Faculty of Computer Science and Electrical Engineering
Fachhochschule Kiel

Master Thesis

Triplet Deep Learning Encoder for Body-Based Person Recognition

Submitted in fulfilment of the requirements for the degree of
Master of Science

Written by
Rathna Srinivas Murthy
Matriculation Number: 932230

In association with
Tagsonomy SL (DIVE)
Calle de Cedaceros 10 Madrid
28014 Spain

Under the supervision of
Prof. Dr. Hauke Schramm, Fachhochschule Kiel
José Carlos González Sánchez, DIVE

June 17, 2020

Abstract

In today's world, video surveillance systems are beneficial to track and identify people and add a prominent value to their safety. Person Re-Identification [1] is a task of identifying a person captured by a single camera across time or by multiple cameras with non-overlapping fields of views has seen greatest difficulty in achieving success. This task has significant challenges due to various scales, viewpoint variations, illumination changes, pose variations and similar appearances. In recent days, it has received a great attention in the Computer Vision research group. In order to achieve better performance of these tasks in surveillance systems, it is always necessary to encode robust representation by extracting similarities and dissimilarities in the data patterns.

The aim of this project work are to (*i*) implement a triplet convolutional neural network using deep metric learning in order to learn an embedding space and extract robust representation of similarities in the data patterns. (*ii*) Evaluate the system performance by experimenting different base architectures, variants of loss functions and the various mining approaches.

The triplet convolutional neural network was implemented using Keras on top of TensorFlow. The network is evaluated using Rank-1, Rank-5 and mAP metrics using Market-1501 public dataset. The results are compared to the state-of-the-art algorithms on this dataset.

The experiments showed that the system trained with ResNet-50 as base architecture and lossless triplet loss with offline triplet mining achieved better performance; Rank-1 accuracy of 92.1% and mAP of 81.9% was achieved. In conclusion, the results were satisfying and showed the capability of triplet neural network to tackle the extraction of robust representation from the data in a feasible and reliable way.

Keywords

Deep Learning, Triplet Convolutional Neural Network, Person Re-Identification, Lossless triplet loss, Rank-1 accuracy, Mean average precision.

Declaration

I, Rathna Srinivas Murthy, do hereby declare that the dissertation titled **Triplet Deep Learning Encoder for Body-Based Person Recognition** has been undertaken for the award of Master of Science. I have completed this study under the supervision of **Prof. Dr. Hauke Schramm, Professor at Fachhochschule Kiel** and **José Carlos González Sánchez, Data Scientist at DIVE**.

I also declare that I have written my own Master's thesis and have not received any other degree or diploma from any university or other tertiary education institute. The information derived from other publications and unpublished works was acknowledged in the text and the list of references is presented in the bibliography.

Ingolstadt, 19th June 2020

Place and Date

Rathna Srinivas Murthy

Acknowledgements

I acknowledge the support of all the people who guided and assisted for successful completion of my Master thesis.

First of all, I would like to thank my Prof. Dr. Hauke Schramm, (Department of Computer Science and Electrical Engineering, Fachhochschule Kiel, Germany) for supervising my thesis work. I am immensely thankful for his priceless support, advices, guidance, valuable comments and suggestions throughout the stages of the thesis work and the final report. The successful completion of the project is only because of his outstanding supervision.

The project was done at the company, Tagsonomy SL (DIVE), Madrid, Spain. I express sincere thanks to my supervisor Jose Carlos González Sánchez, Data Scientist, for helping with the project formulation, necessary guidance and continued support throughout the time frame of the project. His persistent supervision helped me complete my thesis successfully and satisfying the company's expectations. I am also thankful to my colleague Ana Caballero, who helped me in structuring my report and guiding me until the end of successful writing. I thank for the comfort and supportive environment provided by other colleagues at the company.

Lastly, I would like to thank my parents who gave a strong motivation and moral support throughout the completion of the project. A special thanks to my friend Shriram Narayanan who had been a great support and motivation throughout the period.

-Rathna Srinivas Murthy

Contents

Abstract ii

Declaration iii

Acknowledgements iv

Contents v

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 Company . . . 1
- 1.2 Person Re-Identification . . . 3
- 1.3 Aim of the thesis . . . 7

2 Literature Study 8
- 2.1 Introduction to Deep Learning . . . 8
- 2.2 One Shot Learning . . . 15
- 2.3 State-of-the-art . . . 17

3 Methodology 19
- 3.1 Data Processing . . . 19
- 3.2 Triplet System . . . 25
- 3.3 Implementation . . . 35

4 Experiments and Results 38
- 4.1 Evaluation Metrics . . . 38
- 4.2 Experimental Results and Discussion . . . 40

5 Conclusion and Overlook **53**
5.1 Conculsion 53
5.2 Scientific Contribution 54
5.3 Future works 54

Bibliography **55**

List of Figures

1.1 The pipeline of Person Re-Identification system [1] . . . 3
1.2 Challenging examples of Person Re-Identification problem . . . 5

2.1 Biological Neuron versus Artificial Neural Network [2] . . . 9
2.2 Example of a Convolutional Neural Network . . . 11
2.3 Illustration of convolution operation [3] . . . 12
2.4 Example of a Siamese Network . . . 17

3.1 Sample images in the dataset [4] . . . 20
3.2 Example of query images [4] . . . 21
3.3 Number of identities captured by each of 6 cameras . . . 22
3.4 Structure of DataFrame . . . 23
3.5 Examples of Triplets . . . 25
3.6 Example of a triplet . . . 26
3.7 Example of a Triplet Neural Network . . . 27
3.8 MILDNet Architecture [5] . . . 29
3.9 Modified ResNet50 Architecture [6] . . . 30
3.10 The three types of negatives in triplets [7] . . . 31
3.11 Workflow of Offline Triplet Neural Network . . . 32
3.12 Workflow of Online Triplet Neural Network . . . 33
3.13 Non-linear Anchor-Positive and Anchor-Negative distance curve [8] . . . 35

4.1 Examples of ranked list of images for a given query image . . . 39
4.2 Calculation of AP for a query image with 4 true positives . . . 39
4.4 Examples of model trained with MILDNet architecture . . . 43
4.6 Examples of model trained with ResNet-50 architecture . . . 45
4.7 Learning curve for the models trained with triplet loss and lossless triplet loss . . . 47
4.9 Examples of model trained with ResNet-50 and lossless triplet loss architecture . . . 49

List of Tables

3.1 Overview of Person Re-Identification Datasets 20

4.1 Evaluation metrics for systems trained with two different base architectures 41
4.2 Evaluation metrics for model with two different loss functions 47
4.3 Evaluation metrics for model with two different mining approaches . . 51
4.4 State-of-the-art comparison to the thesis work 52

Chapter 1

Introduction

Automatic video surveillance systems are able to monitor different people's activities and behaviours. Tracking and identifying people in different indoor and outdoor environments using surveillance systems is an essential functionality in security, safety and retail applications. Due to various constrains in a particular environment such as variations in background produced by cameras located at different places, variations in appearance as people's deformable postures change over time, uniform clothing and different camera angles, person re-identification still remains a challenging and demanding task. In recent years, computer vision research in this field has received a lot of attention. Most of the person re-identification approaches are based on classification approach [9, 10] and produce a feature representation that suffer from low accuracy. Recently, deep metric learning has seen progress in handling this problem. Deep metric learning aims at learning a distance metric and providing a robust feature representation capturing the similarities and dissimilarities in the data patterns.

This section discusses the person re-identification problem in detail and describes its various applications and challenges. Also, a problem statement is defined that needs to be addressed in the thesis. Finally, the aim of the thesis is mentioned to give an overview of the project.

1.1 Company

Dive is a full stack AI platform that enables businesses working with videos to adopt AI based solutions quickly and economically. They make it simple and easy for organisations to deploy AI capabilities and solutions into their products and processes. The platform comprises of advanced capabilities in the recognition and inspection of faces, human body crops, objects in videos, information retrieval as well as recommender systems. The three areas of focus are Visual Analytics, Interactive

Applications and Precise Recommendation.

Dive's ecosystem of sophisticated artificial intelligence models and algorithms have been developed and productionised over years of cutting-edge research and product development. The combined capabilities of Dive's models provide a broad base of functionality covering all important visual analytics tasks, and these can be freely combined and adjusted to adapt to any realistic use case or limitation.

In addition to the state-of-the-art research, they have invested heavily into the productionisation of the algorithms. The result is the ability to reliably and efficiently process hundreds of time-synchronised video feeds per site at low latency, while performing deep multi-model analysis on each and every frame. Further, they go beyond the traditional single-perspective analysis by performing multi-camera inference - combining the results from many cameras viewing the same location and thereby generating deeper insights into the three-dimensional space.

Dive Customer Journey Analytics

The company's main focus is towards Customer Journey Analytics. Dive's Customer Journey Analytics (CJA) system is a selection and configuration of the company's proprietary models designed to generate insights beyond what is normally available in the retail space. State-of-the-art artificial intelligence provides an opportunity for brick-and-mortar retail to gain access to the metrics, insights and understanding that has until now only been available to online retail.

Dive's CJA system processes thousands of video frames in real-time, so the models which directly process these full-resolution frames have been selected and tuned for extreme efficiency while retaining excellent accuracy. The various state-of-the-art deep learning models in their Customer Journey Analytics includes detecting people, detecting faces, face identity, body identity, sentimental models, demographic models, spatial calibration, 3D pose detection and action classification.

Frame of the project

One of the important use cases for a retail space is to perform customer tracking as they move within the camera field of view in order to gather location statistics such as entry and exit counts, store-based dwell time and zone-based dwell time. In case of a retail space, store-based dwell time is the time spent by a person in a specific store and zone-based dwell time is the time spent in a specific zone. Usually, the videos received from retail space are of low resolution or where faces are too small and facing away from the cameras. In these cases, Dive's CJA uses body-based identification. Good model accuracy across diverse environments, that is, model generalisation is essential for production deployment in the dynamic retail space.

However, most so-called "Re-Identification" models are considered as a classification problem and suffer from low accuracy failing to perform well in many retail use-cases. Hence, the idea of this thesis work is to present an approach that provides a robust representation of similarities in the data patterns and improve the model performance to perform well in most of the retail use-cases.

1.2 Person Re-Identification

1.2.1 Introduction

Person Re-Identification is a task of identifying a particular person captured by a single camera across time or by multiple cameras with non-overlapping fields of views. It is an elementary task in intelligent and automatic video surveillance systems and aims to establish correspondence between images of a person captured by a network of cameras. Re-Identification tasks are different in contrast to the traditional computer vision techniques like classification and detection. Image classification is a supervised learning technique that aims at assigning a class label to an image whereas object detection is a combination of two tasks namely object localization and classification, in other words, it is a technique of drawing a boundary box around an image's object of interest and assigning a class label to each boundary box. In contrast, recognition answers the question who is the person in a given image while re-identification solves the problem of identifying a specific person in multiple cameras or a single camera over a short period of time.

Figure 1.1: The pipeline of Person Re-Identification system [1]

The process of re-identification task consists of different phases. The first phase is to process videos across a set of cameras to detect people by applying object

detection algorithms frame-by-frame and collect the cropped images to prepare a gallery set. Then, the system measures the similarity or correlation of the query image against the entire gallery set and retrieves best matched images based on the similarity measures. The pipeline of the person re-identification system is shown in Figure 1.1. Here, we should remark that the re-identification process cannot be used to find similarities among people over several days due to the change in their visual appearance. In practice, faces cannot always be captured by video surveillance systems. Hence, the body crop of a person is used for this task. It is good to underline that the surveillance cameras span areas from a wide-angle and have non-overlapping fields of view to provide large-scale coverage. A person leaving a field of view and appearing in other views at different locations over a short period need to be matched and distinguished from a set of visually similar but entirely different individuals in those views.

1.2.2 Applications

Person Re-Identification has a range of applications in various fields like security, surveillance, health and criminal investigation.

Person retrieval - Person Re-Identification is correlated to image retrieval technique. In order to achieve this, the embeddings of all the gallery images are generated using the trained system and stored in a large database. Later, a specific person of target identity is provided and its euclidean distance across the entire gallery set is calculated and searched for its instances. Hence, the person re-identification task is useful in scenarios where the aim is to return ranked lists of identities based on their similarities to the query image.

Person tracking in single camera - In a single-camera environment, person re-identification may be helpful as well. In crowded and complex scenarios, person tracking in videos is not a trivial process because of recurrent occlusions and interaction of people. This problem can be overcomed by building a robust person re-identification system across frames in videos. The main idea is to detect different people in one frame and associate the detections to their neighbouring frame detections by comparing the similarity between them. This process is a type of person re-identification.

Cross-camera person tracking - In scenarios having multiple video surveillance systems, person re-identification comes to play in identifying people across multiple cameras to recognise important events and activities and generate insights in case of crowd movement. As soon as a person moves from one camera's position and appears in a different one having non-overlapping field view, person re-identification

is used to provide the correspondence of the person across multiple cameras.

Human Machine Interaction - In case of machine bot, person re-identification is contemplated as "target recognition" where the identity of the respondent is maintained, allowing bot to be constantly aware of the neighboring people.

Customer Journey Analytics - Artificial intelligence provides an opportunity for brick-and-mortar retail to gain access to the metrics, insights and understanding that has until now only been available to online retail. This is being pulled off by analysing customer trends in shopping and spot their touching, surveying of products in different retail stores captured under different cameras. In recent years, most businesses focus on better understanding their customers and creating the best products and experiences.

1.2.3 Challenges

Achieving a robust person re-identification system is intrinsically challenging. The main challenge comes from intra-class variations, that is, variations of appearance of a person across multiple cameras. Also, to control inter-class confusion, that is, many people can look alike in different camera conditions.

Some of the challenging examples are shown in Figure 1.2. The challenging factors of person re-identification task and their effects are as follows:

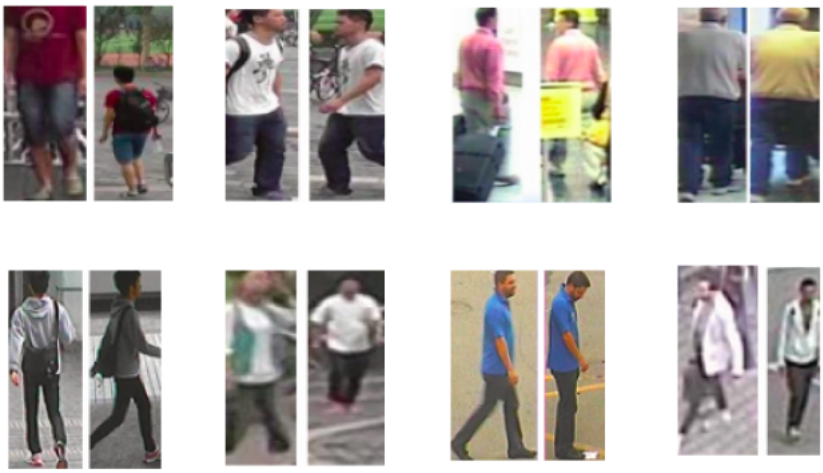


Figure 1.2: Challenging examples of Person Re-Identification problem

Illumination changes - Intensity of daylight, shade, reflected light from colored

surfaces, indoor lighting can vary in different camera views and during the whole course of the day. The same person noticed under different cameras can appear in different colors and shades. This expands the intra-class variation challenging the person re-identification problem.

Low resolution - Many old video surveillance systems consist of very low resolution cameras. Also in practical scenarios, the cost required for the high resolution camera network in all zones could be very expensive resulting in sparse coverage leaving blind spaces. So, limited cameras are installed at high angles and persons are far from the camera resulting in low resolution. Due to the lack of information captured by the cameras, person re-identification has been a challenging task.

Occlusion - In crowded environments, people are completely or partially occluded by other people or objects in the environment. Because of this, some key parts of the person cannot be captured by the cameras leading to a challenge in extracting all the features required for re-identification.

Uniform clothing - The main component for person re-identification is clothing or appearance of the person. In many places, as schools and workplaces, there is a high probability that people have similar clothing since at these places people have uniform clothing. In a crowd, it is obvious to have blue, black and white colors common in clothing. This increases unreliability and ambiguity in the matching process. In such scenarios, it is very difficult to extract discerning features from the visual appearance of different people.

Pose variation - Generally the human body is very deformable in appearance. The system learnt on standing pose will probably fail to detect the same person while running or sitting. The visual changes in appearance and body part localization in the crop image of the person signify the pose variations. It is therefore difficult to retrieve matching persons from the gallery set with pose variations.

Camera viewpoint variation - Since the cameras are installed at high angles, the direction in which people are facing them shows significant variations. From different viewpoint angles the size or shape of the person is different. It is obvious that a three-dimensional view of the person cannot be captured by a single camera. Each camera in fact can capture only a two-dimensional view resulting in partial information of the person's appearance. In terms of shape, images of different people from same viewpoints look more similar than two images of same person from different viewpoints. This is a challenging problem resulting in more intra-class variation and inter-class confusion.

Wide range of candidates in the gallery set - In a public place like an airport or university campus, multiple cameras can capture a massive number of people. There can be a vast amount of identities to match against the query image for person re-identification. It is computationally expensive to match among a large database of identities. Pruning the set of matching identities to the query image can ease this problem.

Change in clothing or accessories - Considering longer time and space between multiple camera views, the chance that people might appear with some changes in appearance or carried objects when viewed from different camera angles. For example, removing a hat or a bag from the back.

Class imbalance among identities for training - Since one person might appear only a few times in the camera network it's difficult to accumulate large number of images of each person. Moreover, it is also feasible that a particular person appears many times in the camera network, then the data amassed is large resulting in class imbalance for a variety of people. The class imbalanced data is usually inadequate to learn a good model.

1.3 Aim of the thesis

The research and development group at Dive would like to investigate various training approaches and build a deep learning model in order to perform well in retail use-cases like customer tracking and gathering their location statistics like entry-exit counts, dwell time, etc.,

The main objectives of the thesis are:

- Implement a triplet convolutional neural network using deep metric learning in order to learn an embedding (feature) space and encode a robust representation of similarities in the data patterns.
- Investigate and evaluate the model performance using different base convolutional neural networks in the triplet system.
- Experiment and evaluate the model performance using different variants of loss functions.
- Analyse and evaluate the two different triplet mining approaches.

Chapter 2

Literature Study

2.1 Introduction to Deep Learning

In the modern decade, deep learning algorithms have become the most popular sub-field of machine learning with representational learning. Deep learning models use complex neural network architectures composed of non-linear transformations to learn hierarchical representation of complex data or features in sequential layers. These algorithms have outperformed the state-of-the-art techniques in the field of Robotics, Computer Vision and Natural Language Processing. In this section, a brief introduction of deep learning and its effectiveness are discussed.

2.1.1 Artificial Neural Network

Neural Networks [11] are a type of machine learning algorithms inspired by biological neurons in human brain. The nervous system in the human body is composed of nerve cells or neurons which are responsible for controlling human behaviors. The structure of a biological neuron is shown in Figure 2.1. A neuron reacts with multiple other neurons and passes information in the form of electrical signals. The neuron receives excitatory and inhibitory inputs through dendrites increasing the membrane potential of the neuron gradually. If the membrane potential reaches a certain threshold, an action potential is established and propagated to the postsynaptic neurons along its axon.

Artificial Neural Network (ANN) is a set of computational units called neurons that are connected to each other by edges associated with weights. The weights are a measure of importance of the connection between the different individual neurons. A group of such neurons are stacked together and represent a layer in the neural network. The rise of neural networks is from a simple single-layer perceptron. Mathematically, this neural network is modeled as a function of inputs, weights and

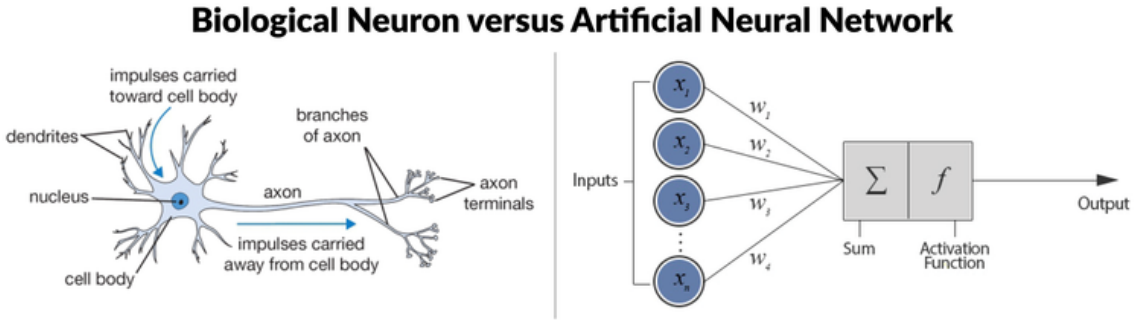


Figure 2.1: Biological Neuron versus Artificial Neural Network [2]

bias. Bias is an additional parameter in neural networks that allows the activation function to shift horizontally along the input axis without altering the shape of the function and helps the model to fit best for the given data. The schematic representation of a single-layer perceptron is shown in Figure 2.1

$$\hat{y} = f(x, w, b) = \begin{cases} 1 & if\ \sigma(\sum_{i=1}^{m} x_i w_i + b) > 0 \\ 0 & otherwise \end{cases} \tag{2.1}$$

In equation 2.1, x is the one-dimensional input vector of size m, w is the weight vector corresponding to the input vector x and b is the bias. The input vector and the weight vector are element-wise multiplied and the values are added. The resulting sum is transformed by an activation function that outputs either 0 or 1.

A single layer perceptron is a linear function which can only learn the linear relationships between the input and the output. The network is learnt on the training data by setting random weights and making predictions. If the predictions are wrong, then the weights are adjusted until the network makes correct predictions and converges. Convergence is achieved only if the input and the output is linearly separable. The learning rule for perceptron to update the weights is,

$$w_{t+1} = w_t + \eta(y - \hat{y})x \tag{2.2}$$

where y is the actual output, $\hat{y}$ is the predicted output and η is the learning rate, which indicates the magnitude of the weight update. Neural networks organised with such topology are also called feed-forward neural networks.

Multi-Layer Perceptrons (MLP) are an extension of single-layer perceptrons having one or more additional layers called hidden layers between the input and the output

layers. If every neuron in a particular layer is connected to every neuron in its previous layer, then it is called a fully connected layer [11]. The hidden layers help the neural network to learn non-linear relationships between the input and output due to non-linear activation functions while propagating the data through the network. The network with just linear activation functions, behaves as a single-layer perceptron in spite of additional hidden layers in them. In MLP, Backpropagation [12] is used as the training algorithm to update the weights instead of just the perceptron learning rule. It is defined as gradient descent optimization algorithm in neural networks.

Backpropagation algorithm has two phases: (1) a forward pass - propagating the activations from input layer to the output layer (2) a backward pass - propagating the errors back from from output layer to the input layer.

At the output layer, backpropagation algorithm computes the gradient (derivative) of the loss function with respect to every output neuron. The gradient is then back propagated to adjust the weights in all the preceding layers until the input layer to reduce the loss. The weight update is done as,

$$w_{t+1} = w_t + \frac{\partial E(w, x, t)}{\partial w} \tag{2.3}$$

where E is a loss function of the weight vector w, the input vector x and the target output t. The loss function penalises the network for making wrong predictions.

2.1.2 Convolutional Neural Networks

In deep learning, Convolutional Neural Network (CNN) [13] is a type of neural networks broadly used in the field of Computer Vision. CNNs have a broad range of applications such as image classification, recognition, semantic segmentation, visual similarity matching, medical imaging, etc., Their use in image classification and object detection has been widely described by authors in [14, 15].

CNNs are motivated by biological activities in which the connectivity pattern between neurons mimic the structure of the visual cortex. Individual neurons respond to stimuli in a compact region known as the "receptive field" of the visual cortex. The receptive fields from different successive neurons overlap partially so that they cover the complete field of vision.

A CNN comprises of an input layer, multiple hidden layers and an output layer as in multi-layer perceptrons. The difference between them is that in CNNs, the hidden layers consist of a sequence of convolutional layers that convolve with the input of

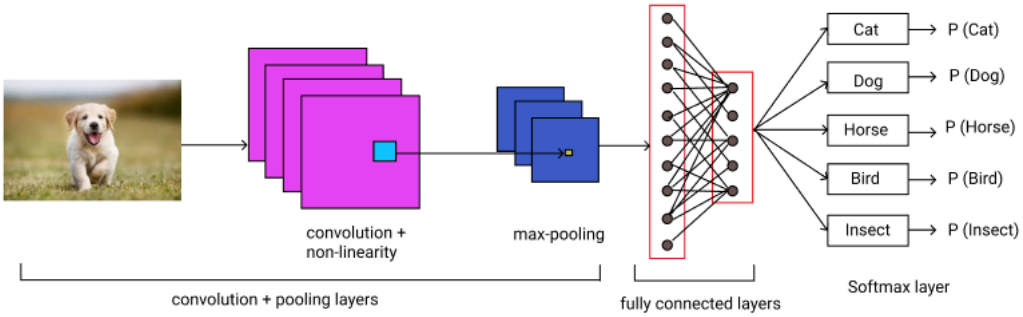


Figure 2.2: Example of a Convolutional Neural Network

the previous layer. Also, a unique feature of CNN is that they assume their input to be images. This allows the network to learn specific patterns in the images.

Each image fed to the network is divided into different segments, each of which is processed by filters to extract specific patterns. CNNs consist of several successive convolutional layers to extract different layers of abstraction in the input image. Networks with more layers can learn complex patterns in the image.

Figure 2.2 shows an example of a convolutional neural network comprising of convolution, pooling and fully connected layers.

Convolution
The main building block of CNN is the convolution layer. It refers to merging two sets of information and generating new information. It is like a fully connected layer where input gets processed by a convolution operation before feeding it to the successive layer. Instead of learning weights for all connections between different layers, convolution operation is performed on each input image using a filter to produce an output called feature map.

A kernel or a filter is a small matrix of weights used to learn features from images at various levels and extract patterns. The kernel is a learnable parameter in this process. This kernel slides over the entire input image. At every position, element-wise matrix multiplication is performed and adds the results to pass on to an activation function and finally outputs a feature map. The area of the filter at every position is called the receptive field. The advantage of a convolution operation is that the same kernel (same matrix of weights and the bias) is used for all the neurons in a hidden layer. Weight sharing radically reduces the number of parameters of CNN.

The illustration of the convolutional operation is shown in Figure 2.3. Each filter

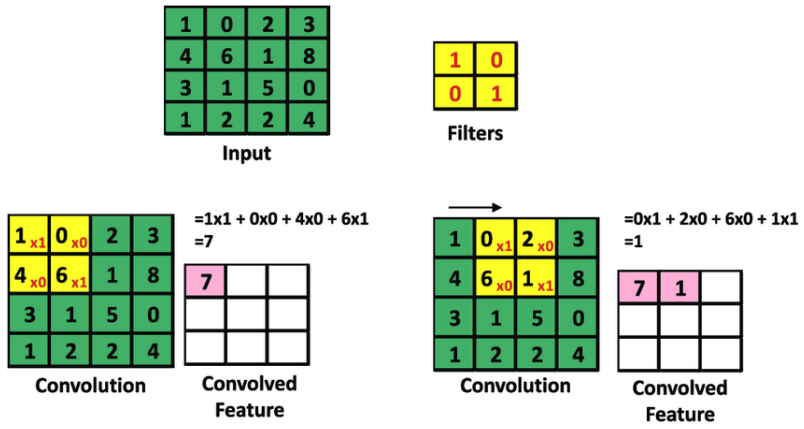


Figure 2.3: Illustration of convolution operation [3]

defined by a weight matrix is capable of detecting a particular pattern i.e, edges, colors, orientations, etc., Therefore, several filters are required to detect multiple patterns in the image to perform tasks like image classification or image recognition. Another advantage of convolutional layers is that they can learn spatial hierarchies of patterns by maintaining the dimensional relationships. For example, incase of person classification, the first convolutional layer can learn basic features like edges and the second layer combines the patterns learnt in the first layer to learn more complex features like eyes, nose, etc., and so on until it learns complex patterns. This allows the convolutional neural network to systematically learn complex and abstract visual patterns in the given input image.

Pooling

Pooling layers are the layers that follow after the convolution layers in CNN. Pooling is performed to downsample the input volume and reduce the dimensionality of the volume. They simplify the information received from the preceding layer and produce a concrete version of the information allowing to extract features at different resolutions. The different types of pooling are max pooling, average pooling and L2-norm pooling. The most common type among them is the max pooling, which returns the highest value in each window of pixels analyzed. It uses a filter of size 2 x 2 and stride of 2. At every window, the maximum feature value is outputted. The size of the feature map volume is reduced while at the same time keeping the significant information. Pooling has no parameters to learn in contrast to convolutional layers. Max-pooling can be replaced by a convolution layer by increasing the stride without loss in accuracy [16].

Activation function

Activation function determines whether or not to activate a neuron by calculating

weighted sum of inputs and adding bias with it. The activation function aims to introduce non-linearity into a neuron's output in complex neural networks. There are many non-linear activation functions which are widely used such as tanh, sigmoid, ReLU, leaky ReLU, parametric ReLU, softmax, etc.,

A commonly used activation function is Rectified Linear Unit (ReLU). It is linear for all positive outputs and zero for all the negative ones. It is easy to compute hence the model takes less time to train and test, converges faster and suffers very less from vanishing gradient problem. The drawback of producing zero for negative values is the problem called "dying ReLU". Leaky ReLU and parametric ReLU are a solution to fix this problem. Leaky ReLU introduces a slight slope for negative inputs rather than a zero as in case of ReLU function.

Backpropogation

In deep learning, the learning process is an optimization problem where the parameters (weights and biases) must be altered so that the loss is minimized. In supervised learning, backpropagation is the most widely used algorithm for training feedforward neural networks. In the process of training a neural network, learning the values of parameters is an important part and is an iterative process. As stated in Section 2.1.1, this process has two phases namely, forward phase and backward phase. The first phase, forward propagation refers to propagating the input data from lower layers to the higher layers to calculate all the intermediate activations and the final output prediction. At the output layer, loss function is used to calculate the error (loss) by comparing the actual output to the predicted output.

The second phase, backward propagation refers to propagating error backwards from output layer through all the hidden layers until the input layer to adjust the values of parameters in order to minimize the error. The ultimate aim of the network is to minimize the error such that the predicted value is equal to the actual value. The model is trained until the loss is minimized and the model outputs good predictions. The neurons in the hidden layer experience only a fraction of total loss, based on its relative contribution to the final output. This process is repeated at each hidden layer and all the parameters are updated accordingly.

Optimization

As stated above, the parameters in the neural network have to be optimized. In practice, they are adjusted using optimizers or iterative optimization algorithms. They are gradient descent algorithms that use backpropagation and compute the derivatives of multiple functions using chain rule. Once the gradients are calculated, these values are used to adjust the weights and biases.

The most common optimization algorithms based on gradient descent are [17] Stochastic Gradient Descent (SGD) [18], Root Mean Square Propagation (RMSProp) [19], Adaptive Moment Estimation (Adam) [20], etc., Adam is an extension of stochastic gradient descent and most widely used optimizer for deep learning tasks in the field of computer vision. It is computationally efficient even for complex optimization algorithms. For each parameter, it decides an adaptive learning rate storing exponentially decaying average of the past gradients.

A usual procedure used in the optimization process is normalizing the inputs and the activations of all the hidden layers. It is known as batch normalization and it normalizes the output of the preceding activation layer by subtracting the mean of the mini-batch and dividing by the standard deviation of the mini-batch. An advantage of batch normalization is that it makes the learning process smoother [21]. This smoothness helps in faster convergence during training.

Regularization techniques
In most cases, the model performs very well on the training data by learning the details but it also learns the noise in the data which can result in poor performance on the unseen data. This problem is denoted as overfitting. Since the main aim of building deep learning models is to do predictions on future data, it is therefore necessary to ensure that the model generalizes well on new unseen data. The problem of overfitting can be overcomed by using the following regularization techniques.

Data Augmentation is the most simple way of reducing overfitting by just increasing the number of training examples. In general, it is not easy to increase the training data as labelling the new data is expensive. The ideal solution is to augment the training examples by randomly applying a set of transformations like rotating, flipping, shifting, scaling, etc.,

L1 and L2 regularization techniques update the cost function by adding an additional penalizing term called regularization term. They are commonly known as weight decay.

$$L(x, y) = \sum_{i=1}^{n}(y - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i| \tag{2.4}$$

$$L(x, y) = \sum_{i=1}^{n}(y - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} (\theta_i)^2 \tag{2.5}$$

Equations 2.4 and 2.5 shows the penalty term added to the loss function. The effect of this term is that the weight values are penalised. Low values are preferred under the assumption that neural networks with small weights lead to simpler models. Therefore, it solves the problem of overfitting.

Dropout is a type of regularization technique. At each iteration, random nodes in some of the hidden layers are removed and the model is trained with the remaining nodes. Each iteration has a different set of nodes resulting in different sets of outputs. This helps in preventing complex adaptation of the same concept in different ways on training data. Dropout is applied only during the training phase, not in validation or testing phase.

Early Stopping is a technique where we split a portion of the training set as a validation set. When the model performance on this validation set is getting worse, the training is stopped.

2.2 One Shot Learning

2.2.1 Classification versus One Shot Learning

The traditional image classification algorithm is a type of supervised learning. These algorithms are used when the outputs are limited to a set of values. In this technique, an input image is passed into a sequence of convolutional blocks and finally at the output layer, a probability distribution over all the classes is generated using a softmax layer as seen in Figure 2.2.

For instance, a classification model to classify four different people namely person1, person2, person3 and person4 generates four probabilities for every input image indicating the probability of the image belonging to each of these four people. Here, each person is considered as a separate class. The main disadvantage of this technique is, a huge number of images are required for each person during the training process. Also, if the model is trained only on the above four classes, it cannot be used to classify a new person. If the model has to categorize the images of the new person as well, a large number of images for this class has to be gathered and the model has to be re-trained including the images of this class. It is extremely difficult to get enough data for each person and the total number of people will be huge as well as dynamically changing. Thus, the cost of data collection and periodical re-training is highly expensive. Hence, person recognition technique is computationally expensive to consider as a traditional classification problem.

In contrast, one-shot learning [22] is a technique where one or few examples are used to classify many new examples in the near future. This technique is very useful in the field of face recognition or person recognition for performing tasks such as identification, verification, tracking and clustering where people must be classified correctly with illumination changes, occlusions and uniform clothing, given one or few examples.

Modern recognition systems approach the problem of one-shot learning for face recognition or person recognition that can address the tasks of verification, recognition and clustering [23].

2.2.2 Siamese Networks

Siamese networks [24] have been historically used to learn embeddings in one-shot learning problems. Mapping of high-dimensional images to a rich low-dimensional learned feature vector representation is called an embedding.

A Siamese network is an architecture having twin neural networks, each network taking a different input, and their outputs are combined to make some prediction. The parameters are shared between the two parallel networks, that is, they are same for the two networks and are updated simultaneously. This architecture is best suited for verification tasks. J. Bromley et al. work [25] first employed this architecture for signature verification in 2005. The example of a Siamese network architecture is shown in Figure 2.4.

The deep CNNs are first trained to discriminate between examples of different classes. The main idea is to first train the model to learn feature vectors representing abstract features in the input images and then compare the two features vectors using L1 or L2 distance and a sigmoid function producing a similarity score between 0 to 1 . The score 0 denotes no similarity between images whereas 1 denotes full similarity between them. This process is also known as Discriminative Learning [26]. The most popular distance function used in literature to solve such problems is the Euclidean distance function [23, 27, 28] calculated using Equation 2.6. During inference time, the trained models are used for verification tasks to predict whether new examples match the template image of each class.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.6}$$

Dimensionality reduction aims at learning a new lower dimensional representation of an input image that preserves the structure of the image such that distances between the output vectors efficiently capture the differences in the input in the feature space. Learning a feature vector representation of an image is an example of dimensionality reduction. The paper of R. Hadsell et al. [29] explores siamese architecture using dimensionality reduction and training the models using contrastive loss.

Contrastive loss evaluates the model performance between pairs of images, in contrast to other loss functions that evaluate the model performance across all input

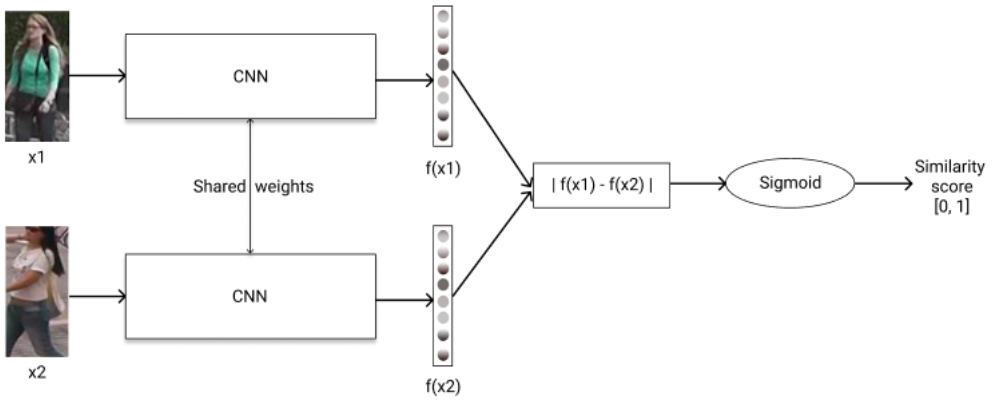


Figure 2.4: Example of a Siamese Network

samples in the training dataset. Given a pair of images, the contrastive loss penalizes the model differently depending on whether the classes of the samples are same or different. Training is done using the standard backpropagation technique explained in Section 2.1.1. If the images belong to the same class, the loss function uplift the models to produce feature vectors that are more similar; and if the classes are different, the loss function uplift the models to produce feature vectors that are dissimilar. The loss function requires a margin prameter m to limit which examples from different pairs are penalized. Choosing the margin parameter m is the main disadvantage of this loss function.

2.3 State-of-the-art

In the progress of deep learning, the idea to see this problem has been changed. One prominent well known example of this is FaceNet [23] system that uses CNN to learn embeddings for faces. The key component of this paper is the use of triple loss as an embedding function for training the CNN. To train such models, triplets are used which consists of 3 inputs, an input (anchor), a positive image and a negative image. The anchor and the positive image belong to the same person whereas anchor and negative images belongs to different person. "The triplet loss optimizes the embedding space such that the data points with the same identity are closer to each other than those with different identities" [27]. The work also presents a novel triplet mining technique to select appropriate triplets during the learning process and contribute to improving the performance of the system.

Current state-of-the-art person re-identification approaches use classification losses to learn deep embeddings [30, 31]. The main disadvantage for classification loss is the increasing number of learnable parameters, as the number of identities increase

during the training phase and most of them will be discarded in the test phase. Some approaches for person re-identification uses a variant of triplet loss to train the systems [32, 33, 34] and has seen moderate success. A different variant of triplet loss has been proposed in the literature for example, fast-approximated triplet loss [35] tries to calculate point-to-set distances rather than point-to-point distances. It considers all images belonging to the same identity as a cluster and re-defines a triplet to anchor, cluster centroid of corresponding identity and centroid of another cluster identity demonstrating efficiency and accuracy. Another example, pose invariant deep metric learning with improved triplet loss [36] adds an additional margin threshold to the original triplet loss defined by Equation 3.3 to further enhance the constraint of the same identity. Also for each input image, posebox and pose estimation confidence scores are calculated, improving the misalignment problem for Person Re-Identification tasks.

A. Hermans et al. e work [27] employ the most popular CNN namely ResNet-50 using triplet loss on CUHK03, Market and MARS datasets and achieved the state-of-the-art results. The ResNet-50 CNN is a prominent system designed for object classification problems and has been trained using ImageNet dataset with 1000 object classes. The main idea of this architecture is to introduce skip connections that skips one or more layers and passes the local features from initial layers until the deeper layers in the network solving the problem of vanishing gradients in deep neural networks.

The work of A. Vishvakarma et al. [5] proposes a challenging novel CNN architecture called MILDNet (Multi-Intermediate Layers Descriptors Net) for visual search and similarity tasks in E-commerce platforms and reachs the current state-of-the-art results in this task. Inspired by the fact that CNN layers are capable of representing the image with increasing levels of abstraction in successive layers, the system compresses the model to a single CNN by coupling activations from multiple intermediate layers called skip connections along with the final layer to capture the features. The output obtained from convolution layers can be portrayed as local features outlining specific image regions. Aggregating these features provide new powerful global features, generalizing the output as a whole.

Chapter 3

Methodology

This chapter presents the dataset used to perform experiments in the thesis work. The pre-processing of the data and the triplet data generation are explained in detail. The design and implementation of the triplet system are discussed in detail. Finally, the details on the pipeline developed throughout the training and testing process is described.

3.1 Data Processing

Processing data is crutial when training deep learning models. Sufficient and appropriate data is necessary to build a successful model. The entire data pre-processing pipeline necessary for training is discussed in detail.

3.1.1 Dataset

In recent years, innumerable datasets have been released for person re-identification task in the literature. A summary of the common available datasets in the literature is shown in Table 3.1.

The datasets available for person re-identification task are categorized into two types: *image datasets* and *video datasets*. The image datasets are further classified into *single-shot datasets* and *multi-shot datasets*. The test set in these cases is further divided into *gallery set* and *query set*. The gallery set is the database of test images and query set consists of query images to test against the gallery images. The gallery set in *single-shot datasets* contains only one true match image for every query identity. ViPeR [37], PRID2011(S) [38] and CUHK01 [39] are the most widely used single-shot datasets. These datasets are relatively small and only include a few hundred identities.

Datasets	#identities	#images	#views
VIPeR	632	1264	2
PRID2011(S)	400	1334	2
CUHK01	971	3884	2
CUHK03	1467	13164	10
Market1501	1501	32217	6
DukeMTMTC-Reid	1812	36441	8
PRID2011(V)	400	24541	2
iLIDS	300	42495	2
Mars	12611	1191003	6

Table 3.1: Overview of Person Re-Identification Datasets

On the other hand, gallery set in *multi-shot datasets* consists of several true match images for every query identity. The most common multi-shot datasets available in the literature are CUHK03 [40], Market1501 [4] and DukeMTMC-Reid [41]. The video datasets are composed of continuous image sequences for each identity. These datasets allow to exploit the temporal information existing in the image sequences. The common video datasets available for person re-identification are iLIDS [42], PRID2011(V) [38], MARS [43]. MARS and PRID2011(V) are the extensions of Market-1501 and PRID2011(S) respectively.

Market-1501 dataset is considered for thesis work due to the following reasons: Firstly, it is the largest available multi-shot image dataset having around 32000 annotated bounding boxes making it available for person re-identification tasks. Secondly, the Deformable Part Model (DPM) [44] is used as a pedestrian detector for automatically extracting bounding box images from the video frames. Third, for each identity there are multiple images under each camera.

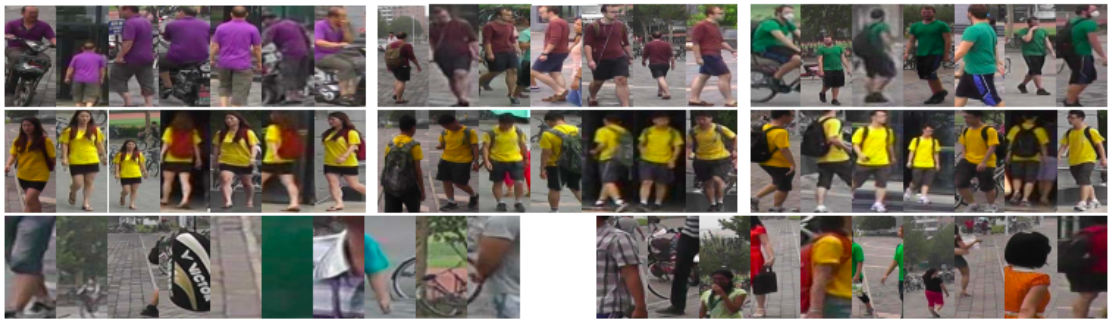


Figure 3.1: Sample images in the dataset [4]

The dataset comprises of images captured in front of a supermarket in Tsinghua University. A total of 6 cameras are used including five 1280×1080 high-resolution cameras, and one 720×576 low-resolution camera. The field-of-view is overlapped among different cameras. Altogether, this dataset comprises of 32,668 annotated bounding boxes of 1,501 different identities. Images of each identity are captured by at most six cameras because of an open environment. Each identity is captured by at least two cameras making it useful for cross-camera search. All images are normalised to a dimension 128×64.

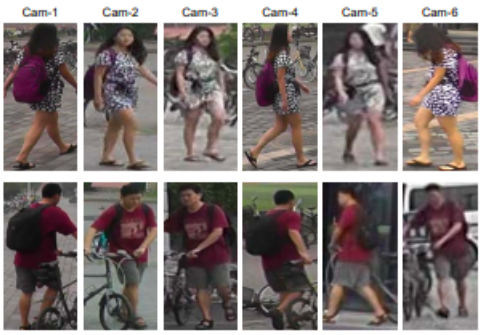


Figure 3.2: Example of query images [4]

Most of the existing datasets use manually cropped bounding boxes, while the Market-1501 dataset applies a state-of-the-art detector called the Deformable Part Model (DPM) [44] on video streams. For each bounding box that is annotated by the DPM, a ground truth bounding box is manually drawn that contains the pedestrian. For both the detected and hand-drawn bounding boxes, the ratio of intersection area to the union area is calculated. If the IOU ratio is larger than 50%, the detected bounding box by DPM is labelled as 'good'; if the ratio is less than 20%, then it is labelled as 'distractor'; otherwise it is labelled as 'junk', defining that this image is of zero influence to the re-identification tasks. Sample images in the dataset are shown in Figure 3.1. The top row in Figure 3.1 shows images of three different identities with unique appearance; the middle row shows three different identities that have similar appearance; the bottom row has some examples of distractors on the left and junk images on the right.

This dataset is randomly divided into train and test sets, containing 750 and 751 unique identities, respectively from the total of 1501 identities. The training set has 12,936 images of the selected 750 identities. The testing set consists of 19,732 images of the selected 751 identities.

For every identity among the 751 identities in the test set, single query image from

every camera is selected forming a query set. Examples of query images of two different identities are shown in Figure 3.2. The remaining images of the test set form the gallery set. Each identity has at most 6 queries, and there are a total of 3368 query images. Figure 3.3 shows the number of identities captured by each of the six cameras. Most of the identities are captured by cameras 1 and 3 and camera 4 captures least identities. It includes identities from both train and test set.

3.1.2 Data labeling and preparation

There is a naming rule for each image in the dataset. For example, each image is named as follows: 0001_c1s1_001051_00.jpg where 'c1' refers to the camera from which it is captured. Here, 'c1' is the first camera and there are 6 cameras. The first four digits '0001' refers to the identity number (each person is given a particular identity number) and 's1' refers to sequence 1 of camera 1. In this context, sequence was automatically defined by the camera as the camera cannot store a whole video that is quite large. Therefore, it splits the video stream into equally large sequences.

Two sequences from the same camera, specifically 'c1s1' and 'c1s2' do not happen exactly at the same time stamp because the starting time of each of six cameras is different. On the other hand, two sequences, namely, 'c1s1' and 'c2s1' almost happen at the same time stamp. '001051' refers to the frame number in the sequence 'c1s1'. The last two digits in the name refers to multiple bounding boxes detected in the same frame for the same identity [45].

The dataset is downloaded from their official project page [45]. The training and testing images are stored in two different directories, *bounding_box_train* and *bounding_box_test* respectively. The selected query images for each test identity are stored

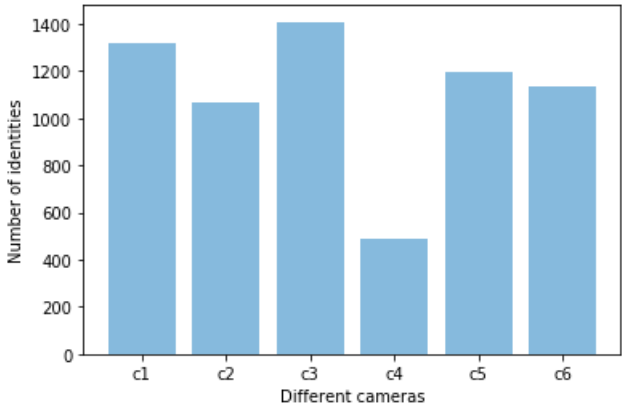


Figure 3.3: Number of identities captured by each of 6 cameras

	file_name	file_path	person_id	camera_id	frame_id	image_size
0	0863_c1s4_047806_02.jpg	./MARKET/bounding_box_train/0863_c1s4_047806_0...	0863	c1	047806	(128, 256)
1	1052_c5s2_142974_01.jpg	./MARKET/bounding_box_train/1052_c5s2_142974_0...	1052	c5	142974	(128, 256)
2	0703_c5s2_055430_02.jpg	./MARKET/bounding_box_train/0703_c5s2_055430_0...	0703	c5	055430	(128, 256)
3	0299_c5s1_067223_02.jpg	./MARKET/bounding_box_train/0299_c5s1_067223_0...	0299	c5	067223	(128, 256)
4	1313_c3s3_054678_01.jpg	./MARKET/bounding_box_train/1313_c3s3_054678_0...	1313	c3	054678	(128, 256)
5	1012_c6s2_124643_06.jpg	./MARKET/bounding_box_train/1012_c6s2_124643_0...	1012	c6	124643	(128, 256)
6	0261_c1s1_055581_03.jpg	./MARKET/bounding_box_train/0261_c1s1_055581_0...	0261	c1	055581	(128, 256)
7	0647_c2s2_030887_01.jpg	./MARKET/bounding_box_train/0647_c2s2_030887_0...	0647	c2	030887	(128, 256)
8	0780_c6s2_103843_06.jpg	./MARKET/bounding_box_train/0780_c6s2_103843_0...	0780	c6	103843	(128, 256)
9	0184_c1s5_012536_02.jpg	./MARKET/bounding_box_train/0184_c1s5_012536_0...	0184	c1	012536	(128, 256)

Figure 3.4: Structure of DataFrame

in a seperate directory.

The preliminary step in the data preparation is extracting the necessary metadata specified by the naming rule of every image. The metadata extracted for the training process are the *person's identity number*, *camera id* and the *frame id* associated with the image. Finally, the metadata extracted along with the filename, filepath and image size are stored in a table-like data structure called *DataFrame* and saved to disk. The process is repeated for both train and test sets. Here, the *person's identity number* is considered as the ground truth labels needed for generating triplets during the training process. The images are resized to 256 x 128 during the training process. Figure 3.4 shows the structure of the data frame.

3.1.3 Data Augmentation

Because the training set consists of few samples, it is essential to expand the training set with new, plausible images.

Data augmentation is an approach to artificially generate more training data from the existing data. This is accomplished by applying various image transformations to the training samples and creating new and unique training samples. The augmentation techniques can also help in improving the ability of the model to generalize well reducing overfitting. The different image transformations include operations such as scaling (zooming), shift (translation), rotation, flip (reflection) and much more.

The Keras open-source deep learning library running on top of TensorFlow, provides the capability to apply data augmentation automatically during training the neural

network. This is achieved by using their ImageDataGenerator class. The data transformation techniques applied on the training samples are as follows: The images are randomly shifted in one direction, such as horizontal and vertical directions by maximum 20% of the pixels. Images are rotated by a random angle in the range between 0° and 25°, randomly zoomed in the range between 80% (zoom in) and 120% (zoom out) by adding new pixel values around the image or interpolating pixel values respectively. Rotations and zooming leading to an out-of-range object are corrected by a translation. Finally, images are flipped horizontally and are specified by a random boolean value.

3.1.4 Triplet Data Generator

A data generator is used to randomly select triplets, an anchor, positive example and a negative example "on-the-fly" at the starting of each epoch during the training process.

A generator function is a function that acts like an iterator and returns an iterator with yield statement instead of a return statement. These iterators compute the value of every item only when needed by the program. This process is useful as the neural network is trained with batches of images instead of images as a whole.

First, the triplet data generator samples random batch of input images. For each anchor (input) image, a positive sample with the same identity number and different camera id as the anchor and a negative sample with different identity number irrespective of camera id as the anchor are selected with the help of DataFrame created in Section 3.1.2 and stored in three different arrays. Each resulting array volume is of size (batch_size, 256, 128, 3). Then, the data transformations explained in Section 3.1.3 are applied to these triplet arrays to perform data augmentation. Finally, the generator yields the modified batch of triplet data for use in the current training step and repeating the same process iteratively until the end of every epoch. It is observed that random sampling of triplet images has few semi-hard and hard triplets sufficient for the first iteration of the learning process.

Figure 3.5 shows few examples of triplets yielded by the triplet data generator. The order of images in all of the triplet examples shown are as follows: anchor, positive and negative images respectively. Figure 3.5(a) is an example of easy triplet as the identities have unique appearance, Figure 3.5(b) is an example of semi-hard triplet as the anchor and negative samples have similar appearances, Figure 3.5(c) is an example of hard triplet as the identities almost have the same clothing whereas Figure 3.5(d) is an example where the positive image contains occlusion adding noise to the training process.

(a) Easy triplet. (b) Semi-hard triplet. (c) Hard triplet. (d) Triplet with occlusion.

Figure 3.5: Examples of Triplets

3.2 Triplet System

This section provides a detailed description of the triplet system, different base architectures, variants of loss function and the triplet mining techniques mainly implemented in the thesis work.

3.2.1 Triplet Neural Networks

The idea of contrastive loss discussed in Section 2.2.2 can be further extended to three examples called the triplet loss. *Triplet loss* was first introduced by F. Schroff et al. from Google in the paper "FaceNet: A Unified Embedding for Face Recognition and Clustering" [23]. Unlike contrastive loss, triplet loss considers a triplet, one input image called an anchor, one positive example having same class as anchor and one negative or non-matching example having a different class. Figure 3.6 shows an

example of a triplet.

Figure 3.6: Example of a triplet

The model trained with triplet loss outputs a feature vector or embedding that has meaningful euclidean relationship such that the images of same class produce feature vectors that have smaller distances in the feature space, they can be clustered together, allowing verification; images of different class produce feature vectors that have large distances in the feature space allowing discrimination from other classes. In other words, a system with triplet loss directly learns a mapping from images to a compact embedding space where euclidean distances directly correspond to their similarity or dissimilarity [23]. For an input image x, the embedding of the image is denoted by $f(x)$. This embedding belongs to a d-dimensional space, $f(x) \in R^d$.

As the main aim of the thesis work is to provide a robust representation of similarities in the data patterns, this technique is incorporated to ensure that the image of a particular person, is closer to all other images of the same person forming well separated cluster in the embedding space which moreover, are significantly discriminated from images of different people in the embedding space. This can be mathematically explained by Equation 3.1:

$$||f(x_i^a) - f(x_i^p)||^2 < ||f(x_i^a) - f(x_i^n)||^2, \forall\ f(x_i^a), f(x_i^p), f(x_i^n) \in \tau \tag{3.1}$$

where x_i^a is the anchor image, x_i^p is a positive example, x_i^n is a negative example, $||f(x_i^a) - f(x_i^p)||^2$ is the l2-distance between anchor and positive, $||f(x_i^a) - f(x_i^n)||^2$ is the l2-distance between anchor and negative and τ is the set of all possible triplets in the training set.

To prevent the model from learning a trivial solution an extra margin is added such that the clusters of each class are separated by a margin. Adding this to Equation 3.1 results in,

$$||f(x_i^a) - f(x_i^p)||^2 + \alpha < ||f(x_i^a) - f(x_i^n)||^2, \forall\ f(x_i^a), f(x_i^p), f(x_i^n) \in \tau \tag{3.2}$$

where α is the margin parameter forced between positive and negative pairs.

The loss that is minimised is called the triplet loss and is given by Equation 3.3. This is also referred to as deep metric learning [46].

$$L(A, P, N) = \sum_{i=1}^{n} max(||f(x_i^a) - f(x_i^p)||^2 - ||f(x_i^a) - f(x_i^n)||^2 + \alpha, 0) \tag{3.3}$$

which pushes $d(a, p)$ to 0 and $d(a, n)$ should be greater than $d(a, p) + \alpha$.

Example of a triplet neural network is shown in Figure 3.7. As the system trained with triplet loss considers three images (a triplet) as input, they are passed into three parallel convolutional neural networks sharing the same weights as shown in Figure 3.7. The triplet system outputs three embeddings, that is, one embedding for each of the input image in the triplet. Finally, we compute the triplet loss for the three embeddings produced by the system as output and backpropagate the error into the system and update the parameters. During training, this loss function penalizes the system such that the distance between the positive examples is less and the distance between the negative examples is more. The process is repeated until the loss is minimized and the system converges.

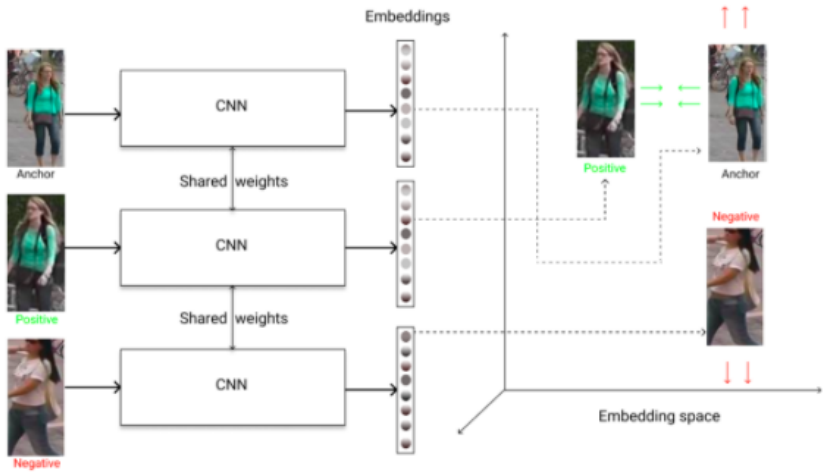


Figure 3.7: Example of a Triplet Neural Network

Triplet neural networks have several advantages over the standard neural networks.

- Triplet neural networks can learn without fully labelled data just with data pairs or triplets. They are more applicable for weakly supervised cases where the training data has no proper labels. Instead it just has information about pairwise or triplet wise relationships among the training samples.

- These networks have better generalization ability in comparison to the normal neural networks. In case of discrete classes in training and testing dataset, these learned models can easily be applied to the unseen classes in the test set.
- It is difficult to train a good classifier when the number of classes is very large around hundreds or thousands or when there are only a few images per class. An alternative solution for this is to train a triplet neural network.
- It is more feasible to update the triplet models continuously. When new classes are included to the training set, these models are directly updated with images of the new class which is not possible for the model trained with classification loss.
- While generating triplets, the negative samples can be varied for the same anchor and positive pairs resulting in more number of triplets while training triplet neural networks.

3.2.2 Base Network Architecture

The two well known network architectures that outperformed in different Computer Vision applications are experimented as the base architecture in the triplet neural network. The performance of both the base networks are compared and discussed in Section 4.2.1.

The first network architecture explored is a novel CNN architecture called MILDNet [5] that has seen significant progress in the field of Visual Search and Similarity in Ecommerce platforms. Figure 3.8 shows the network architecture.

The MILDNet architecture uses VGG16 [47] as its base convolutional neural network that is pre-trained on ImageNet dataset. The VGG16 consists of five convolutional blocks each having two or three convolutional layers and a max-pooling layer with two fully connected layers at the end and a softmax layer as its output. Here, only the five convolutional blocks are considered excluding the fully connected layers at the end. Additionally, features are extracted from four intermediate layers apart from the last convolutional block, just after the max-pooling layer in each convolutional block; also called skip connections. The main idea to add these skip connections is to provide new additional global features by aggregating the local features after each convolutional block.

Global average pooling (GAP) [48] is used to aggregate and flatten the features from these intermediate layers. GAP is a technique that calculates the average output of each feature map and has no trainable parameters. They are added to reduce

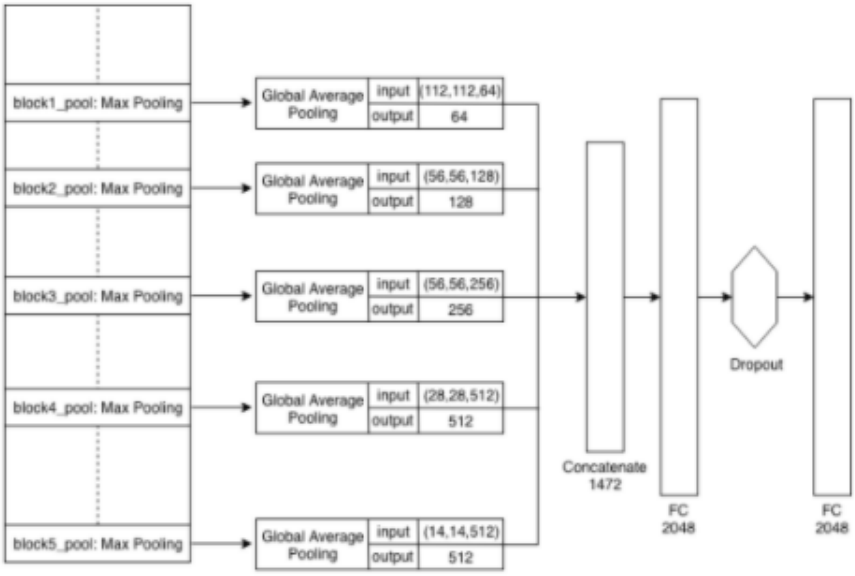


Figure 3.8: MILDNet Architecture [5]

the number of parameters in the network and minimize overfitting. GAP layers are a type of dimensionality reduction technique where given a feature map volume of size $height \times width \times channels$, is reduced to a size of $1 \times 1 \times channels$. Finally, the flattened features are concatenated to a dimension of 1472-d feature vector which is then passed through a FC - dropout - FC layer to generate the desired feature vector of 2048 dimension. The architecture of MILDNet is shown in Figure 3.8.

The second network architecture explored is the ResNet-50 [49], the most groundbreaking work in the field of Computer Vision and Deep Learning in recent years. Many Computer Vision applications such as Object Detection and Recognition have seen significant progress in their performance by taking advantage of the powerful representational ability of ResNet.

The work of He et al. [49] proposes the famous ResNet architecture, also known as Deep Residual Network in 2015. To solve many complex problems, increasing the network depth by simply stacking layers together doesn't work well because of vanishing gradient problems. When the gradient is back propagated to its preceding layers, continuous multiplications makes the gradient extremely small saturating their performance or even degrading rapidly. To overcome this problem, authors introduced "identity shortcut connection" that skips one or more layers. This idea can be implemented in a CNN, by just adding skip connections as shown in the Residual Learning block in Figure 3.9. The skip connections push the new layer to

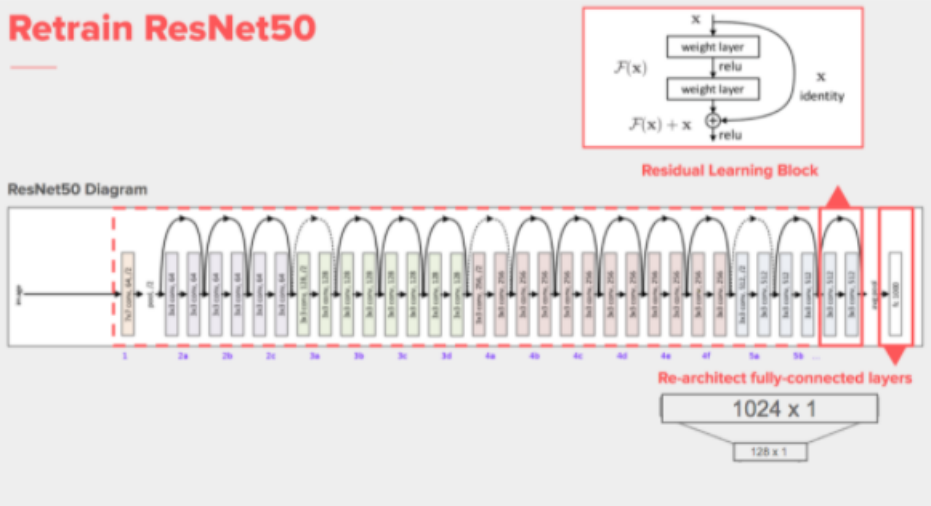


Figure 3.9: Modified ResNet50 Architecture [6]

learn something new and different from what the input has already encoded. It also minimizes the vanishing gradient problem by allowing the gradient flow from top layers to the bottom by means of identity connections without any changes.

The ResNet-50 is a deep CNN model with 50 layers that is pre-trained on ImageNet dataset. It consists of five different stages each with a convolution and an identity block. Each convolution block and the identity block contains 3 convolutional layers and followed by a softmax layer as its output. Here, only the five stages that are pre-trained on ImageNet weights are considered discarding the last softmax layer. In addition, two fully connected layers are added to tackle this problem. The first fully connected layer has 1024 units which is followed by batch normalization and ReLU and the second layer has 128 units, the final desired feature vector. The architecture of modified ResNet-50 used in this work is shown in Figure 3.9.

3.2.3 Triplet Mining

Generally, generating all possible triplets from the training set results in a huge number of triplets that easily satisfies Equation 3.3. These triplets would not contribute to the training and always result in slower convergence during the learning process [23]. For this reason, it is always crucial to select proper triplets during the learning process that may contribute to the improvement of the learning process.

There are three categories of triplets: easy triplets, hard triplets and semi-hard triplets [7].

- **Easy triplets:** triplets with loss 0, that is, $||f(a) - f(p)||^2 + margin <$

$||f(a) - f(n)||^2$

- **Hard triplets:** triplets in which negative image is more similar to the anchor image than the positive image, $||f(a) - f(n)||^2 < ||f(a) - f(p)||^2$
- **Semi-hard triplets:** triplets in which negative image is not similar to the anchor as positive image but still results in positive loss, $||f(a) - f(p)||^2 < ||f(a) - f(n)||^2 < ||f(a) - f(p)||^2 + margin$

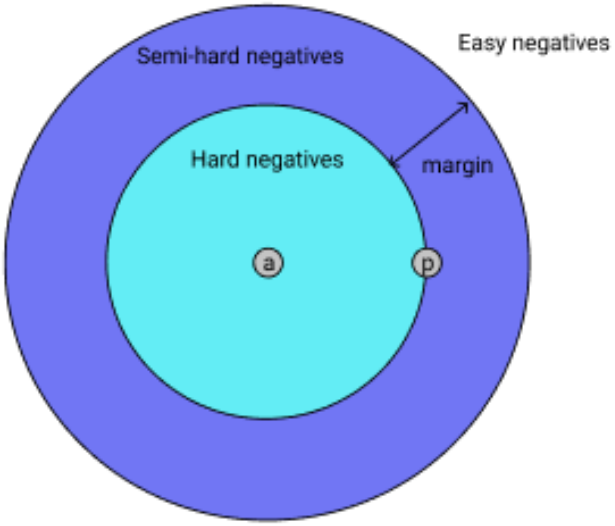


Figure 3.10: The three types of negatives in triplets [7]

Each of these three categories depend on the position of the negative that is relative to position and given anchor. It is obvious that, if the loss function is 0 for easy triplets, then the model does not learn much from them. In contrast, hard triplets generate high loss and produce a big impact on network parameters. If the dataset is mis-labelled then hard triplets add too much weight collapsing the embedding space. So, it is always necessary to choose some triplets within a batch that are a combination of both easy and hard, that is, semi-hard triplets. Semi-hard triplets lead to better generalization and fast convergence. In particular, these categories can be extended as: easy negatives, hard negatives or semi-hard negatives.

Figure 3.10 shows the three different categories for negatives in the embedding space. The selection of correct triplets for training is an important factor for attaining good performance and greatly impacts our metrics. Therefore, the approach of directly learning the mapping of face images to their embeddings using triplet loss and later

using these embeddings for either verification or identification tasks such as FaceNet, is the basis for all the modern and state-of-the-art methods [23]. "Models trained from scratch as well as pretrained ones, using a variant of the triplet loss to perform end-to-end deep metric learning outperforms most other published methods by a large margin" [27].

Offline Triplet Mining

It is a technique to generate triplets by mining them offline, that is, at the start of each epoch. The goal of this method is to randomly select images from the training set and for each image, that is, an anchor, randomly select a positive example and a negative example. The idea is to train the model for one iteration by randomly selecting the triplets offline. After that, computing embeddings for all the images in the training set and selecting hard triplets based on the returned ranked list of k-nearest images in the embedding space for each of the identity in the train set and re-train the model for the second iteration. The generation of triplets for the second iteration is explained in detail in the experimental section 4.2.3. Ideally, a list of triplets are generated. Then, batches of these triplets of size B are created and passed into three parallel models sharing the same weights. Finally, we compute $3B$ embeddings and the loss of these B triplets, then backpropagate the error into the network [7]. The workflow of the Offline mining technique is shown is Figure 3.11.

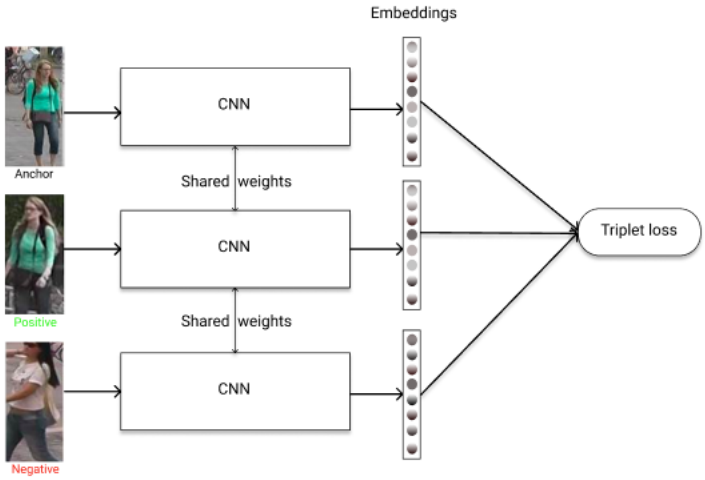


Figure 3.11: Workflow of Offline Triplet Neural Network

Online Triplet Mining

This technique was first introduced in the paper of F. Schroff et al. called FaceNet [23] in 2015. The goal of this approach is to create the triplets spontaneously

for every batch of inputs, instead of generating them offline. The process goes as follows: A batch of B images are selected and passed into the model and generate B embeddings.

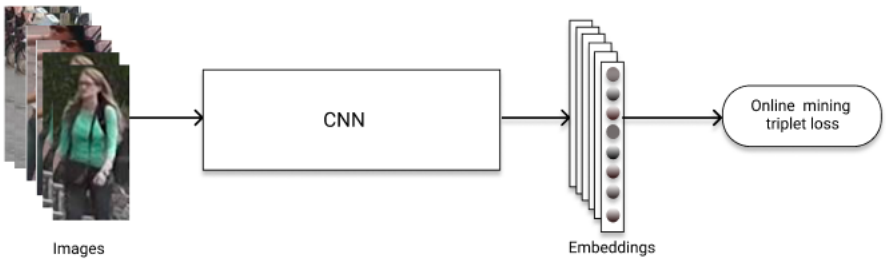


Figure 3.12: Workflow of Online Triplet Neural Network

A valid triplet is defined as follows: A triplet (a, p, n) with a and p having same label but are different while n has a different label than a. The core idea is to form batches by randomly sampling images from the training data. For each batch, P classes are randomly selected and then from each class K images are randomly selected resulting in a batch of PK images. A typical value for K is K=4.

There are two different strategies in online triplet mining: batch all and batch hard mining.

- **Batch all:** In this category, we select all valid triplets and average the loss on all semi-hard and hard triplets. This generates a total of $PK(K-1)(PK-1)$ triplets, for every batch of PK images, PK anchors, $K-1$ possible positives for each anchor and $PK-K$ possible negatives are possible. The key factor to be noted here is to reject easy triplets, remember that these are the triplets that result in loss = 0, since averaging on them would make the overall loss very small.

- **Batch hard:** In case of batch hard category, for each anchor the hardest positive example, biggest distance of $d(a, p)$ and the hardest negative example, smallest distance of $d(a, n)$ are selected within the batch. This technique generates a total of PK triplets. Also, the selected items are the hardest triplets within the batch.

3.2.4 Triplet Loss and Lossless Triplet Loss

In recent years, the field of Computer Vision has made a lot of research in deep metric learning and has seen progress using triplet loss. As stated in Section 3.2.1, the triplet loss is calculated using Equation 3.3. The main problem in Equation 3.3 is that, during the 1^{st} iteration of training, for an easy triplet, the loss reduces to 0 losing a lot of information. As long as the negative example is further away from the positive example by a margin, there will be no gain in the learning algorithm.

Here is an illustration to explain the following: Consider the distance between anchor and positive, $d(a, p)$ as 1.2, distance between anchor and negative $d(a, n)$ as 2.4 and margin m as 0.2. Substituting these values in Equation 3.3, we get:

$$L = max(1.2 - 2.4 + 0.2, 0) = max(-1, 0) = 0 \tag{3.4}$$

Similarly, for any value of $d(a, p)$ greater than 1, the loss is 0. In this scenario, pushing the anchor closer to a positive example and reducing $d(a, p)$ is hard for the learning algorithm. As a result, after certain epochs during training, the training loss converges to 0 clearly not improving its learning process [50].

In order to capture the lost information below 0, the loss function calculated in the N-dimension feature space has to be controlled efficiently. For this, the final feature vector in the top layer of the CNN model has to be normalised using L2-normalization. As a result, all the feature vectors lie on the surface of the unit hypersphere with which the maximum distance between any two vectors are known to be the range [0, 4]. The squared euclidean distance between normalised feature vectors are proportional to their cosine similarity as given by Equation 3.5.

$$\left|\left|\frac{A}{||A||} - \frac{B}{||B||}\right|\right|^2 = \left|\left|\frac{A}{||A||}\right|\right|^2 + \left|\left|\frac{B}{||B||}\right|\right|^2 - 2\frac{AB}{||A|| \cdot ||B||} = 2 - 2\frac{AB}{||A|| \cdot ||B||} \tag{3.5}$$

Another main advantage is that the value of squared euclidean distance between normalised vectors is guaranteed to be in the range [0, 4] which solves one of the difficulties in tuning the margin parameter in Equation 3.3 and cluster the data points since the embedding constantly changes during the training process.

Instead of triplet loss which calculates the linear cost, the lossless triplet loss proposes a non-linear cost function [50]. The non-linear cost function is defined by the Equation 3.6.

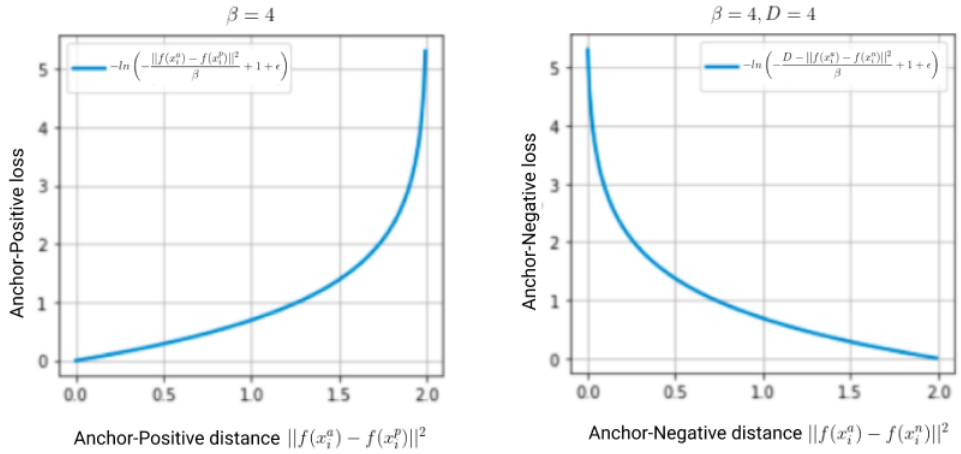


Figure 3.13: Non-linear Anchor-Positive and Anchor-Negative distance curve [8]

$$L(A, P, N) = \sum_{i=1}^{n} \left[-ln\left(-\frac{||f(x_i^a) - f(x_i^p)||^2}{\beta} + 1 + \epsilon \right) - ln\left(-\frac{D - ||f(x_i^a) - f(x_i^n)||^2}{\beta} + 1 + \epsilon \right) \right] \quad (3.6)$$

where *first term* is an associative term focusing on pulling the anchor closer to the positive point and the *second term* is the contrastive term focusing on pushing the anchor farther away from the negative point in the $n-$dimensional feature space. ϵ is added to prevent the $ln(0)$ and β is a scaling factor. The maximum distance between any two points in the feature space is 4 as squared euclidean distance is calculated between them. Because of the non-linear cost, the lossless triplet loss does not produce a 0 loss even for easy triplets. Also, the non-linear cost saves much effort from choosing hard triplets during the learning process and faster convergence in the learning process. The paper of L. Utkin et al. [51] uses an ensemble of triplet neural networks trained with lossless triplet loss and has seen success in improving the accuracy of diagnostics of lung cancer.

3.3 Implementation

This section presents the details on the pipeline developed throughout the training and testing process in the project work.

3.3.1 Training

Due to the small size of the dataset, the weights pre-trained on ImageNet dataset are used throughout the experiments. Batch size is set to 32 as it is a good starting point.

The batch size could not be increased to 64 due to memory constraints while training with GPU. The input size of the image is $256 \times 128 \times 3$. As stated in Section 3.1.4, the generator yields batches of triplets after applying data augmentation techniques and the resulting volume is fed into the model for learning process. Since batches of images are used during the learning process, the final 4-dimensional input volume is of size $32 \times 256 \times 128 \times 3$. Throughout the experiments, Adam optimizer [52] is used as an optimization algorithm. Initially, the learning rate is set to 1e-4 and the learning rate is decayed by 0.5 after every 50 epochs.

The first 10 layers of the architecture are frozen and the remaining layers are open for training to avoid overfitting problem. Keras library running on top of TensorFlow is used as the deep learning framework. Experiments are performed using different base architectures, variants of loss function and mining techniques. The loss is monitored and discussed their results in Section 4.2. The testing set is used for validation during the learning process and monitored their performance after every epoch.

3.3.2 Nearest Neighbour Search

The trained model is then utilized to get a ranked list of matching identity images from the entire gallery set by searching through the feature or the embedding space. The person matching is based on squared euclidean distance (L2-distance) of the feature vector extracted from the final layer of the network.

For every query image in the test set, the ranked list of images is obtained by calculating the squared euclidean distance of the query image against all the images in the gallery set and choosing the k-nearest neighbour neighbours to the query image in the embedding space. The runtime complexity to perform this task is $O(n^2)$. In order to reduce the runtime complexity, approximate nearest neighbours are chosen rather than the exact nearest neighbours. The runtime complexity is now reduced to $O(nk)$. By definition, for a query image q, approximate nearest neighbour finds a point $p' \in P$ whose distance is at most c times the distance from the query image to its actual nearest point $p \in P$. In most of the cases, approximate nearest neighbour is almost the same as the k-nearest neighbours.

An open-source library by Spotify called ANNOY is used to calculate the nearest neighbours for this task [53]. This algorithm is based on random generation of projections and trees. To compute the nearest neighbours, the algorithm splits the data points into half and repeats the process recursively until each set consists of k items and the value of k is set explicitly. To further improve the results, a priority queue from the root node is used to search the tree. The priority queue is sorted by

the smallest margin for the path from the root node of the tree. For every set of k items, multiple trees are built. In each set of k items, duplicate points are eliminated and for the remaining points, the distances are computed from the gallery set.

3.3.3 Development Environment

The operating system used for the thesis is Linux and Python is the programming language used throughout the thesis. As discussed earlier, Keras is the deep learning framework used for the thesis work. Keras is a high-level neural network API written in Python and capable of running on top of TensorFlow [54]. It is flexible and allows the implementation of neural networks in a few number of lines with high levels of abstraction.

Usually, neural network training requires a vast amount of computation. Deep learning training can be done using accelerators such as GPUs or TPUs which is used in the field of High Performance Computing (HPC) for more than a decade. The training is carried out on NVIDIA GeForce RTX 2080 Ti (11 GB) GPU provided by our company Tagsonomy SL in Madrid.

The python programming environment used is JupyterLab, the next generation framework for interactive computing and machine learning. It provides an improved interface to Jupyter Notebooks. JupyterLab consists of file browser, console, terminal, text editors, markdown editors, and so on. Notebooks are mainly used as it combines code, graphics, visualizations and text that run on a browser.

Chapter 4

Experiments and Results

This chapter explains the evalutaion metrics used in the thesis work to evaluate the performance of the system. The experiments performed on different base architectures, variants of loss function and various mining techniques and their results are explained in detail.

4.1 Evaluation Metrics

In literature, Rank-k accuracy and mean average precision (mAP) are the most widely used metrics to evaluate and investigate the performance of re-identification algorithms. The ranks at different k leads to another metric called Cumulative Matching Characteristics curve. CMC is used as a measure of $1 : m$ performance of identification systems. The system returns the list of k-identities from the gallery set based on their distances to the query image sorted from smaller to large values. It evaluates the ranking capabilities of any identification system. In other words, it shows the probability that a query identity appears in the returned rank list of results. This evaluation metric is usually valid for single-shot image datasets $(1 : 1)$.

For a given query image, Rank-k accuracy is calculated as follows:

$$Rank@k = \begin{cases} 1 & if\ top-k\ ranked\ list\ has\ at\ least\ one\ match \\ & same\ as\ query\ identity \\ \\ 0 & otherwise \end{cases} \tag{4.1}$$

where k is the number of nearest neighbours in the ranked list of images.

The final rank-k accuracy is calculated by averaging the accuracy over all the query identities. For example, Rank-1 accuracy is the percentage of times the top match

Figure 4.1: Examples of ranked list of images for a given query image

has the same identity as query image; Rank-5 accuracy is the percentage of times that at least one of the top five matches has the same identity as query image and so on.

However, for multi-shot image datasets, Rank-k metrics is biased as it fails to provide a reasonable comparison of the quality of different ranked lists. Also, precision and recall are not considered. For all query examples in Figure 4.1, the Rank-1 and Rank-5 accuracy are set to 1 failing to provide a reasonable comparison of the quality of the two ranked lists. Therefore, it is good to calculate the average precision of ranked lists as it is a good metric which provides a fair comparison of different ranked lists.

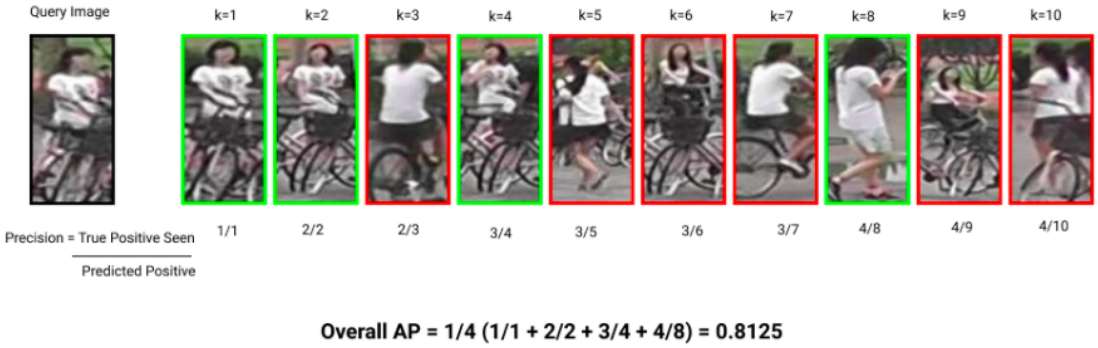


Figure 4.2: Calculation of AP for a query image with 4 true positives

The average precision for the ranked list of a query image is calculated as follows:

$$AP@k = \frac{1}{Total\ \ TP} \sum_{i=1}^{k} \frac{TP\ seen}{i} \tag{4.2}$$

where $Total\ TP$ refers to the total number of true positives for a query image and $TP\ seen$ refers to the number of true positives that are seen till k. The true positive is an outcome where the model correctly predicts the identity as the query's identity. The calculation of average precision for a query image is shown in Figure 4.2. Then, the average precision (AP) values of all the query images are averaged to obtain the mAP which considers both the precision and recall, thus yielding a more comprehensive evaluation metric. The formula for mAP is given by Equation 4.3.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.3}$$

where N is the number of query images.

4.2 Experimental Results and Discussion

This section describes the various experiments performed and discusses their results. The experiments are performed on different base CNN architectures, variants of triplet loss function and techniques of triplet mining. Each experiment section addresses the model performance on the technique handled and concludes the model with best results during the thesis work.

4.2.1 MILDNet versus ResNet-50

This experiment discusses the model behaviour by analysing and evaluating the two different state-of-the-art convolutional neural networks as base networks in the triplet architecture. As the aim of the thesis is to encode a robust representation, that is, an efficient feature vector by extracting important patterns from the data, it is crucial to select an appropriate convolutional neural network as base architecture in the triplet system. The two convolutional neural networks considered are 1) MILDNet: A Lightweight Single Scaled Deep Ranking Architecture [5], a novel architecture that is greatly compact and has seen remarkable progress for visual product matching and recommendations in E-commerce platforms and 2) ResNet-50: Residual Networks [49], the fundamental breakthrough architecture and winner of ImageNet challenge in 2015. The design of both the network architectures are discussed in Section 3.2.2.

Figure 3.7 shows the triplet architecture where three parallel convolutional networks are assembled sharing the same weights. The triplets are selected offline at the beginning of each epoch. The model is trained using each of the two CNN models as base networks. Since the initial layers in any convolutional neural networks are known to capture generic features that need not be trained, the first 10 layers are freezed and the rest of the layers are open during training with each CNN

architecture. For the first 10 layers, the weights pre-trained on ImageNet dataset are used. The input size of the image is 256 x 128 x 3. The triplet loss function given by equation 3.3 is used in both the experiments. The batch size is set to 32 and the model is trained with Adam optimizer and initial learning rate is set to 1e-4. The learning rate is decayed by 0.5 after every 50 epochs. The model is trained for 100 epochs for both CNN experiments; the training of each of the models took approximately 8 hours.

After training, the performance of both the models are measured by calculating the Rank-1, Rank-5 and mean average precision (mAP) metrics on the test set. During the inference phase, only one base CNN model is used to produce the embeddings since the three parallel models in the triplet architecture share the same weights. There is no need to generate triplets during the testing phase as the model performance is just measured by returning the ranked k-nearest neighbours of a query image based on their euclidean distance to all the gallery images in the embedding space.

Metrics

Table 4.1 shows the evaluation metrics on the test set for each of the two base networks.

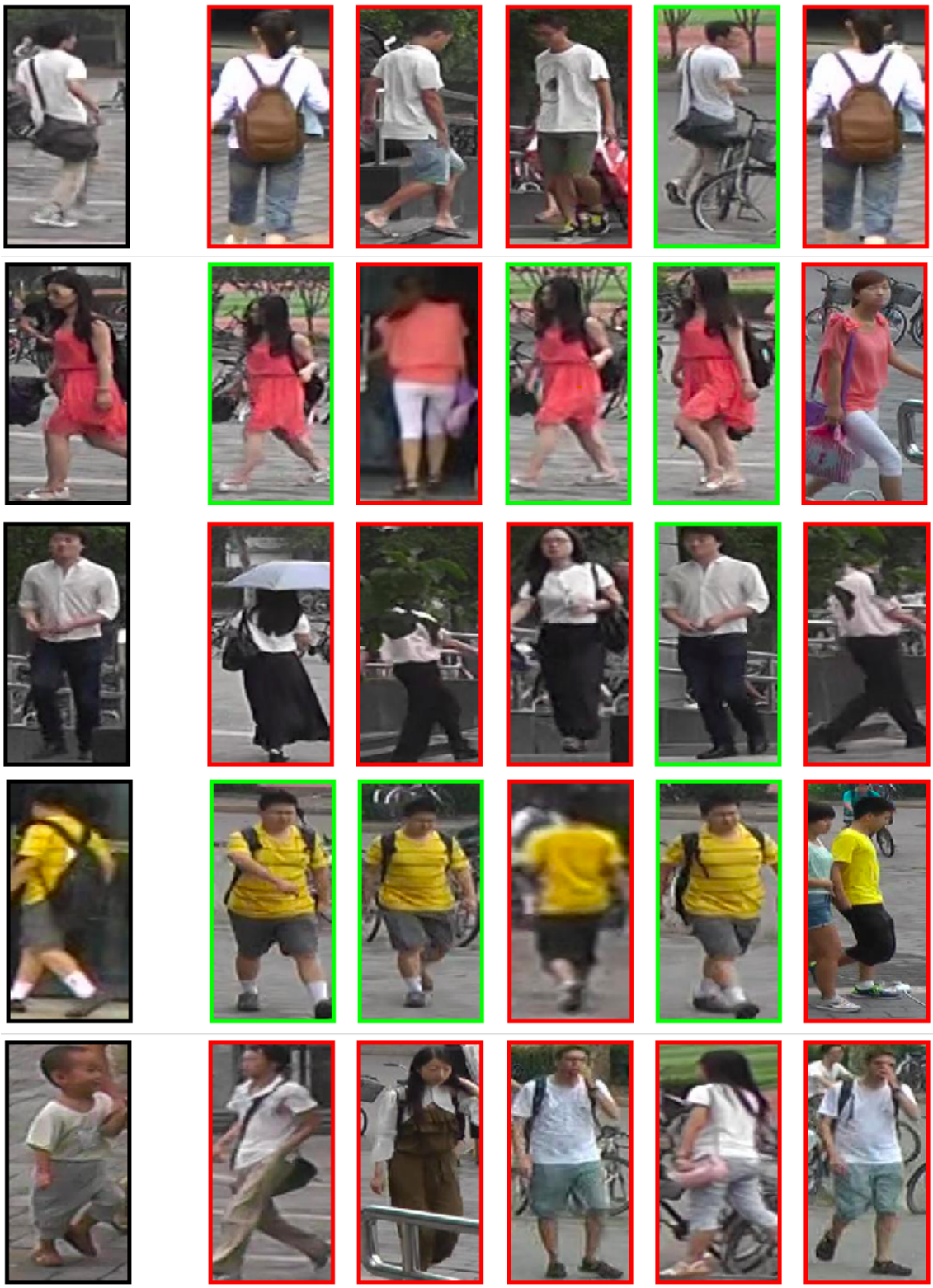
Base architecture / Metrics	**Rank-1**	**Rank-5**	**mAP**
MILDNet	79.5	85.4	70.3
ResNet-50	86.6	91.2	75.9

Table 4.1: Evaluation metrics for systems trained with two different base architectures

Figure 4.4 shows the visual results of ranked k-nearest neighbours for the model trained with MILDNet as its base network. Figure 4.6 shows the visual results of ranked k-nearest neighbours for the model trained with ResNet-50 as its base network.

Discussion

The analysis of the experiments on base architecture indicate that the model trained with ResNet-50 architecture performs better than the model trained with MILDNet as base network. It is clearly evident from the evaluation metrics in table 4.1 and visual results in figure 4.4 and figure 4.6. In deep convolutional neural networks, different levels of abstraction are captured at different levels of layers. The output of convolutional layers are interpreted as local features representing particular regions

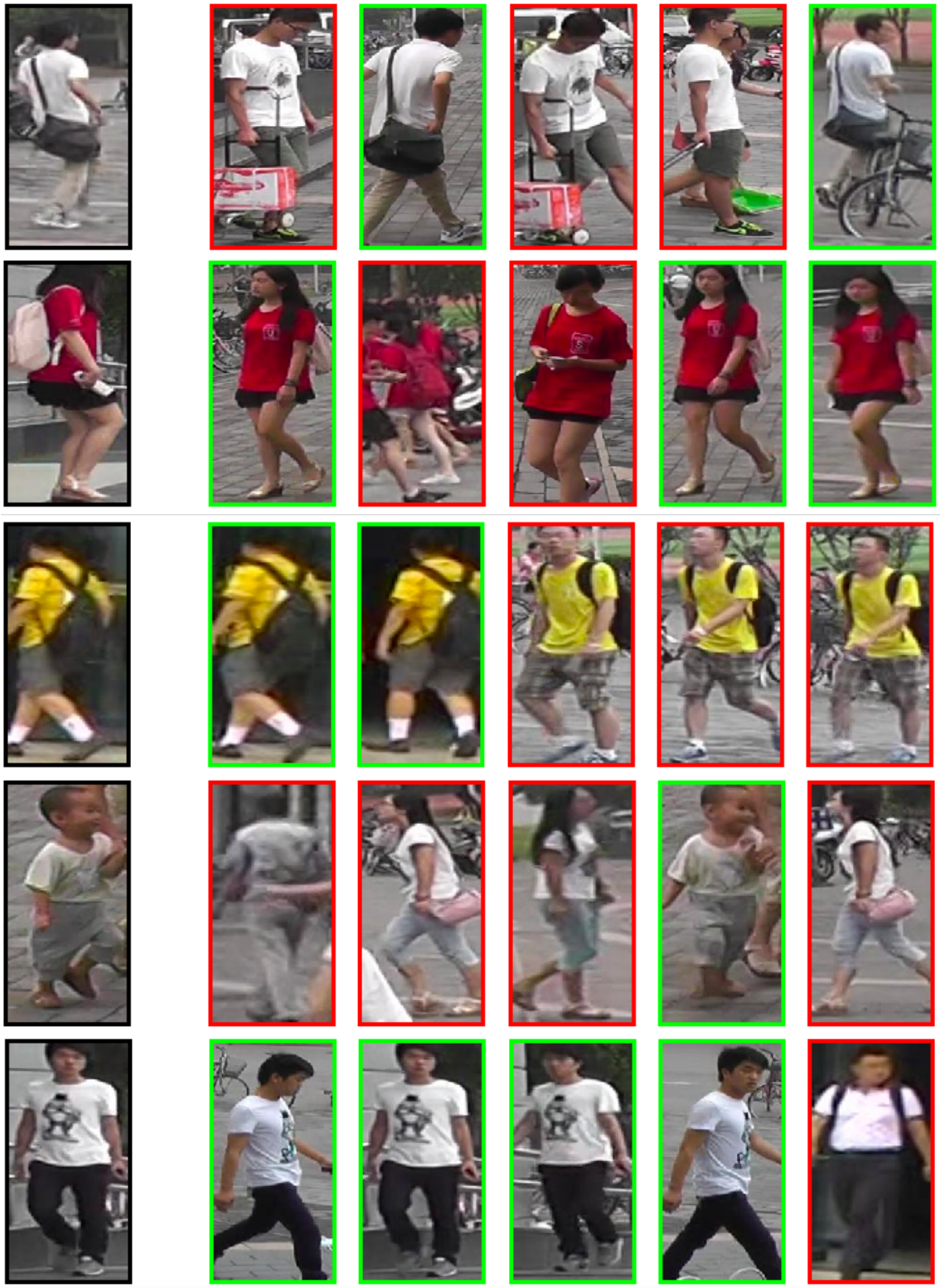


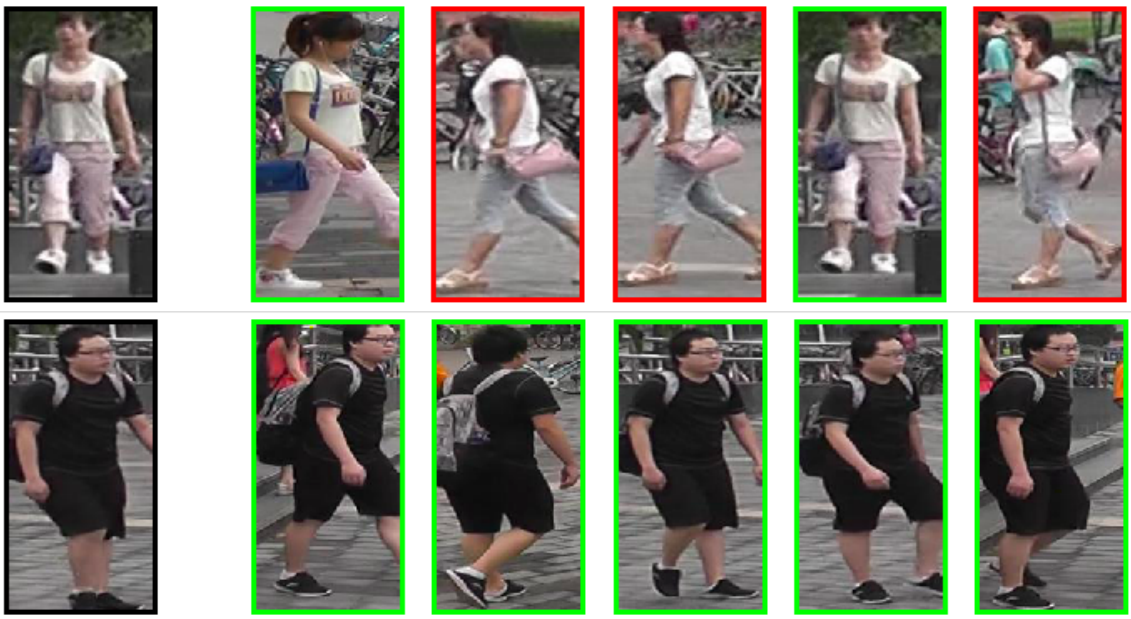
Images with black border represent query images, images with green border are the correct matches to the query image and images with red border are the incorrect matches to that of the query image.

Figure 4.4: Examples of model trained with MILDNet architecture

in an image. Since multiple features are present at multiple layers, it is not sufficient to consider only the features from its preceding layers to learn features in current layers and solve such complex problems. In order to overcome this problem, the idea is to add skip connections between layers that help traverse information from low level layers to high level layers in deep neural networks. So, learning complex features in the later layers becomes more accurate and efficient.

In MILDNet architecture, skip connections are added just after max-pooling layer in each convolutional block to extract additional features apart from the final output. There are 5 skip connections in MILDNet architecture still failing to learn and ex-





Images with black border represent query images, images with green border are the correct matches to the query image and images with red border are the incorrect matches to that of the query image.

Figure 4.6: Examples of model trained with ResNet-50 architecture

tract complex local features which is clearly seen in Figure 4.4. Clearly, the model is sensitive to the color and appearance of the person. It is able to distinguish between colors but the model is getting confused when people have similar appearance, especially when people are wearing the same color dress. Due to this, the top match of many query images have a false positive reducing the performance of the model.

In case of ResNet-50 architecture, skip connections are added after every two convolutional layers pushing the current layer to learn something new and different and aggregating the features until the later layers in the deep network. There are a total of 16 skip connections in the network benefitting to learn more complex local features and increasing the performance. The ranked k-nearest neighbours in Figure 4.6 shows that most of the top match are identical to that of the query image. It is obvious that the network is able to return the ranked list matching to the identity but is still sensitive to factors like similar appearance and occlusions. The mAP has also been improved by 5% compared to the previous model.

4.2.2 Triplet Loss versus Lossless Triplet Loss

This section discusses the model performance by investigating and estimating the two variants of triplet loss functions. In general, loss function in deep neural networks evaluates the performance of the model by measuring the error between model's output and a desired output given specific data. Unlike these loss functions, the loss function used in training a triplet model has to directly learn the mapping of a low dimensional representation from images to a compact embedding space where euclidean distances directly correspond to the measure of similarity and dissimilarity of images. Therefore, it is necessary to select an appropriate loss function to train the triplet architecture. The two variants of triplet loss considered are 1) Triplet loss, the loss function first introduced in FaceNet: A Unified Embedding for Face Recognition [23] and achieved the state-of-the-art results in face recognition and 2) Lossless triplet loss, a non-linear loss function introduced by M. O. Arsenault [50]. A detailed description of both these loss functions are discussed in Section 3.2.1 and 3.2.4.

From the previous experiment, it is clear that RestNet-50 as the base network performed better than the MILDNet in the triplet architecture. Hence, ResNet-50 is considered to be the base network in all the remaining experiments. In this experiment, a model is trained with ResNet-50 and lossless triplet loss and its performance is compared with the model trained with ResNet-50 and triplet loss in the previous experiment. Similar to the previous experiment, the triplets are selected offline at the beginning of each epoch; the first 10 layers are freezed and the remaining layers are open for training; all other experiment set up remains the same. The only difference in the experiment is the non-linear cost function called lossless triplet loss.

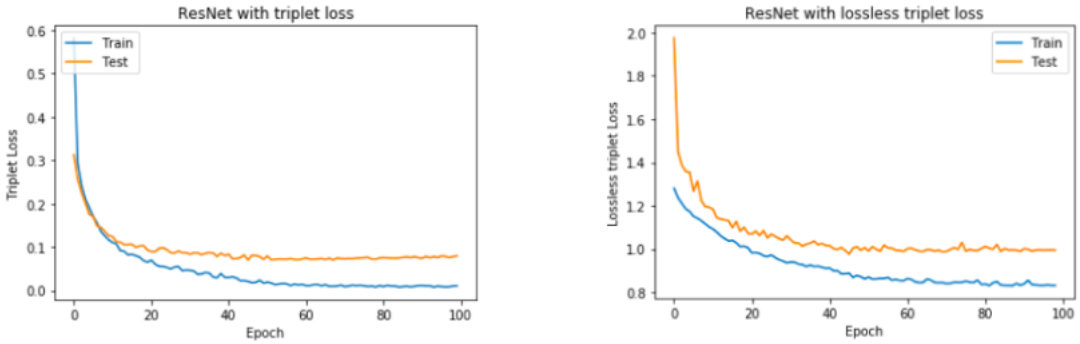
As stated in Section 3.2.4, lossless triplet loss function has the capability to capture the lost information below 0 by efficiently controlling the compact euclidean feature space. Because of the non-linear cost, the loss function does not produce a 0 loss even for easy triplets and yields faster convergence in the learning process.

Once the training is completed, for every query image in the test set, the system returns the ranked list of nearest neighbours from the gallery set and the Rank-1, Rank-5 and mean average precision metrics are calculated.

Metrics

Table 4.2 shows the evaluation metrics on the test set for each of the variants of triplet loss function.

The metrics in Table 4.2 shows that the Rank-1 accuracy has increased by 6% and Rank-5 accuracy is increased by 5%. There is a significant increase in the mean



(a) Model with triplet loss function. (b) Model with lossless triplet loss.

Figure 4.7: Learning curve for the models trained with triplet loss and lossless triplet loss

Loss function / Metrics	**Rank-1**	**Rank-5**	**mAP**
Triplet loss	86.6	91.2	75.9
Lossless triplet loss	92.1	96.5	81.9

Table 4.2: Evaluation metrics for model with two different loss functions

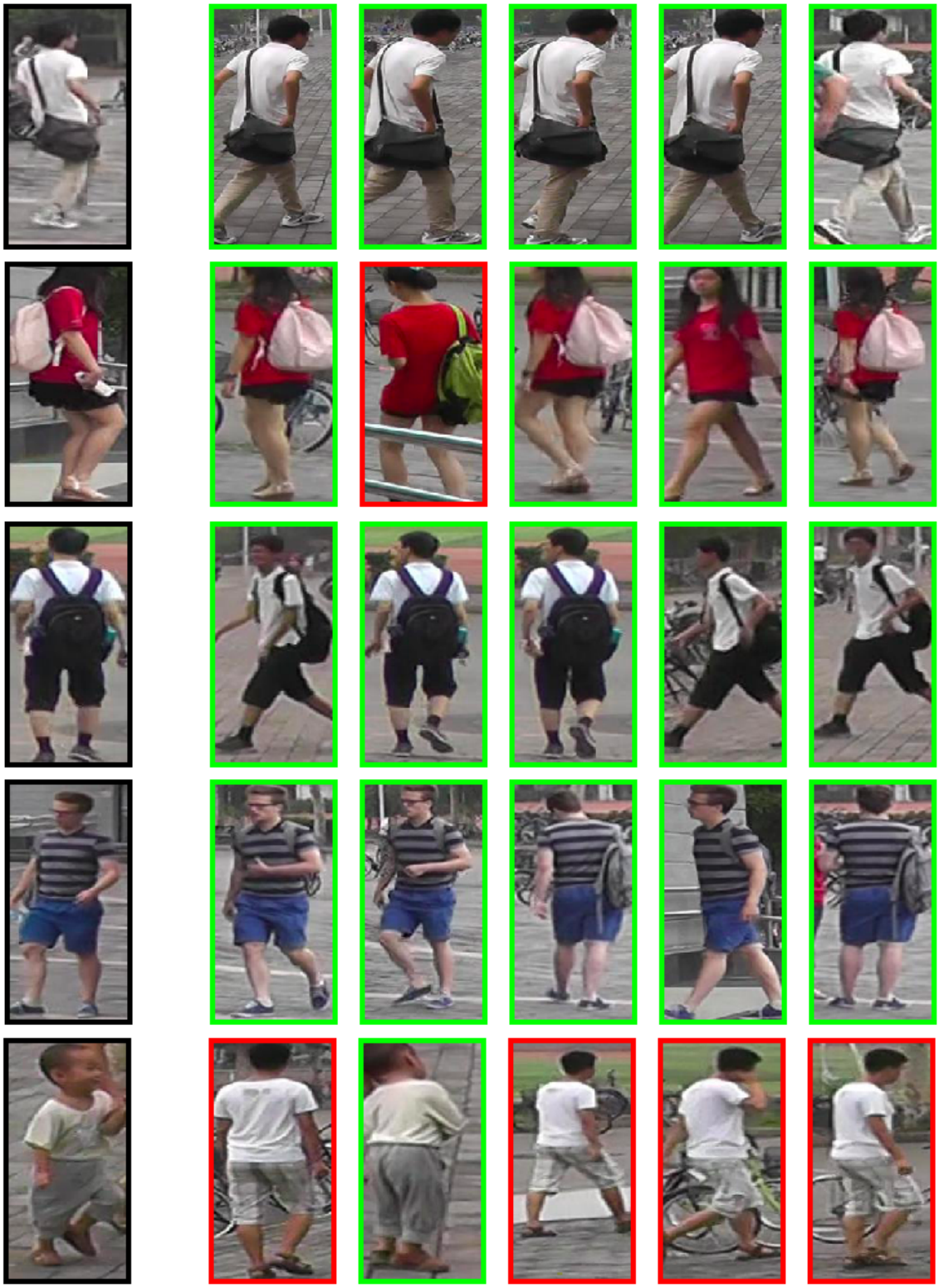
average precision as well.

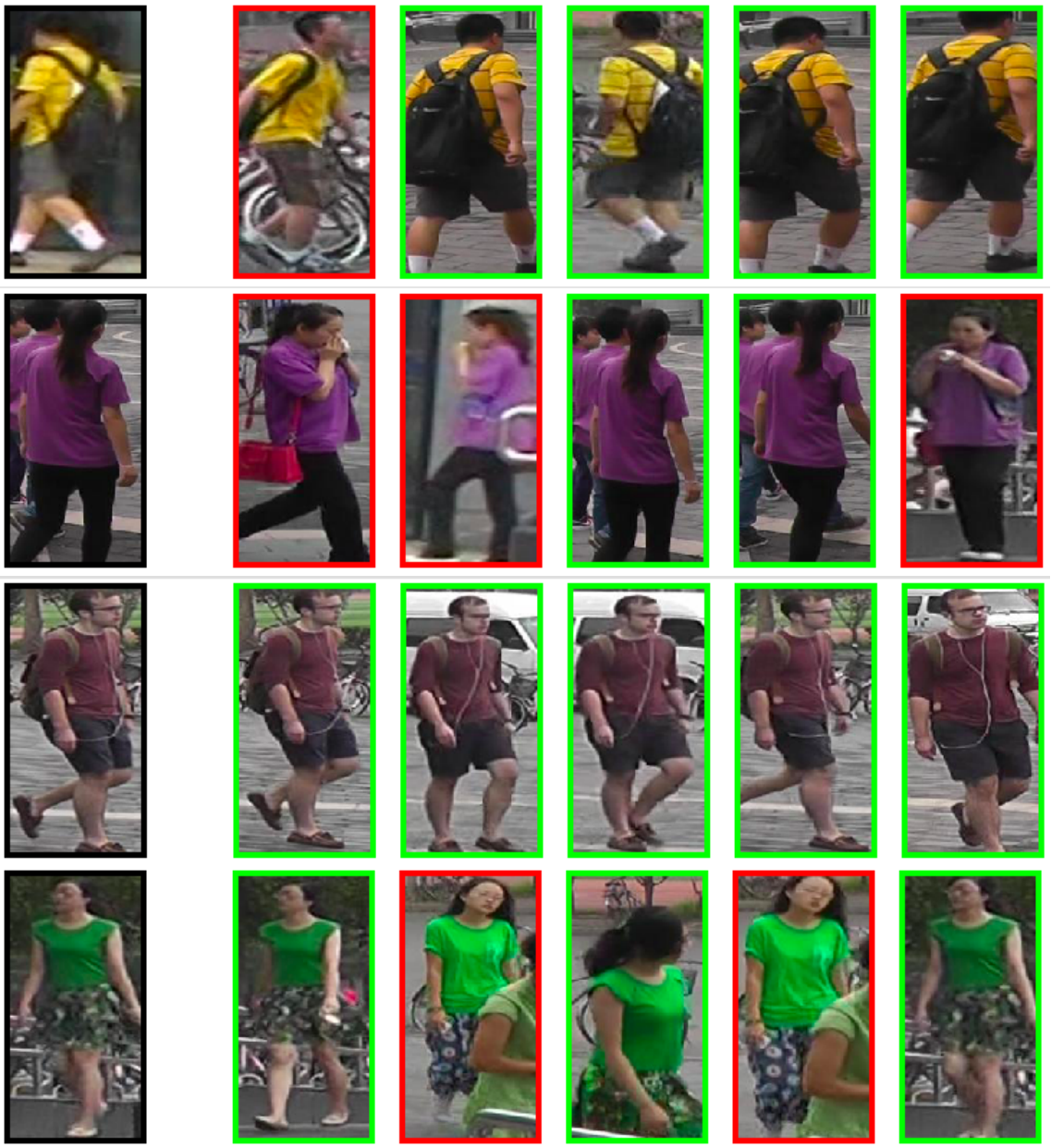
Figure 4.9 shows the visual results of ranked k-nearest neighbours for some of the query images for the model trained with lossless triplet loss.

Discussion

The experiment indicates that the model trained with lossless triplet loss performs better than the model trained with triplet loss. It is also evident from the evaluation metrics in Table 4.2 and visual results in Figure 4.9.

Figure 4.7 shows the learning curves of the models trained with triplet loss and lossless triplet loss. Figure 4.7(a) shows the learning curve of the model trained with triplet loss. It is clear from the graph that, after 50 epochs the loss converges to 0 during training. This means that the model should perform very well. But in fact, it is not the case. According to the triplet loss in Equation 3.3 , for any easy triplet the model reduces the loss to zero losing a ton of information. As long as the negative example is farther away from the positive example by a margin, there is no gain in the learning algorithm. As a result, the training loss converges to 0 clearly not improving its learning process.





Images with black border represent query images, images with green border are the correct matches to the query image and images with red border are the incorrect matches to that of the query image.

Figure 4.9: Examples of model trained with ResNet-50 and lossless triplet loss architecture

Introducing the non-linear cost function has eliminated this problem. Figure 4.7(b) shows that the loss does not reduce to 0 even after 100 epochs. Thus, the ability

of the lossless triplet loss to capture the lost information below 0 has improved the model's learning ability to learn complex local features and encode a robust feature vector by extracting patterns from the data. This is clearly evident from the visual results shown in Figure 4.9. The ranked k-nearest results shows that almost all the top matches are identical to that of the query image and has significantly increased the performance of its Rank-1 accuracy. It is clearly able to distinguish between the identities even with similar appearance and occlusions. For most of the examples, the ranked list of k-nearest neighbours have correct matches to that of the query identity. These results have almost increased the Rank-5 accuracy by 5% compared to the model trained with triplet loss. The mAP has almost increased by 5% improving the performance of the system.

4.2.3 Offline Triplet Mining versus Online Triplet Mining

For successful training of the triplet neural networks, it is important to select relevant and appropriate triplets. Choosing appropriate triplets also achieves faster convergence in the learning process. As discussed in Section 3.2.3, there are two different types of mining triplets during the training process. They are 1) Offline triplet mining - process where the triplets are selected at the beginning of each epoch. 2) Online triplet mining - process where the triplets are selected on the fly after passing a batch of images into a single model.

Figure 3.7 shows the workflow of the model for offline triplet mining. During the first iteration of the training, a batch of 32 images (anchor) are randomly selected from the training set. For every anchor image, a positive image which has the same identity as that of anchor image and a different camera id and a negative image which has different identity as that of anchor image are randomly selected from the training set [55]. It results in a batch of 32 triplets (anchor, positive and negative) which are then passed into three parallel neural networks sharing the same weights. The model outputs 3 x 32 embeddings and each embedding is of length 128-dimension. Then, the loss and the gradients are computed and then back propagate the gradients into all the three networks. Random sampling of triplets results in at least 15% of hard triplets within a batch. The process is repeated for all the remaining batches in the epoch. The model is trained for 100 epochs in the first iteration.

After the first iteration, for every image in the training set, 10 nearest neighbours are returned based on their euclidean distance in the embedding space. Among the 10 nearest neighbours, the neighbours with different identity to that of the input image and their distances are noted. Based on these results, 30% of hard triplets are selected for the second iteration of training. In the second iteration, 70% of triplets in a batch are randomly selected similar to the first iteration. For the remaining

30% of the triplets, hard triplets are selected based on the model results in the first iteration. For every identity in the training set, the incorrect matches in their nearest neighbours list are noted. The negative images for hard triplets are picked from the incorrect matches. Hard triplets generate high loss and produce a big impact on the network parameters. If too many hard triplets are included in the batch, the model adds too much weight collapsing the embedding space. Therefore, it is important to choose appropriate amount of hard triplets within a batch for successful performance of the model.

Figure 3.12 shows the workflow of the model for online triplet mining. In this experiment, the triplets are selected on the fly for each batch of inputs. A batch of B images are selected randomly from the training set; here B = 64. Within each batch, 8 identities are randomly selected and for each identity 8 images are randomly selected; resulting in a batch of 64 images. The model consists only of a single network and the batch of images are passed into the model to obtain 64 embeddings. All valid triplets within the batch are selected as described in Section 3.2.3. And the loss is averaged over all the triplets within a batch.

ResNet-50 is used as the base architecture with input image size as 256 x 128 x 3. The first 10 layers are freezed and the remaining layers are open for training similar to the previous experiments. The remaining other experiment set up remains the same.

Once the training is completed, for every query image in the test set, the system returns the ranked list of nearest neighbours from the gallery set and the Rank-1, Rank-5 and mean average precision metrics are calculated.

Metrics

Table 4.3 shows the evaluation metrics on the test set for each of triplet mining techniques.

Mining Technique / Metrics	**Rank-1**	**Rank-5**	**mAP**
Online mining	91.3	95.8	79.7
Offline mining	92.1	96.5	81.9

Table 4.3: Evaluation metrics for model with two different mining approaches

Discussion

The metrics in Table 4.3 shows that there is not much significant difference with the two approaches. The Rank-1 accuracy for online mining approach is only 0.8% less

than the offline mining; Rank-5 accuracy is reduced by 0.7% and mAP is reduced by 1.2%. Since the overall experimental setup for both the experiments are the same, the difference mainly could come from random sampling of triplets in offline mining. Since the same triplets are not chosen in both the models during training, there is a slight gap in the metrics for this approach. For the dataset used in the thesis work, clearly offline mining approach works better than online mining technique.

From all of the above experiments, the results indicate that the model trained with ResNet-50 as base network and lossless triplet loss with offline triplet mining gives the best results and is considered as the final successful model for this thesis work.

4.2.4 State-of-the-art Comparison

Method	**Year**	**Rank-1(%)**	**mAP(%)**
SGGNN [56]	2018	92.3	82.2
Aligned Re-Id [57]	2017	91.8	79.3
OSNet [58]	2019	94	83.4
AANet [10]	2019	93.7	82.9
Thesis work	-	92.1	81.9

Table 4.4: State-of-the-art comparison to the thesis work

Chapter 5

Conclusion and Overlook

5.1 Conculsion

The aim of this project work are to (*i*) implement a triplet convolutional neural network using deep metric learning in order to learn an embedding space and extract robust representation of similarities in the data patterns, (*ii*) Evaluate the system performance by experimenting different base architectures in the triplet model, (*iii*) Investigate the performance of the model using variants of loss functions, (*iv*) Experiment the model performance using the two triplet mining approaches.

The triplet convolutional neural network was implemented using Keras on top of TensorFlow. The network was evaluated using Rank-1, Rank-5 and mAP metrics using Market-1501 public dataset. The results were compared to the state-of-the-art algorithms on this dataset.

It was demonstrated that model trained with ResNet-50 as base network performed better than the model trained with MILDNet as base network in triplet model. Especially, the results were improved because of more number of skip connections in ResNet-50 compared to that of MILDNet as skip connections provide the ability to learn more new and complex features.

The model trained with lossless triplet loss clearly outperformed the traditional triplet loss in encoding a robust representation from the data patterns. The problem of reducing the loss to 0 and losing a ton of information has been eliminated by using lossless triplet loss, a non-linear cost function and increased the ability of the model to capture the lost information below 0. Also, it is clearly seen that there is no much difference in training the system with two different types of triplet mining.

In conclusion, the experiments showed that the system trained with ResNet-50 as base architecture and lossless triplet loss with offline triplet mining achieved better performance; Rank-1 accuracy of 92.1% and mAP of 81.9% was achieved. The results were satisfying and showed the capability of triplet system to tackle the extraction of robust representation from the data in a feasible and reliable way.

5.2 Scientific Contribution

A new non-linear variant of loss function, that is, lossless triplet loss was introduced in the thesis work inspired by the work of M. O. Arsenault in his blog post [50]. The detailed description of the loss function and the experimental results are explained in Section 3.2.4 and 4.2.2 respectively. This loss function has not been implementated for person re-identification problem in the literature earlier. But, it has been used for different problem in the literature. For example, the paper of L. Utkin et al. [51] uses an ensemble of triplet neural networks together with lossless triplet loss and has seen success in improving the accuracy of diagnostics of lung cancer. The speciality of this is that it introduces non-linearity and captures the lost information below loss 0 and results in faster convergence and better performance.

Even though research and experiments in the thesis work are done to solve Person Re-Identification problem, the company can further extend this approach to solve problems in other domains like Visual Search and Recommendations of fashion and other products in E-commerce platforms and achieve state-of-the-art results.

5.3 Future works

- In the present study, the Market-1501 dataset was used which has only around 32000 images. The use of a larger dataset with more number of identity examples could improve the results and generalization of predictions.
- A manual annotation of body crops from the video frames would decrease uncertainties associated with the ground truth labelling and eliminate occlusions and mis-alignment in the annotated body crops.
- Also, the research can be further extended using a video re-id dataset to handle this problem. The rich temporal information contained in the video sequences can be the next direction to explore this task and improve the results.
- Applying Semantic Segmentation algorithms on body crops for background removal can further improve the results.
- This approach can be further extended to solve problems in other domains like Visual Search and Recommendations of fashion and other products in E-commerce portals.

Bibliography

[1] Y. Chen, "Person re-identification in images with deep learning," 2018.

[2] K. Willems, "Deep learning in python." https://www.datacamp.com/community/tutorials/deep-learning-python, 2019.

[3] W. Ng, B. Minasny, M. Montazerolghaem, J. Padarian, R. Ferguson, S. Bailey, and A. Mcbratney, "Convolutional neural network for simultaneous prediction of several soil properties using visible/near-infrared, mid-infrared, and their combined spectra," *Geoderma*, 07 2019.

[4] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Computer Vision, IEEE International Conference on*, 2015.

[5] A. Vishvakarma, "Mildnet: A lightweight single scaled deep ranking architecture," *CoRR*, vol. abs/1903.00905, 2019.

[6] "Resnet-50 architecture for deep learning." https://roamerworld.blogspot.com/2019/05/resnet50-architecture-for-deep-learning_24.html, 2019.

[7] O. Moindrot, "Triplet mining." https://omoindrot.github.io/triplet-loss, 2018.

[8] E. Y. Cheu, "Learning embedding space with triplets." https://www.linkedin.com/pulse/learning-embedding-space-triplets-eng-yeow-cheu-ph-d-/, 2018.

[9] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," *CoRR*, vol. abs/1905.00953, 2019.

[10] C.-P. Tay, S. Roy, and K.-H. Yap, "Aanet: Attribute attention network for person re-identifications," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1st ed., 1995.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[14] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection.," in *ECCV (4)* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9908 of *Lecture Notes in Computer Science*, pp. 354–370, Springer, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[16] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR (workshop track)*, 2015.

[17] S. Ruder, "An overview of gradient descent optimization algorithms.," 2016.

[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010* (Y. Lechevallier and G. Saporta, eds.), (Heidelberg), pp. 177–186, Physica-Verlag HD, 2010.

[19] Y. N. Dauphin, H. d. Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, (Cambridge, MA, USA), p. 1504–1512, MIT Press, 2015.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[21] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," 2018. cite arxiv:1805.11604Comment: To appear in NIPS'18.

[22] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3630–3638, Curran Associates, Inc., 2016.

[23] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[24] V. Kumar B G, G. Carneiro, and I. Reid, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[25] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network.," in *NIPS* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), pp. 737–744, Morgan Kaufmann, 1993.

[26] B. . Juang and S. Katagiri, "Discriminative learning for minimum error classification (pattern recognition)," *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043–3054, 1992.

[27] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017.

[28] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *CoRR*, vol. abs/1711.08184, 2017.

[29] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping.," in *CVPR (2)*, pp. 1735–1742, IEEE Computer Society, 2006.

[30] W. Li, X. Zhu, and S. Gong, "Person re-identification by deep joint learning of multi-loss classification," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, p. 2194–2200, AAAI Press, 2017.

[31] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[32] S. Khamis, C.-H. Kuo, V. K. Singh, V. D. Shet, and L. S. Davis, "Joint learning for attribute-consistent person re-identification," in *Computer Vision - ECCV 2014 Workshops* (L. Agapito, M. M. Bronstein, and C. Rother, eds.), (Cham), pp. 134–146, Springer International Publishing, 2015.

[33] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Learning to rank in person re-identification with metric ensembles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1846–1855, 2015.

[34] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li, "Embedding deep metric for person re-identification: A study against large variations," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9905 of *Lecture Notes in Computer Science*, pp. 732–748, Springer, 2016.

[35] Y. Yuan, W. Chen, Y. Yang, and Z. Wang, "In defense of the triplet loss again: Learning robust person re-identification with fast approximated triplet loss and label distillation," 2019.

[36] M. Chen, Y. Ge, X. Feng, C. Xu, and D. Yang, "Person re-identification by pose invariant deep metric learning with improved triplet loss," *IEEE Access*, vol. 6, pp. 68089–68095, 2018.

[37] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *Computer Vision – ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), (Berlin, Heidelberg), pp. 262–275, Springer Berlin Heidelberg, 2008.

[38] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof, "Person Re-Identification by Descriptive and Discriminative Classification," in *Proc. Scandinavian Conference on Image Analysis (SCIA)*, 2011.

[39] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *ACCV*, 2012.

[40] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[41] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3774–3782, 2017.

[42] T. Wang, S. Gong, X. Zhu, and S. Wang, "Person re-identification by video ranking," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 688–703, Springer International Publishing, 2014.

[43] Springer, *MARS: A Video Benchmark for Large-Scale Person Re-identification*, 2016.

[44] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627–1645, 2010.

[45] L. Zheng and Shen, "Market-1501 dataset." http://www.liangzheng.com.cn/Project/project_reid.html, 2015.

[46] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," 2014.

[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[48] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[50] M.-O. Arsenault, "Lossless triplet loss." https://towardsdatascience.com/lossless-triplet-loss-7e932f990b24, 2018.

[51] L. Utkin, A. Meldo, M. Kovalev, and E. Kasimov, "An ensemble of triplet neural networks for differential diagnostics of lung cancer," in *2019 25th Conference of Open Innovations Association (FRUCT)*, pp. 346–352, 2019.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[53] E. Bernhardsson, "Annoy-approximate nearest neighbors oh yeah." https://github.com/spotify/annoy, 2019.

[54] "Keras." http://en.wikipedia.org/w/index.php?title=Estimation_lemma&oldid=375747928.

[55] D. Shankar, S. Narumanchi, H. A. Ananya, P. Kompalli, and K. Chaudhury, "Deep learning based large scale visual recommendation and search for e-commerce," *CoRR*, vol. abs/1703.02344, 2017.

[56] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[57] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," 11 2017.

[58] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.