# PERSON RE-IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

A THESIS SUBMITTED TO
THE BOARD OF GRADUATE PROGRAMS
OF
MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS

BY

FATİH AKSU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL AND ELECTRONICS ENGINEERING PROGRAM

SEPTEMBER 2021

Approval of the Board of Graduate Programs

Prof. Dr. Cumali Sabah

Chairperson

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

Assoc. Prof. Dr. Murat Fahrioğlu

Program Coordinator

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Cem Direkoğlu

Supervisor

**Examining Committee Members**

| | | |
|---|---|---|
| Assoc. Prof. Dr. Boran Şekeroğlu | Near East University, Information Systems Engineering | |
| Asst. Prof. Dr. Cem Direkoğlu | METU NCC, Electrical and Electronics Engineering | |
| Asst. Prof. Dr. Meryem Erbilek | METU NCC, Computer Engineering | |

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name : Fatih Aksu

Signature :

# ABSTRACT

## PERSON RE-IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

Aksu, Fatih
Master of Science, Electrical and Electronics Engineering Program
Supervisor: Asst. Prof. Dr. Cem Direkoğlu

September 2021, 78 pages

Person re-identification is an important computer vision topic particularly for surveillance system applications. The main aim of person re-identification is to identify previously observed individuals over a camera network with non-overlapping occurrences. Various approaches have been introduced to overcome this problem. This study has two distinct key contributions. First, a new part-based method for person re-identification, which combines semantically partitioned body part masks with a convolutional neural network, is proposed. With this study, it is observed that the proposed body part-based system has promising results, and has advantages over the baseline person re-identification systems. The second contribution of this study is to investigate and compare potential lightweight convolutional neural networks suitable for person re-identification task. Results show that some lightweight convolutional neural networks can be used instead of more generalized deeper networks. Well-designed lightweight convolutional neural networks may have higher accuracy with lower computational cost for person re-identification.

Keywords: Person Re-identification, Semantic Segmentation, Lightweight Convolutional Neural Networks

# ÖZ

## EVRİŞİMLİ SİNİR AĞLARI KULLANARAK KİŞİYİ YENİDEN TANIMLAMA

Aksu, Fatih
Yüksek Lisans, Elektrik – Elektronik Mühendisliği
Tez Yöneticisi: Dr. Öğr. Üyesi Cem Direkoğlu

Eylül 2021, 78 sayfa

Kişiyi yeniden tanımlama özellikle güvenlik sistemleri uygulamaları için önemli bir bilgisayar görüşü görevidir. Kişiyi yeniden tanımlamanın ana amacı daha önce kaydedilen bir kişiyi o kamera ağı üzerinde tekrar tanımlamaktır. Bu problemi çözmek için farklı yaklaşımlar sunulmuştur. Bu çalışmada iki farklı konuda katkıda bulunulmuştur. İlk olarak, anlamsal bölümlenmiş vücut bölümleri ile evrişimli sinir ağlarını birleştiren kişiyi yeniden tanımlama için bölüm bazlı yeni bir metot sunulmuştur. Bu çalışma ile vücut bölümleri bazlı sistemin umut verici sonuçlar verdiği ve temel kişiyi yeniden tanımlama sistemlerinden daha avantajlı olduğu görüşmüştür. Bu çalışmanın ikinci katkısı ise kişiyi yeniden tanımlama görevi için uygun olabilecek muhtemel hafif evrişimli sinir ağlarını incelemek ve karşılaştırmaktır. Sonuçlar göstermektedir ki genelleştirilmiş, derin evrişimli sinir ağları yerine bazı hafif evrişimli sinir ağları kullanılabilir. İyi dizaynlanmış hafif evrişimli sinir ağları kişiyi yeniden tanımlama için daha düşük hesaplama maliyetiyle daha yüksek doğruluk payına ulaşabilir.

Anahtar Kelimeler: Kişiyi Yeniden Tanımlama, Anlamsal Bölümleme, Hafif Evrişimli Sinir Ağları

Dedicated to three women in my life;

My mother,

My sister,

And my love.

## ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Asst. Prof. Dr. Cem Direkoğlu for his great guidance and help. I also would like to thank my family for their great support. Finally, I would like to thank Ezgi Uzun for all her love and motivation.

I would also like to thank Middle East Technical University Northern Cyprus Campus for supporting us with Scientific Research Project (SRP) Fund (Grant no: FEN-19-D-3).

# TABLE OF CONTENTS

## LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

## LIST OF ABBREVIATIONS

ABBREVIATIONS

| | |
|---|---|
| ANN | Artificial Neural Network |
| CCTV | Closed-Circuit Television |
| CMC | Cumulative Matching Characteristics |
| CNN | Convolutional Neural Network |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| FC | Fully Connected |
| FNN | Feed Forward Neural Network |
| GAN | Generative Adversarial Network |
| LSTM | Long Short Term Memory |
| mAP | Mean Average Precision |
| ML | Machine Learning |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |

# CHAPTER 1

# INTRODUCTION

Surveillance video analysis takes a great part in public safety and security. The number of Closed-Circuit Television (CCTV) cameras are increasing constantly to keep humans safe. These CCTV cameras are recording non-stop but examining these recordings for humans is a long and exhausting process. Especially, uncertainty of time and location that is needed to be examined makes it even harder since all the recordings of all the cameras needed to be examined for a long period of time. Therefore, the aid of computers is required to shorten these examine time as short as possible to find the person of interest before it is too late. Moreover, the performance of CCTV operators can decrease over time because of task disengagement or vigilance decrement. Vigilance can be defined as the ability to sustain attention over time. Donald et al. in 2015 [1] showed that vigilance decrement occurs in CCTV operators and causes decrease in performance. In another study conducted by Donald and Donald in 2015 [2] showed that one third of the operators disengaged after 30 minutes of a 90 minutes recording. This disengagement decreased the detection performance. Therefore, the aid of computers is essential in surveillance video analysis to prevent human sourced performance decrements.

Person re-identification is a computer vision topic that aims to find an observed person over a camera network. An image of a person of interest which is stated as query image is compared with other images captured by the other cameras in the network to find the matched or similar images. Generally, the algorithm outputs different sized candidate lists for humans to select from instead of examining hours of recordings. In literature, different kind of methods are used. Earlier works [3]–[6] extract hand-crafted features and measure the distance between these features to find the images that have minimum distance with query image which implicates that those

images are similar and can belong to the same person. With the increase in computational power, nowadays it is possible to implement more complex algorithms such as deep learning algorithms. Deep learning methods are started to be used in person re-identification. Since person re-identification is generally an image-based system, Convolutional Neural Networks (CNNs) are used mostly in literature. Finding a good architecture for different systems are important and in literature there are many different architectures used in person re-identification. Beside the architecture of backbone network, different methods are also added to improve overall result. These methods can be classified into different categories. Identification models [7]–[9], verification models [10]–[12], distance metric-based models [13]–[15], part-based models [16]–[18] are some example categories.

In this thesis, as a first study, a part-based person re-identification model is introduced that performs better than a baseline model with promising performance. The proposed method is based on body part segmentation and feature weighting using a Gaussian filter.

The second study, in this thesis, is about the depth and complexity of the networks also vary greatly in literature. There are some common backbone architectures that are used in most works such as ResNet-50 [19], VGG-16 [20], and ImageNet [21]. These networks are very deep networks which have many parameters and very hard to compute without professional hardware. Therefore, in this thesis, lightweight networks that are easy to compute such as OSNet [22], NASNet [23], DCTI [24], and MobileNetV2 [25] are experimented and compared for person re-identification (Person Re-ID). Results show that lightweight networks can achieve better results with much less computational complexity.

## 1.1 Motivation

Various networks and methods are used in Person Re-ID. Particularly part-based methods proved to be effective for this task. Some part-based works [13], [18] partition the image at the beginning, others [26] partition at the middle, and some [16], [17], [27] partition at the end. As the location of partition differs, the partition method also differs. However, in general, the image is partitioned in a horizontal manner which means that researchers partition the image into different number of equal horizontal layers. Since the bounding boxes that contain the images of people differ most of the time and the size and positions of people also vary, this method causes misalignments. This part misalignment problem can be seen in Figure 3.1. To align the necessary parts correctly, a body part-based partitioning method is used in this work by using a state-of-art body part segmentation algorithm, namely, Cross Domain Complementary Learning (CDCL) [28]. With the help of CDCL, body part maps of the individuals are extracted. From these maps, binary part masks are created, and then a Gaussian filter is applied to smooth these masks and prevent the feature loss. Finally, these masks are applied to feature maps to extract semantically partitioned local body part features. By this way, the network is trained according to body parts instead of horizontal parts that solves the misalignment problem.

Another important aspect, in person re-identification, is the computational cost of the models. Some models require high computational power to be computed and for researchers and users with limited resources, it is hard to use, reimplement and improve this kind of networks. For instance, widely known networks like ResNet-50 [19], VGG-16 [20], and ImageNet [21] have 25.1M, 134.2M, 60M number of parameters correspondently. Despite there are many Person Re-ID frameworks proposed based on these complex networks, they are computationally not efficient particularly during the training stage. Therefore, some potential lightweight CNNs are experimented in this thesis for Person Re-ID. In this thesis, we hypothesized that well-designed lightweight networks may perform better than complex networks.

## 1.2 Objectives

The objectives of this study are as follows:

- To find a good partitioning method in order to prevent misalignments of body parts
- To improve the partitioning method in order to prevent potential feature losses
- To examine different lightweight networks to decrease the computational cost

## 1.3 Overview of the Thesis

This thesis consists of five chapters. In this first chapter, introduction, motivation and objectives has summarized and will follow by the overview and related publications. Chapter 2 will explain the basic concepts of deep learning and convolutional neural networks which are the basis of this study. Throughout the study, several network architectures and methods demonstrated. In this section, concepts needed to understand the architectures and methods will be explained in detail. Moreover, literature review related to this study will be also presented in Chapter 2.

Chapter 3 will explain the study with the title of Body Part-Based Person Re-Identification With Semantic Segmentation And Gaussian Filtering. In this chapter, a new approach will be demonstrated which uses a semantic segmentation approach for part-based person re-identification. Method itself, implementation details, experiments and results will be described in this chapter.

Chapter 4 will investigate the possible lightweight convolutional neural network to be used in person re-identification task. The architecture of networks and implementation purposes will be explained in details. Moreover, experiments, comparisons, and results also will be demonstrated in this chapter.

Chapter 5 will present conclusions and future work of this study. The methods and results of this study will be summarized in this section with the recommended directions of future research.

## 1.4 Publications Related to This Study

F. Aksu and C. Direkoğlu, "Lightweigth Convolutional Neural Networks for Person Re-Identification," *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2021, pp. 1-5, doi: 10.1109/HORA52670.2021.9461269.

F. Aksu and C. Direkoğlu , "Person Re-Identification in Surveillance Videos using Deep Learning based Body Part Partition and Gaussian Filtering", *Avrupa Bilim ve Teknoloji Dergisi*, pp. 291-296, Nov. 2020, doi:10.31590/ejosat.823257

# CHAPTER 2

# BASIC CONCEPTS AND LITERATURE REVIEW

This chapter will discuss the basic concepts of deep learning, convolutional neural networks, and also will review the related work to provide background and basis for the investigations and analysis reported in this study.

## 2.1 Deep Learning and Convolutional Neural Networks

The term Deep Learning (DL) came up from a revolutionary idea of Neural Networks which is a subsection of Machine Learning (ML) which itself is a subsection of Artificial Intelligence (AI). The dream of creating an intelligent being for humans probably goes far back beyond our estimations. Nil J. Nilsson, in his book [29] shows a humanoid robot design in the form of a medieval knight sketched by Leonardo Da Vinci in 1495. Moreover, he also mentions about the "artificial animals" that is written in the introduction of Thomas Hobbes' book Leviathan published in 1651. Nilsson mentions a few more automated or artificially intelligent models proposed through history. Furthermore, there are lots of science fiction novels, stories, movies about AI dates back to 19$^{th}$ century. These visions made it possible to creation of DL. Even some researches [30], tries to find its source far back in 300 BC with Aristotle and its Associationism, most of the researchers' consensus on Perceptron as the seed of neural networks and DL.

A Perceptron, which is an artificial neuron, is first proposed by Rosenblatt [31], a psychologist from Cornell University, in 1958. Rosenblatt and his group created

Perceptron based on human nervous system, especially visual system. The aim of the machine was to recognize the letters of the Rosenblatt. Perceptron consist of four units; sensory unit, projection unit, association unit and response unit. Sensory unit collects data, projection unit transmits collected data from sensory unit to association unit. Association unit takes collected data and add different weights linearly then apply linear transform onto the thresholded sum and pass it to response unit [30]. Perceptron is illustrated in Figure 2.1. Even the figure shows a sigmoid activation function which is a non-linear function, the first perceptron was using a linear function. In time, non-linear functions are started to be used as activation function and this is one reason that fastened the improvement of Neural Networks.

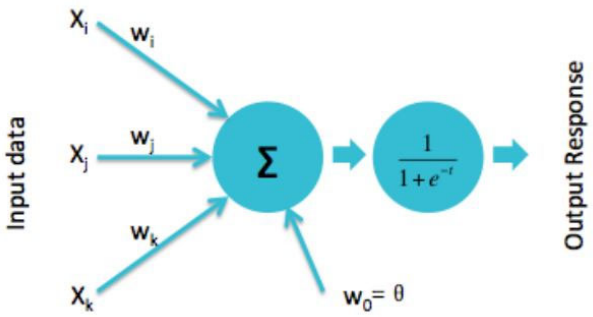


Figure 2.1. Illustration of Perceptron [30]

### 2.1.1 Artificial Neural Networks

Perceptron cannot be considered as a neural network but it can rather be considered as a single neuron. Combination of multiple neurons creates a neural network. The term Artificial Neural Network (ANN) represents an artificially created neural network if a perceptron is considered as an artificial neuron since it is created based on the human nervous system. Even a human or animal neuron is not the same with an artificial neuron in terms of working principle, it is close enough to call this

machine learning algorithms as ANN. These basic units, artificial neurons are connected in parallel and in series to make an ANN. Before directly explaining an ANN, it is better to start with the working principles of a single neuron in detail.

As explained for Perceptron, a neuron takes multiple inputs which are called features and adds different weights to them to find which features are the most important. Then, sum these features which were multiplied with weights and feeds it to an activation function [32]. There are several activation functions used in ANNs or other ML and DL algorithms. Equation 2.1 shows the formula of a single neuron. *y* represents the output. *f* represents the activation function. *w* represents the weights of each input. *x* represents the input features. *b* represents the bias element which is used most of the time for each neuron. *n* is the number of features, i.e. size of input.

$$y = f(\sum_{i=0}^{n} \omega_i x_i + b) \tag{2.1}$$

The output y is generally put into a threshold for classification. For example, for binary classification, sigmoid function is used as activation function. Sigmoid function gives an output between 0 and 1. A threshold is decided which is generally 0.5. It means that the values above 0.5 are classified as 1 and values below 0.5 are classified as 0. In a perceptron, there is only one single neuron whose input is real inputs and whose output is real outputs since there are no hidden layer. However, in ANNs this threshold is performed at the final layer for classification and only the hidden layer takes the real inputs. Figure 2.2 shows the architecture of an ANN. ANNs with several layers are called Deep Neural Network (DNN) [33], [34]. Architecture of a DNN with three hidden layers is shown in Figure 2.3. Hidden layers in the middle take the output of previous hidden layer as their input and give output to the next hidden layer. Each circle in the hidden layers represent a neuron and each line represent a connection. Each line has a different weight and each neuron has a

different bias element. The number of hidden layers is subject to decision, generally with empirical studies. Layers may have different activation functions or all layers may have the same activation function. The general approach is having same activation functions for all hidden layers and a different activation function for output layer.

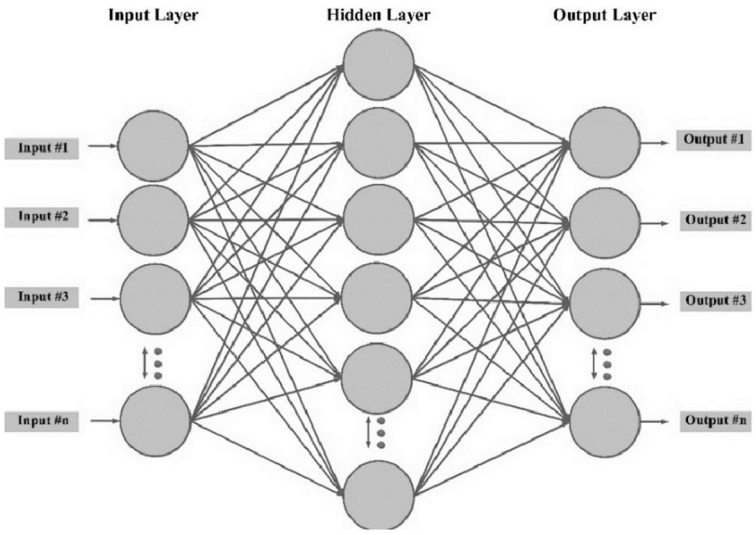


Figure 2.2. Architecture of an ANN [35]

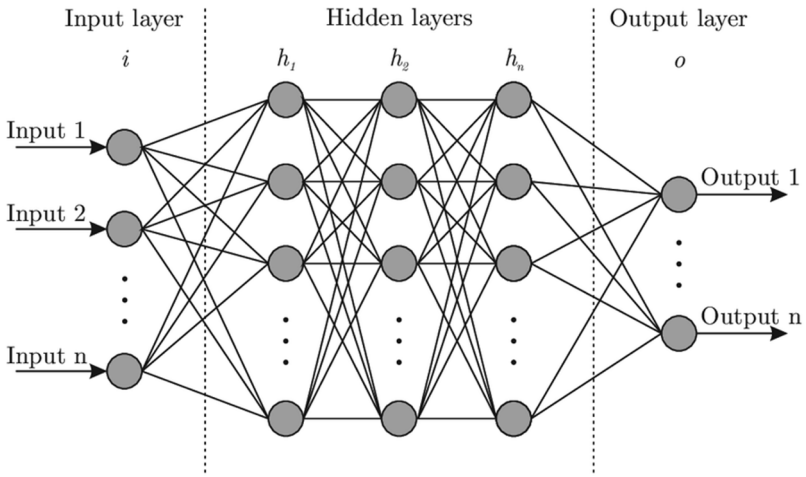


Figure 2.3. Architecture of a DNN with 3 hidden layers [36]

Several different number of activation functions are used in ML and DL algorithms. Some of these activation functions are linear, sigmoid, tanh, ReLU, and Leaky ReLu [37]. Figure 2.4 shows the formulas and graphs of four activation functions. ReLU outputs zero if the input is negative and pass the input to output if it is positive. Leaky ReLU is similar to ReLu but it is not outputting zero if input is negative but gives a small linear output. According to Andrew NG, sigmoid is a good option to be used at the output layer of the network for binary classification. He also mentions that tanh activation is superior than sigmoid except the output layer of the binary classification. Finally, he emphasizes that ReLU is the most popular activation function and it is the most commonly used one in literature [38].

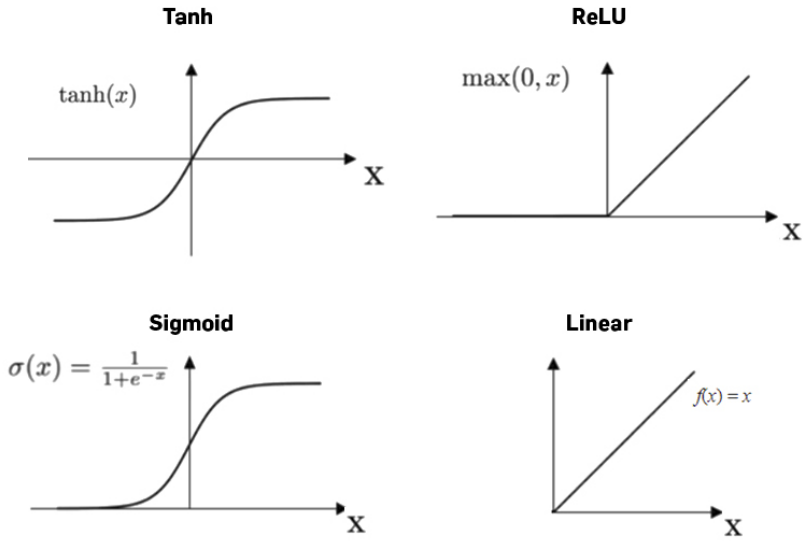


Figure 2.4 Activation functions [37]

Number of neurons in a hidden layer should also be decided, and this is an important decision for architecture engineering. There is no obligation to have a same number of neurons in each hidden layer. After building the network, all the weights are initialized randomly in general and inputs are fed into network and mathematical computations are done according to Equation 2.1. This is a step of forward

propagation. Outputs are compared with the ground truths and a loss is computed according to a cost function. Then, backward propagation (backpropagation) is done to change the weights according to decrease the loss. It continues for a predetermined number of epochs or until it converges. The details of these steps are explained in section 2.1.4. The type of neural network shown in Figure 2.3 is called Feed Forward Neural Network (FNN). There are also different types of neural networks. One of the most important type of neural networks is Recurrent Neural Network (RNN) which is used for time series or text data and Convolutional Neural Network (CNN) which is used for unstructured data such as audio, image or video. The next section explains the CNNs which is the basis of this work.

### 2.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are first introduced by Fukushima in 1988 [39]. However, because of the computational cost, it did not spread much. Later, in 1990s, LeCun et al. used CNNs on handwritten digits classification [40]. The biggest breakthrough happened in 2012 with the AlexNet [41] which won the ImageNet Large Scale Visual Recognition Challenge [42]. After that, CNNs are started to be used widely. CNNs are implemented for visual images. CNNs as its name states take their power from convolution operation. Convolution operation, depending on its scale, consider a pixel and the pixels around it. This enables to extract local features in a continuous manner.

CNNs have a different architecture than other neural networks since the inputs are images instead of numbers. Moreover, there are no neurons but convolution filters instead, in its hidden layers. Filters are applied to images, for input, or to feature maps, for hidden layers and the weights of the filters are tuned in training phase. These convolutional filters act like neurons of ANN. It is applied to the images or

feature maps and the results is fed into an activation function. The main difference between CNN and ANN is this. A general architecture of CNN is depicted in Figure 2.5.

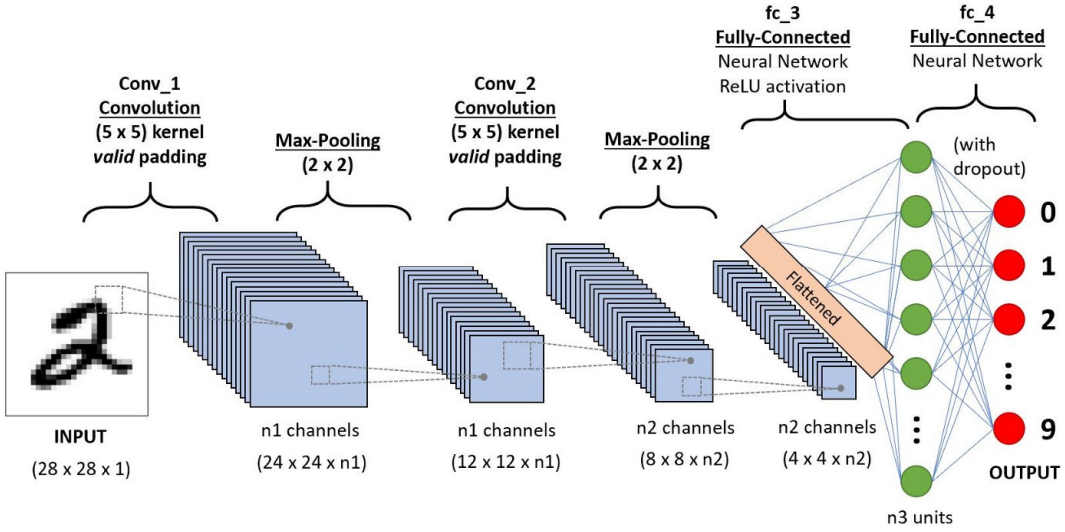


Figure 2.5. An example architecture of CNN [43]

In the figure above, input is a handwritten digit with a size of 28x28x1 where the height and width are 28 pixels and there is one color channel. In chromatic images, channel size is 3 belong to red, green and blue channels. Several number of convolution filter or kernel are applied to input image where the *n1* is the number of filters. Filter and kernel are used interchangeably in literature. The filter size is 5x5 in this case. Filter sizes should be equal to each other in the same layer but they can be different between layers. The output size of the convolutional layer is 24x24xn1 in this case since valid padding is used with *n1* filters. Valid padding is the padding of zeros around the image to keep the output size as same as the input size. Then a max pooling layer is applied to decrease the feature maps' size and also extract correlated feature between further points. Pooling operation is important since it decreases computational costs and increases training speed of CNN. Then, another

convolutional layer and max pooling layer are applied. Finally, the final feature map is flattened and connected to a fully-connected layer. This fully connected layer is connected to a softmax layer for classification. These are the main layers used in CNNs. There are various other layers, for instance, batch normalization layer, dropout layer, average pooling layer etc. The next section explains some of the common layers used in CNNs.

### 2.1.3 Common Layers of Convolutional Neural Networks

#### 2.1.3.1 Convolutional Layer

Convolutional layer is the building block of CNN. It applies cross-correlation operation to its inputs. Although its name is convolutional neural networks and the layer is called convolutional layer, cross-correlation operation is applied which is basically the same operation with convolution but without inversing the matrix. The dimension of the convolution changes according to the input. For image data, as in this work, 2D convolution is used. In 2D convolutional layer, a 2D filter is created with a filter size depending on the architect. Then, this filter is applied to image starting from the upper left most part of the image shifted through right until the end of image. Then it goes back to left most part but with shifting one pixel down. It continuous until the image is completely scanned. When the filter is applied on the first part of the image, every value of the filter is multiplied with the corresponding pixel and the results are summed up. This creates the first value of feature map. Then filter shifts, same operation is applied and it creates second value of feature map. Figure 2.6 shows the one step of convolution operation for better understanding. The edges of filter and image should overlap. This causes output to reduce in size by $n$-1 where $n$ is the size of the filter. To prevent this, zero padding can be applied which is the addition of zero values to the edges of the input. Filter shift left to right and up to down with a stride. Stride means how many pixels the kernel will be shifted. A

general formula indicates the output size shown in Equation 2.2 where $n_o$ is the output size $n_i$ is the input size $p$ is padding, $k$ is kernel size and $s$ is stride. Input or output sizes can belong to height or width of the image. Height and width do not require to be the same for input and output images but it is required to be square for kernels and it is preferred to be an odd number for kernel size.

$$n_o = \frac{n_i+2p+k}{s} + 1 \tag{2.2}$$

There are $k^2$ weights needed to be tuned for a filter with a size of $k$ x $k$. Total number of parameters is $k^2$+1 with the bias. Backpropagation step tries to find correct parameters which decreases the loss as much as possible. Therefore, filters learn how to extract features. The filters in the earlier layers extract simple features while filters in the later layers extract complex features. This is because in the earlier layers' filters are applied on specific parts of the image while in later layers image becomes more compact with the size decrement because of convolution and pooling layers.

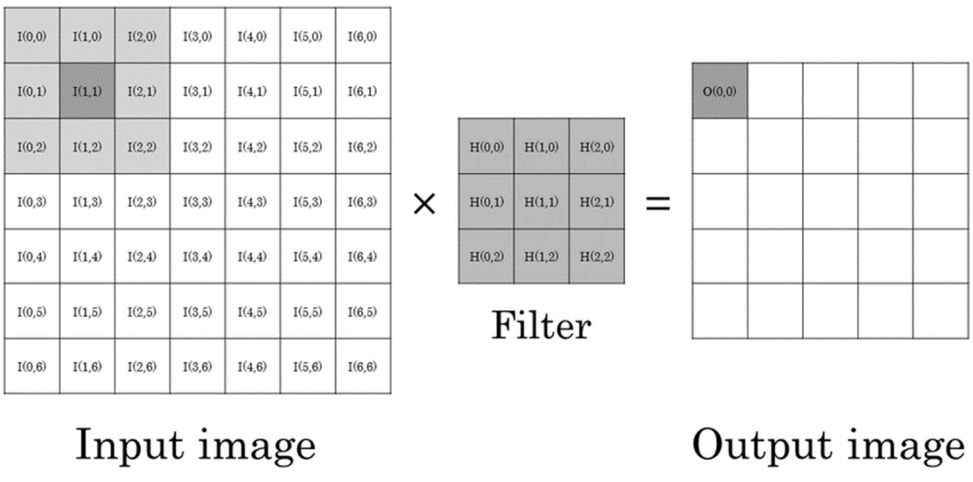


Figure 2.6. Convolution operation [44]

After applying convolution operation, an activation function is applied to the output of the convolutional layer. ReLU activation function got popular over time and used widely for CNNs. [45]. These activation functions are applied elementwise which means that functions are applied to each value of the output separately so the output size does not change. Until now, one filter of one layer is explained. In general, in one layer, more than one filter is used and the total output size multiplies with the filter number since each filter outputs a different feature map. Each convolutional layer has different number of filters in general.

#### 2.1.3.2 Batch Normalization Layer

When the dataset is small, then input data is completely fed into network. However, when dataset is large or input size is large or available memory is small, inputs are fed into network as mini batches. Batch means a portion of the data. The batch size depends on the available memory as well as the designer's choice since it affects the training process of the network. In this work, batch size is decided as 32 most of the time. These batches fed into network and goes through the layers. In each training step, weights are adjusted according to loss. This cause significant changes in the inputs of hidden layers since they are the outputs of the previous hidden layer. Huge changes in inputs of layers causes huge changes in the weights of that layers. Therefore, network gets hard to train and converge in effective weights. Batch normalization can prevent this from happening. It standardizes the inputs of a layer by normalizing it with the mean and standard deviation of the batch. Generally, it is applied after each convolutional layer and also to the input layer. Batch normalization significantly reduce epochs and also helps to generalize the network by providing some regularization. Generalization is also an important task which prevents overfitting which means that the network is tuned very precisely for a specific data. However, input data can change and a good network should handle the input variations.

#### 2.1.3.3 Dropout Layer

Dropout is another regularization technique that generalizes network better and prevents overfitting. Regularization is an important task in neural networks as stated and there are various techniques to generalize the network. However, most of the techniques require extra computational cost which causes network to work slower. Dropout is one of the best techniques that does not increase computational cost, on the contrary it decreases. In dropout layer, some of the input elements are zeroed out with a previously decided probability. In every forward propagation, some randomly chosen elements with a probability in the dropout layer are dropped and computation continue. By this way, it allows to add all the parts of the inputs values to network and make it more generalized.

#### 2.1.3.4 Pooling Layers

Pooling layers are also an essential part of the CNNs. Pooling layers are generally used for down sampling which decreases the size of the input of next convolutional layer. Another important task of pooling layers is making the feature maps more compact so correlated far away pixels are also generates features. Pooling layer works in a similar fashion with convolutional layer. It shifts through the feature maps with a stride. However, pooling filters do not have weights and the application is a little different than convolutional layer. Three types of pooling layers are common in the most CNNs which are max pooling layer, average pooling layer and global average pooling layer. Max pooling filter choses the maximum value of the pixels where the filter is applied and eliminates other values. It is used for choosing the most important features and sweeping the less important features. Average pooling takes the average of the elements where the filter is applied. An example of the max pooling and average pooling operations are demonstrated in Figure 2.7.

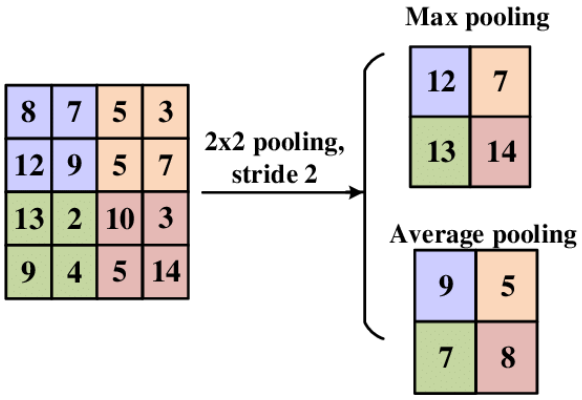


Figure 2.7. Example of max pooling and average pooling [46]

Global average pooling layer takes the average of entire feature map. There is no shifting filter in this pooling type. It is generally used at the end of the network to convert feature maps into a feature vector. After applying global average pooling, the size of the feature vector is equal to channel size of the feature maps. This allow networks to work with any input size since the size of the last layers do not depend on input size but the channel number.

#### 2.1.3.5 Fully Connected Layer

Fully connected (FC) layers are used after the convolutional layers in general. They have the same layer shape of FNNs. After the convolutional layers, the final feature map is flattened and connected to fully connected layers. There may be one or more fully connected layers connected together. Flattened layer works as a feature vector and it is similar to the input layer of an FNN. In general, global average pooling operation is applied to the final feature map and feature vector is obtained. This feature vector is connected to a FC layer and this FC layer is connected to a softmax layer for classification. This approach is taken also in this work, mainly.

#### 2.1.3.6 Softmax Layer

Softmax layer is the last layer of the network to be used for multi-class classification. Number of nodes in softmax layer is equal to number of classes in the dataset. Softmax activation function is applied which is shown in Equation 2.3. $\delta$ indicates the probability of the class. $x$ denotes the features and n denotes the number of classes. It gives the probability of each class between 0 and 1 and the sum of the probabilities are equal to 1. Therefore, it shows that which class that the input image can belong with which probability. After softmax layer, an argmax operation is applied to find the maximum value among the probabilities which indicates the maximum probable class that image belongs to.

$$\delta(x_i) = \frac{e^{x_i}}{\sum_j^n e^{x_j}} \tag{2.3}$$

### 2.1.4 Training and Testing

Training is the process of learning of the network. After creating the network, it is trained with a trainset which is a portion of the overall dataset to learn the suitable parameters that gives desired results. The chosen dataset is break into two, three or four portions, mostly three, before training and testing. A general approach is the dividing dataset into three with a ratio of %60 training, %20 validation and 20% testing [47]. These ratios are changeable and it is the decision of the researcher. However, in bigger dataset which contains hundreds of thousands of data or even millions of data, ratio is generally %90 training, 5% validation and 5% testing [47]. Trainset is the data that are used for training the network for adjusting the parameters of the filters, neurons etc. Validation set is used for hyperparameter tuning which are the parameters do not belong the filters but belong to higher processes like network architecture or optimizer parameters or learning rate etc. Finally, test set contains the

data that the network never seen before in training or validation processes to get an unbiased result. After splitting the data, some transformations are applied to the train set to prepare data for network. Some common transformations are resizing, changing the size of the inputs, normalization, standardizing the values of the inputs generally between 0 and 1, and converting inputs into tensors. Some data augmentation techniques are also applied in this step which are random flip, random crop etc. Some images are flipped horizontally or vertically if it does not break the information of input to have more artificially created different inputs. For instance, horizontal flip can be used on person re-identification since human body is symmetrical horizontally, but it is not a good choice of flipping vertically since no camera will capture people upside down. Random crop is cropping a part of the image to have different images belong to different parts of the original image. These transformations are applied to mini batches, since the dataset in general, is too big to fed into network at once. After transformation, a minibatch is fed into network and forward propagation starts. Transformed image goes into every layer of the network consecutively and necessary computation is done according to layer type. At the end, in case of multi-class classification task, probabilities belong to different classes are computed. Then, the cost is computed according to ground truth of the image and computed probabilities of the classes. There are various loss functions that compute losses and some of the important ones are explained in section 2.1.4.1. After loss is computed the backward propagation starts to adjust the parameters of the layers according to an optimizer. Several optimizers are explained in section 2.1.4.2. These processes continue until every data in the train set is fed into network and computed back. After every data is processed, one epoch is completed. Epoch means the number of iteration that the overall train set is applied to these processes. Epoch is a choice of the designer, and if it is too small, there may be not enough time for network to learn necessary parameters. If it is too large, network can overfit and work for only that portion of the data which is trainset. Another important function is scheduler which decreases the learning rate after some epochs and there are various ways to do it which are explained in section 2.1.4.3.

#### 2.1.4.1 Loss Functions

##### 2.1.4.1.1 Conventional Loss Functions

There are some loss functions that are used widely in many machine learning (ML) algorithms like mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE) etc. These are the regression loss functions which are used when the output is not a category but a continuous number. These are not widely used in classification but rather for regression. However, it is a good idea to mention these conventional loss functions briefly. Mean absolute error commonly called as L1 loss. It is measured as the average of the sum of absolute differences between predictions and ground truths. The mathematical formulation of MAE is depicted in Equation 2.4 where $n$ indicates the number of training examples and $y_i$ is the ground truth of the $i^{th}$ example and $\hat{y}_i$ is the prediction for the $i^{th}$ example.

$$\frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2.4}$$

Mean squared error also called quadratic error or L2 loss is measured as the average of the sum of squared differences between predictions and ground truths. The mathematical formulation of MSE is depicted in Equation 2.5. Root mean squared error as its name states is the root of the MSE. Its mathematical formulation is shown in Equation 2.6. The meanings of the variables are the same as above.

$$\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{2.5}$$

$$\sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{2.6}$$

#### 2.1.4.1.2 Cross-Entropy Loss

Cross-entropy loss is one of the commonly used lost function. It is used for classification tasks. It is also called logarithmic loss or log loss. As its name states, it is a logarithmic function which increases the loss when the predicted probability diverges from the actual ground truth. Figure 2.8 shows and example graph of cross-entropy loss when the ground truth is 1. When the predicted output is close to one, the loss decreases through zero and when the predicted probability is 1 the loss is zero since it is the correct prediction. On the other hand, if the probability of the prediction is close to zero, the loss increases exponentially through infinite. This is the main advantage of the cross-entropy loss since it speeds up the learning by adjusting the loss logarithmically. The mathematical formulation of cross-entropy loss is shown in Equation 2.7. $\hat{y}_i$ is the output of softmax layer and $n$ is the number of examples.

$$-\frac{1}{n}\left(\sum_{i=1}^{n} y_i \log \hat{y}_i\right) \tag{2.7}$$

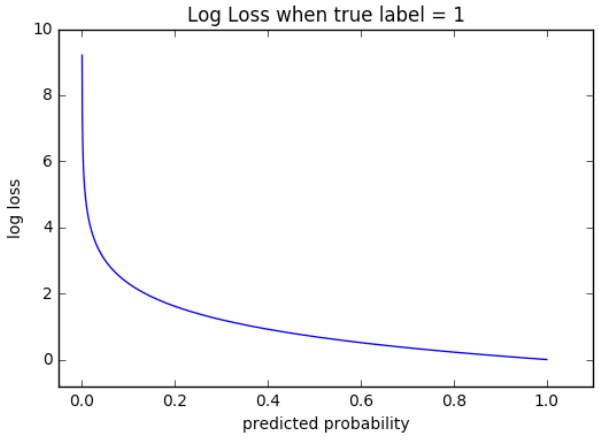


Figure 2.8. Graph of log loss with ground truth of 1

#### 2.1.4.1.3 Triplet Loss

Triplet loss is a little bit different than other loss functions explained above. If the number of different classes are not too high cross-entropy loss is a good option. Most of the classification problems use cross-entropy loss for example ImageNet dataset has 1000 different classes and generally most of the researchers use cross-entropy loss. However, when the number of classes increases, it is hard to use a softmax layer which have the same number of nodes with number of classes. In this case, triplet loss is a better approach. Its main aim is decreasing the distance between intraclass elements and increase the distance between interclass elements. Therefore, training of triplet loss also differs from others. Instead of feeding one input, three inputs are used; one of them the original input called anchor, other one is an input belongs to same class called positive, and another one belongs to different class called negative. These three inputs are fed together and the formula depicted in Equation 2.8 is applied to compute loss.

$$\max(0,\ D(a,p) - D(a,n) + margin) \tag{2.8}$$

D indicates the distance between two elements. Two distance are computed; one of them is between anchor and positive, and the other one is between anchor and negative. Distance metric can be changed according to architecture. L1 distance or L2 distance can be used or another approach is using 1 – cosine similarity. Cosine similarity measures the similarity between two vectors. Triplet loss tries to make distance between anchor and positive close to zero and makes the distance between anchor and negative to be greater than distance between anchor and positive plus margin. Margin is a hyperparameter which needed to be chosen by the designer. Triplet loss is also used in person re-identification. In section 2.2, the example works can be found.

#### 2.1.4.2 Optimizers

The main aim of a neural network is to find parameters that causes minimum loss. This is a process of forward and backward propagation. The process of minimizing loss is called optimization. Several different optimization algorithms are used in ML and DL models. Most of the algorithms works with mini batches. It is hard to compute the loss of entire dataset and update the parameters by using an optimizer to minimize the loss. Therefore, an optimization is performed after each cycle. For a cycle, a single element can be used or a subset of the dataset which is called a mini-batch can be used. One forward propagation of this mini-batch computes the loss and this loss is used by optimization algorithm. Most commonly used optimization functions are gradient descent, RMSprop and ADAM optimizers.

##### 2.1.4.2.1 Gradient Descent

Gradient descent has three different versions, namely, batch gradient descent, mini-batch gradient descent, and stochastic gradient descent. Batch gradient descent takes the entire train set's cost to compute the gradients and update the parameters whereas stochastic gradient descent takes only one training example's cost and does the same processes. Mini-batch gradient descent is better than other two since it takes the cost of a mini-batch and does the same processes. In backpropagation step, the gradients of the parameters are computed according to cost of the mini-batches. Then the weights are updated with the formula shown in Equation 2.9. $\theta_j$ stands for the weights and biases of neurons or convolutional filters. $\alpha$ is the learning rate and $J$ is the cost of the training examples.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.9}$$

The gradient of cost computed for each parameter and that parameter is updated with a learning rate. If the learning rate is too big, the optimization algorithm may not converge to minimum loss while if it is too small it may take very long time to converge. Therefore, it is important to find a suitable learning rate. Another approach is starting the learning rate with a bigger value and decrease it later. This approach is explained in section 2.1.4.3.

There is another version of gradient descent that contains momentum. It is generally referenced as gradient descent with momentum. It is designed to accelerate the optimization. Gradient descent works based on gradients but it may cause to deviate and may go upwards instead of downwards. This is caused by the lack of a history variable. Momentum adds a history variable which is based on the former gradients. This is the reason why the algorithm gets the name momentum. The mathematical formula of the gradient descent with momentum is shown in Equation 2.10 and 2.11. Where $\beta$ is another hyperparameter to be tuned but generally most of the researchers use 0.9.

$$v_{d\theta} = \beta v_{d\theta} + (1 - \beta) d\theta \tag{2.10}$$

$$\theta = \theta - \alpha v_{d\theta} \tag{2.11}$$

#### 2.1.4.2.2 RMSprop

RMSprop is similar to gradient descent with momentum but with an important additional term. It is implemented to prevent the vanishing and exploding gradient problems. It normalizes the momentum part with the root mean square of the gradients. By this way, it slows down the exploding gradients and speeds up the vanishing gradients. Equation 2.12 and 2.13 show the mathematical formula of

RMSprop. To prevent confusion, the letter *s* is used instead of *v* of the gradient descent with momentum formula. The square of gradients are element wise squares. $\beta$ and $\alpha$ are hyperparameters needed to be tuned. $\theta$ indicates the parameters which are weights and biases. $d\theta$ is the derivative of the parameters.

$$s_{d\theta} = \beta s_{d\theta} + (1 - \beta) d\theta^2 \tag{2.12}$$

$$\theta = \theta - \alpha \frac{d\theta}{\sqrt{s_{d\theta}}} \tag{2.13}$$

#### 2.1.4.2.3 ADAM

ADAM stands for adaptive moment estimation. It is implemented by Kingma and Ba in 2015 [48]. It can be said that it is a combination of momentum and RMSprop since it has the properties of both of the algorithms. ADAM was the state-of-the-art optimization algorithm when it is published and it is still one of the most commonly used optimizers in literature, if it is not the most. The mathematical formulas belong to ADAM are shown below. It has three portions; momentum-like portion, rmsprop-like portion and the update portion. There are three hyperparameters that are needed to be tuned. One of them is $\alpha$ which is the learning rate and the other two are $\beta_1$ and $\beta_2$. *v* belongs to momentum portion and *s* belongs to rmsprop portion. $\theta$ indicates the parameters and $d\theta$ indicates the derivatives of the parameters.

$$v_{d\theta} = \beta_1 v_{d\theta} + (1 - \beta_1) d\theta \tag{2.14}$$

$$s_{d\theta} = \beta_2 s_{d\theta} + (1 - \beta_2) d\theta^2 \tag{2.15}$$

$$\theta = \theta - \alpha \frac{v_{d\theta}}{\sqrt{s_{d\theta}}} \tag{2.16}$$

#### 2.1.4.3 Schedulers

Schedulers are used for adjusting the learning rate during the training process. If the same learning rate that is decided at the beginning is used for the whole training process, network may learn too slow or may not learn precisely. Small learning rate causes learning process to be slow and large learning rate causes not to converge. Therefore, schedulers changes learning rate in some milestones during the training. There are several different schedulers but three of them are explained in this section which are the well-known ones and also mentioned throughout this thesis. These schedulers are called StepLR, MultiStepLR and ReduceLROnPlateau.

StepLR reduces learning rate after some predefined number of epochs by a factor of decay. For instance, with a starting learning rate of 0.1 and a step size of 10 and decay factor of 0.1, learning rate will be 0.01 at the $10^{th}$ epoch, 0.001 at the $20^{th}$ epoch, 0.0001 at the $30^{th}$ epoch and so on. MultiStepLR decreases the learning rate for some predefined epochs. It does not reduce learning rate after equal number of epochs. The epochs when the learning rate will be reduced can be defined by the designer with different number of milestones. For example, with a starting learning rate of 0.1 and a decay factor of 0.1 with the milestones 20, 50, 90, learning rate will be 0.01 at the $20^{th}$ epoch, 0.001 at the $50^{th}$ epoch and 0.001 at the $90^{th}$ epoch. The final scheduler is ReduceLROnPlateau. It is a dynamic scheduler that reduces learning rate at not predefined epochs. It decreases the learning rate when the loss stops decreasing for a predefined number of epochs to below of a predefined threshold. For example, with a starting learning rate of 0.1, a threshold of 0.0001, a decay factor of 0.1 and a patience of 3 which is the number of epochs to wait; learning rate will be 0.01 when the difference between present loss and previous loss is lower than 0.0001 for 3 epochs. This process will continue until the training is completed.

## 2.2 Literature Review for Person Re-identification

Person re-identification is an important task in surveillance systems. Its main objective is to find a previously captured person over the camera network. It is an area researched for 2 decades. Person re-identification before taking its name was researched as multi-camera tracking. One of the earliest works published by Huang and Russell in 1997 [49], was using Bayesian formula to predict the appearance of an object that is observed previously from another camera. According to Zheng, Yang and Hauptmann in 2016 [50], the term person re-identification is first used in a multi-camera tracking paper published by Zajdel, Zivkovic and Kröse in 2005 [51]. After that, person re-identification started to be researched with its name. Earlier works before DL era, were using hand-crafted features to compute the distances between features and predict whether they belong to same identity or not. These works are explained under the name of hand-crafted methods in section 2.2.1. Then, ML algorithms are started to be used with these hand-crafted features and also to extract and choose features from beginning. Later, DL algorithms are started to be used with the improvements in computational power and the increase in usage of CNNs.

Different approaches are taken with DL algorithms in person re-identification and they are categorized differently in different survey papers. For example, Lavi, Serj and Ullah in 2018 [52], categorizes the DL methods of person re-identification as based on classification and based on Siamese networks. Wang et al. in 2018 [53], categorizes DL methods as CNN-based methods, CNN and RNN-based methods, GAN-based methods and Hybrid methods. Another review published by Wu et al. in 2019 [54], categorizes DL methods for person re-identification as identification model, verification model, distance metric-based deep model, part-based deep model, video-based deep model, data augmentation-based deep model and other perspectives. Furthermore, person re-identification methods are also categorized according to dataset. There are image-based and video-based models. Image-based

models have also different type of datasets like RGB-D images with depth information or infrared images. However, these types of cameras are not mainstream. Therefore, in this section, RGB type image-based methods are reviewed to narrow it down. This thesis is also based on an image-based system. The subsections are hand-crafted methods in section 2.2.1, identification-based methods in section 2.2.2, verification-based methods in section 2.2.3 and part-based methods in section 2.2.4.

### 2.2.1 Hand-crafted Methods

Hand-crafted methods are based on hand-crafted features before DL era. The recent methods extract features with neural networks. However, before neural networks, researchers were extracting features by themselves. These features vary between researchers. Some of the commonly used features are color histograms, edge histograms, texture channels etc. One of the earliest works published by Gheissari et al. in 2006 [5] uses color and salient edgel histograms for matching. Even though they use spatial-temporal information, they match the images based on the histograms. This work separated the person re-identification task from multi-camera tracking [50]. Gray and Tao [3] used eight color channel which are individual channels of RGB, YCbCr and HS and nineteen texture channels which are the results of convolution with Gabor [55] and Schmid [56] filters. These color and texture channels are also used a number of other works [57]–[60]. Zhao [61] et al. used 32-dimensional LAB color histograms and 128-dimensional scale-invariant feature transform (SIFT) descriptors which are extracted from 10x10 patches with a stride of 5. These features are also used by Zhao et al. in their further researches [62], [63]. Later, these features are also used by Shen et al. in 2015 [64]. In 2012, Layne et al. [65] extracted semantic features of images which show the characteristics of the person such as dress, skin color etc. There were 15 attributes in total. Then, they combined low level visual features with semantic features to match the images.

### 2.2.2 Identification-based Methods

Identification-based methods take classification approach on person re-identification task. In this method, every individual is considered as a different class. In most of the papers, softmax classification, explained in section 2.1.3.6, is used. In 2016, Wu et al. [7] created Feature Fusion Net (FFN) which is the combination of CNN with hand crafted features. First, they extract features with CNN and crafted color and texture histograms, then they concatenated these two feature vectors into a fully connected layer and used a softmax layer to obtain predictions. In their CNN, there are five convolutional layers each followed by a pooling layer. Their hand-crafted features contain color histograms of RGB, HSV, LAB, XYZ, YCbCr and NTSC channels and also 8 Gabor [55] and 13 Schmid [56] texture filters. These features are obtained from 16 horizontally partitioned stripes and each feature has a 16-dimensional histogram. These combined features are called Ensemble of Local Features (ELF). The overall architecture is shown in Figure 2.9.

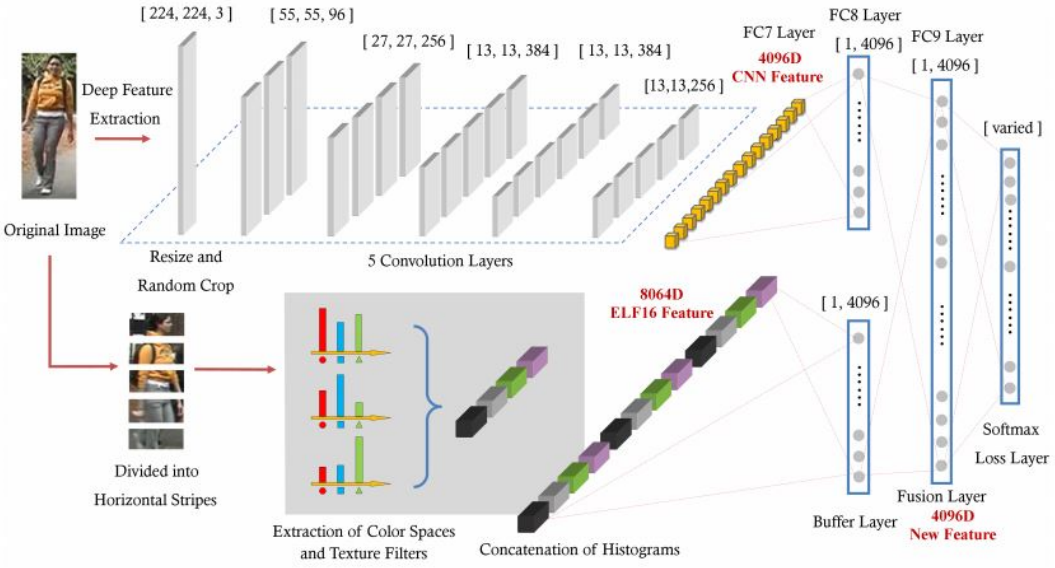


Figure 2.9. Architecture of FFN [7]

Xiao et al. in 2016 [9], implemented a CNN and trained it with multiple datasets to achieve a general network. They also use a domain guided dropout layer instead of a conventional dropout layer. In their dropout layer, the neurons that are non-effective are eliminated according to their domain. They achieved state-of-the-art results on most of the datasets with a large margin. Jin et al. in 2017 [66] proposed a CNN contain nine convolutional layers, four max pooling layers, one fully-connected layer, one feature reweighting layer and finally one softmax layer. They trained the network with combination two different losses which are identification loss and center loss. In 2018, Zheng, Zheng and Yang [67] introduced a Pedestrian Alignment Network (PAN) to align the bounding box better by decreasing the background and preventing the body part losses. They used this network to identify individuals and the results were very promising. Lin et al. in 2019 [68] combined attribute features and identity features together for identifying individuals Figure 2.10. They first extract features from a CNN and from these features predict attributes like gender, cloth types, cloth colors etc. Then, they combined features of CNN with the predicted attributes with re-weighting and fed them to a softmax layer for identification. CNN is trained with two different losses which are attribute loss and identification loss.

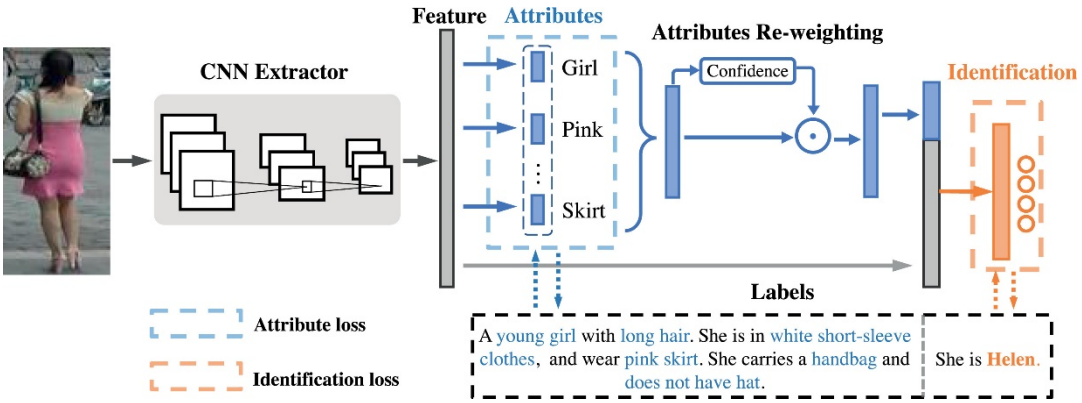


Figure 2.10. Architecture of [68]

### 2.2.3 Verification-based Methods

One of the earliest papers that uses DL on person re-identification [69] uses classification as a method of verification. It extracts features of two images and uses a softmax layer with two nodes to classify as same or different. Figure 2.11 shows the architecture of the network. There are two convolutional layers followed by max pooling layers and one height factoring and maxout grouping layers in between. After them, there is a fully-connected layer and a softmax layer for classification. Although they use softmax layer since the classes are same and different it is considered as a verification-based method. They named the network as Filter Pairing Neural Network (FPNN). Several other papers such as [10], [11] also use this binary classification approach for verification in their study.

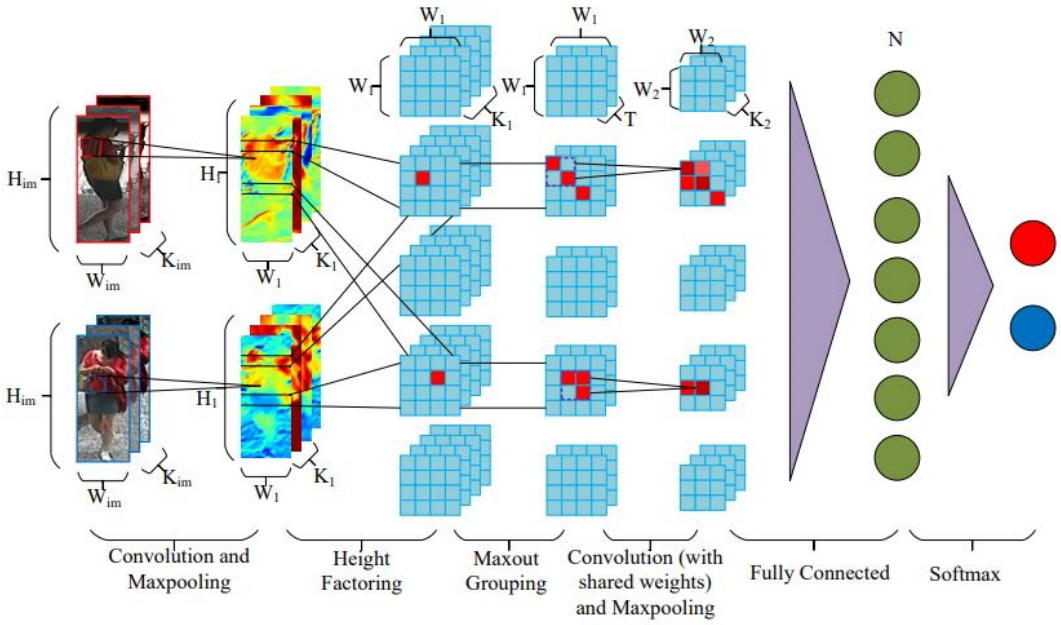


Figure 2.11. Architecture of [69]

Another one of the earliest paper using DL on person re-identification [70] is also uses a verification-based method. It extracts features with three Siamese CNN and

combines them and measure the similarity between two images. These three SCNNs are applied to three parts of two different input images. Figure 2.12 shows the broader architecture of the algorithm. Siamese networks are similar to triplet loss but instead of three inputs it takes two inputs as pair. It extracts the features of these input pairs and computes a distance or similarity metric. Loss is calculated according to this metric then network is trained. They use cosine similarity as similarity metric. The SCNNs contain two convolutional layers each followed by a normalization and max pooling layer and a fully connected layer at the end.

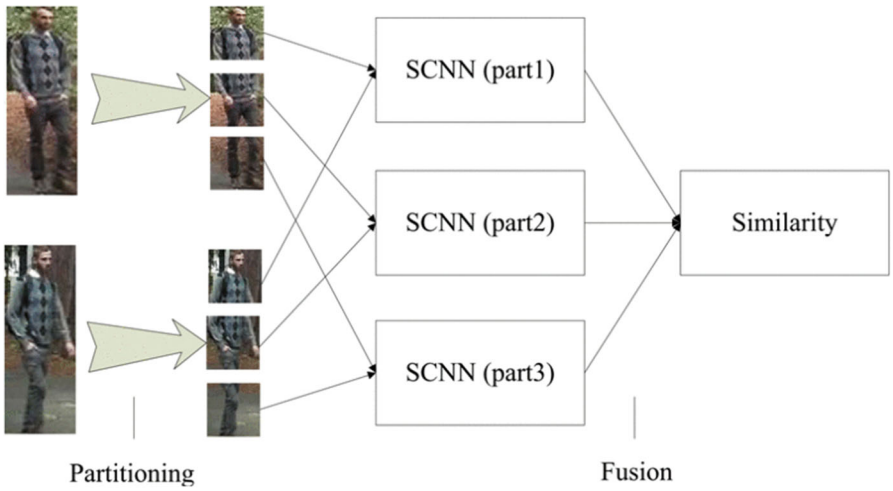


Figure 2.12. Architecture of [70]

Some papers [71]–[73] use a combination of both verification and identification methods. Qian et al. [73] use a similar method and train two network for an input pair and computes the similarity. The difference is that they also use this network as classification and connect softmax layers at the end of them. Another two papers that use combination of identification and verification are published by Zheng et al. in 2017 [72] and Geng et al. in 2016 [71].

### 2.2.4 Part-based Methods

Part-based methods are not a different category than identification-based and verification-based methods. But it is mentioned separately since part-based methods are an important part of this thesis is on part-based methodology. Part-based methods also need to take an identification or verification approach at the end and these methods may also use part-based approach. An important paper published by Sun et al. in 2018 [27] use the part-based method. In most of the papers, partition occurs at the beginning of the network and each part, generally, fed into different networks and concatenated at the end for classification or verification. Sun et al. take a different approach and they partition the image at the end of the network. This method decreases the computational cost significantly since only one network is needed to be trained. They use a backbone network, ResNet-50 in their case but they mention any network would be suitable, and they partition the final feature map into 6 horizontal stripes and take their averages by using global average pooling. Then, they fed these features into 6 different classifiers to have 6 predictions and they use max voting technique to decide which class the image belong to. It needed to be remembered that class here indicates the identity. The architecture is shown in Figure 2.13. They also use a part alignment algorithm to align the misaligned parts partition horizontally. They use a similarity check in intra-part and inter-part and if the value is similar to a neighbor part more than its own part, that value is shifted to vector it belong to. They achieve state-of-the-art results with this algorithm.

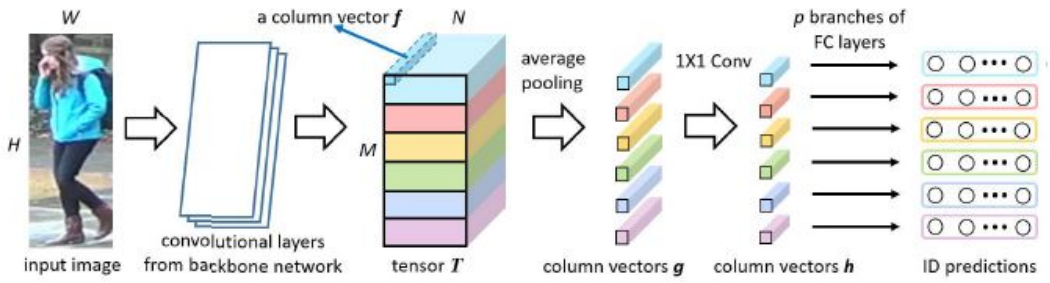


Figure 2.13. Architecture of [27]

Cheng et al. in 2016 [13] also take a part-based approach with a verification-based method. They use a CNN with five branches in parallel. One the branches takes the whole image globally and the other four takes a part of horizontally partitioned four equal stripes. The final features at the fully connected layers of these five branches are concatenated into one global fully connected layer and triplet loss is calculated with two other networks with the inputs of one similar and one different images. The architecture is shown in Figure 2.14. Figure shows the one of the three networks used for triplet loss.

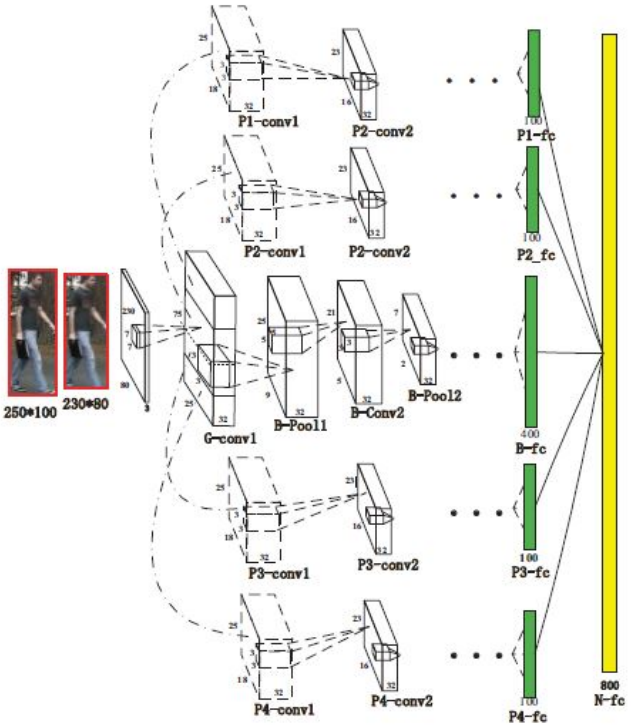


Figure 2.14. Architecture of [13]

Li, Zhu and Gong [74] also uses a CNN with multiple branches but instead of combining the feature at the end and compute loss with combined features, they use a multi-loss approach. They use one global and three local branches and they only

combine three local branches together and predict the class of the person and make another classification with the global branch. Shi et al. [75] partition the input into three overlapping branches on the contrary to above works. Then, they feed these three parts into three parallel branches of the network and combine the final features at the final fully connected layer. Varior et al. [76] take a different approach and instead of using a CNN they use a Long Short Term Memory (LSTM) which is a network model of RNN. Liu et al. [77] also use LSTM but they use it to produce part aware attention features. They use this parts in CNNs and use triplet loss to verify individuals. Attention-based models are also used in [78] and [79]. Most of the recent researches take a part-based approach and also most of them use attention methods.

# CHAPTER 3

# BODY PART-BASED PERSON RE-IDENTIFICATION WITH SEMANTIC SEGMENTATION AND GAUSSIAN FILTERING

Part-based models are effective, and widely used in literature as shown in section 2.2. Various ways of the part-based models were implemented and used by researchers. However, a great challenge in part-based models is part misalignment as seen in Figure 3.1. Body parts of the person should be in align properly so that models can work efficiently. Researchers are trying different methods to align body parts as stated in section 2.2.

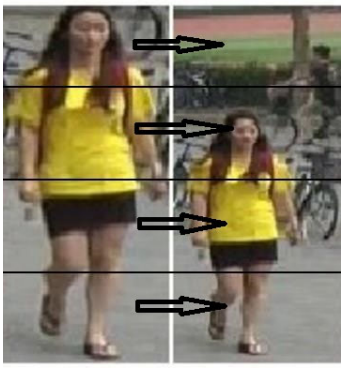

Figure 3.1. Part misalignment

In this work, semantic body part segmentation is used to prevent misalignment, and also to decrease the effect of background features which are almost same for all images taken from the same camera. The segmented body parts are converted into masks. One mask is created for each segmented body part. Segmentation is a difficult task and it gets more difficult when the segmentation applied on body parts.

Moreover, segmenting body part images, recorded by low resolution CCTV cameras at a distance, makes it even more difficult. Therefore, masks are smoothed by using Gaussian filter to prevent features losses caused by deficient masks. Rest of this chapter will explain the proposed method in details and also experiment setup and results of the experiments.

## 3.1 Method

Segmenting body parts is a hard topic and requires a very effective model. Therefore, a state-of-the-art algorithm, namely CDCL [28] is used for body part segmentation. The body part maps which are the output of the CDCL algorithm are taken and then, Gaussian filter is applied on them to create smooth masks. The feature extraction is done by using ResNet-50 network [19]. After that, smoothed masks are applied to feature maps to extract local body part features. A global average pooling is applied to take the average of each part separately. Finally, a fully connected layer followed by a softmax layer is located to the end of the network. Figure 3.2 shows the overall architecture of the proposed method. The upper and lower branches of the figure indicate two different networks. Upper branch belongs to main network and lower one belongs to CDCL until gaussian masking part. Masks are applied via multiplication.

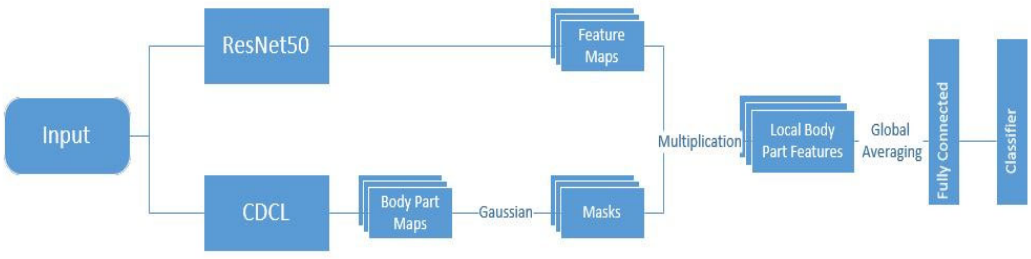


Figure 3.2. Overall architecture of the proposed method

### 3.1.1 Body Part Segmentation

CDCL is used to segment body parts. CDCL has option to decide how many body parts will be segmented. In this work, 7 parts are segmented via CDCL which are head, chest, arms, forearms, upper and lower legs, and background. CDCL use ResNet-101 network as its backbone followed by three head networks which output body part maps, key point maps, and part affinity fields. In this work, only body part maps are obtained since other outputs are not necessary. Figure 3.3 shows the overall architecture of CDCL.

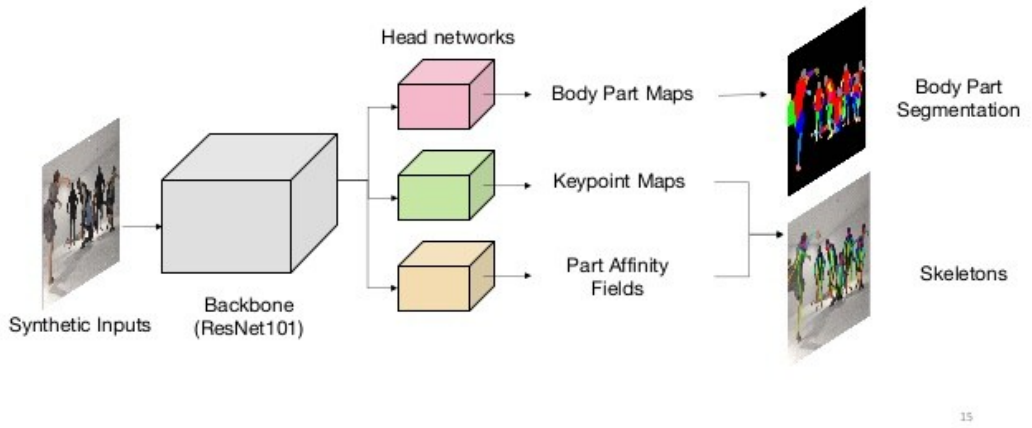


Figure 3.3. Overall architecture of CDCL [28]

CDCL can segment body parts of more than one person in the same image. However, the dataset used in this work have images belong to one person only with a bounding box. Therefore, all the images in the training, validation and test sets are fed into the algorithm separately. The algorithm outputs body part maps of each input image. Figure 3.4 shows the input and output images of the CDCL algorithm. As seen in the figure, input image is segmented into 7 parts. Each part is shown with a different color for demonstration. The original output of the CDCL is a matrix with the size of input image. Each pixel has a value between 0 and 6 corresponding to body parts.

Figure 3.4. Input and output images of CDCL

### 3.1.2 Mask Creation and Smoothing

Local features are extracted by using masks in this work. These masks are created by using the segmented body parts. The output of the CDCL which has a value between 0 and 6 for each pixel is taken first and for each value a different binary mask is created. These masks have 1 at the desired part locations and 0 at the rest of the image. Since there is 7 segmented parts, 6 belong to body and 1 belong to background, 7 different masks are created, 6 of which belong to body parts. However, these binary masks cannot be applied directly to features of the image. In DL algorithms, features are very important and it is required to be used as efficient as possible. Since existing segmentation algorithms still need improvement, CDCL can also output deficient segmented parts and these deficiencies occurs as 0 in the masks. When these binary masks are applied to feature maps, the features in the portions which have 0 value are disappeared. Therefore, a smoothing method with Gaussian filtering is proposed. By using Gaussian filters on binary masks, the near portions of the detected parts which have a value of 1 which can still belong to the same part are also taken into consideration. Moreover, as the nature of the Gaussian filtering, the pixels far from the desired part have small values on the contrary to the pixel next to the desired parts which have close values to 1. Figure 3.5 shows four of

the binary masks (a), smoothed masks (b), and an example of masks applied to input image (c). As seen in the figure, CDCL cannot segment the parts very well. For some parts, only a small portion of them are detected. If we apply these binary masks, it can be seen that, for example in the head portion, only a small part of the cheek will be taken into consideration. The features in that part of the cheek is not enough to show discriminative properties. However, when masks are smoothed with Gaussian filtering, more area is taken into account as seen in part (b). These masks are applied to the global feature maps at the output of the ResNet-50 network but for demonstration purposes, they are applied to input images in part (c) and it can be seen that right parts of the body are masked in this way.

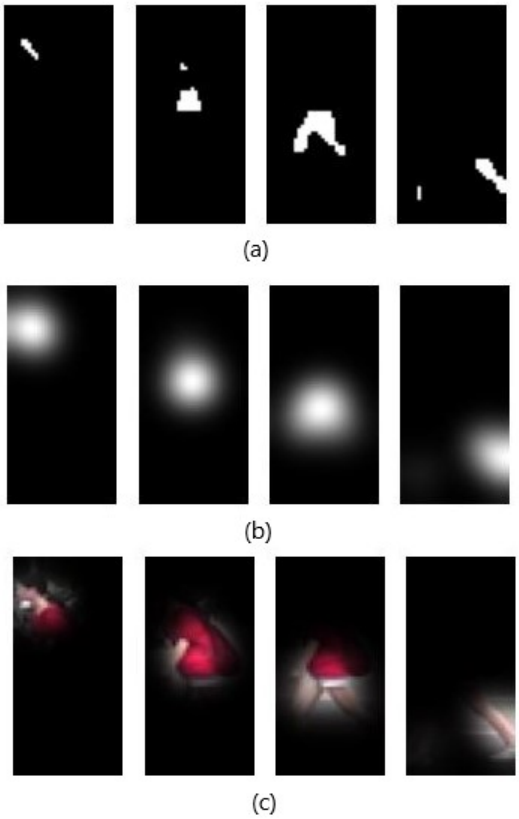


Figure 3.5. Binary mask (a), smoothed masks (b), mask applied on input image (c)

### 3.1.3 Global Feature Extraction

Feature extraction is one of the most important steps in any machine learning / deep learning algorithms, since the whole process is built on finding good patterns in features that distinguishes the classes from each other. These features are generally extracted by using a neural network in deep learning algorithms. The type and architecture of the network depends on the data in general and varies for most of the cases. However, there are some deep networks trained on huge datasets and generalized a little more than others. These networks are proved themselves by achieving good results in many different applications. Therefore, one of this commonly known networks, ResNet-50 namely, is used in this work as a backbone network. It is widely used in person re-identification literature and achieves remarkable results for the most of them.

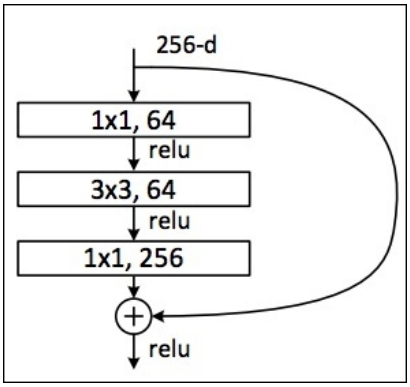


Figure 3.6. Bottleneck building block of ResNet-50/101/152 [19]

ResNet-50 is a 50-layer convolutional neural network with residual branches. The most important feature of this network is the residual branches that it has. A residual branch is a shortcut between input and output of some number of layers. These layers that are shortcut are called bottlenecks. Figure 3.6 shows the implementation of these bottlenecks used in ResNet-50/101/152. Residual branches are used to prevent

vanishing / exploding gradient problems as well as the degradation problem. In deeper networks, the gradients can decrease gradually and eventually gets closer to zero which makes it nearly impossible to learn anything. Another problem is when the gradients increase when layers get deeper and achieves astronomical numbers which is again a great problem that makes the network useless. One solution for these problems is normalization of weights in the beginning (or in the intermediate layers) and the other solution is residual branches. Residual branches also solve the degradation problem which can be defined as the decrement of accuracy in training after some epochs. Therefore, these residual branches allow a network to have deeper layers and ResNet-50 has been proven itself in many datasets with different input sized data.

ResNet has different versions with different number of layers. The overall architectures of some of this Residual Networks are demonstrated in Table 3.1. ResNet-50 is shown in the middle column. It starts with a 7x7 convolutional layer with a stride of 2. A 3x3 max pooling layer follows with again a stride of 2. ResNet-50 has 4 grand layers which have different number of bottlenecks under them. They are stated as conv2, conv3, conv4, and conv5 in the table. The first layers of each grand layer have a stride of 2 and the rest of them have a stride of 1. Therefore, the output size decreases only to half after each grand layer. After these layers, an average pooling layer follows. This average pooling layer is a global average pooling layer which means that it takes the average of whole feature map into one single number. Then, a fully connected layer with a size of 1000 is connected to this averaged feature vectors. In this work, the size of FC layer changed to 751 since there are 751 different people in the dataset.

Table 3.1. Architecture of Residual Networks [19]

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

### 3.1.4 Local Feature Extraction and Classification

Local features, meaning that the features belong to some part of image instead of the whole image, are used in part-based person re-identification algorithms. Therefore, local features are extracted after global features are extracted. It is achieved by applying the masks that are created with CDCL's body part maps and smoothed with gaussian filtering on to the global features extracted by ResNet-50 network. These masks are applied at the end of the network right before the global average pooling layer. After the masks are applied, several feature maps are obtained according to mask number. These feature maps are averaged by themselves, separately, and feature vectors are obtained. These final feature vectors belong to different parts of the body are connected into different classifiers. Each part has it is own FC and softmax layers. Softmax layer is used for classification task. In this work, two different classification method is experimented. First one is the one explained above, different classifiers for different parts and result is obtained by max voting technique between classifiers. The other one is using a shared classifier which means that the final averaged features are concatenated all together and connected to one FC and softmax layer. Results showed that usage of shared classifier increases the accuracy.

## 3.2 Experiments

All the experiments are conducted on a cloud computing system, namely Google Collaboratory or Google Colab [80], in short. It provides free access for several CPU and GPUs based on a dynamic usage limit. The hardware that are allowed to use changes frequently. Therefore, in the experiments, time consumption is not measured but flop size is considered when it is needed. Several different experiments are conducted but only three of them are demonstrated in this thesis which have the most impact belongs to three different proposed methods. The rest of this section explains the dataset that is used and implementation details of experiments. Finally, the next section shows the results of the experiments.

### 3.2.1 Dataset

Market-1501 [81] dataset is used for all of the experiments in this work. It is a commonly known dataset in person re-identification. It is collected from six cameras that are showing in front of a supermarket in Tsinghua University. Five of these cameras are high resolution (1280x1080) cameras while one of them is low-resolution (720x576). It is created because the datasets available in 2015 were limited in scale, its bounding boxes were drawn by hand which is not realistic, and also have only one ground truth and one query image for each identity. It has 32,668 images of 1,501 different person. These images are the bounding boxes that are drawn by an algorithm called Deformable Part Model [82] which creates misalignments since it is not robust, and this makes it more realistic. 12,185 of these images are used for training, 751 of them are used for validation and 19,732 are used for testing.

Figure 3.7. Sample images from Market-1501 dataset [81]

There are 3,368 query images belongs to 750 different people. Each of this query images of a person are taken from a different camera which means that one person has maximum six query images since there are six cameras in total. Testing is done by finding the images belong to same person as query image in the test set which has 19,732 bounding boxes. 2,793 of these images in test set are distractors which means that they have not an image of a person at all or have only a small part of it which makes it impossible to identify correctly. The third row of the Figure 3.7 shows examples of these distractors. These distractors have less than 20% overlapping area with the actual person's image. If this overlapping area is above 50% it is taken as good image. The values between these two is marked as junk which are not included in evaluation.

### 3.2.2 Implementation Details

The algorithm is implemented with Pytorch [83] which is a python library created for deep learning algorithms especially. First, body part maps are extracted via CDCL to be used in masks. These maps are extracted outside of the main algorithm and saved to gain from time and used ready in main algorithm. These maps are taken into network as body part maps then, converted into masks and applied to feature maps inside of the network. A custom dataset function is created to feed the network

with both images and maps belong to same image. Before fed into network, images are transformed by resizing to 512x256 then converting into tensors which are the multidimensional matrices of Pytorch. Finally, the images are normalized with a mean of 0 and standard deviation of 1. Images are fed into network as mini-batches with a size of 32 for the final experiments. This batch size is chosen empirically. Network is chosen as ResNet-50 as stated before. Pre-trained version of it is used in this work which means that the parameters of the network is trained before with a large dataset, ImageNet [42] dataset in this case. The final FC layer of ResNet-50 is changed with a 751 sized FC layer and the network fine tuned which means that every parameter of the network is trained. Transfer learning methods have two main versions one of them is training all the parameters and the other one is training only the later layers and keep the former layers as they are. First option is chosen in this work to adjust the model more precisely to the dataset and increase the overall accuracy. As explained before, the output of the last convolutional layer of the network is taken and partitioned by applying masks. This outputs seven different local feature maps. These feature maps are pooled by global average pooling layer. Two different experiments are conducted; one with separate classifiers and other one with shared classifier. The network is trained for 30 epochs. Stochastic gradient descent is used as the optimizer with a learning rate starting 0.1 and decay by a factor of 0.1 for every 10 epochs. Cross entropy loss is chosen as the loss function since a classification approach is used.

## 3.3 Results

Results are shown in two different metrics which are Cumulative Matching Characteristics, CMC and Mean Average Precision, mAP. CMC shows the probability that the query image found in different sized candidate lists. Rank-1, Rank-5, and Rank-10 lists are generally used to indicate first, first five, and first ten guesses correspondingly. In other words, it shows that what is the probability of

finding the right person in first guess, or in first five guesses, or in first ten guesses. One correct guess in the candidate lists is accepted for that rank. In some old works, Rank-20 list is also used but, in recent works and also in this work, it is not included to the results since it is not a good indicator anymore. mAP is the mean of the average precision which is the area under the precision-recall curve. It shows a more general result than CMC since it is considering both precision and recall. The formulation of mAP can be found in Equations 3.1 and 3.2. *GTP* is the number of ground truth positives. *n* indicates the total number of images. *r* is 1 if the retrieved image at rank *k* is relevant, otherwise it is 0. $P_k$ is the precision at rank *k*. Finally, *N* is the number of query images. Results are shown in Table 3.2.

$$AP = \frac{1}{GTP}\sum_{k}^{n} r_k \frac{(P_k + P_{k-1})}{2} \tag{3.1}$$

$$mAP = \frac{1}{N}\sum_{i}^{N} AP_i \tag{3.2}$$

Table 3.2. Results of the part-based experiments

| **Model** | **Rank-1** | **Rank-5** | **Rank-10** | **mAP** |
|---|---|---|---|---|
| Baseline Method #1 (ResNet-50) [19] | 0.858 | 0.949 | 0.969 | **0.670** |
| Baseline Method #2 (Horizontal partition with individual classifiers) [27] | 0.796 | 0.912 | 0.941 | 0.590 |
| Semantic partition with individual classifiers (no Gaussian filtering) (Proposed Method) | 0.817 | 0.912 | 0.940 | 0.468 |
| Semantic partition with Gaussian filtering and individual classifiers (Proposed Method) | 0.845 | 0.940 | 0.961 | 0.634 |
| Semantic partition with Gaussian filtering and shared classifier (Proposed Method) | **0.870** | **0.952** | **0.971** | 0.668 |

First row of the Table 3.2 belongs to the ResNet-50 network purely. It is shown as a baseline. However, the more precise baseline is in the second row. It belongs to a horizontal partitioned model which partition at the end of the network as this work and uses separate classifier for each part. Partition has some downsides like it makes some data augmentation techniques unusable. Moreover, using masks makes it even harder since the masks should fit the images perfectly. Therefore, the main comparison is made with the baseline #2. Other three rows belong the proposed methods of this section. First one belongs to semantic partition using masks but without smoothing, so basically using binary masks with separate classifiers for each part. It achieves 0.817, 0.912, and 0.940 in Rank-1, Rank-5 and Rank-10 accuracies; and 0.468 in mAP. It does not improve when it is compared with the baseline #2 which achieves 0.796, 0.912, and 0.940 in Rank-1, Rank-5, and Rank-10 accuracies; and 0.590 in mAP. Furthermore, baseline #2 is much better than first proposed method in mAP. The fourth row belongs to second proposed method which is again semantic partition but with smoothing this time and also again with separate classifiers. It achieves 0.845, 0.940, and 0.961 in Rank-1, Rank-5, and Rank-10 accuracies and 0.634 in mAP. Results show that it surpasses baseline #2 with 0.049 in Rank-1, 0.028 in Rank-5, 0.020 in Rank-10 and 0.044 in mAP. In other words, second method has a 4.4% more mAP than baseline #2. The last row belongs to the third proposed method which is semantic partition with smoothed filtering but this time using shared classifier instead of individual classifiers for each part. It achieves 0.870, 0.952, 0.971 in Rank-1, Rank-5, Rank-10 accuracies and 0.668 in mAP. It has the highest results in Rank-1, Rank-5 and Rank-10 accuracies. The mAP is 8% higher than the baseline #2. These results show that it is promising to use semantic partition method with gaussian smoothing applied to masks instead of horizontal partition.

## 3.4 Conclusion

Part-based methods are important for person re-identification task. However, most of the works use horizontal partition which causes misalignment. Semantic partition on the contrary, prevents this misalignment problem and also exclude the background to obtain more efficient features. Results show that using semantic partitioning instead of horizontal partitioning improves the accuracy. Moreover, smoothing the masks that extract semantic parts also improves the final results. Finally, using a shared classifier instead of separate classifier for each part also increases the accuracy.

# CHAPTER 4

# LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORKS FOR PERSON RE-IDENTIFICATION

Convolutional neural networks first introduced by Fukushima in 1988 [39]. However, because of the computational cost, it did not spread much. Later, in 1990s, LeCun et al. used CNNs on handwritten digits classification [40]. The biggest breakthrough happened in 2012 with the AlexNet [41] which won the ImageNet Large Scale Visual Recognition Challenge [42]. After that, CNNs are started to be used widely in any area. However, networks are becoming deeper and more complex as time passes. These deeper networks have many parameters to train, and it is needed powerful hardware. Since not all the researchers or intuitions have powerful devices, it is hard to reimplement and use these deeper networks. The training time and test time highly depend on the hardware. Therefore, computational cost is generally compared by the parameters of the network or the total flops which are the additions/multiplications. For example, ResNet-50 [19] has 25.1M parameters and 3.8G flops in total. VGG-16 [84] has 134.2M parameters and 15.5G flops in total. Another network, PCB [27] has 26.8M parameters. These networks are widely used in person re-identification but as explained, it is hard to implement and improve with limited resources. Thus, in this chapter, some commonly known lightweight networks are implemented and tested in Market-1501 [81] dataset to observe how well they perform for person re-identification task. Some of these networks were designed for object recognition, and some of them are designed directly for person re-identification problem.

## 4.1 Lightweight Convolutional Neural Networks

Four different lightweight convolutional neural networks are implemented and compared in this work. These networks are called DCTI [24], NASNet [23], MobileNetV2 [25], and OSNet [22]. This section explains the architectures and aims of these networks in details.

### 4.1.1 DCTI

DCTI is proposed by Truong, Nguyen and Tran in 2018 in the paper titled "Lightweight deep convolutional network for tiny object recognition" [24]. Their main aim was implementing a lightweight network for small scale datasets which is not very suitable for heavy networks. Moreover, deeper networks are also not very efficient when the input size is small like in the datasets CIFAR-10 and CIFAR-100 [85]. Therefore, they proposed a new network called DCTI which has 7.63M parameters in total. They achieved 94.34% accuracy on CIFAR-10 dataset and 73.65% on CIFAR-100 dataset.

DCTI has five phases of convolutional layers. Each phase has different number of 3x3 convolutional layers. After each phase 2x2 max pooling layers are used. Moreover, after each convolutional layer, dropout and batch normalization layers are used. Overall architecture of DCTI is shown in Figure 4.1. The input size is chosen as 32x32x3. Input is fed to the first phase which has two 3x3 convolutional layers. Using two 3x3 convolutional filters is the same with using one 5x5 filter. However, by using two 3x3 filters decreases the parameter numbers and also makes the network deeper. One 3x3 filter has 9 parameters, two 3x3 filter has 18 parameters, but one 5x5 filter has 25 parameters. By this way, 7 parameters are reduced. Because of this, they used this approach for each phase. After first phase, they used 2x2 max

pooling layer to decrease the size of the feature maps two times. Max pooling also reduce variance and computational complexity. It also extracts low level features from neighborhood. The structure of the network repeats with similar approach with different number of convolutional layers. Second phase has two 3x3 convolutional layers. Third phase has three 3x3 convolutional layers. Fourth phase has two 3x3 convolutional layers. Finally, fifth phase has one 3x3 convolutional layers. After these phases, they use a global average pooling layer to feed feature maps directly to feature vectors. At the end, a fully connected layer and a softmax layer is connected for classification.

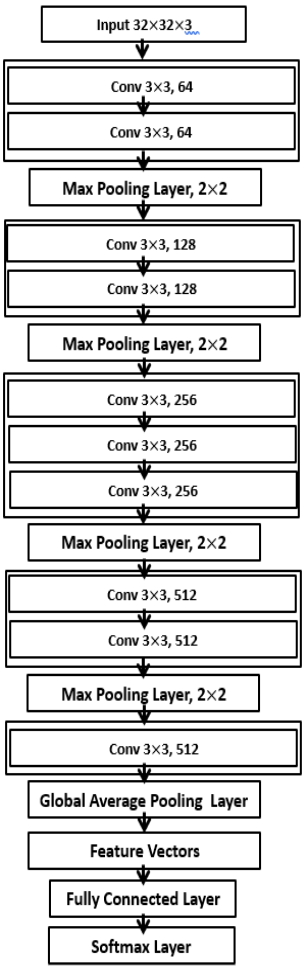


Figure 4.1. Overall architecture of DCTI [24]

In this work, some properties of DCTI are modified to obtain better results. The first change is the input size. Input size kept original as in dataset which is 128x64x3. Since the network is already lightweight and there are less number of parameters compared to other deeper networks. Keeping input size as 128x64x3 does not increase the computational cost much. On the contrary, reducing the input size may cause loss of some important features from images. Another change is made at the end in the fully connected and softmax layers. Since there are 751 different people in Market-1501 dataset, the parameters are changed to 751. Final layer should be the number of class since it is a classification problem. The final change is in dropout layers. Dropout layers are omitted empirically after each phase to increase the accuracy. However, omitting dropout layers increase the number of parameters and flops. It is an important trade-off and it depends on the aim and hardware. Results showed that DCTI is not a good choice for person re-identification even after trying different trade-offs. Results are demonstrated in section 4.3 in details.

### 4.1.2 NASNet

NASNet is proposed by Zoph et al. in 2018 in the paper titled “Learning Transferable Architectures for Scalable Image Recognition” [23]. Their main aim was learning an efficient model directly based on the dataset by using a neural architecture search algorithm which uses reinforcement learning methods. Since image classification models requires an important amount of time for architecture engineering, they did consider a method like this. However, searching a valuable model is an expensive work, they first search for a model that fits a small dataset which is CIFAR-10 and expand it by multiplying layers to a larger dataset which is ImageNet in their work. They first search the best convolutional layer which they called cell and then stack different number of cells together with different parameters to create the final model. They achieve 2.4% error rate on CIFAR-10 dataset and 82.7% top-1 accuracy on ImageNet dataset.

The general, coarser architecture of NASNet is shown in Figure 4.2. CIFAR-10 architecture is the original architecture that is also used in this work. The network consists of two different cells which are called normal cell and reduction cell. Reduction cell follows N consecutive normal cells. There are 2 reduction cells and 2xN normal cells in the general architecture. Type of the normal cell and reduction cell changes in different models. NASNet has three different models namely, NASNet-A, NASNet-B and NASNet-C. In this work, NASNet-A is used which is the best model according to their experiments. The cell architecture of NASNet-A is shown in Figure 4.3. The number of consecutive normal layers, *N*, is chosen experimentally and different architectures gives different results. For this work, *N* is chosen as 4.

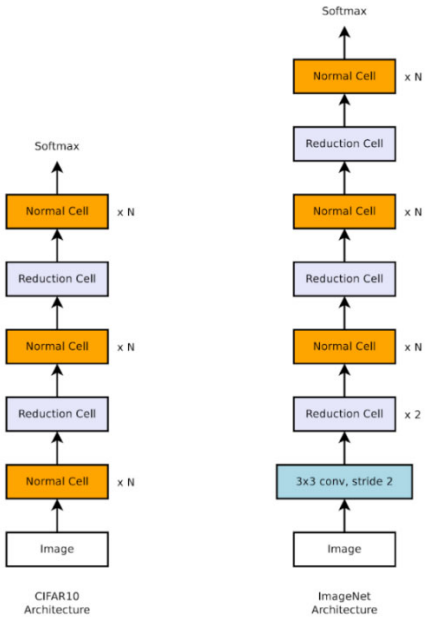


Figure 4.2. General architecture of NASNet [23]

There are different NASNet models created in their work and NASNet-A (4@1056) is chosen for this work. 'A' indicates the model type where the architecture can be

found in Figure 4.3. ‘4’ indicates that four normal cells are repeated before reduction cell and ‘1056’ indicates the number of filters in the penultimate layer of the network. There are five blocks in each cell which have different layers. These layers are seperable layers, identity layers, average pooling layers, and max pooling layers. Seperable layers have four convolutional layers; two 3x3 and two 1x1 layers. One 1x1 convolutional layer follows one 3x3 convolutional layer. After that there is a batch normalization layer and another 3x3 and 1x1 convolutional layer follows and one final batch normalization layer. These six layers creates one seperable layer. Some seperable layers have a 5x5 convolutional layer instead of 3x3. These blocks take inputs from previous hidden state and the one before of the previous hidden states. This architecture gives good results on person re-identification. The result are shown in section 4.3.

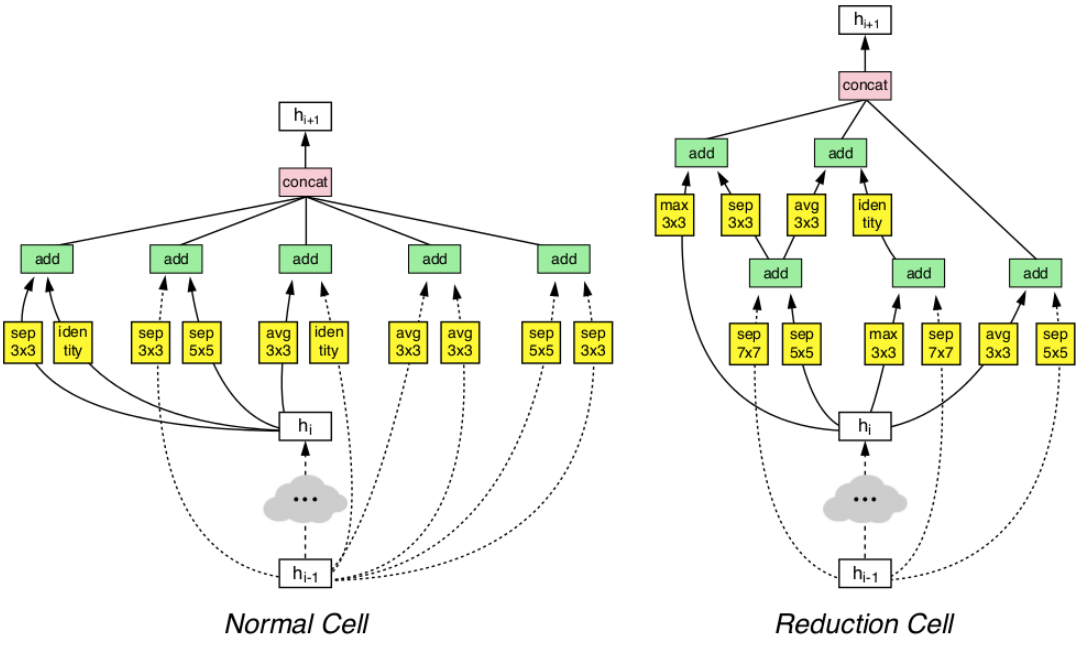


Figure 4.3. NASNet cell architectures of NASNet-A model [23]

### 4.1.3 MobileNetV2

MobileNetV2 is implemented by Sandler et al. in 2018 [25]. This network is mainly implemented to be used in mobile environments since these environments do not have high computational power. They achieved 72% top-1 accuracy on ImageNet dataset. Their main contribution is the inverted residual with linear bottleneck. The overall architecture of MobileNetV2 can be found in Table 4.1. It starts with a 3x3 convolutional layer and continuous with several number of bottlenecks consecutively. *t* column of the table indicates the output to input channels ratio of the fist 1x1 convolutional layer of the bottleneck. *c* indicates the output channels of the overall bottleneck. *n* indicates the number of repeats of the identical bottlenecks. Finally, *s* indicates the stride. The architecture of bottlenecks is shown in Figure 4.4. Bottlenecks consist of two 1x1 convolutional layers and one 3x3 convolutional layer in between. When stride is 1, they added the input to the output of the bottleneck which is called a residual branch.

Table 4.1. Overall architecture of MobileNetV2 [25]

| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 32 | 1 | 2 |
| $112^2 \times 32$ | bottleneck | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | bottleneck | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | bottleneck | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | bottleneck | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | bottleneck | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | bottleneck | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | bottleneck | 6 | 320 | 1 | 1 |
| $7^2 \times 320$ | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2 \times 1280$ | avgpool 7x7 | - | - | 1 | - |
| $1 \times 1 \times 1280$ | conv2d 1x1 | - | k | - | |

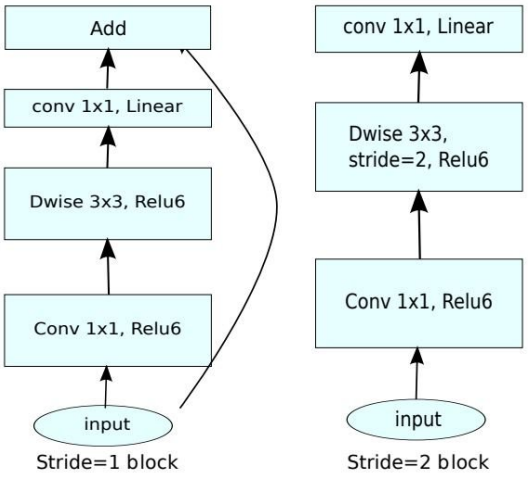


Figure 4.4. Architecture of bottlenecks of MobileNetV2 [25]

### 4.1.4 OSNet

OSNet is proposed by Zhou et al. in 2019 in the paper titled "Omni-Scale Feature Learning for Person Re-Identification" [22]. It is a special network created for person re-identification task. Since person re-identification is an instance-level recognition problem, it relies on discriminative features. Features can be depending on different scales and combination of different scales. Therefore, they created OSNet which stands for omni-scale network. OSNet is a lightweight network which has the least number of parameters among other three networks. Besides that, it also gives the best results. The overall architecture of OSNet is shown in Table 4.2. It has 5 convolutional stages, 2 transition stages and global average pooling and fully connected layers at the end. The main contribution of this network is its bottlenecks. 3 convolutional stages have 6 bottlenecks in total. The architecture of bottlenecks is shown in Figure 4.5.

Table 4.2. Overall architecture of OSNet [22]

| stage | output | OSNet |
|---|---|---|
| conv1 | 128×64, 64<br>64×32, 64 | 7×7 conv, stride 2<br>3×3 max pool, stride 2 |
| conv2 | 64×32, 256 | bottleneck × 2 |
| transition | 64×32, 256<br>32×16, 256 | 1×1 conv<br>2×2 average pool, stride 2 |
| conv3 | 32×16, 384 | bottleneck × 2 |
| transition | 32×16, 384<br>16×8, 384 | 1×1 conv<br>2×2 average pool, stride 2 |
| conv4 | 16×8, 512 | bottleneck × 2 |
| conv5 | 16×8, 512 | 1×1 conv |
| gap | 1×1, 512 | global average pool |
| fc | 1×1, 512 | fc |
| # params | | 2.2M |
| Mult-Adds | | 978.9M |

In the first and last convolutional stages, there are no bottlenecks. In the first stage, there is a 7x7 convolutional layer with a stride of 2 followed by a 3x3 max pooling layer. The last stage contains a 1x1 convolutional layer. Transition stages have 1x1 convolutional layers followed by 2x2 average pooling layers. In the convolutional layers in the middle 2 bottlenecks occur which are concatenated consecutively. These bottlenecks have 4 different branches. Each branch has a different scale. These scales are 3, 5, 7, and 9. These scales indicates the receptive field of the filters. However, instead of using 3x3, 5x5, 7x7, and 9x9 convolutional filters, they use different number of 3x3 filters consecutively. It decreases the parameters size while keeping the receptive field the same. After each branch, there is an aggregation gate. Before aggregation gate, scales are homogenous which means that each scale have the same weight. The aggregation gate gives different weight to branches so they have a heterogenous combination. After that, outputs of all the branches are combined with a residual block to give an output.

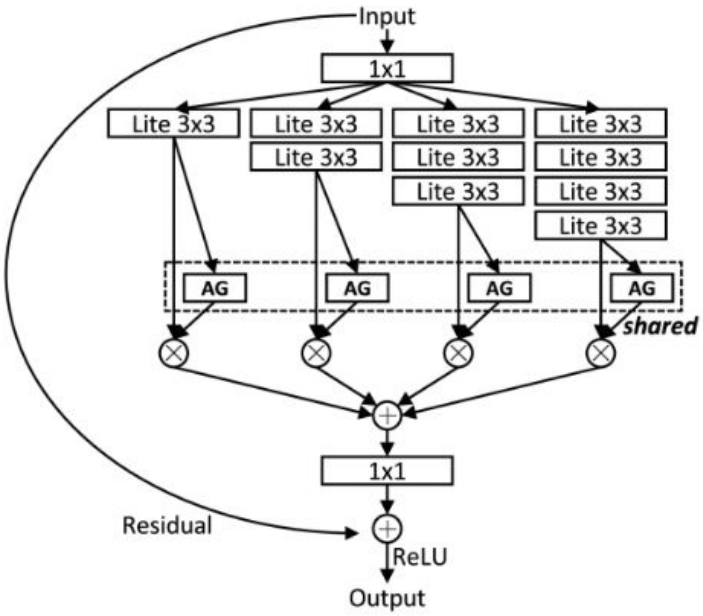


Figure 4.5. Architecture of bottlenecks of OSNet [22]

## 4.2 Experiments

Experiment are conducted on a cloud environment which is called Google Colab. Because of the Google Colab's dynamic usage limits, hardware that are allowed to be used is changing frequently. Therefore, comparison is made according to parameter numbers and number of flops. Experiments are done on Market-1501 dataset. The details of the dataset are explained before in the section 3.2.1. Networks are implemented with Pytorch library. The first network, DCTI, is reimplemented from scratch with some changes that are mentioned in section 4.1.1. Other three networks' implementations are taken from the Torchreid [86] library. Only the last layers are changed according to the number of classes (number of individuals). All the networks are trained with a batch size of 32 for 30 epochs. Loss function is chosen as cross entropy loss as it is approached as a classification problem and the networks are created for this purpose. As optimizer, stochastic gradient descent is used.

Learning rate is chosen as 0.1 at first. ReduceLROnPlateau function is used as scheduler. This function decreases the learning rate by 0.1 when the loss does not decrease for 4 epochs. L2 distance is used while testing to compute the distance between query image and gallery images. All the networks are trained from scratch for a fair comparison. The results are shown in the next section.

## 4.3 Results

Four networks are trained and tested on Market-1501 dataset with the details explained in the previous section. Two different metrics are used which are CMC and mAP whose details can be found in section 3.3. Table 4.3 shows the number of parameters and number of flops of the networks. Table 4.4 shows the results of the experiments.

Table 4.3. Number of parameters and number of flops of the lightweight networks

| **Model** | **Parameters** | **Flops** |
|---|---|---|
| ResNet-50 | 25,046,831 | 677,931,759 |
| DCTI | 8,022,447 | 2,056,823,023 |
| NASNet | 4,232,978 | 99,959,103 |
| MobileNetV2 | 2,224,960 | **55,627,776** |
| OSNet | **2,193,616** | 255,199,776 |

The measured final total number of parameters and total number of flops are demonstrated above after all the modifications. ResNet-50 has 25M parameters and 678G flops in total. DCTI has the highest number of parameters and flops with 8M parameters and 2G flops between lightweight networks. NASNet comes later with

4.2M parameters and nearly 100M flops. MobileNetV2 has the least number of flops which is 55.6M flops and it has 2.2M parameters. Finally, OSNet is the one with the least number of parameters which is 2.19M and it has 255.2M flops.

Table 4.4. Results of the experiments of lightweight networks

| **Model** | **Rank-1** | **Rank-5** | **Rank-10** | **mAP** |
|---|---|---|---|---|
| ResNet-50 | 0.455 | 0.690 | 0.767 | 0,241 |
| DCTI | 0.433 | 0.666 | 0.749 | 0.209 |
| NASNet | 0.593 | 0.787 | 0.848 | 0.332 |
| MobileNetV2 | 0.498 | 0.725 | 0.800 | 0.250 |
| OSNet | **0.628** | **0.823** | **0.876** | **0.350** |

The rank-1, rank-5, rank-10 and mAP results of the networks are shown above. In general, mAP is a more sophisticated metric than CMC. Therefore, when the mAP results are compared, OSNet has the highest results with the least number of parameters and the DCTI has the lowest results with the highest number of parameters. Moreover, OSNet also achieves the best results according to the metrics. NASNet comes second after OSNet in mAP. ResNet-50 is better than only DCTI in mAP and worse than any other lightweight networks.

## 4.4 Conclusion

General, deeper networks that are proven themselves on other tasks such as image classification are used widely for person re-identification task. However, results show that the number of parameters is not an indication for the accuracy. When the network is task specific as OSNet, it gives the best results with least number of

parameters. Therefore, building a specific network instead of a deeper general network gives better results. However, it is a time-consuming approach to find an effective network specific to a task and results of the NASNet show that architecture search algorithms can also give acceptable results. Implementing more sophisticated search algorithms may be a good approach for further research.

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

Person re-identification is an important and popular topic in computer vision literature and surveillance systems. Different approaches have been proposed over two decades. In this study, these different approaches are investigated, and a new part-based approach is introduced and implemented. Most of the works in literature partition the images or features into equal horizontal stripes or use an attention method. Furthermore, most of the researches partition the images in the beginning. In this study, partition is done at the end of the network on the final feature maps. Moreover, instead of partitioning the features into equal horizontal parts, semantic partition is done with a separate state-of-the-art network. Partitioned body part masks are multiplied with final feature maps and local feature vectors are obtained. Instead of using separate classifier for each part, a shared classifier is used. Results show that this proposed method increased the rank-1 accuracy 7%, rank-5 accuracy 4%, rank-10 accuracy 3% and mAP 8% compared to the work with horizontal partition and individual classifiers.

Second part of the thesis focus on potential lightweight convolutional neural networks that can be used on person re-identification task. Four different networks are reimplemented and experimented. These networks are called DCTI, NASNet, MobileNetV2 and OSNet. Results showed that OSNet is the most suitable network for person re-identification. It has the least number of parameters which is 2.19M

parameters and achieves 0.628 in rank-1 accuracy, 0.823 in rank-5 accuracy, 0.876 in rank-10 accuracy and 0.350 in mAP. It can be concluded that specifically tailored networks like OSNet gives the best results. The second-best option is NASNet. It has the second least number of parameters which is 2.22M parameters. It achieves 0.593 in rank-1, 0.787 in rank-5, 0.848 in rank-10 accuracies and 0.332 in mAP. These results show that networks build with another AI algorithm, reinforcement learning in this case, can also achieve good results. They also proved that the deeper networks do not always the best choice. Well-designed lightweight networks can achieve better results than general deep networks.

## 5.2 Future Work

Literature on person re-identification is growing constantly. It is mostly growing through mostly part-based methods. In this study, we have also observed that it is effective to use part-based approach for person re-identification. Moreover, we have seen that specifically tailored lightweight CNNs give better results than generally built deeper networks. Computational cost is an important challenge for DL methods and it is sensible to use lightweight networks to overcome this challenge. For further research, it seems a good approach to use a reinforcement learning method to create a specific network specifically for Person Re-ID and use it as backbone network. Multiple datasets can be used to create the model for generalization in a topic. It is not only a good approach for person re-identification but it can also be used for any task. Combining part-based method with an automatically created network specifically tailored for Person Re-Id task can give state-of-the-art results in person re-identification.

## REFERENCES


[1] F. Donald, C. Donald, and A. Thatcher, "Work exposure and vigilance decrements in closed circuit television surveillance," *Applied Ergonomics*, vol. 47, pp. 220–228, Mar. 2015, doi: 10.1016/j.apergo.2014.10.001.

[2] F. M. Donald and C. H. M. Donald, "Task disengagement and implications for vigilance performance in CCTV surveillance," *Cognition, Technology and Work*, vol. 17, no. 1, pp. 121–130, Feb. 2015, doi: 10.1007/s10111-014-0309-8.

[3] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Oct. 2008, vol. 5302 LNCS, no. PART 1, pp. 262–275. doi: 10.1007/978-3-540-88682-2_21.

[4] Y. Cai and M. Pietikäinen, "Person Re-identification based on global color context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6468 LNCS, no. PART1, pp. 205–215. doi: 10.1007/978-3-642-22822-3_21.

[5] N. Gheissari, T. B. Sebastian, P. H. Tu, J. Rittscher, and R. Hartley, "Person reidentification using spatiotemporal appearance," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 1528–1535. doi: 10.1109/CVPR.2006.223.

[6] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, "Person re-identification by symmetry-driven accumulation of local features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2360–2367. doi: 10.1109/CVPR.2010.5539926.

[7] S. Wu, Y. C. Chen, X. Li, A. C. Wu, J. J. You, and W. S. Zheng, “An enhanced deep feature representation for person re-identification,” May 2016. doi: 10.1109/WACV.2016.7477681.

[8] L. Wu, Chunhua Shen, and A. van den Hengel, “Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification,” *Pattern Recognition*, vol. 65, pp. 238–250, May 2017, doi: 10.1016/j.patcog.2016.12.022.

[9] T. Xiao, H. Li, W. Ouyang, and X. Wang, “Learning Deep Feature Representations with Domain Guided Dropout for Person Re-identification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1249–1258. Accessed: Jun. 15, 2021. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2016/html/Xiao_Learning_Deep_Feature_CVPR_2016_paper.html

[10] L. Wu, C. Shen, and A. van den Hengel, “PersonNet: Person Re-identification with Deep Convolutional Neural Networks,” Jan. 2016, Accessed: Jun. 15, 2021. [Online]. Available: http://arxiv.org/abs/1601.07255

[11] E. Ahmed, M. Jones, and T. K. Marks, “An improved deep learning architecture for person re-identification,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Oct. 2015, vol. 07-12-June-2015, pp. 3908–3916. doi: 10.1109/CVPR.2015.7299016.

[12] W. Li, R. Zhao, T. Xiao, and X. Wang, “DeepReID: Deep filter pairing neural network for person re-identification,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Sep. 2014, pp. 152–159. doi: 10.1109/CVPR.2014.27.

[13] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, “Person re-identification by multi-channel parts-based CNN with improved triplet loss

function," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 1335–1344. doi: 10.1109/CVPR.2016.149.

[14] A. Hermans, L. Beyer, and B. Leibe, "In Defense of the Triplet Loss for Person Re-Identification," Mar. 2017, Accessed: Jun. 15, 2021. [Online]. Available: http://arxiv.org/abs/1703.07737

[15] C. Su, S. Zhang, J. Xing, W. Gao, and Q. Tian, "Deep attributes driven multi-camera person re-identification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9906 LNCS, pp. 475–491. doi: 10.1007/978-3-319-46475-6_30.

[16] Y. Suh, J. Wang, S. Tang, T. Mei, and K. M. Lee, "Part-aligned bilinear representations for person re-identification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Sep. 2018, vol. 11218 LNCS, pp. 418–437. doi: 10.1007/978-3-030-01264-9_25.

[17] H. Yao, S. Zhang, R. Hong, Y. Zhang, C. Xu, and Q. Tian, "Deep Representation Learning with Part Loss for Person Re-Identification," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2860–2871, Jun. 2019, doi: 10.1109/TIP.2019.2891888.

[18] D. Li, X. Chen, Z. Zhang, and K. Huang, "Learning Deep Context-aware Features over Body and Latent Parts for Person Re-identification," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 7398–7407, Oct. 2017, Accessed: Jun. 15, 2021. [Online]. Available: http://arxiv.org/abs/1710.06555

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on*

*Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2015. Accessed: Jun. 15, 2021. [Online]. Available: http://www.robots.ox.ac.uk/

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," Mar. 2010, pp. 248–255. doi: 10.1109/cvpr.2009.5206848.

[22] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2019, vol. 2019-October, pp. 3701–3711. doi: 10.1109/ICCV.2019.00380.

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. v. Le, "Learning Transferable Architectures for Scalable Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, Jul. 2017, Accessed: Jun. 15, 2021. [Online]. Available: http://arxiv.org/abs/1707.07012

[24] T. D. Truong, V. T. Nguyen, and M. T. Tran, "Lightweight deep convolutional network for tiny object recognition," in *ICPRAM 2018 - Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods*, 2018, vol. 2018-January, pp. 675–682. doi: 10.5220/0006752006750682.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[26] B. Xie, X. Wu, S. Zhang, S. Zhao, and M. Li, "Learning Diverse Features with Part-Level Resolution for Person Re-Identification," *Lecture Notes in*

*Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12307 LNCS, pp. 16–28, Jan. 2020, Accessed: Jun. 15, 2021. [Online]. Available: http://arxiv.org/abs/2001.07442

[27] 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 *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Sep. 2018, vol. 11208 LNCS, pp. 501–518. doi: 10.1007/978-3-030-01225-0_30.

[28] K. Lin, L. Wang, K. Luo, Y. Chen, Z. Liu, and M. T. Sun, "Cross-Domain Complementary Learning Using Pose for Multi-Person Part Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 3, pp. 1066–1078, Mar. 2021, doi: 10.1109/TCSVT.2020.2995122.

[29] N. J. Nilsson, *The Quest for Artificial Intelligence*. Cambridge University Press, 2009.

[30] H. Wang and B. Raj, "On the Origin of Deep Learning," Feb. 2017, Accessed: Jul. 15, 2021. [Online]. Available: https://arxiv.org/abs/1702.07800v4

[31] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, Nov. 1958, doi: 10.1037/H0042519.

[32] S. S. Tirumala, S. Ali, and C. P. Ramesh, "Evolving deep neural networks: A new prospect," *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016*, pp. 69–74, Oct. 2016, doi: 10.1109/FSKD.2016.7603153.

[33] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the Number of Linear Regions of Deep Neural Networks," *Advances in Neural Information*

*Processing Systems*, vol. 4, no. January, pp. 2924–2932, Feb. 2014, Accessed: Sep. 06, 2021. [Online]. Available: https://arxiv.org/abs/1402.1869v2

[34] M. Mirzaey, M. Behdad, and Y. Hojatpour, "Applications of Artificial Neural Networks in Information System of Management Accounting," 2017.

[35] F. Bre, J. M. Gimenez, and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, pp. 1429–1441, Jan. 2018, doi: 10.1016/J.ENBUILD.2017.11.045.

[36] "Activation Function - AI Wiki." https://docs.paperspace.com/machine-learning/wiki/activation-function (accessed Jul. 15, 2021).

[37] A. NG, "Activation functions ." https://www.coursera.org/learn/neural-networks-deep-learning/lecture/4dDC1/activation-functions (accessed Sep. 06, 2021).

[38] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, Jan. 1988, doi: 10.1016/0893-6080(88)90014-7.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[40] A. Krizhevsky, … I. S.-A. in neural, and undefined 2012, "Imagenet classification with deep convolutional neural networks," *proceedings.neurips.cc*, Accessed: Jul. 11, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[41] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp.

211–252, Sep. 2014, Accessed: Jun. 28, 2021. [Online]. Available: http://arxiv.org/abs/1409.0575

[42] "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Jul. 16, 2021).

[43] C. Baskin, N. Liss, E. Zheltonozhskii, A. M. Bronshtein, and A. Mendelson, "Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform," Jul. 2017, Accessed: Jul. 16, 2021. [Online]. Available: http://arxiv.org/abs/1708.00052

[44] R. Zaheer and H. Shaziya, "GPU-based empirical evaluation of activation functions in convolutional neural networks," *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018*, pp. 769–773, Jun. 2018, doi: 10.1109/ICISC.2018.8398903.

[45] H. Yingge, I. Ali, and K. Y. Lee, "Deep neural networks on chip - A survey," *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*, pp. 589–592, Feb. 2020, doi: 10.1109/BIGCOMP48618.2020.00016.

[46] A. NG, "Train / Dev / Test sets - Practical Aspects of Deep Learning | Coursera." https://www.coursera.org/lecture/deep-neural-network/train-dev-test-sets-cxG1s (accessed Aug. 05, 2021).

[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, Accessed: Jul. 18, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[48] T. Huang, S. R.- IJCAI, and undefined 1997, "Object identification in a bayesian context," *Citeseer*, Accessed: Jul. 19, 2021. [Online]. Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.9216&rep=rep1&type=pdf

[49] L. Zheng, Y. Yang, and A. G. Hauptmann, “Person Re-identification: Past, Present and Future,” Oct. 2016, Accessed: Jul. 19, 2021. [Online]. Available: https://arxiv.org/abs/1610.02984v1

[50] W. Zajdel, Z. Zivkovic, and B. J. A. Kröse, “Keeping track of humans: Have I seen this person before?,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, pp. 2081–2086, 2005, doi: 10.1109/ROBOT.2005.1570420.

[51] B. Lavi, M. F. Serj, and I. Ullah, “Survey on Deep Learning Techniques for Person Re-Identification Task,” Jul. 2018, Accessed: Jul. 19, 2021. [Online]. Available: https://arxiv.org/abs/1807.05284v3

[52] K. Wang, H. Wang, M. Liu, … X. X.-C. T. on, and undefined 2018, “Survey on person re-identification based on deep learning,” *Wiley Online Library*, vol. 3, no. 4, pp. 219–227, Dec. 2018, doi: 10.1049/trit.2018.1001.

[53] D. Wu *et al.*, “Deep learning-based methods for person re-identification: A comprehensive review,” *Neurocomputing*, vol. 337, pp. 354–371, Apr. 2019, doi: 10.1016/J.NEUCOM.2019.01.079.

[54] I. Fogel and D. Sagi, “Gabor filters as texture discriminator,” *Biological Cybernetics 1989 61:2*, vol. 61, no. 2, pp. 103–113, Jun. 1989, doi: 10.1007/BF00204594.

[55] C. Schmid, “Constructing models for content-based image retrieval,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, doi: 10.1109/CVPR.2001.990922.

[56] B. Prosser, W. S. Zheng, S. Gong, and T. Xiang, “Person re-identification by support vector ranking,” 2010. doi: 10.5244/C.24.21.

[57] W. S. Zheng, S. Gong, and T. Xiang, “Reidentification by relative distance comparison,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 653–668, 2013, doi: 10.1109/TPAMI.2012.138.

[58] A. J. Ma, P. C. Yuen, and J. Li, “Domain Transfer Support Vector Ranking for Person Re-identification without Target Camera Label Information.” pp. 3567–3574, 2013. doi: 10.1109/ICCV.2013.443.

[59] W.-S. Zheng, X. Li, T. Xiang, S. Liao, J. Lai, and S. Gong, “Partial Person Re-Identification.” pp. 4678–4686, 2015.

[60] R. Zhao, W. Ouyang, and X. Wang, “Unsupervised Salience Learning for Person Re-identification.” pp. 3586–3593, 2013. doi: 10.1109/CVPR.2013.460.

[61] R. Zhao, W. Ouyang, and X. Wang, “Person Re-identification by Salience Matching.” pp. 2528–2535, 2013. doi: 10.1109/ICCV.2013.314.

[62] R. Zhao, W. Ouyang, and X. Wang, “Learning Mid-level Filters for Person Re-identification.” pp. 144–151, 2014.

[63] Y. Shen, W. Lin, J. Yan, M. Xu, J. Wu, and J. Wang, “Person Re-Identification With Correspondence Structure Learning.” pp. 3200–3208, 2015.

[64] R. Layne, T. M. Hospedales, and S. Gong, “Towards person identification and re-identification with attributes,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7583 LNCS, no. PART 1, pp. 402–412, 2012, doi: 10.1007/978-3-642-33863-2_40.

[65] H. Jin, X. Wang, S. Liao, and S. Z. Li, “Deep person re-identification with improved embedding and efficient training,” *IEEE International Joint*

*Conference on Biometrics, IJCB 2017*, vol. 2018-January, pp. 261–267, Jan. 2018.

[66] Z. Zheng, L. Zheng, and Y. Yang, "Pedestrian alignment network for large-scale person re-identification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 10, pp. 3037–3045, Oct. 2019, doi: 10.1109/TCSVT.2018.2873599.

[67] Y. Lin *et al.*, "Improving person re-identification by attribute and identity learning," *Pattern Recognition*, vol. 95, pp. 151–161, Nov. 2019, doi: 10.1016/J.PATCOG.2019.06.006.

[68] W. Li, R. Zhao, T. Xiao, and X. Wang, "DeepReID: Deep Filter Pairing Neural Network for Person Re-Identification." pp. 152–159, 2014.

[69] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," *Proceedings - International Conference on Pattern Recognition*, pp. 34–39, Dec. 2014, doi: 10.1109/ICPR.2014.16.

[70] M. Geng, Y. Wang, T. Xiang, and Y. Tian, "Deep Transfer Learning for Person Re-identification," *2018 IEEE 4th International Conference on Multimedia Big Data, BigMM 2018*, Nov. 2016, Accessed: Jul. 20, 2021. [Online]. Available: https://arxiv.org/abs/1611.05244v2

[71] ZhengZhedong, ZhengLiang, and YangYi, "A Discriminatively Learned CNN Embedding for Person Reidentification," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, Dec. 2017, doi: 10.1145/3159171.

[72] X. Qian, Y. Fu, Y.-G. Jiang, T. Xiang, and X. Xue, "Multi-Scale Deep Learning Architectures for Person Re-Identification." pp. 5399–5408, 2017.

[73] W. Li, X. Zhu, and S. Gong, "Person Re-Identification by Deep Joint Learning of Multi-Loss Classification," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, pp. 2194–2200, May 2017,

Accessed: Jul. 20, 2021. [Online]. Available: https://arxiv.org/abs/1705.04724v2

[74] H. Shi *et al.*, "Embedding Deep Metric for Person Re-identification: A Study Against Large Variations," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 732–748, 2016, doi: 10.1007/978-3-319-46448-0_44.

[75] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, "A Siamese Long Short-Term Memory Architecture for Human Re-identification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9911 LNCS, pp. 135–153, 2016, doi: 10.1007/978-3-319-46478-7_9.

[76] H. Liu, J. Feng, M. Qi, J. Jiang, and S. Yan, "End-to-end comparative attention networks for person re-identification," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3492–3506, Jul. 2017, doi: 10.1109/TIP.2017.2700762.

[77] W. Li, X. Zhu, and S. Gong, "Harmonious Attention Network for Person Re-Identification." pp. 2285–2294, 2018.

[78] C. Wang, Q. Zhang, C. Huang, W. Liu, and X. Wang, "Mancs: A Multi-task Attentional Network with Curriculum Sampling for Person Re-identification." pp. 365–381, 2018.

[79] "Google Colaboratory."

[80] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable Person Re-identification: A Benchmark," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1116–1124. doi: 10.1109/ICCV.2015.133.

[81] 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, doi: 10.1109/TPAMI.2009.167.

[82] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *Advances in Neural Information Processing Systems*, vol. 32, Dec. 2019, Accessed: Jun. 28, 2021. [Online]. Available: http://arxiv.org/abs/1912.01703

[83] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Jul. 11, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[84] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009, Accessed: Jul. 11, 2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

[85] K. Zhou and T. Xiang, “Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch,” *arXiv*, Oct. 2019, Accessed: May 02, 2021. [Online]. Available: http://arxiv.org/abs/1910.10093

**TEZ İZİN FORMU** / THESIS PERMISSION FORM

**PROGRAM** / PROGRAM

Sürdürülebilir Çevre ve Enerji Sistemleri / Sustainable Environment and Energy Systems ☐

Siyaset Bilimi ve Uluslararası İlişkiler / Political Science and International Relations ☐

İngilizce Öğretmenliği / English Language Teaching ☐

Elektrik Elektronik Mühendisliği / Electrical and Electronics Engineering ☒

Bilgisayar Mühendisliği / Computer Engineering ☐

Makina Mühendisliği / Mechanical Engineering ☐

**YAZARIN** / AUTHOR

**Soyadı** / Surname : Aksu

**Adı** / Name : Fatih

**Programı** / Program : Electrical and Electronics Engineering

**TEZİN ADI** / TITLE OF THE THESIS (**İngilizce** / English) : Person Re-Identification using Convolutional Neural Networks

**TEZİN TÜRÜ** / DEGREE: **Yüksek Lisans** / Master ☒ **Doktora** / PhD ☐

1. **Tezin tamamı dünya çapında erişime açılacaktır.** / Release the entire work immediately for access worldwide. ☒

2. **Tez <u>iki yıl</u> süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **<u>two years</u>**. * ☐

3. **Tez <u>altı ay</u> süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **<u>six months</u>**. * ☐

**Yazarın imzası** / Author Signature ..... **Tarih** / Date 08/09/2021

**Tez Danışmanı** / Thesis Advisor Full Name: Cem Direkoğlu (Asst. Prof. Dr.)

**Tez Danışmanı İmzası** / Thesis Advisor Signature: .....

**Eş Danışmanı** / Co-Advisor Full Name: –

**Eş Danışmanı İmzası** / Co-Advisor Signature: –

**Program Koordinatörü** / Program Coordinator Full Name: Murat Fahrioğlu (Assoc. Prof. Dr.)

**Program Koordinatörü İmzası** / Program Coordinator Signature: .....