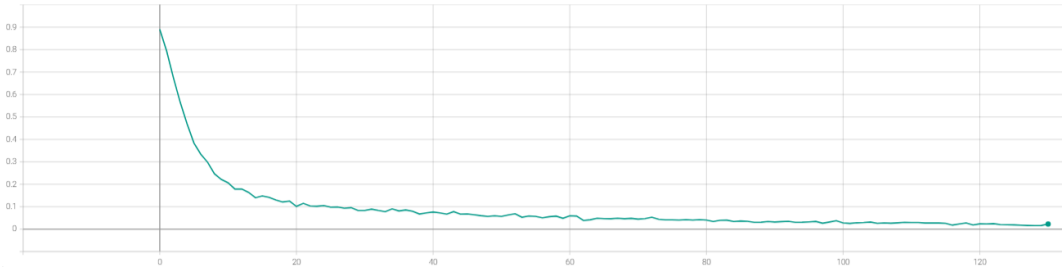


Figure 7.3: Graph showing the loss during training of the Multi-Stream network. Start Loss: 0.88874, End Loss: 0.01071
Logged with TensorBoard

After training the network, it was put though the same test as the Dual-Net. This scored an accuracy of *75.2%* on the test set. The test showed that the clustering of the feature vectors, showed an inter ID distance of *0.7* min and *2.03* max. It also showed an intra ID distances of *2.9* min and *1.7* max for the output feature vectors of the network.

Network	Tested on	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Multi-Stream	Both	75.27	0.69 / 2.04	0.28	2.910383e-10 / 1.68	0.28

Table 7.2: Stats for the Multi-Stream

7.2.3 Iteration 1 of Dual-TransReID Identification

TransReID had proved itself effective in the preliminary test, with an accuracy of 90.2% (see section 7.1.) This test is similar to the first test with a dual network, this time both networks would be the TransReID. This is to see if splitting up the data between Luggage and person would improve the system, compared to having luggage and person in one bounding box. TransReID compared to the other networks tested, has a much larger output vector. The TransReID network architecture has an output vector of 3840, this is for each network in this dual network setup. Meaning that the real output of this system is a feature vector of 7680. After training the system on the dataset made for this project, the system scored an accuracy of 94.8%. This test also showed that the network also had an inter ID max and min distance of, *23.5* and *51.6* respectively. With an intra ID min and max of, *5.1* and *46.7* respectively. These high distances can be attributed to the high feature vector dimensionality, which leads to extreme distances between feature vectors. Therefore, a normalisation and reduction of the dimensionality of the feature vector had to be done, so a fair comparison between the other tests. This gives a new inter ID min and max of, *22.1* and *50.8* respectively, and intra ID distances of *3.0* min and *44.7* max.

Network	Tested on	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Dual-TransReID	Both	94.78	23.54 / 51.60	3.34	5.15 / 46.70	7.23

Table 7.3: Stats for the Dual-TransReID

7.2.4 First Iteration evaluation

From the first iteration of tests, it can be seen that the addition of another network with information from the luggage images improves the accuracy. There is an increase in accuracy, when compared with the solo networks.

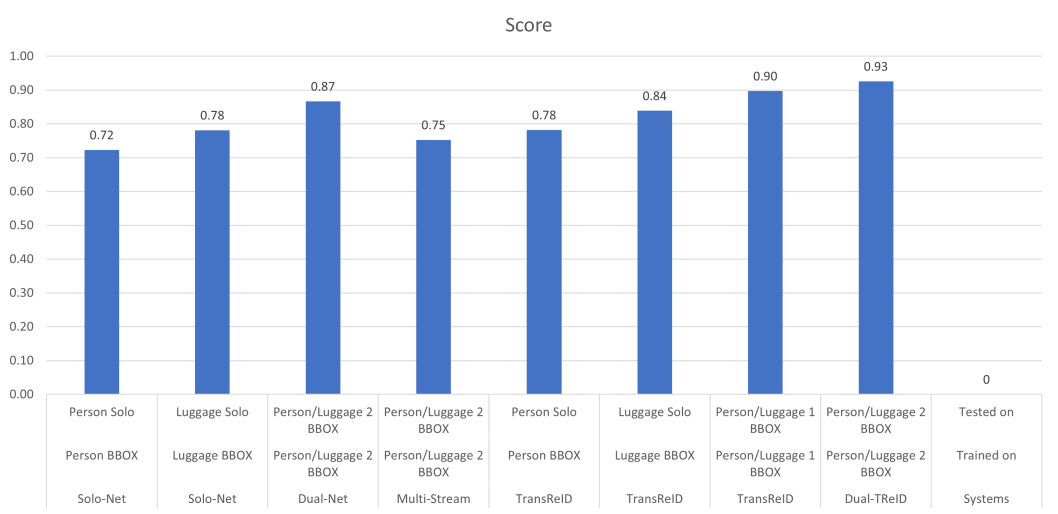


Figure 7.4: Current accuracy of the first iteration of tests and baseline score from TransReID

Network	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Dual-TransReID	94.8%	23.54 / 51.60	3.34	5.15 / 46.69644	7.23
Dual-Net	86.7%	0.19 / 2.10	0.20	1.4551915e-11 / 2.01	0.52
Multi-Stream	75.3%	0.69 / 2.04	0.28	2.910383e-10 / 1.68	0.28

Table 7.4: Stats for the three primary tests done for the first iteration.

In figure 7.4 and the table 7.4, it can be seen that the addition of luggage information increases the accuracy. This is whether or not it is given in a separate bounding box or contained in the same bounding box as the person it belongs to. Specifically seen in the tests with TransReID, it scored an accuracy of *90%* with a bounding box containing both person and the luggage they were carrying. Though when splitting the persona and luggage into two separate bounding boxes and running them though a dual-TransReID network. The Dual-TransReID scored an accuracy of *93%*, this increase in accuracy is also proven with the Dual-Net test. The Dual-Net test accuracy increased by *15%*, when comparing ReID'ing a bounding box with only the person to ReID'ing a bounding boxes with a person and luggage. Though there was a big drop in accuracy, when training the system in a multi-stream setup. A suspected reason is the lost function needed to describe two different inputs at the same time, possibly requiring a much longer training time. As for the Inter and Intra distances, they are much harder to compare. The output of Dual-TransReID is a feature vector with a dimensionality of 7680, where the Dual-Net and Multi-Stream both have a dimensionality of 256.

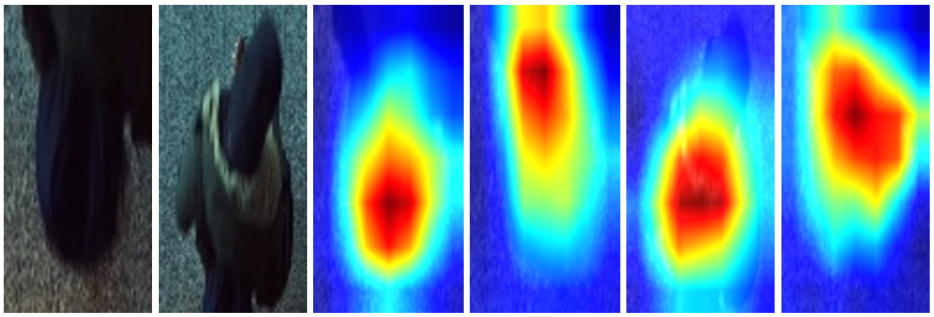


Figure 7.5: Grad-CAM[17] heat maps, the first two images are the input to the networks. The two middle are is the Multi-Stream visualisation of the last convolution layer. The last two images are the Dual-Net visualisation.

Grad-CAM heat map shows the attention or activation of a convolution layer on a given input[17]. In this case, the last convolution layer was used to generate the heat maps seen in 7.5. Here it can be seen that the Multi-Stream attention heat maps are a bit smaller than the Dual-Net heat maps. This shows a wider area of attention taken into account for the dual-net compared to the Multi-Stream network.

7.3 Second Iteration

Addition of MLP neural network after the original networks to help narrow in the feature vectors. This MLP consists of 3 layers with nodes corresponding to the output of the network before. This addition of a follow-up MLP, should help guide the output to be more descriptive after the concatenation of the two networks[43]. This MLP network consists of 2 hidden layers and then an output layer. The purpose of the MLP is to do a dimensionality reduction on the feature vector and possibly improve the fusion of two networks[44]. The First hidden layer has 256 neurons, followed by the second hidden layer with 128 neurons. The last output layer is also at 128 neurons and contains the feature vector for the MLP. Meaning there are 50944 parameters for the MLP, with 96 of them being none-trainable parameters.

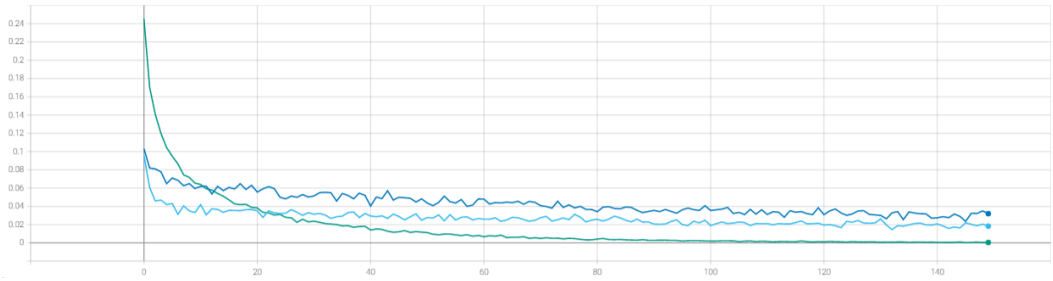


Figure 7.6: Graph showing the loss during training of the three networks. Green: TransReID, dark blue: Multi-Stream, light blue: Dual-Net.

Network	Start Loss	Epoch of half Loss	End Loss
Dual-TransReID	0.2455	3rd Epoch	0.0005
Dual-Net	0.1031	24th Epoch	0.03203
Multi-Stream	0.0958	2nd Epoch	0.01833

Table 7.5: Showing the loss drop for the trained networks

During the training, it was noted that the TransReID loss was much greater and fell much lower during training. This can be attributed to the higher dimensionality feature vector output of the TransReID architecture. With the larger feature vector, there is more data to work with and therefore the loss function starts off higher, but is able to generalise more based on it..

7.3.1 Second Iteration evaluation

The results of the addition of a follow-up MLP, trained on the output of each test in the previous iteration of tests. With the addition of an MLP all networks had an increase in accuracy and an improvement in other metrics. While also having an effect on the intra and inter distances. The TransReID showed an accuracy increase of *0.9%* and the intra also got grouped closer togeather. The intra STD of the TransReID went from *7.2* to *0.1*, as the min and max also decreased. But in this case the inter distances also decreassed, with these inter distances, it is more useful to have higher distances. This seems to be a trend with the Dual-Net and Multi-Stream networks too, as their accuracy did increase and both inter and intra distance decreased. Dual-Net accuracy increased by *2.2%* from *86.7%* to *88.9%*, dual net was the only network where the min intra distance increased. Multi-Stream accuracy increased by *1.4%* from *75.3%* to *76.7%*, it also had the least change in Intra STD.

There can be multiple reasons for all of these changes, coming from the inclusion of an MLP network. For the Multi-Stream the small change in Intra STD, can be attributed to the fact the two sides of the networks were trained as one. This method of training already produced a feature vector that worked together after the concantenation. The Dual-Net had the smallest change in inter STD, likely attributed to the training method the Dual-Net went though. Its training focused on differentiation

Network with MLP	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Dual-TransReID-MLP	95.7%	0.91 / 1.66	0.18	0.05 / 0.94	0.10
Dual-Net-MLP	88.9%	0.35 / 1.68	0.19	2.3267341e-06 / 1.46	0.28
Multi-Stream-MLP	76.7%	0.06 / 1.68	0.19	1.9633164e-05 / 1.52	0.21
Network without MLP	Accuracy (R1) %	Inter ID min / max	Inter STD	Intra ID min / max	Intra STD
Dual-TransReID	94.8%	23.54 / 51.60	3.34	5.15 / 46.69644	7.2346725
Dual-Net	86.7%	0.20 / 2.10	0.20	1.4551915e-11 / 2.01	0.52
Multi-Stream	75.3%	0.69 / 2.04	0.28	2.910383e-10 / 1.68	0.28

Table 7.6: Stats for the second iteration

In the table above it can be seen the improvements an MLP Network provides.

8 - Discussion

In this chapter, an exploration of things in the report will be discussed. The idea of this discussion is to explore what could have been done differently and what changes could have been made to get different results. simCLR was not tested with the newly created dataset and the structure needed for that system was outside the scope of resources for this report.

8.0.1 Results

The results for ResNet architecture show a form of improvement, though the results also show greater accuracy when the TransReID is employed in a Dual-Net system. Multiple aspects could improve the results e.g. Optimization Algorithm, Loss Function and datasets.

8.0.2 Dataset

There are a few limitations with the dataset, as currently there are very few IDs only coming from two cameras. With the current dataset, the variation is very limited. Making a larger more varied dataset could help the networks better generalise for the task of Re-Identification. This dataset is also only a top view of people and luggage, this could also affect the accuracy of the networks. For the dataset a test for the augmentations should have been done, these augmentations expand the range of representations and aids the network to generalise during training. What types of augmentations should be researched, to extend the dataset.

8.0.3 Evaluations and methods

The methods used to evaluate the tests and baselines are the rank-1 and inter/intra distances for the outputs of the tested systems. The general measurements between all tests are the Rank-1, this represents the accuracy of the network giving descriptive feature vectors for a given input. This metric represented the best first case guess for each system when classifying one output to a collection of feature vector ID centers. More methods of evaluating the short comings of the networks should have been used and researched to get a better understanding of what was going on.

8.1 Future work

This section will shortly go through things that could be looked into, if more work is dedicated to this paper in the future. One of the main challenges for the proposed systems is the lack of datasets carrying both labeled person and luggage. The step before doing object detection is something to be looked into. Currently there are not many datasets out there, so making a system for that specified object detection would be interesting.

Another aspect of the dataset that needs to be looked into is the method of linking luggage to a person. Some of these cases can be done with a distance measurement, between person and luggage. This method would in theory work well on backpacks and luggage carried closely to the person it belongs to. Though luggage such as rolling suite cases carried behind the person it belongs to, would prove more difficult to link with that person. People walking right behind that luggage could be wrongly linked with the use of a simple distance measurement.

Then comes the network architectures, more networks should be tried in a dual-net setup. More research into methods of fusing and training Multi-Stream networks. This extends into late or early fusion, as well as researching multi-stage fusion.

9 - Conclusion

The focus of this report was to research Computer Vision regarding person Re-identification in sectors such as airports. Usage for this topic was the general purpose of further Re-identification, with differently structured datasets. The research here aims to help this challenge by investigating sets of bounding boxes meant for Re-Identification people with luggage.

For this report an initial problem formulation was made:
How can a Computer Vision system be used to Re-identify individual people with the use of the luggage they carried?

With this initial problem formulation, research was made into past approaches of this topic. In chapter 2 past methods were analysed and used for setting the approach for the tests and methods in this report. With these previous solutions evaluated, a continuation of delimitations was created in chapter 3. These delimitations determine the approach of the following sections and serves as guidelines for the rest of the testing and implementation of tests done in this paper.

Out from this a final problem formula was made:
"How can a network use a bounding box set of person and luggage to Re-ID them across multiple instances, what architecture works well for generating descriptive feature vectors."

Following this final problem formula, the next step was designing the tests and Networks to be evaluated against the baseline. These design decisions and considerations in chapter 4, outlines the network architecture used throughout the paper. This section also looked into the needed aspects for both the dataset and modifications networks like TransReID needed.

The following chapter 5 goes through multiple optimization factors for training and improving the effectiveness of the networks tested. In this chapter, multiple loss functions and optimisation algorithms used in training were analysed and described. The implementation of the different approaches are described in chapter 6.

Chapter 7 goes through the different tests and methods for improving the net-

works and their results from the tests. Here the TransReID network went through a preliminary test in section 7.1, that showed a Rank-1 accuracy of *90%*. The first iteration of tests, showed an improvement when using the multi bounding box explained in section 4.2. Here the Dual-TransReID network achieved an accuracy *94.8%*, on the dataset with two bounding boxes. This improvement was also proved with the ResNet-50 architecture. When a Resnet-50 network trained on person solo dataset, it scored an accuracy of *72.3%* and *78.1%* on luggage solo dataset. When it is tested as a Dual-Net configuration, it achieved an accuracy of *86.7%*.

The second iteration of tests focused on improving the fusion of the features from the past iteration of tests. This was done with an addition of an MLP network, this had a lesser yet positive effect on the accuracy of all networks. The increase in accuracy across all networks was *0.9%* to *2.2%* with the addition of an MLP. With this additional follow-up network, the STD of the output feature vectors decreased. The most drastic decrease were the Dual-TransReID, that network had an STD drop from *7.2* to *0.1*. This change can be attributed to the feature dimensionality being reduced from 7680 to 128.
A better fusion method, could possibly further improve the networks. More research should be done. However there is sertainly merit to dividing a bounding box containing both person and luggage up into two bounding boxes that contain a person and luggage separately.

Bibliography

[1] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. "Beyond Triplet Loss: A Deep Quadruplet Network for Person Re-identification". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.145. URL: https://doi.org/10.1109/cvpr.2017.145.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *arXiv preprint arXiv:2002.05709* (2020).

[3] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *arXiv preprint arXiv:2006.10029* (2020).

[4] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. "TransReID: Transformer-Based Object Re-Identification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15013–15022.

[5] Chi Su, Jianing Li, Shiliang Zhang, Junliang Xing, Wen Gao, and Qi Tian. "Pose-Driven Deep Convolutional Model for Person Re-identification". In: (Oct. 2017). DOI: 10.1109/iccv.2017.427. URL: https://doi.org/10.1109/iccv.2017.427.

[6] Di Wu, Si-Jia Zheng, Xiao-Ping Zhang, Chang-An Yuan, Fei Cheng, Yang Zhao, Yong-Jun Lin, Zhong-Qiu Zhao, Yong-Li Jiang, and De-Shuang Huang. "Deep learning-based methods for person re-identification: A comprehensive review". In: *Neurocomputing* 337 (Apr. 2019), pp. 354–371. DOI: 10.1016/j.neucom.2019.01.079. URL: https://doi.org/10.1016/j.neucom.2019.01.079.

[7] Alex Krizhevsky. "One weird trick for parallelizing convolutional neural networks". In: (Apr. 2014). arXiv: 1404.5997 [cs.NE].

[8] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: (Sept. 2014). arXiv: 1409.1556 [cs.CV].

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: (Sept. 2014). arXiv: 1409.4842 [cs.CV].

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: (Dec. 2015). arXiv: 1512.03385 [cs.CV].

[11] Stefan Schneider, Graham W. Taylor, Stefan Linquist, and Stefan C. Kremer. "Past, present and future approaches using computer vision for animal re-identification from camera trap data". In: *Methods in Ecology and Evolution* 10.4 (2019), pp. 461–470. DOI: https://doi.org/10.1111/2041-210X.13133. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13133. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13133.

[12] URL: https://www.image-net.org/.

[13] Thommen George Karimpanal and Roland Bouffanais. "Self-organizing maps for storage and transfer of knowledge in reinforcement learning". In: (Nov. 2018). arXiv: 1811.08318 [cs.AI].

[14] Jinrui Yang, Jiawei Zhang, Fufu Yu, Xinyang Jiang, Mengdan Zhang, Xing Sun, Ying-Cong Chen, and Wei-Shi Zheng. "Learning To Know Where To See: A Visibility-Aware Approach for Occluded Person Re-Identification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 11885–11894.

[15] Tianyu He, Xu Shen, Jianqiang Huang, Zhibo Chen, and Xian-Sheng Hua. "Partial Person Re-Identification With Part-Part Correspondence Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9105–9115.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: (June 2017). arXiv: 1706.03762 [cs.CL].

[17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: (June 2017). arXiv: 1706.03762 [cs.CL].

[19] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers". In: (June 2021). arXiv: 2106.10270 [cs.CV].

[20] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. "Discriminative unsupervised feature learning with exemplar convolutional neural networks". In: (June 2014). arXiv: 1406.6909 [cs.LG].

[21] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. "Unsupervised feature learning via non-parametric instance-level discrimination". In: (May 2018). arXiv: 1805.01978 [cs.CV].

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with Contrastive Predictive Coding". In: (July 2018). arXiv: 1807.03748 [cs.LG].

[23] Philip Bachman, R Devon Hjelm, and William Buchwalter. "Learning representations by maximizing mutual information across views". In: (June 2019). arXiv: 1906.00910 [cs.LG].

[24] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive Multiview Coding". In: (June 2019). arXiv: 1906.05849 [cs.CV].

[25] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum Contrast for unsupervised visual representation learning". In: (Nov. 2019). arXiv: 1911.05722 [cs.CV].

[26] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S M Ali Eslami, and Aaron van den Oord. "Data-efficient image recognition with Contrastive Predictive Coding". In: (May 2019). arXiv: 1905.09272 [cs.CV].

[27] Aske R. Lejbølle, Benjamin Krogh, Kamal Nasrollahi, and Thomas B. Moeslund. "One-To-One Person Re-Identification For Queue Time Estimation". In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 1706–1710. DOI: 10.1109/ICIP40778.2020.9191293.

[28] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. "Scalable person re-identification: A benchmark". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1116–1124.

[29] Haobo Lyu, Hui Lu, Lichao Mou, Jonathon Wright, Li Xuecao, Xinlu Li, Xiao Zhu, Jie Wang, Le Yu, and Peng Gong. "Long-Term Annual Mapping of Four Cities on Different Continents by Applying a Deep Information Learning Method to Landsat Data". In: *Remote Sensing* 10 (Mar. 2018), p. 471. DOI: 10.3390/rs10030471.

[30] *CUDA Zone*. 2021. URL: https://developer.nvidia.com/cuda-zone.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: (Dec. 2015). arXiv: 1512.03385 [cs.CV].

[32] Shruti Jadon. "An Overview of Deep Learning Architectures in Few-Shot Learning Domain". en. In: (2020). DOI: 10.13140/RG.2.2.31573.24803/1. URL: http://rgdoi.net/10.13140/RG.2.2.31573.24803/1.

[33] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. *Qualitatively characterizing neural network optimization problems*. 2015. arXiv: 1412.6544 [cs.NE].

[34] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[35] Chunlin Wang, Jianyong Sun, Wanjin Xu, and Xiaolin Chen. "Depth Learning Standard Deviation Loss Function". In: *Journal of Physics: Conference Series* 1176 (Mar. 2019), p. 032050. DOI: 10.1088/1742-6596/1176/3/032050.

[36] Pengfei Fang, Pan Ji, Lars Petersson, and Mehrtash Harandi. "Set Augmented Triplet Loss for Video Person Re-Identification". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 464–473.

[37] Z. Ming, J. Chazalon, M. M. Luqman, M. Visani, and J. Burie. "Simple Triplet Loss Based on Intra/Inter-Class Metric Learning for Face Verification". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 1656–1664. DOI: 10.1109/ICCVW.2017.194.

[38] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. "A Comprehensive Survey of Loss Functions in Machine Learning". In: *Annals of Data Science* (Apr. 2020). DOI: 10.1007/s40745-020-00253-5. URL: https://doi.org/10.1007/s40745-020-00253-5.

[39] C. Zhao, X. Lv, Z. Zhang, W. Zuo, J. Wu, and D. Miao. "Deep Fusion Feature Representation Learning With Hard Mining Center-Triplet Loss for Person Re-Identification". In: *IEEE Transactions on Multimedia* 22.12 (2020), pp. 3180–3195. DOI: 10.1109/TMM.2020.2972125.

[40] Mariusz Kurowski, Andrzej Sroczyński, Georgis Bogdanis, and Andrzej Czyżewski. "An Automated Method for Biometric Handwritten Signature Authentication Employing Neural Networks". In: *Electronics* 10.4 (Feb. 2021), p. 456. DOI: 10.3390/electronics10040456. URL: https://doi.org/10.3390/electronics10040456.

[41] Alexander Hermans*, Lucas Beyer*, and Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". In: *arXiv preprint arXiv:1703.07737* (2017).

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: (Mar. 2015). arXiv: 1503.03832 [cs.CV].

[43] Xianzhi Du, Mostafa El-Khamy, Jungwon Lee, and Larry Davis. "Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 953–961.

[44] Peixiang Huang, Runhui Huang, Jianjie Huang, Rushi Yangchen, Zongyao He, Xiying Li, and Junzhou Chen. "Deep Feature Fusion with Multiple Granularity for Vehicle Re-identification." In: *CVPR Workshops*. 2019, pp. 80–88.

[45] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[46] *TensorFlow Core: Machine Learning for Beginners and Experts*. URL: https://www.tensorflow.org/overview.

[47] *PyTorch an open source machine learning framework*. URL: https://pytorch.org/.

[48] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and Realtime Tracking with a deep association metric". In: (Mar. 2017). arXiv: 1703.07402 [cs.CV].