The Influence of Colour on Image Classification and Face Recognition

Comparing the Performance of Convolutional Neural Networks Using Multiple Datasets, Network Architectures, and Image Chromaticities

Master Thesis

Vincent A. Rüegge

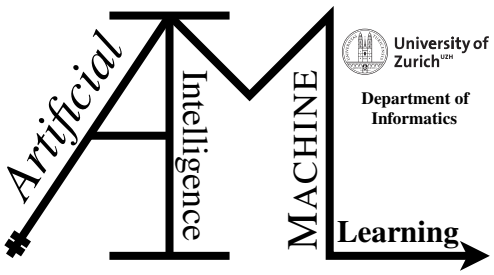
15-700-966

Submitted at

February 28 2021

Thesis Supervisor

Prof. Dr. Manuel Günther


Artificial
Intelligence
MACHINE
Learning
University of Zurich[UZH]
Department of Informatics

Master Thesis

Author: Vincent A. Rüegge, vincent@rueegge.ch

Project period: September 01 2020 - February 28 2021

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

Throughout the writing of this thesis I have received support, assistance, and guidance and I would like to express my appreciation to everyone contributing to my work.

I thank my supervisor, Professor Doctor Manuel Günther, for his continuing support and expertise. Our discussions and your feedback were invaluable to me and deepened my understanding of Deep Learning in both, theory and practice. You allowed me to work freely with an omnipresent guidance and support. Thank you.

I thank my parents for creating an environment that allowed me to focus on my work. Thank you for your counsel and sympathetic ear.

I also thank my sister for proofreading my work. Your feedback was greatly appreciated and your visits on the weekend provided healthy distractions.

Abstract

Recognition of colour is an essential part of our visual system and impressive neural resources are devoted to colour vision. With the emergence of Convolutional Neural Networks (CNNs) from Artificial Intelligence (AI) and AI's goal to simulate vision, the influence of colour on artificial image classification and face recognition gained further attraction in recent years and remains subject to much debate until today. In this thesis, we compare the performance of CNNs using four datasets, six network architectures, and two image chromaticities to investigate on the influence of colour on image classification and face recognition. To extend our analysis beyond the application of pre-existing network architectures, we implement a novel CNN architecture, R-G-B, that learns convolutional filters on the colour channels of an RGB image separately and fuses those layers later in the network architecture, allowing us to investigate on the influence of processing colour within CNNs. Our results suggest that colour does improve performance on both, image classification and face recognition at the expense of computational costs. Furthermore, R-G-B networks are applicable but do not improve performance and are outperformed by regular networks trained on either RGB or greyscale images.

Zusammenfassung

Das Erkennen von Farben ist ein wesentlicher Bestandteil unseres Gehirns und ein Grossteil der neuronalen Ressourcen wird dafür eingesetzt. Mit dem Aufkommen von Convolutional Neural Networks (CNNs) aus dem Bereich Artificial Intelligence (AI) und dem Ziel von AI, Sehen zu simulieren, gewann der Einfluss von Farbe auf künstliche Bild- und Gesichtserkennung in den letzten Jahren an Bedeutung und ist auch heute noch Gegenstand kontroverser Diskussionen. In dieser Arbeit vergleichen wir die Leistungsfähigkeit von CNNs im Hinblick auf den Einfluss von Farbe auf Bild- und Gesichtserkennung. Dazu verwenden wir vier Datensätze, sechs Netzwerk-Architekturen, und zwei Bild-Chromatizitäten. Um unsere Analyse in der Anwendung von bestehenden Netzwerk-Architekturen zu erweitern, implementieren wir eine neue Netzwerk-Architektur, R-G-B, welche die Netzwerk-Filter auf den einzelenen Farbkanälen eines RGB Bildes anwendet und später in der Netzwerk-Architektur zusammenführt. Somit können wir den Einfluss von Farbverarbeitung innerhalb eines Netzwerkes untersuchen. Unsere Resultate zeigen, dass Farbe die Leistungsfähigkeit erhöht, sowohl für Bild- wie auch für Gesichtserkennung, allerdings auf Kosten der Berechnungszeit. Zusätzlich zeigen unsere Resultate, dass R-G-B Netzwerke anwendbar sind, die Leistungsfähigkeit jedoch nicht erhöhen und sowohl von regulären Netzwerken trainiert auf RGB Bildern, wie auch von regulären Netzwerken trainert auf Grauwert Bildern in der Leistung übertroffen werden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual object recognition is one of the most important functions of the brain and is extensively studied in the field of cognitive science. For the visual system to work properly, it must be able to recognise objects of various shapes, textures, and colours (Rossion and Pourtois, 2004). According to Polyak (1957), the perception of colour in the human visual system coevolved with the evolution of brightly coloured plants to facilitate food gathering, and more specifically, the search and recognition of natural objects such as fruit and vegetables. Therefore, the recognition of colour is an essential part of our visual system and impressive neural resources are devoted "[...] to color vision and the perceptual salience of color [...]" (Wurm et al., 1993, p. 899). Interestingly, many people with colour vision deficits state, that they do not experience any difficulties in day-to-day life (Wurm et al., 1993) and are only made aware of their deficits after taking a colour test (Steward and Cole, 1989). Nevertheless, various studies concluded that colour is an important cue in object and face recognition (Rossion and Pourtois, 2004; Wurm et al., 1993; Yip and Sinha, 2002) and Tanaka and Presnell (1999) specifically introduced the term colour diagnosticity to describe the degree to which an object is associated with a specific colour. They showed that the influence of colour on object recognition is relevant for objects with high colour diagnosticity, e.g. bananas compared to objects with low colour diagnosticity, e.g. lamps.

With the emergence of Convolutional Neural Networks (CNNs) from Artificial Intelligence (AI) and AI's goal to simulate vision (Waltz, 1982) as cited in Perrott and Hamey (1991), the influence of colour on artificial image classification and face recognition gained further attraction in recent years and remains subject to much debate until today. CNNs are often used in image classification and face recognition because they are well designed to process images of various chromaticities (Chollet, 2018; Simard et al., 2003) and they achieved state-of-the-art results in both, image classification (Russakovsky et al., 2015) and face recognition (Cao et al., 2018). Although there is existing research investigating on the influence of colour on image classification and face recognition, it is not yet explicitly clear whether colour improves image classification and face recognition performance. Various studies suggest that colour positively influences the performance of image classification and face recognition (Arandjelović, 2012; Buhrmester et al., 2019; Clark et al., 2019; Jones and Abbott, 2006; Jones and Rehg, 2002; Liu et al., 2010; Liu and Liu, 2008; Lu et al., 2018; Rajapakse et al., 2004; Shih and Liu, 2006; Socher et al., 2012; Torres et al., 1999; Yang and Liu, 2008). On the other hand, multiple studies show that networks trained on greyscale images outperform networks trained on images that combine red, green, and blue colour information (RGB), suggesting that colour does not add performance enhancing information (Bui et al., 2016; Sachin et al., 2017; Xie and Richmond, 2018).

In this thesis, we therefore aspire to mitigate the discrepancies about the influence of colour on image classification and face recognition. We identify and address the following gaps currently present in the literature: (1) The use of CNNs in image classification and face recognition without the comparison between RGB and greyscale images. (2) The comparison between RGB and

greyscale images without the use of CNNs. (3) The use of CNNs and the comparison between RGB and greyscale images without the use of multiple purely CNN-based feature extractors. Therefore, we conduct our experiments using multiple purely CNN-based feature extractors and compare the performance of RGB and greyscale images on image classification and face recognition. Additionally, we extend our analysis by proposing a novel CNN architecture, R-G-B, that processes each colour channel in an RGB image separately. Currently, the fusion of the three colour channels in an RGB image is made in the first convolutional layer of a CNN. However, researchers never questioned this behaviour. Due to the nonlinear behaviour of the network, it is unclear whether it might be better to learn convolutional filters on the colour channels separately and fuse those channels later in the network architecture. Thus, our proposed novel CNN architecture allows us to further investigate on the influence of processing colour information within CNNs. We conduct a total of 19 experiments spanning over four datasets, six network architectures, and two image chromaticities. More specifically, we trained and evaluated our networks on *CIFAR-10*, *ImageNet*, *VGGFace2*, and *Labeled Faces in the Wild* data. For our networks we used LeNet, ResNet18, ResNet34, and their R-G-B counterparts. We trained the original architectures on both, RGB and luma transformed greyscale images.

The thesis is structured as follows: In Chapter 2, we highlight the related work regarding the influence of colour on image classification and face recognition. In Chapter 3, we provide an overview over computer vision, specifically image classification and face recognition. Chapter 4 explains key concepts with regard to colour in both, the human visual system and computer vision. In Chapter 5, we explain the origin, evolution, and the building blocks of CNNs. Chapter 6 outlines the four datasets used throughout the thesis and explains how we normalised the data. Chapter 7 explains the architectures of the original networks and our novel R-G-B networks. In Chapter 8, we provide an overview over all the experiments and describe the experimental setup. In Chapter 9, we report the results obtained with each experiment, which we discuss in Chapter 10, highlighting the thesis' limitations, and suggesting directions for future work. Finally, in Chapter 11, we draw conclusions from our work and summarise the key findings.

Chapter 2

Related Work

Regarding image classification, Buhrmester et al. (2019) investigated on the impact of different colour spaces and data augmentation techniques on four distinct datasets. They evaluated the performance on a custom CNN with two convolutional layers with max pooling and three fully connected layers. They found that colour information played an important role in classifying images. Furthermore, they showed that colour information was increasingly relevant in datasets with a high number of classes and that certain classes benefited more from colour information than other classes did. For example, deer, rabbits, foxes, beavers, ships, plain landscapes, and deserts all benefited from colour information. Additionally, they showed that greyscale images transformed using the luma transformation are especially well suited for classes where texture is of great importance. Such classes included e.g. trees, flowers, cats, cars, and people. One possible extension which they mentioned in their study would be to consider state-of-the-art network architectures and compare the performance on various colour spaces.

Clark et al. (2019) investigated seven input formats passed to a ResNet34 and their effects on terrain classification. Among others, they compared RGB and greyscale images and found that colour improved validation accuracy but also led to over-reliance and poor testing accuracy. They suggested to use greyscale images over colour images to avoid overfitting the training data. For their study, they assumed that the network architecture does not have an impact on the relative results between the different input formats. Therefore, they used the ResNet34 architecture for all their experiments. The influence of using multiple architectures thus remains subject to further investigation.

Xie and Richmond (2018) pre-trained an Inception-V3 network on *ImageNet* after converting the images into their greyscale counterparts using the luma transformation. They evaluated their network by classifying greyscale chest X-ray images and found that the pretrained network on greyscale images outperformed the network on colour images in terms of both, speed and accuracy. Their results suggest that colour is not a critical feature in image classification. However, they also found that colour did help to distinguish between certain classes. For example, images belonging to the ice-cream class were predicted more accurately using colour images, whereas images belonging to the pier class were predicted more accurately using greyscale images. The comparison between different CNN architectures and their performance remains subject to further investigation.

Bianco et al. (2017) demonstrated the impact of colour balancing, a procedure used to map device-dependent RGB values into a device-independent colour space, on CNN-based texture classification. Thus, they aimed to correct for different lighting conditions and examined the impact on the accuracy in recognising textures. They used various CNN architectures and showed that the effectiveness of colour balancing is not proven. Additionally, since colour balancing is part of the image preprocessing, the study is more situated in the domain of image processing rather than image classification (Perrott and Hamey, 1991).

Sachin et al. (2017) analysed the effect of different colour spaces on scene classification. They used a pretrained CNN (Places-CNN) in combination with classifiers such as Random Forests and Extra Tree Classifiers to classify scenes into eight different scene categories. They obtained best results using greyscale images and showed that the colour space affected the overall and class-wise accuracy. Because they worked with a relatively small dataset, they chose to use a pretrained network instead of training a CNN from scratch. Additionally, their classifier was not purely CNN-based.

Bui et al. (2016) investigated on the performance of greyscale images and RGB images on image classification. They applied a Convolutional Recursive Neural Network on RGB images and their greyscale converted counterparts. Additionally, they compared their approach to more traditional classifiers such as Random Forests and Support Vector Machines. The results showed that greyscale images led to a higher accuracy compared to RGB images across all classifiers at less computational cost. Their Convolutional Recursive Neural Network classifier outperformed both, the Random Forests and the Support Vector Machines. The effect of a purely CNN-based classifier remains subject to further investigation.

Bo et al. (2013), Socher et al. (2012), Bo et al. (2011), and Lai et al. (2011) experimented with image classification with regard to RGB-D images. RGB-D images are images captured by a depth camera. The camera is capable of providing a corresponding depth map that holds three-dimensional information to the two-dimensional original image. Whereas current image classification applications are limited to two-dimensional images, typically in greyscale, the usage of RGB-D images can take full advantage of colour-coded information and depth channels. Bo et al. (2013) showed superior object recognition results using Linear Support Vector Machines. Socher et al. (2012) introduced a network based on a combination of Convolutional and Recursive Neural Networks and applied it to RGB-D images. They obtained state-of-the-art results and demonstrated the applicability of Convolutional and Recursive Neural Networks to the domain of depth images. Bo et al. (2011) reached an improvement of 10% – 15% in accuracy over the state-of-the-art using Linear Support Vector Machines. Lai et al. (2011) used Linear Support Vector Machines, Gaussian Kernel Support Vector Machines, and Random Forests and demonstrated that combining colour-coded information and depth information substantially improved the quality of the results. Although the usage of RGB-D images is not directly comparable to the usage of RGB images, the results suggest that colour-coded information has an impact on the overall performance.

LeCun et al. (2004) assessed the applicability of Nearest Neighbour methods, Support Vector Machines, and CNNs for recognising greyscale images of toys with invariance to pose, lighting, and surrounding clutter. Thus, they aimed to detect and recognise three-dimensional objects in images primarily from shape information. They found that CNNs outperformed the other techniques. However, the comparison between RGB and greyscale images has to be further addressed.

Jones and Rehg (2002) constructed histogram colour models, operating on the colour of a single pixel, to classify skin and non-skin pixels. Their skin classifier reached high accuracy in detecting images containing naked people which could be used for low-level feature extraction and image indexing and retrieval. Their results suggest that colour can be a powerful cue for detecting people in images. However, the power of colour in general image classification remains unclear. Furthermore, their classifier is not directly comparable to a CNN as it does not extract features on multiple levels.

Regarding face recognition, Lu et al. (2018) proposed a novel colour space consisting of one luminance component and two chrominance components which they tested using CNNs on three datasets. They achieved better performance compared to state-of-the-art colour spaces, suggesting that colour does have an impact on face recognition performance. However, they did not directly compare their novel colour space to the greyscale colour space.

Arandjelović (2012) investigated on the discriminative power of colour-based invariants under

large illumination changes between training and query data. They applied Canonical Correlation Analysis to compute the similarity between sets of colour-based invariants. Their results obtained on a large database with extreme illumination variability suggest that colour may significantly improve greyscale-based matching algorithms. The effect in combination with CNNs has to be further investigated.

Liu et al. (2010) presented a novel face recognition method that extracts features in the colour image discriminant colour space. They derived three colour component images and applied three different image encoding methods to extract features from each component image. The experiments showed the effectiveness of their approach and suggest that colour-coded information is beneficial. However, they did not use CNNs to extract features from the images and did not compare their results to greyscale images.

Liu and Liu (2008) applied an Enhanced Fisher Model to extract features in a novel hybrid colour space. The hybrid colour space combined the R component of the RGB colour space and the chromatic components I and Q of the YIQ colour space. Their results showed improved face recognition performance due to the complementary characteristics of the component images. Although they did not apply CNNs and compare the performance to greyscale images, their results suggest that colour does have discriminatory power.

Yang and Liu (2008) developed a colour image discriminant model to unify the colour image representation and recognition tasks into one framework. Thus, they tackled the current trend of first choosing a colour image representation scheme and then evaluating its effectiveness using a recognition method. The authors argued that such a separate strategy cannot guarantee the chosen colour image representation scheme to be best suited for the subsequent recognition method. Their results demonstrate the effectiveness of the proposed model and suggest that colour-coded information can improve face recognition performance. They did not apply CNNs but reported superior performance of their approach compared to greyscale images.

Jones and Abbott (2006) explored the feature extraction from colour images for face recognition. Therefore, they extended Gabor filters to the hypercomplex (quaternion) domain and quantified the effectiveness of these filters based on an elastic graph implementation extended to colour images. They compared monochromatic and colour images and showed an improvement of 3% - 17% in recognition accuracy over monochromatic images when using complex Gabor filters. They did not apply CNNs to extract features but their results suggest that colour is beneficial in face recognition.

Shih and Liu (2006) presented a colour configuration YQC_r, where Y and Q are from the YIQ colour space and C_r is from the YC_bC_r colour space. The new colour configuration was effective for face recognition using the Enhanced Fisher Linear Discriminant model. Their results suggest that colour is important in face recognition tasks. However, the usage of CNNs as feature extractors and the comparison to greyscale images has to be further investigated.

Rajapakse et al. (2004) used Non Negative Matrix Factorization to compare colour images and greyscale images. Their results showed improved accuracy of colour images compared to greyscale images when facial expressions and illumination variations were present in the data. The impact of using CNNs as feature extractors remains to be investigated.

Torres et al. (1999) used Principal Component Analysis to compare colour images and images with only the luminance information present. They showed that using colour information improved the recognition rate. However, the usage of CNNs remains to be further investigated.

Chapter 3

Computer Vision Compendium

Computer vision research started with the pioneering work of Roberts (1963) who built a computer program capable of displaying a three-dimensional representation of a two-dimensional image. Within computer vision, there are multiple sub-fields, one of which is recognition. According to Szeliski (2010), analysing a scene and recognising all of the objects present in this scene is the most challenging visual task we might ask a computer to perform, to which there are several reasons. The real world is made up of a vast amount of different objects. Furthermore, these objects can appear under various circumstances such as illumination, pose, or occlusion, just to name a few. Additionally, the intra-class variability of an object, e.g. the different breeds of dogs, makes it unlikely to perform an exhaustive matching against a database including every variety of every class. The same challenge holds true for face recognition. Designing a system capable of recognising faces affected by variances of illumination, pose, facial expression, occlusion, etc. is non-trivial. In this thesis, we focus our attention on image classification and face recognition, two sub-fields of computer vision as illustrated in Figure 3.1. In this chapter, we describe the goals and challenges of image classification and face recognition and outline the steps necessary to perform each.

3.1 Image Classification

Image classification can be described as the ability of a computer to recognise patterns and objects present in an image (Albon, 2018). Within computer vision, **object recognition** in general is considered to be one of the most challenging areas (Bui et al., 2016; Grauman and Leibe, 2011; Socher et al., 2012), and is of great interest to research and industry in many application domains. Despite recent success (Russakovsky et al., 2015), image classification remains susceptible to various challenges. For example, objects of the same category can appear in multiple variations, depending on e.g. illumination, pose, camera viewpoint, occlusion, and background clutter which complicate object recognition tasks (Grauman and Leibe, 2011). Furthermore, colour-coded information has been shown to be susceptible to noise, illumination, and the quality of the capturing device (Ebner, 2007), further contributing to the variety of object representations the recognition systems must be able to deal with.

Image classification consists of two stages. A learning stage in which the algorithm builds the classifier by analysing the training data and a classification step in which the classifier is used to make predictions on validation or test data (Han et al., 2011). In image classification with CNNs for example, the image is propagated through each layer of the network where features such as edges, corners, lines, textures, or shapes are extracted in each layer. The CNN then outputs a vector of scores, one for each category present in the dataset. Ideally, the highest score within

the output vector represents the actual label of the corresponding category of the input. However, before training, the CNN is unlikely to do so. To perform at a reasonable level and learn from its predictions, it requires a measure of performance that can be used to optimise its internal parameters, i.e. its weights. Therefore, during training, the CNN uses a loss function to compute a measure such that a higher number represents incorrect predictions and the CNN's confidence of these incorrect predictions, or correct predictions and the CNN's insecurity of these correct predictions. More formally, a measure representing the distance between the predicted label and the actual label. For use cases with more than two classes, the **cross-entropy loss** is often applied as a measure of performance (Wu, 2020). After propagating the input and calculating the cross-entropy loss, the CNN applies **backpropagation** which passes the loss backwards through the network and calculates the gradients (Goodfellow et al., 2016). The gradients indicate for each weight by which amount the loss changes if the weight is increased by a small amount. Thus, based on the gradients, the weights can be adjusted to minimise the loss. The adjustment is performed using e.g. **stochastic gradient descent** (SGD), an optimisation technique that can be described as "[...] showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly." (LeCun et al., 2015, p. 437). This procedure is then repeated over many small subsets of the training data until the average of the loss stops decreasing (Howard and Gugger, 2020; LeCun et al., 2015). The classifier is then applied on validation or test data to complete the second stage of image classification and assess the network's ability to correctly classify previously unseen data.

3.2 Face Recognition

Face recognition is one of the most preferred biometric systems for verification and identification of individuals and dates back as far as 1974 with the development of the first automated face recognition system by Kanade (1974). Contrary to face detection, which can be described as detecting the presence of a human face in an image, face recognition is concerned with identifying a specific individual in a passive, non-intrusive manner (Chokkadi and Bhandary, 2019; Lawrence et al., 1997; Huang et al., 2008). Because of the wide range of application domains, face recognition is one of the most researched areas in computer vision (Oloyede et al., 2020). It can be applied in border security, surveillance, law enforcement, access control, computer graphics, and psychology (Naik, 2014; Oloyede et al., 2020). It has been shown, that face recognition systems perform best under controlled conditions. However, the challenge remains to construct face recognition systems capable of dealing with conditions resembling real-life more closely. Such conditions include varying illumination, pose, facial expression, occlusion, plastic surgery, low image resolution, ageing, hair, facial wear, and motion and are termed *face recogntion in the wild* (Jain and Li, 2011; Oloyede et al., 2020; Syafeeza et al., 2014). Under such conditions, face recognition is according to Taigman et al. "[...] at the forefront of the algorithmic perception revolution." (2014, p. 1).

Typically, a face recognition system consists of four stages. These are face detection, preprocessing (or normalisation or alignment), feature extraction (or representation), and recognition (or matching). However, the exact definition of each stage varies in literature (Jain and Li, 2011; Naik, 2014; Oloyede et al., 2020; Taigman et al., 2014). Since our face recognition experiments are conducted with the ***Bob*** framework (Anjos et al., 2012, 2017; Günther et al., 2012, 2016), we henceforth use the terminology and procedural description used by *Bob*. Figure 3.2 illustrates the face recognition stages for **face verification** and **face identification** respectively.[1]

Although we focus on face verification, we briefly describe both procedures to highlight the differences. In face verification, the system tries to prove whether a person's claimed identity is

[1] https://www.idiap.ch/software/bob/docs/bob/docs/stable/bob/bob.bio.base/doc/struct_bio_rec_sys.html

true. Therefore, face verification is a one-to-one setting. In face identification, the system tries to associate a person's identity with an identity from a set of identities in the system's database. Therefore, face identification is a one-to-many setting (Jain and Li, 2011; Naik, 2014; Syafeeza et al., 2014). Face verification can be compared to image classification as both aim to predict the class/identity of a given image/person. However, regarding Deep Learning, there are differences worth mentioning. Whereas image classification is an end-to-end process in which data is fed into the network to produce a prediction, face verification is a two-step problem. First, the system is trained similarly to an image classification system. However, to assess the generalisability beyond the set of training identities, an intermediate layer, e.g. the second to last layer, of the network is used as a representation on which a threshold on a distance measure is applied to decide whether two images are from the same identity (Schroff et al., 2015). The *Pre-processor* processes the raw biometric data to make it fit for modeling purposes and make recognition easier. One example of pre-processing is to crop the face image and remove the background. The *Feature Extractor* transforms the face image into a numerical representation by extracting the most important features from the image. This is achieved similarly to how CNNs extract features in image classification problems, where features such as edges, corners, lines, textures, or shapes are extracted in each layer. The *Model Database* stores the numerical representation for each person alongside the person's ID. The *Matcher* then compares a new input, a **probe**, to one (for verification) or all (for identification) representations in the *Model Database* and produces a similarity score for each comparison. Finally, the *Decision Maker* decides whether the probe and the representation match (for verification), i.e. whether the similarity score is above or below a pre-defined threshold. For identification, the *Decision Maker* decides which representation best describes the probe. In this thesis for example, we first trained our networks to predict identities on *VGGFace2* data and then tested the performance on *Labeled Faces in the Wild* data by calculating the distance-cosine between the intermediate layer (a vector of size 256) of the probe and the corresponding numerical representation stored in the *Model Database*. CNNs are frequently used for face recognition tasks (Chokkadi and Bhandary, 2019; Jain and Li, 2011), since they provide for partial invariance with regard to translation, rotation, scale, and deformation (Lawrence et al., 1997; Naik, 2014) and according to Oloyede et al. (2020), they have been the most preferred classification technique in recent times. Therefore, we use CNNs to conduct our face verification experiments and investigate on the influence of colour on face recognition.

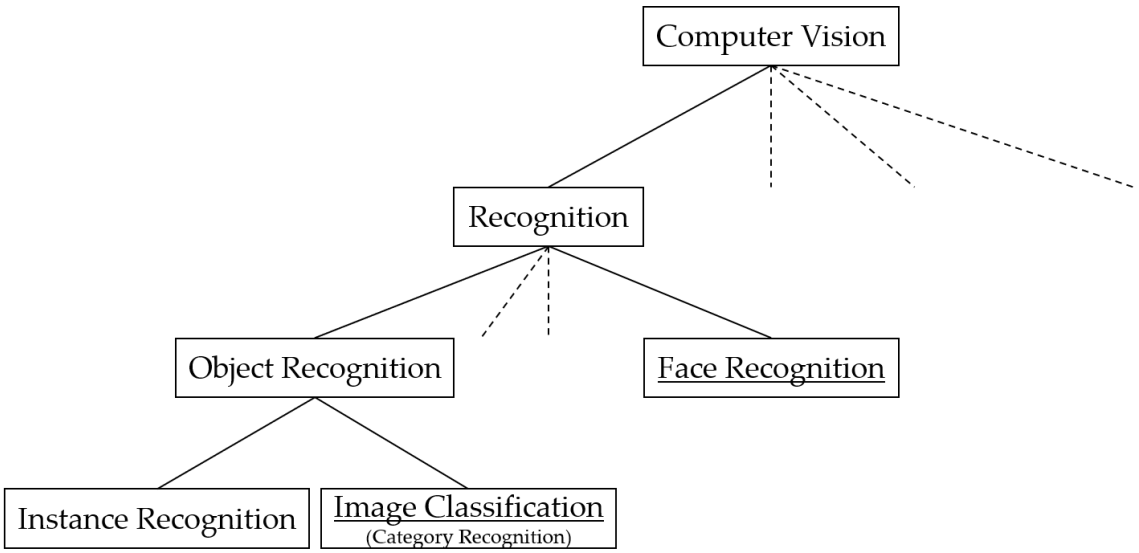
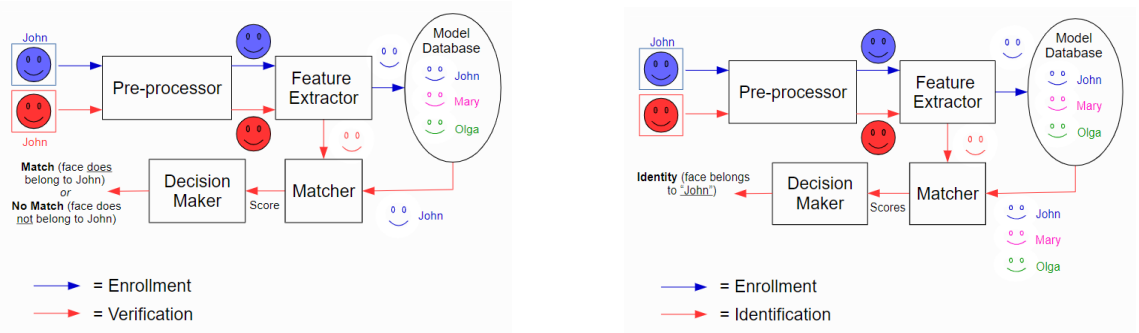


Figure 3.1: Image classification and face recognition as subfields of computer vision



(a) Face verification

(b) Face identification

Figure 3.2: Face verification and identification with *Bob* [1]

Chapter 4

Colour in Image Classification and Face Recognition

The human visual system can distinguish between thousands of colours but only between about two dozen shades of grey (Gonzales and Woods, 2008), which leads us to believe that colour must be an important part in helping us to interpret scenes and recognise objects therein. According to Rossion and Pourtois (2004), Wurm et al. (1993), and Yip and Sinha (2002), colour seems to improve image classification and face recognition performance. However, as highlighted in Chapter 2, the influence of colour on artificial image classification and face recognition is still subject to much debate. In this chapter, we briefly describe the evolution of the human colour vision system and its adaptation in image processing. We then explain the definition of a colour space, a key concept with regard to colour specification. More specifically, we discuss the RGB colour space and show how to transform images from RGB to greyscale using the luma transformation. This is an essential part of our experimental setup. Finally, we outline the current state of colour in image classification and face recognition. Is it perceived as a performance enhancing feature or just a computational burden?

4.1 From the Retina to the Monitor

Compared to the modern human's option to go grocery shopping in the nearest supermarket, our ancestors had to rely far more on their ability to gather food. In order to spot for example ripe red berries and distinguish them from their green and unripe counterparts, humans thus had to develop some form of colour vision. Polyak (1957) argued that the ability of humans to perceive colour thus coevolved with the evolution of brightly coloured plants. Because an exhaustive description of the evolution of the human visual system would go beyond the scope of this thesis, we will focus on some of the core concepts needed to understand colour from both, a biological and a technical point of view. The **retina** is the innermost membrane of the eye and its surface contains two types of receptors: **cones** and **rods**. There are around six to seven million cones and 75 to 150 million rods present on the surface of the retina. The cones are located in the central part of the retina and are highly sensitive to colour. Because each cone is connected to its own nerve end, they can be used to resolve fine details. The rods on the other hand are distributed over a larger area and multiple rods are connected to the same nerve end. They serve to give a general picture of the field of view, are not involved in colour vision, but are sensitive to low levels of illumination. This explains why we can perceive objects as brightly coloured during daylight (relatively high illumination) when the cones are active but only as colourless forms during the night (relatively low illumination) when the rods are active. The cones can be further divided into

three principal categories. These categories roughly represent the colours red (R), green (G), and blue (B), the so-called **primary colours**. It is not surprising then that the most commonly used colour space in image processing is called RGB (Gonzales and Woods, 2008). But what exactly is a colour space?

4.2 Colour Spaces and the Luma Transformation

A **colour space** (or colour model or colour system) is a specification of a coordinate system and a subspace within that system where each colour is represented by a single point. Thus, colours can be specified in a standardised way. The most commonly used colour space oriented towards colour monitors and colour video cameras is **RGB**. Therefore, when we talk about colour in respect to computer vision, the RGB colour space is omnipresent. It is based on a three-dimensional cartesian coordinate system where each of the three primary colours red, green, and blue is located at a corner of the system. The secondary colours cyan, magenta, and yellow are located at the three other corners between the primary colour corners, whereas black is located at the origin and white at the corner furthest away from the origin. Figure 4.1 displays the RGB colour space. As illustrated in the figure, the line between the colour black and the colour white is the so called **greyscale**. Every point on this line is represented with equal RGB values. Note that, for convenience, the colours in the figure have been normalised to be in range [0,1]. Images represented in the RGB colour space therefore consist of three component images (**channels**), one for each primary colour. When displayed on an RGB monitor, the three component images are combined on the screen to produce a composite colour image (Gonzales and Woods, 2008). Greyscale images on the other hand, are composed of a single component image instead of three (Sachin et al., 2017). There exist multiple algorithms to transform an RGB image into its greyscale counterpart. In their study, Kanan and Cottrell (2012) described thirteen of such methods. However, in this study we focus on the **luma transformation** (or luminance transformation) that is designed to match human brightness perception by using a weighted combination of the RGB channels. The luma transformation is described in Equation (4.1). We chose the luma transformation because of its availability in PyTorch and its wide usage in image processing software (Kanan and Cottrell, 2012). The question remains, whether RGB images or their transformed greyscale counterparts lead to superior classification and recognition performance?

$$L = R \times \frac{299}{1000} + G \times \frac{587}{1000} + B \times \frac{114}{1000} \tag{4.1}$$

4.3 The Current State

There are multiple studies suggesting that colour information can be beneficial to artificial image classification and face recognition (Arandjelović, 2012; Buhrmester et al., 2019; Clark et al., 2019; Jones and Abbott, 2006; Jones and Rehg, 2002; Liu et al., 2010; Liu and Liu, 2008; Lu et al., 2018; Rajapakse et al., 2004; Shih and Liu, 2006; Socher et al., 2012; Torres et al., 1999; Yang and Liu, 2008). However, multiple other studies show that colour does not help and that colour only introduces computational costs (Bui et al., 2016; Sachin et al., 2017; Xie and Richmond, 2018). This has partially been attributed to the fact that colour is susceptible to noise, lighting conditions, and the quality of the capturing device. Interestingly, some studies also found the impact of colour on image classification to be dependent on the class to be predicted (Buhrmester et al., 2019; Xie and Richmond, 2018). That is, certain objects of a class are better suited to be associated with a specific colour which in turn can boost classification performance for this class. The degree to which

an object is associated with a specific colour is called **colour diagnosticity**, a term introduced by Tanaka and Presnell (1999). It can be observed that the impact of colour on artificial image classification and face recognition is not entirely clear.

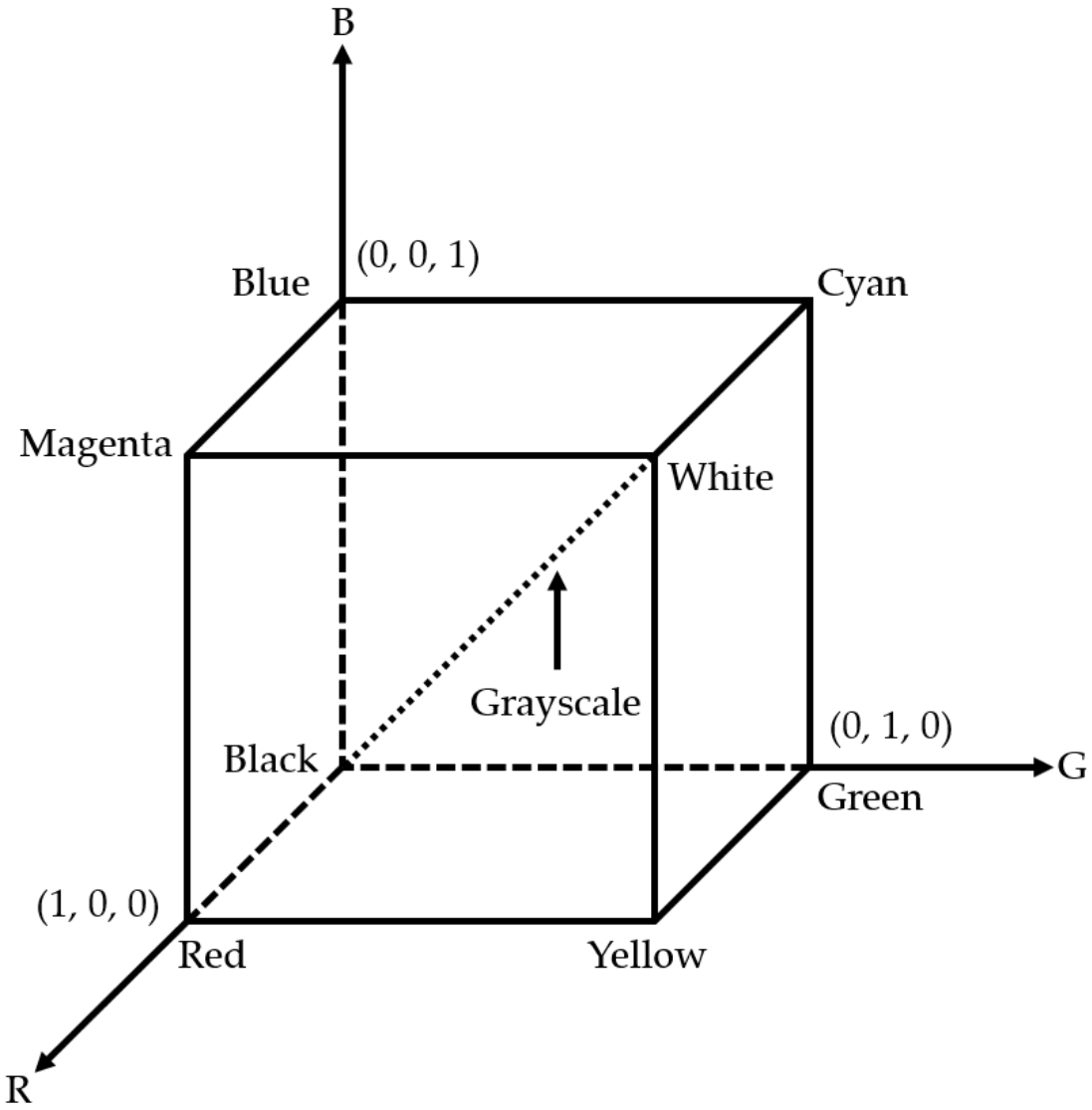


Figure 4.1: RGB Colour Space; Gonzales and Woods (2008)

Chapter 5

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are often used in computer vision applications (Chollet, 2018; Simard et al., 2003) and they are well designed to process images of various chromaticities. They achieved state-of-the-art results in both, image classification (Russakovsky et al., 2015) and face recognition (Cao et al., 2018). In this chapter, we first outline the origin and evolution of CNNs. We then explain typical building blocks of CNNs that are used throughout the thesis. Additionally, we describe batch normalisation, a common regularisation technique applied in residual networks.

5.1 Origin and Evolution

In the 1950s, a small subset of **Artificial Intelligence** (AI) called **Machine Learning** (ML) revolutionised several fields. **Neural Networks** (NN), a subfield of ML, arose and spawned a novel area today called **Deep Learning** (DL). Within DL, CNNs are a form of supervised learning algorithms that can be used to e.g. classify images and recognise faces (Alom et al., 2018). In the early 1960s, the idea of connecting input units to local receptive fields, and the almost simultaneous discovery of locally sensitive, orientation-selective neurons in the cat's visual system (Hubel and Wiesel, 1962) sparked the invention of CNNs. Such local receptive fields enabled neurons to extract visual features such as oriented-edges, end-points, and corners and connect these features to construct more complex patterns. Thus, an important step towards image processing was made. When exactly the first CNN was introduced is not entirely clear. According to Nielsen (2015), CNNs date back as far as the 1970s. Contradictorily, Alom et al. (2018) date the origin back to the 1980s with the first CNN network structure proposed by Fukushima (1988). However, the proposed network was not widely used due to computation power limitations which played a crucial role in the popularity of CNNs as an efficient learning approach for computer vision (Alom et al., 2018). The excitement and optimism regarding NNs in general lasted for the entirety of the decade, but faded in the 1990s due to the rise of other techniques such as Support Vector Machines (Nielsen, 2015). The general opinion was that learning useful, multistage feature extractors with little prior knowledge was infeasible (LeCun et al., 2015). Simard et al. even mentioned that "In 2000, [...] the term "neural networks" in the submission title was negatively correlated with acceptance." (2003, p. 1). For CNNs, the breakthrough came later in the 1990s with the establishment of the modern subject of CNNs proposed by Lecun et al. (1998) (Nielsen, 2015). Their success in the handwritten digit classification problem led various researchers to further improve CNNs and obtain state-of-the-art results in many image classification tasks (Alom et al., 2018). Therefore, CNNs were widely applied since the early 2000s, mainly for detection, segmentation, and recognition of objects and regions in images (LeCun et al., 2015). In 2012, the *ImageNet* competition (Russakovsky et al., 2015) further proved CNNs applicability on large scale

datasets and their superior performance compared to more traditional techniques. The usage of CNNs led to an almost 50% reduction in error rates (Krizhevsky et al., 2012; LeCun et al., 2015).

5.2 Building Blocks

5.2.1 Input

For image classification and face recognition tasks, the **input** to a CNN is typically an image. The image is stored as a multi-dimensional array of size $c \times n \times m$, where c is the number of channels in the image, n is the height, and m is the width of the image. Each cell in the matrix represents the pixel intensity for the corresponding pixel of the image (LeCun et al., 2015). The number of channels depends on the image's chromaticity. An RGB image consists of three channels, one for each primary colour. A greyscale image consists of only one channel representing the pixel intensity along the greyscale. The height and width of the image define the number of pixels present in the image and are typically referred to as the image's **resolution** (Dumoulin and Visin, 2016). For example, an RGB image from the *CIFAR-10* dataset is of size $3 \times 32 \times 32$, whereas its converted greyscale counterpart is of size $1 \times 32 \times 32$. Figure 5.1 illustrates the structure of RGB and greyscale images. For simplicity, we will only consider square images throughout the thesis, that is images with $n = m$.

5.2.2 Convolution

The goal of a CNN is to extract features from an image and classify the image accordingly. A feature is a visually distinctive attribute such as an edge, a corner, a texture, a line, a particular shape, or in combination, an object or a face. However, a feature extractor is needed to transform the raw data, such as pixel values in an image, into a suitable representation that can be used for classification and recognition. Whereas traditional ML techniques require careful engineering and domain expertise to design such a feature extractor, a CNN can be fed with raw data and automatically transform the input in such a way that it can extract features (e.g. horizontal and vertical edges) from it, as illustrated in Figure 5.2. This is a key advantage of CNNs over traditional ML approaches as it eliminates the need for hand-crafted feature extractors (Alom et al., 2018; Lecun et al., 1998; LeCun et al., 2015; LeCun and Bengio, 1995). To extract the features, the CNN transforms the raw input using a **convolution**. A convolution is a mathematical linear operation between matrices (Howard and Gugger, 2020). Hence, the name Convolutional Neural Network. The convolution slides a **kernel**, a little matrix, across an image. Thereby, the convolution multiplies each cell of the kernel with each cell in the corresponding area of the image. This area is called the **local receptive field**. The results are then added together to produce an output, a **feature map** (Goodfellow et al., 2016). Since each cell within a feature map is only connected to the local receptive field, each feature map extracts partial information of the image including its spatial representation. This is important because of the strong two-dimensional local structure present in images. In other words, the pixels that constitute the image are highly correlated with regard to their location (Lecun et al., 1998). A convolution can exploit this knowledge and take into account the input topology (Nielsen, 2015; Simard et al., 2003). Furthermore, because each cell of the kernel is used at every position of the input, a convolution only learns one set of parameters rather than learning a separate set of parameters for every receptive field. This is referred to as **parameter sharing** and results in CNNs being much more efficient than fully-connected operations in terms of memory requirements (Alom et al., 2018; Goodfellow et al., 2016; Krizhevsky et al., 2012; Nielsen, 2015). It also allows the CNN to detect the same feature regardless of its location in the image (Chen et al., 2015), and improves the overall generalisability of the network

(LeCun and Bengio, 1995). Figure 5.3 shows the application of a kernel of size 3×3 across one channel of a 4×4 image (light blue) to produce a feature map of size 2×2 (light green). Notice, the dark blue area in the image represents both, the kernel and the receptive field. In the case where $c > 1$, the convolution needs a separate kernel for each channel. That is, the number of kernels has to be equal to the number of incoming channels. Multiple kernels stacked together are sometimes referred to as a **filter**. Therefore, a filter is a three-dimensional collection of kernels applied to a three-dimensional collection of input channels where the depth (number of kernels and number of channels) is the same. Figure 5.4 illustrates a convolution for $c > 1$. We can see that the number of incoming channels is equal to the number of kernels in each filter, that is $c_1 = c_2 = 3$, and that the number of filters determines the number of feature maps and outgoing channels, $k_{1,2} = f_{1,2} = c_3 = 2$. Furthermore, we can see that each filter, k_1 and k_2, produces its own feature map, f_1 and f_2 respectively. Whereas the number of filters determines the number of outgoing channels, the size of the kernels, the **padding**, and the **stride** determine the height and width of the outgoing channels. Typical kernel sizes are 3×3 and 5×5, although various other sizes can also be applied. However, even kernel sizes are seldom used in practice because they require different amounts of padding on the top/bottom and left/right of the input (Howard and Gugger, 2020). If padding is applied, additional pixels are added around the outside of the input to increase its height and width. Typically, pixels of zeroes are added. Padding ensures that the height and width of the feature map are the same as the height and width of the input. Figure 5.5 illustrates the application of a 3×3 kernel over a 5×5 input with padding added to the input to produce a feature map of size 5×5. The stride defines the distance between two consecutive positions of the kernel. It constitutes a form of subsampling and can also be interpreted as a measure of how much of the output is retained. Figure 5.6 displays the application of a 3×3 kernel over a 5×5 input with a stride of two to produce a feature map of size 2×2. In general, the relationship between the output o, the input i, the kernel k, the padding p, and the stride s can be described using Equation (5.1) (Dumoulin and Visin, 2016). The sequential usage of convolutional layers allows for the extraction of different feature levels. The first convolutional layer extracts low-level features such as edges, corners, textures, and lines. The second convolutional layer extracts higher-level features such as particular arrangements of edges and shapes. The last convolutional layer extracts the highest-level features, e.g. the objects and faces to be classified (LeCun et al., 2015; Ren et al., 2012).

$$o = \lfloor \frac{i + 2p - k}{s} \rfloor + 1 \tag{5.1}$$

5.2.3 Activation Function

The output of a convolution (or if present a batch normalisation), is fed into a non-linearity, or **activation function** before being passed to the next convolution. Thus, the network is able to learn more complex models (Zheng et al., 2014). In Deep Learning, the most common activation function is the rectified linear unit, **ReLU**, due to its superior performance regarding training time (Namatēvs, 2017). Krizhevsky et al. (2012) for example, mentioned their training time to be up to six times faster on the *CIFAR-10* dataset compared to an equivalent network with a hyperbolic tangent (Tanh) activation function. The ReLU activation is described in Equation (5.2) (Howard and Gugger, 2020).

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{5.2}$$

5.2.4 Pooling

According to Namatēvs, "[...] **pooling** ensures that the network focuses on the most important patterns." (2017, p. 43). A pooling layer is typically placed right after the activation function to perform subsampling on the feature map and reduce the feature map's resolution and complexity for subsequent layers (Albawi et al., 2017; Nielsen, 2015). This in turn makes the output more robust to shifts and distortions in the input and further improves computational efficiency (Dumoulin and Visin, 2016; Goodfellow et al., 2016; LeCun and Bengio, 1995; Nielsen, 2015). In some sense, pooling works similarly to a convolution as it subsamples parts of the input to produce a reduced output according to some function. The most common pooling operation is **max pooling** [1], which outputs the maximum value of each subsample. Figure 5.7 illustrates the max pooling operation with a 2×2 filter and a stride of 2. Since both, convolution and pooling rely on the assumption that some function is repeatedly applied to subsets of the input, the relationship between the output o, the input i, the kernel k, and the stride s of a pooling operation can be formulated as the relationship of a convolution without padding. The relationship is described in Equation (5.3) (Dumoulin and Visin, 2016).

$$o = \lfloor \frac{i-k}{s} \rfloor + 1 \tag{5.3}$$

5.2.5 Full-Connection and Output

Another type of layer of a CNN is a **fully-connected** layer. It connects every cell of the output volume of the previous layer to a neuron. To map the three-dimensional output volume to a one-dimensional input volume of the fully-connected layer, the output volume is flattened. That is, all the rows of each feature map are concatenated to produce a one-dimensional string of cells. If, for example, the output volume of the previous convolutional layer consists of three feature maps of size 12×12, the flattened input to the fully-connected layer will consist of $3 \times 12 \times 12 = 432$ cells (Nielsen, 2015). These 432 cells are then connected to e.g. ten **output** neurons of the fully-connected layer to predict ten classes present in the data. It is also possible to stack multiple fully-connected layers and train another classifier or make predictions (Namatēvs, 2017). Essentially, the last layer of a CNN is a traditional NN (Albawi et al., 2017). Figure 5.8 illustrates the three-dimensional output volume being flattened into a one-dimensional string of cells. Each cell is then connected to each output neuron to construct the fully-connected layer and predict ten classes. For simplicity, only two of the flattened cells are connected to each output neuron.

5.3 Batch Normalisation

CNNs are interconnected because a layer's input depends on the output of the previous layers. Thus, a layer's input distribution also depends on changes during training, as the parameters of previous layers are updated. This phenomenon is referred to as **internal covariate shift** and is responsible for slower training times by requiring lower learning rates and careful parameter initialisation. To address this issue, Ioffe and Szegedy (2015) proposed to apply **batch normalisation**, a method to make normalisation part of each layer within the network. Thus, each batch of images being processed by the network is normalised per layer, which subsequently reduces the internal covariate shift and allows to use higher learning rates and be less careful about initialisation. The normalisation is achieved by calculating the average mean μ and variance σ^2 of the

[1] https://cs231n.github.io/convolutional-networks//#pool

activations of a layer and normalising each activation X according to Equation (5.4) to produce the normalised activation X', with ϵ being a constant added to the variance for numerical stability. Since the network might not require every layer to have $\mu = 0$ and $\sigma^2 = 1$, batch normalisation adds two learnable parameters γ and β to scale and shift the normalised activations and allow them to have any mean or variance, independent from the mean and variance of previous layers (Howard and Gugger, 2020; Ioffe and Szegedy, 2015).

$$X' = \gamma \frac{(X - \mu)}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{5.4}$$

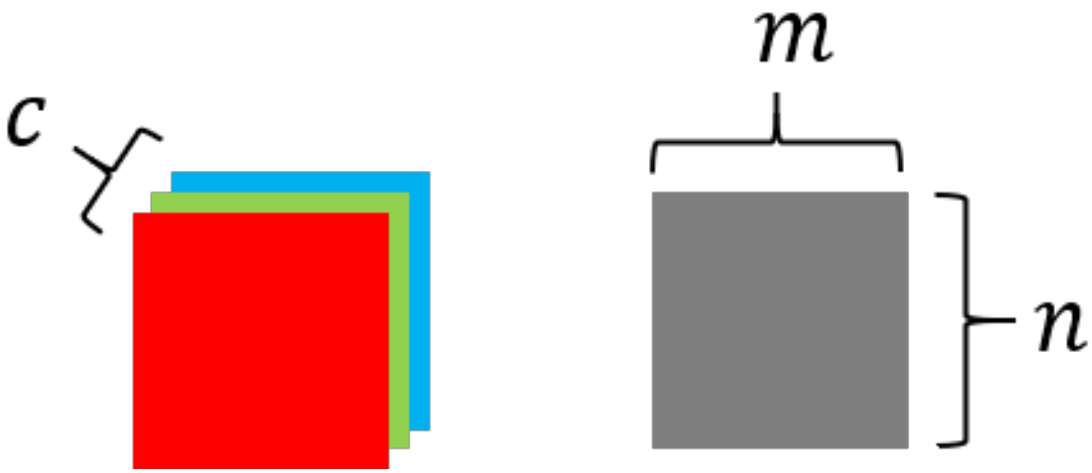
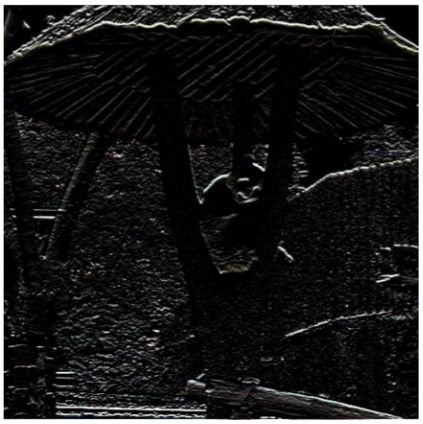


Figure 5.1: Structure of an RGB image (left) and a greyscale image (right). c represents the number of channels, n the height, and m the width of the image

(a) Input image



(b) Horizontal edge

(c) Vertical edge

Figure 5.2: Input image with different filters visualised; Wu (2020)

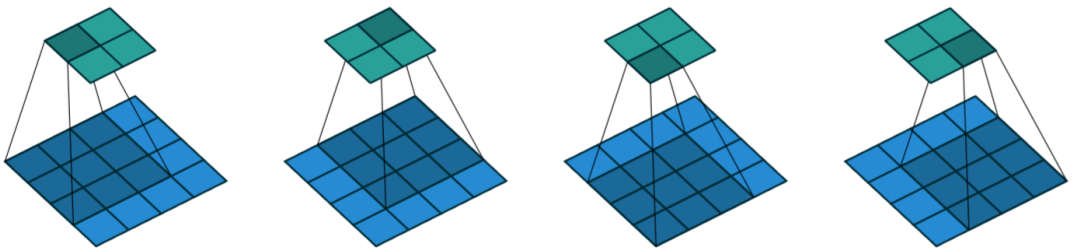


Figure 5.3: Illustration of a convolution operation; Dumoulin and Visin (2016)

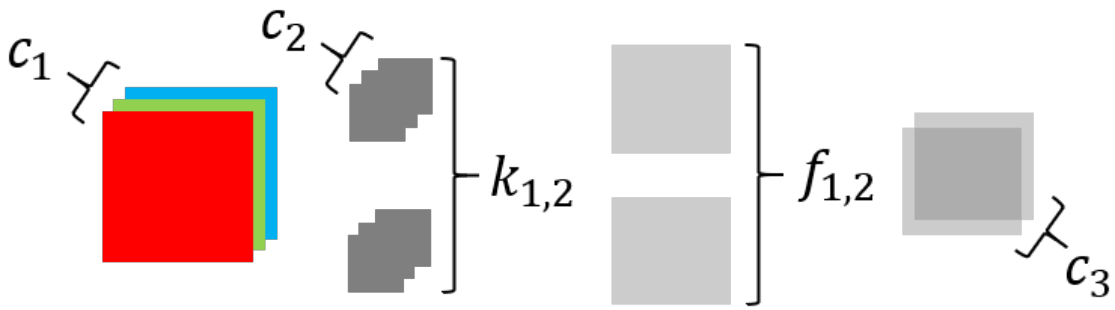


Figure 5.4: Convolution operation over multiple channels. c_x represents the channels, k_x the filters, and f_x the feature maps

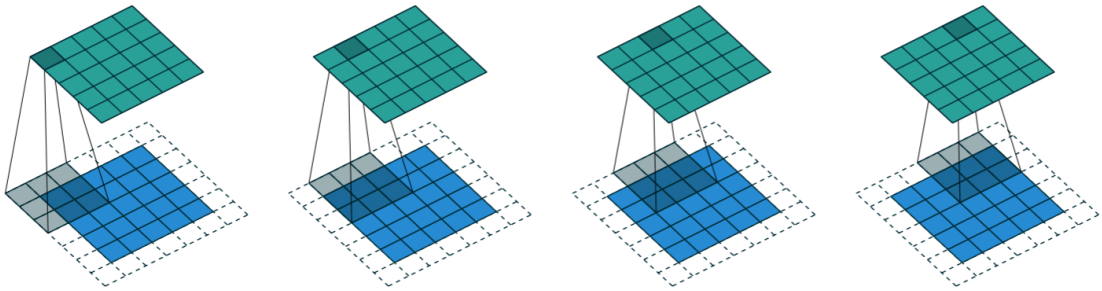


Figure 5.5: Convolution operation with padding; Dumoulin and Visin (2016)

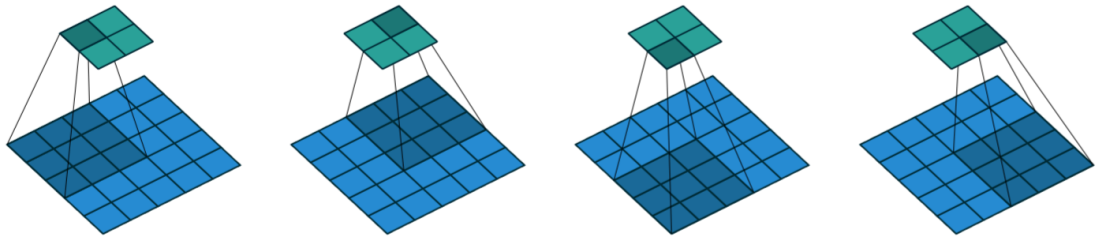


Figure 5.6: Convolution operation with stride; Dumoulin and Visin (2016)

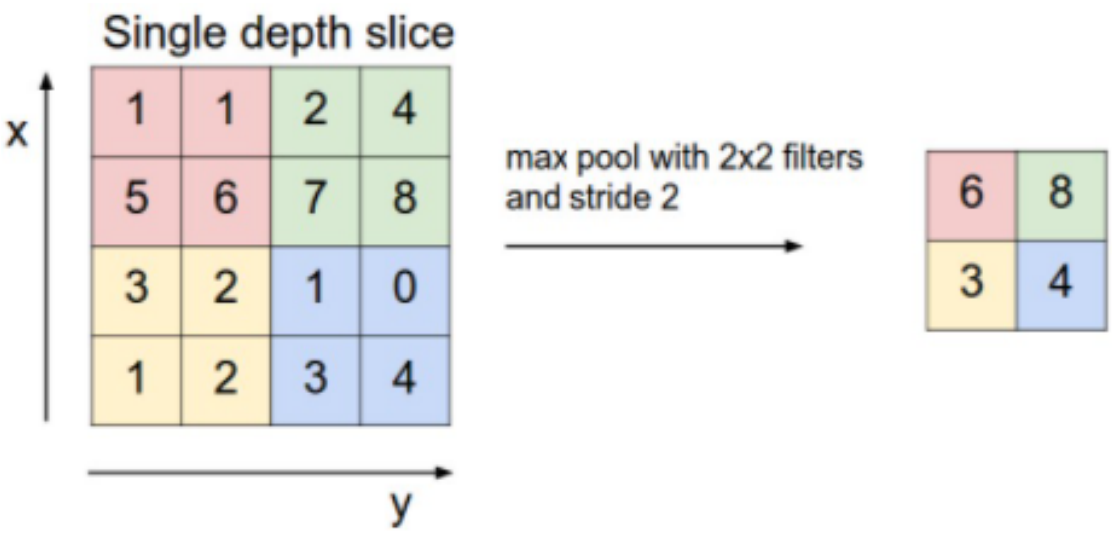


Figure 5.7: Sample max pooling operation[1]

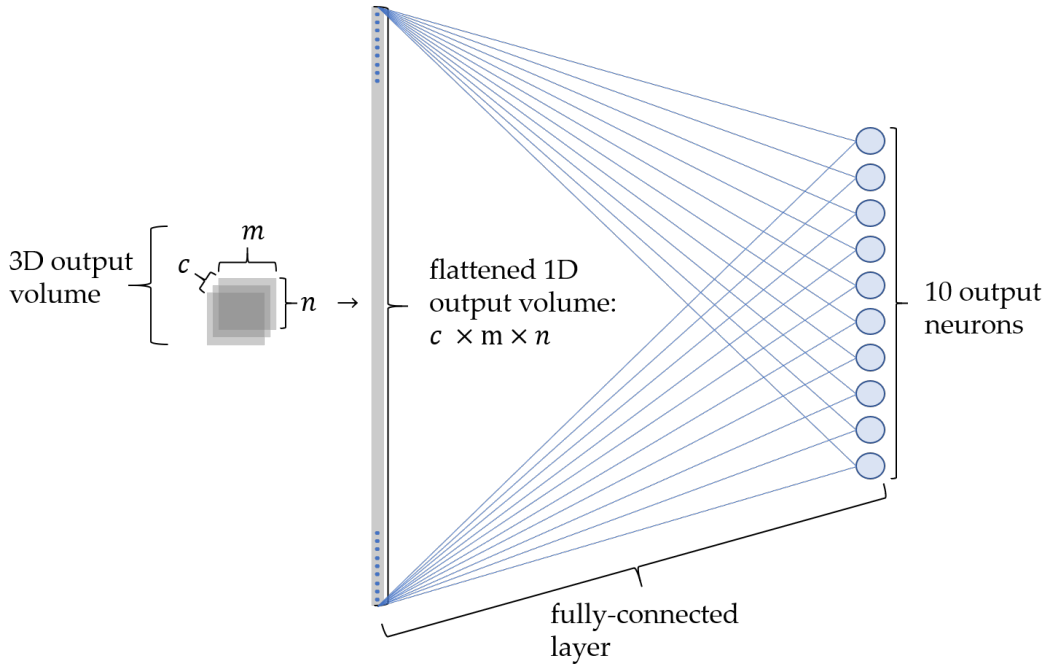


Figure 5.8: Fully-connected layer after a flattened convolutional output volume. c represents the channels, n the height, and m the width of the image

Chapter 6

Datasets

In this chapter, we provide an overview over the different datasets used throughout the thesis and explain how we normalised them for our experiments. Specifically, we outline the size, structure, and image format of each dataset and illustrate sample images. In total, we use four datasets to train, validate, and test our networks. We conduct our first experiments on *CIFAR-10*, a relatively small dataset allowing us to develop the training and validation procedure within reasonable time. We then use *ImageNet* as our second dataset to conduct further image classification experiments on a larger, more challenging dataset. As our third dataset, we use *VGGFace2* to train and validate our networks with regard to face recognition. Finally, we test the networks we trained on *VGGFace2* on our fourth dataset, *Labeled Faces in the Wild*.

6.1 CIFAR-10

CIFAR-10 is a labelled subset of the tiny images dataset (Torralba et al., 2008) and consists of 60'000 colour images of size 32×32. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and is divided into a training set and a testing set containing 50'000 and 10'000 images respectively. The 60'000 images are equally divided into ten classes, leading to 5'000 images per class in the training set and 1'000 images per class in the testing set. The classes are designed to be mutually exclusive. In other words, each image belongs to exactly one class. Figure 6.1a shows ten sample images of the *CIFAR-10* dataset. We chose to use *CIFAR-10* because of its relatively small size and because it has been used extensively in research and many authors show state-of-the-art results with this specific benchmark dataset (Cubuk et al., 2018; DeVries and Taylor, 2017; Dutt et al., 2020; Gastaldi, 2017; Goodfellow et al., 2013; Huang et al., 2017, 2019; Krizhevsky and Hinton, 2010; Phong and Ribeiro, 2020; Real et al., 2019; Wistuba et al., 2019; Yamada et al., 2018; Zagoruyko and Komodakis, 2016; Zoph and Le, 2016).

6.2 ImageNet

ImageNet (Deng et al., 2009) is a large scale database of full resolution images built upon the structure of WordNet (Miller, 1995). The database is divided into subtrees (e.g. plant, flora, plant life) which consist of sets of synonyms (synsets) or classes. At the time of writing, the database consists of almost 15 million images belonging to almost 22'000 synsets.[1] For our experiments, we used the *2012 ImageNet Large Scale Visual Recognition Challenge* (ILSVRC2012) dataset, a subset of the entire *ImageNet* database. Figure 6.1b shows ten sample images of one class of the *ImageNet*

[1] http://www.image-net.org/

dataset. This subset consists of over 1.2 million full-resolution colour images for the training set and 50'000 full-resolution colour images for the validation set. There are 1'000 classes in total with 732-1'300 images per class for the training set and 50 images per class for the validation set. *ImageNet* is a challenging, large-scale dataset that has been used extensively in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) to train state-of-the-art networks. The challenge has been run annually since 2010 and has become a standard benchmark for large scale object recognition and object detection. It shows how the state-of-the-art accuracy improved over the years, showcasing the progress in large scale object recognition and object detection (Russakovsky et al., 2015).

6.3 VGGFace2

A major contributing factor to the success of CNNs in computer vision is the availability of large quantities of training data. For image classification tasks, the introduction of *ImageNet* was instrumental and further advanced research in this area. However, for face recognition tasks, the availability of a comparable dataset was lacking until the introduction of *VGGFace* by Parkhi et al. (2015). The dataset contains 2'622 identities with 1'000 images per identity resulting in a total of 2.6 million images in the entire dataset. Three years later, Cao et al. (2018) introduced ***VGGFace2***, a dataset for recognising faces across pose and age. It contains 9'131 identities with an average of 362.6 images per identity resulting in a total of 3.31 million full-resolution colour images in the entire dataset. Figure 6.1c shows ten sample images of one person of the *VGGFace2* dataset. *VGGFace2* aims to introduce a new large-scale face dataset with large variations in pose, age, illumination, ethnicity, and profession. Additionally, it provides more balanced data with regard to ethnicity and gender compared to *VGGFace*. *VGGFace2* is divided into two splits, one for training having 8'631 classes and one for validation having 500 classes. For our experiments, we only had access to the 8'631 identities in the training set. Thus, for each identity, we randomly split the images into a training set (90% of the data) and a validation set (10% of the data).

6.4 Labeled Faces in the Wild

In order to test the performance of a face recognition system under uncontrolled circumstances, Huang et al. (2008) introduced the ***Labeled Faces in the Wild*** dataset. The dataset aims to provide images spanning the range of conditions present in everyday life. Therefore, the dataset includes variety in factors such as pose, lighting, race, accessories, occlusion, and background. The dataset contains 5'749 identities and a total of 13'233 full-resolution colour images. Figure 6.1d shows ten sample images of the *Labeled Faces in the Wild* dataset. Additionally, the dataset comes in two protocols, **View 1** for algorithm development and **View 2** for performance reporting. Since we developed our networks on *VGGFace2*, we exclusively used View 2 to report the performance of our networks.

6.5 Data Normalisation

In total, we applied three normalisation procedures to our data to change the image size and section, chromaticity, and range of pixel intensity values. Since we only considered square images for our experiments, we cropped the *ImageNet* and *VGGFace2* images to be of size 224×224 as suggested in Krizhevsky et al. (2012), Simonyan and Zisserman (2014), and Cao et al. (2018) respectively. We also cropped the *Labeled Faces in the Wild* images to be of size 224×224 to match

the size of the cropped *VGGFace2* images the network was trained on. Additionally, for *ImageNet*, we randomly cropped and resized[2] the images. For *VGGFace2*, we normalised the images with respect to the center eyes and the center mouth locations. We computed these two points based on the annotated landmarks (left eye coordinates, right eye coordinates, tip of the nose coordinates, left mouth corner coordinates, and right mouth corner coordinates) of each image annotated with the **Multitask Cascaded Convolutional Network** (MTCNN) described in Zhang et al. (2016). We then used *Bob* (Anjos et al., 2012, 2017; Günther et al., 2012, 2016) to crop and align each image to the predefined size such that the two derived landmarks (center eyes and center mouth) would be located at the (x, y)-coordinates (112, 80) and (112, 170) respectively. Figure 6.2a illustrates the provided MTCNN landmarks (blue dots), our derived normalisation landmarks (green dots), and the resulting normalised image with the predefined landmark coordinates (red dots). An excerpt of the normalised images can be found in Figure A.1 in the Appendix. For *Labeled Faces in the Wild*, we used *Bob* to crop and align each image to the predefined size such that the right eye (subject's perspective) would be located at the (x, y)-coordinate (60, 70) and the left eye (subjetc's perspective) at the (x, y)-coordinate (160, 70). Figure 6.2b illustrates an original image and the resulting normalised image with the predefined eye locations (red dots). To convert the images into greyscale, we applied PyTorch's own greyscale conversion function. The function translates a three-channel RGB image into a one-channel greyscale image using the ITU-R 601-2 luma transformation, described in Equation (4.1) and applied in e.g. Buhrmester et al. (2019) and Xie and Richmond (2018). To normalise the images with regard to the range of pixel intensity values, we standardised them to be in range $[-1, 1]$. A normalised range of pixel intensity values often allows a network to distinguish between classes more easily, since it reduces the variability within each class (Bishop, 2006). Common normalisation techniques include **standardisation** and **mean normalisation** with the former showing superior performance (Pal and Sudeep, 2016). Both techniques transform an input X into its normalised counterpart X' using the mean of the training set μ, and the standard deviation of the training set σ. Equations (6.1) and (6.2) illustrate the standardisation and the mean normalisation respectively. Whereas both of these procedures require to compute μ and/or σ, transforming the images into PyTorch Tensors scales them to be in range $[0, 1]$. This allowed us to set μ and σ in Equation (6.1) equal to 0.5 to scale the data in range $[-1, 1]$ and make it centered around 0 without the need to compute μ or σ.

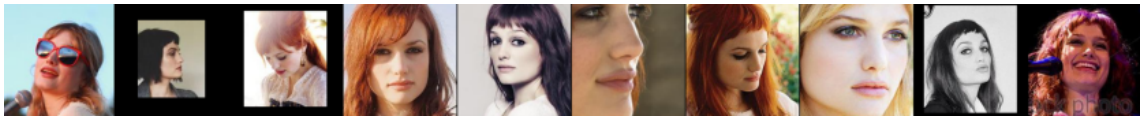
$$X' = \frac{(X - \mu)}{\sigma} \tag{6.1}$$

$$X' = X - \mu \tag{6.2}$$

[2] https://github.com/pytorch/examples/blob/master/imagenet/main.py#L122

(a) *CIFAR-10*

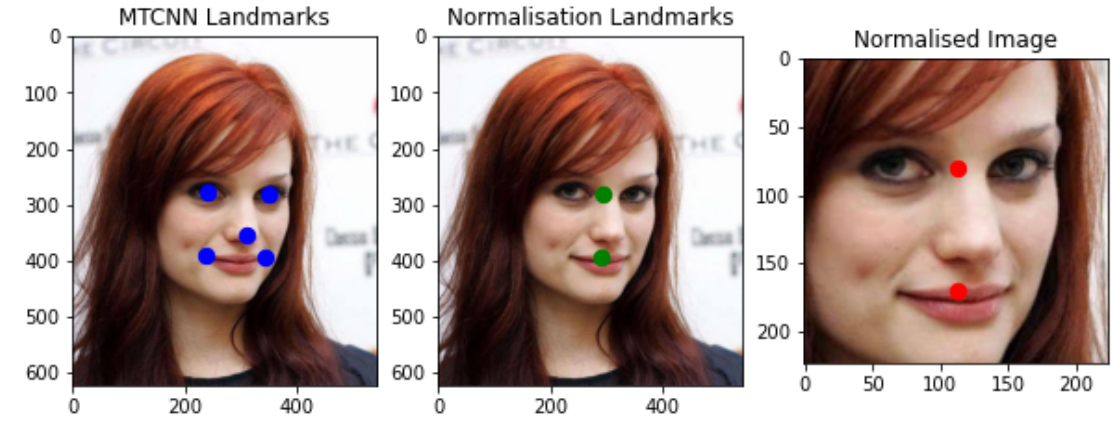
(b) *ImageNet*



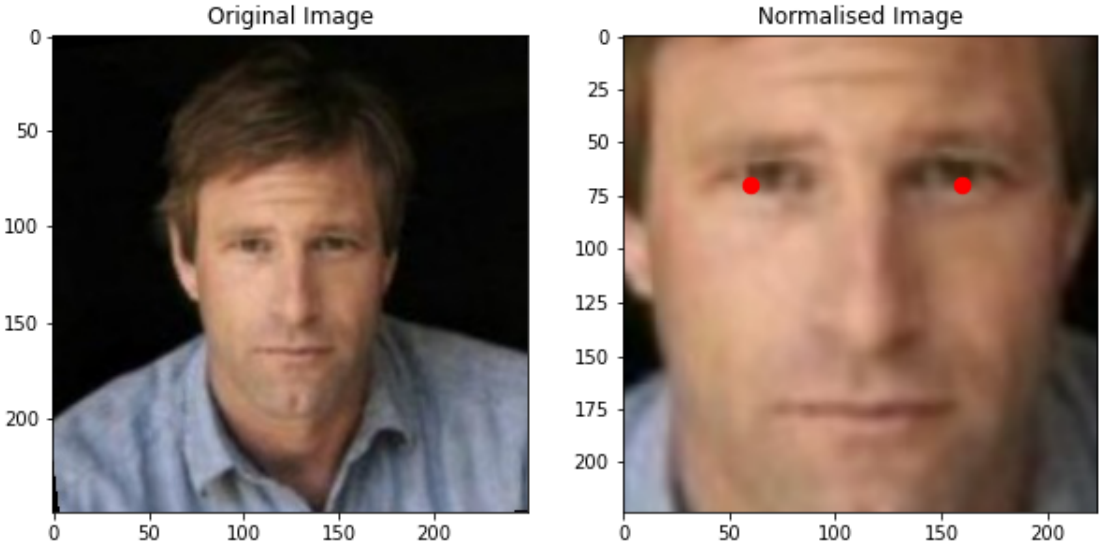
(c) *VGGFace2*

(d) *Labeled Faces in the Wild*

Figure 6.1: Sample images of each dataset



(a) *VGGFace2*



(b) *Labeled Faces in the Wild*

Figure 6.2: Face normalisation

Chapter 7

Networks

In this chapter, we explain the networks and their architectures used throughout the thesis. The chapter is divided into a section detailing pre-existing network architectures (LeNet & Residual Networks) and a section describing our novel network architecture (R-G-B) which processes the RGB channels of an image separately. For each of the pre-existing networks, we outline the original architecture and concepts as well as the changes made to suit the goal of this thesis. For our novel architecture, we first draw a comparison to the concept of depthwise separable convolutions and show how they are different to our architecture. Second, we explain the core concepts of our architecture and the constraints it is bound by. Finally, we illustrate the implementation of our novel architecture on the basis of the LeNet architecture and explain the differences between the original and our novel architecture.

7.1 Pre-existing

7.1.1 LeNet

LeNet is a network introduced by Lecun et al. (1998) and is considered to be one of the first CNNs (Nielsen, 2015). Due to its wide usage and simplicity it is well suited for benchmarking and furthermore, in our case, for developing and testing the training and validation procedure. The original LeNet consists of seven layers, not counting the input and was trained on greyscale images of size 32×32. The seven layers are made up of four convolutional layers, two average pooling layers, and a fully-connected layer, whereas each activation is passed through a scaled hyperbolic tangent activation function. Denoting convolutional layers as C_x, pooling layers as S_x, activations as A, and fully-connected layers as F_x, LeNet can be summarised as: $C_1 \rightarrow A \rightarrow S_1 \rightarrow C_2 \rightarrow A \rightarrow S_2 \rightarrow C_3 \rightarrow A \rightarrow C_4 \rightarrow A \rightarrow F_1$. For our experiments, we used the ReLU activation function instead of the scaled hyperbolic tangent used in the original paper. As shown by Krizhevsky et al. (2012), using ReLU instead of Tanh can result in up to six times faster training without reducing the error rate. Additionally, having the same input size as in the original paper allowed us to implement C_4 as a fully-connected layer as described in Lecun et al. (1998). Thus, our implementation of LeNet can be summarised as: $C_1 \rightarrow A \rightarrow S_1 \rightarrow C_2 \rightarrow A \rightarrow S_2 \rightarrow C_3 \rightarrow A \rightarrow F_1 \rightarrow A \rightarrow F_2$. Figure 7.1 shows the architecture of our implementation of LeNet. F denotes the height (and width) of the kernel, D denotes the number of incoming channels, K represents the number of outgoing channels (the number of filters applied), and $\#\ Params$ represents the number of parameters in the corresponding layer. The three convolutional layers are highlighted in colours. The first row of a convolutional layer denotes the convolution itself and the second row denotes the biases present in this layer.

7.1.2 Residual Networks

As demonstrated by He et al. (2015), Ioffe and Szegedy (2015), Simonyan and Zisserman (2014), and Szegedy et al. (2015), deeper networks usually outperformed shallower networks in the challenging *ImageNet* classification task. Based on these findings, one could assume that improving a network's performance is as simple as adding more layers to the network. Counterintuitively, the study conducted by He et al. (2016) showed that deeper networks resulted in higher training and testing errors compared to shallower networks on both, *CIFAR-10* and *ImageNet*. One reason for this is the **degradation problem**, a phenomenon characterised by a saturating and then decreasing accuracy as the network's depth increases (He et al., 2015). This phenomenon has been partially attributed to optimisation challenges such as **vanishing and shattered gradients** (Balduzzi et al., 2017; Monti et al., 2018) and suggests that the solvers might have difficulties in approximating identity functions (He et al., 2016). To tackle this problem and thus allow networks to go deeper, He et al. (2016) proposed a novel network architecture, the residual network, **ResNet**. Figure 7.2 illustrates the training and validation error achieved on *ImageNet* with both, plain networks (left) and ResNets (right), where the thin lines represent the training error and the bold lines represent the validation error. It can be observed, that going deeper with ResNets does not decrease the network's performance. The main idea is to allow the network to easily learn the identity function if necessary by passing the identity forward through the network in so-called **shortcut connections**, as illustrated in Figure 7.3. Thus, the network will at least be able to learn the identity function and additionally have the potential to learn more complex functions. More formally, the authors denote $H(x)$ as an underlying function to be fitted by a few stacked layers of a network, with x denoting the input to these layers. Based on the hypothesis that multiple nonlinear layers can asymptotically approximate complicated functions (Montufar et al., 2014), one can hypothesise that the same layers can asymptotically approximate the residual functions, i.e., $H(x) - x$ (as long as the input and output are of the same dimension). Thus, instead of expecting the stacked layers to approximate $H(x)$, the authors explicitly let these layers approximate the residual function $F(x) \coloneqq H(x) - x$. The original function $H(x)$ thus becomes $F(x) + x$. The authors hypothesise that although for the network it should be possible to approximate both functions, the ease of learning might be different. In practice, this can be achieved by directly passing the input x forward through the network. The additional forward passing of the input is realised through the shortcut connections. These shortcut connections skip one or more layers, perform identity mappings, and add the input to the output of the stacked layers, right before the activation function. In addition to potential ease of learning, the shortcut connections do neither add extra parameters nor computational complexity to the network. In the case of **ResNet18** for example, the shortcut connections skip two layers, each consisting of a convolution, a batch normalisation, and a ReLU activation. The two skipped layers are often referred to as a **residual block**. Figure 7.3 depicts two layers grouped as one residual block with a parallel shortcut connection. He et al. (2016) define multiple consecutive blocks with the same dimensions as a layer. Since the term layer has already been used to define the ensemble of different operations, we define the consecutive blocks with the same dimensions as **districts**. ResNet18 for example, consists of four districts, each of which consists of two blocks, and each block consists of two layers. Each of those layers consists of a convolution, a batch normalisation, and a ReLU activation. This results in $(4 \times 2 \times 2) + 2 = 18$ layers in total, where the $+2$ references the first layer before the first district, consisting of a convolution and a max pooling, plus the fully-connected layer after the fourth district. It is worth mentioning that there are two types of residual blocks, an **identity block** and a **convolutional block**. The identity blocks are used whenever the input at the beginning of a block and output at the end of a block are of the same dimension. Whenever the dimension changes, a convolutional block is used. In a convolutional block, the shortcut connection does not simply forward the input x, but additionally transforms the dimension of x to match the dimension of the following layer. This is done by a $\mathbf{1 \times 1}$ **convolution** introduced and described in Lin et al. (2013), followed by a batch normal-

isation. Depending on how many residual blocks are contained within each district and how the layers inside the residual blocks are composed, various sizes of ResNets can be constructed. The most common being ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. Figure 7.4 illustrates the first layer and the first two districts of our ResNet18 architecture. F denotes the height (and width) of the kernel, D denotes the number of incoming channels, K represents the number of outgoing channels (the number of filters applied), and $\#\ Params$ represents the number of parameters in the corresponding layer. Two consecutive rows, one operation, mimic a convolutional layer followed by a batch normalisation. The parameters of a batch normalisation are equal to the number of filters K of the corresponding convolutional layer multiplied by two, representing the two learnable parameters γ and β. Two consecutive operations represent a block. Two consecutive blocks represent a district and are accentuated using the same colour (e.g. district one is accentuated in blue, district two in orange). Two consecutive rows framed by dashed lines represent a shortcut connection. The entire architecture can be found in the Appendix, Figure A.2. We implemented the ResNets according to the official implementation provided by PyTorch. [1]

7.2 Novel R-G-B

7.2.1 Inspiration

Currently, the fusion of the three colour channels in an RGB image is made in the first convolutional layer of a CNN. However, researchers never questioned this behaviour. Due to the nonlinear behaviour of the network, it is unclear whether it might be better to learn convolutional filters on the colour channels separately and fuse those channels later in the network architecture. In this thesis, we therefore investigate a novel architecture, based on common CNN architectures, to handle colour channels with CNNs separately. The novel architecture (1) takes RGB images as input, (2) separates the images into their colour channels before the first convolutional layer, (3) propagates the channels sequentially through the CNN, and (4) fuses the convolved channels together before the fully-connected layer at the end of the CNN. Thus, the CNN processes each colour channel individually and derives colour-specific features. This approach can partially be compared to the application of a **depthwise separable convolution**, first introduced by Sifre and Mallat (2014). Depthwise separable convolutions are a form of factorised convolutions that split the convolution operation into two steps. The first step consists of a depthwise convolution, i.e. a convolution performed independently over each channel of the input. The second step consists of a pointwise convolution, i.e. a set of 1×1 convolutions, to transform the number of channels from the first step into the desired number of outgoing channels of the convolution operation. This factorisation reduces computational cost and network size (Chollet, 2017; Howard et al., 2017). Figure 7.5 illustrates a conventional convolution and the two steps composing a depthwise separable convolution. We can see that the output of the depthwise separable convolution is the same as the output of the conventional convolution. This is due to the pointwise convolution that fuses the channels together to produce the desired output volume. However, our approach is different as it does not fuse the layers back together until later in the network. Thus, we keep processing the channels individually throughout the entirety of the network architecture. In essence, every input fed to our network is passed through the network three times, once for each channel. Thus, our approach does not reduce computational cost but rather increases it.

[1] https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html

7.2.2 Concepts & Constraints

Figure 7.6 illustrates the comparison of the original and our novel approach to process RGB images over one convolutional layer and one fully-connected layer using the architecture of LeNet. Since the RGB channels are split and processed separately, we named our approach **R-G-B**. In the original architecture, the network takes an RGB image as input and applies six filters to the image, each with a depth of three. Each of the six filters produces a feature map. The six feature maps are then stacked together to produce the output volume of the first convolutional layer. The output volume is flattened and each cell is connected to a neuron in the fully-connected layer. In the R-G-B architecture, the network takes an RGB image as input and splits the image into its three colour channels. For simplicity, Figure 7.6 illustrates the propagation of the three channels as a parallel process. Each channel is convolved with $\lfloor \frac{k}{3} \rfloor$ filters, where k is equal to the number of filters in the corresponding original architecture. Thus, in the case of R-G-B-LeNet, each channel is convolved with $\frac{6}{3} = 2$ filters. Each of the two filters produces a feature map. The two feature maps are then stacked together to produce the output volume of the first convolution. After every channel produced its output volume, the three volumes are concatenated together to produce the overall output volume. This output volume is flattened and each cell is connected to a neuron in the fully-connected layer. Figure 7.7 shows the architecture of our implementation of R-G-B-LeNet. F denotes the height (and width) of the kernel, D denotes the number of incoming channels, K represents the number of outgoing channels (the number of filters applied), and $\#\ Params$ represents the number of parameters in the corresponding layer. The three convolutional layers are highlighted in colours. The first row of a convolutional layer denotes the convolution itself and the second row denotes the biases present in this layer. This approach allows us to transform the original architecture into its R-G-B counterpart by keeping the number of parameters in the network comparable. The number of parameters can be interpreted as the **network's capacity**, i.e. the network's ability to fit a wide variety of functions. Since we aspire to compare original networks with their R-G-B counterparts, the networks' capacity must be comparable. Thus, training on the same data, both network architectures are equally likely to underfit, overfit, or fit the data (Goodfellow et al., 2016; Wu, 2018). The number of parameters can be calculated using the formula described in Equation (7.1). P refers to the number of parameters in a layer, F refers to the kernel size, D refers to the number of incoming channels, and K refers to the number of outgoing channels (the number of filters applied) in that layer. The $+1$ represents the bias typically associated with each filter (Unnikrishnan et al., 2018).

$$P = ((F \times F \times D) + 1) \times K \tag{7.1}$$

7.2.3 Implementation

We illustrate a pseudocode implementation of the original LeNet and its R-G-B counterpart in Figure 7.8a and Figure 7.8b respectively. Whereas we initialise the architecture once in the original LeNet, we initialise each convolutional layer three times in the R-G-B-LeNet, once for each channel. Thus, R-G-B-LeNet consists of nine convolutional layers, whereas each channel is propagated through three of them. To propagate the channels separately, we transform the input $[batch\_size, 3\ \ channels, height, width]$ into a list of shape $[batch\_size, 1\ \ channel, height, width]$ for each channel in the *split* method. Then, we propagate each channel through the network and save the output in a temporary list $temp$. We concatenate $temp$ in the *concatenate* method to produce $[batch\_size, in\_features, height, width]$, where $in\_features$ is equal to the number of incoming channels expected by the subsequent fully-connected layer. This batch of images is then propagated through the fully-connected layers and returned. Figure 7.9 illustrates the *split* and *concatenate* methods on the basis of a sample batch containing two images of size 2×2. For

simplicity, we did not change the shape of the concatenated channels. In reality, the shape of the concatenated channels would be equal to the shape of the volume being passed to the first fully-connected layer of the network. In the case of LeNet for example, the shape of the concatenated channels would be $[batch\_size, 120, 1, 1]$. With this approach, we are able to construct an R-G-B-LeNet with a total of 61'738 parameters compared to a total of 62'006 parameters present in the original LeNet. The difference in the total number of parameters can be explained using Equation (7.1). In the first convolutional layer of the R-G-B-LeNet we have $((5 \times 5 \times 1) + 1) \times 2$ parameters for each channel, whereas in the original LeNet, we have $((5 \times 5 \times 3) + 1) \times 6$ parameters. Thus, the first convolutional layer in the R-G-B-LeNet has 300 parameters less compared to the original LeNet. Additionally, in the second convolutional layer, for the R-G-B-LeNet we get $((5 \times 5 \times 2) + 1) \times 16$ parameters for each of the three channels. In the original LeNet we get $((5 \times 5 \times 6) + 1) \times 16$ parameters. Whereas the three channels, with depth 2 each, offset the one channel with depth 6, we have 2×16 more biases in the R-G-B-LeNet compared to the original LeNet in the second convolutional layer. Added up, we have -300 parameters from the first convolutional layer and +32 parameters from the second convolutional layer resulting in a difference of 62'006-61'738=268 parameters.

Although the same principles and transformations apply to the R-G-B-ResNet architecture, we had to reengineer parts of the original ResNet architecture to allow for roughly the same number of parameters being present in the original and the R-G-B-ResNet. Since ResNets do not include bias terms in their convolutional layers, Equation (7.1) can be reduced to Equation (7.2). As we keep the size of each kernel $F \times F$ the same, we can either change D or K to manipulate the total number of parameters. Because we concatenate the three colour channels together in the end, we can have either D or K to be a third of the original values in every convolutional layer, but not both at the same time. The only exception being the first convolutional layer as it expects only one incoming channel and applies a third of the filters of the original architecture. However, changing K does not only impact the number of parameters for the current convolutional layer but also the number of parameters of the subsequent convolutional layer because $K_{conv_x} = D_{conv_{x+1}}$. In order to apply our R-G-B approach to the strict architectural pattern of a ResNet and taking $K_{conv_x} = D_{conv_{x+1}}$ into consideration, we reengineered the ResNet architecture to use a different architectural pattern in the first district compared to districts two, three, and four. We introduce the ***CrosshairBlock*** to build district one, taking D_{conv_x} incoming channels and producing $K_{conv_x} = D_{conv_x} \times 3$ outgoing channels in the first layer and $D_{conv_{x+1}} = K_{conv_x}$ incoming channels producing $K_{conv_{x+1}} = D_{conv_x}$ outgoing channels in the second layer. This results in a crosshair-like channel-pattern, hence the name. Figure 7.10 illustrates the architectural design of a *CrosshairBlock*, highlighted in red. For districts two, three, and four we introduce another type of block, the ***ChannelBlock***. It takes D_{conv_x} incoming channels and produces $K_{conv_x} = D_{conv_x} \times 6$ outgoing channels in the first layer of the first block and $D_{conv_{x+1}} = K_{conv_x}$ incoming channels producing $K_{conv_{x+1}} = \frac{K_{conv_x}}{3}$ outgoing channels in the second layer of the first block. Figure 7.10 illustrates an example of a *ChannelBlock*, highlighted in green. Although *ChannelBlocks* can look like *CrosshairBlocks* in the second block, they are only used to build districts two, three, and four, whereas *CrosshairBlocks* are strictly used to build district one. Figure 7.10 also illustrates the first layer and the first two districts of the R-G-B ResNet18 architecture. F denotes the height (and width) of the kernel, D denotes the number of incoming channels, K represents the number of outgoing channels (the number of filters applied), and $\#\ Params$ represents the number of parameters in the corresponding layer. Two consecutive rows, one operation, mimic a convolutional layer followed by a batch normalisation. The parameters of a batch normalisation are equal to the number of filters K of the corresponding convolutional layer multiplied by two, representing the two learnable parameters γ and β. Two consecutive operations represent a block. Tow consecutive blocks represent a district and are accentuated using the same colour (e.g. district one is accentuated in blue, district two in orange). Two consecutive rows framed by dashed lines repre-

sent a shortcut connection. The entire architecture can be found in the Appendix, Figure A.3.

$$P = (F \times F \times D) \times K \tag{7.2}$$

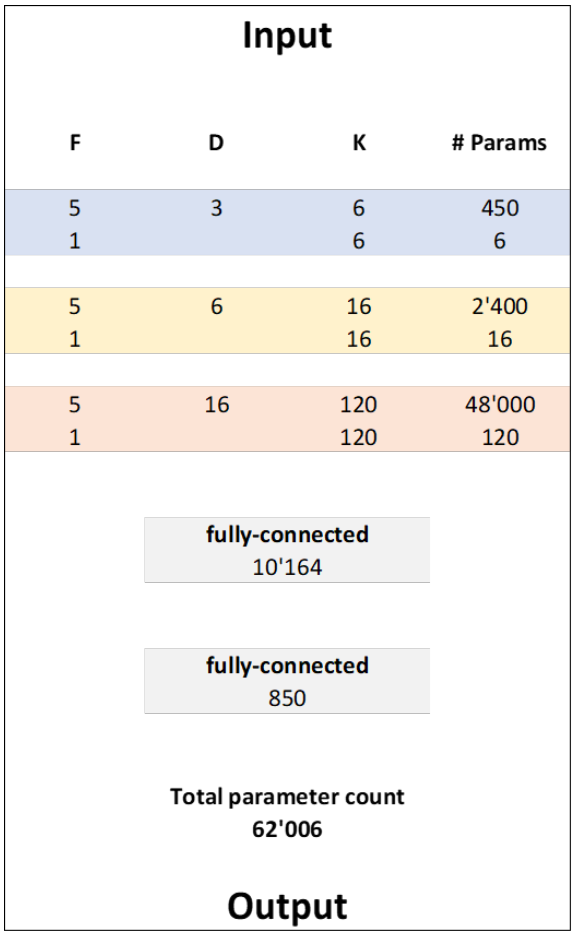


Figure 7.1: LeNet architecture

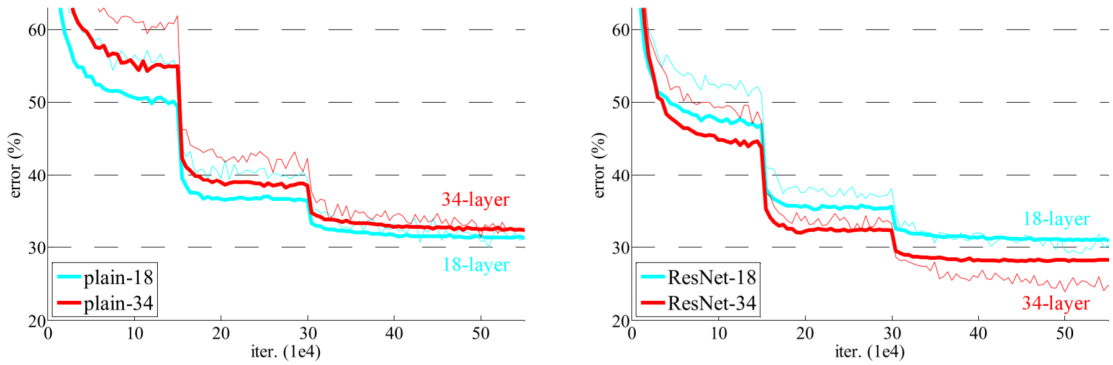


Figure 7.2: Training and validation error for plain (left) and residual (right) networks on *ImageNet*. The thin lines represent the training error and the bold lines represent the validation error; He et al. (2016)

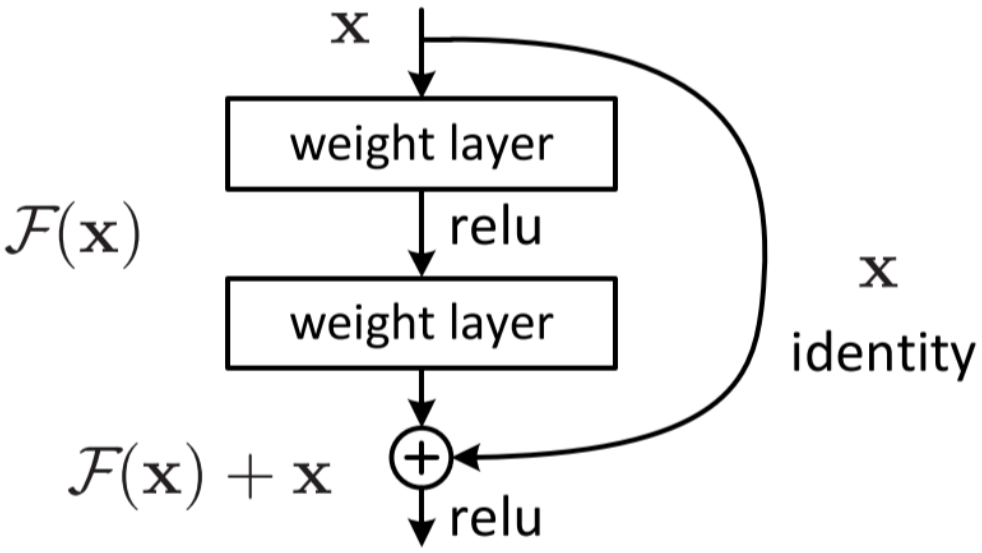
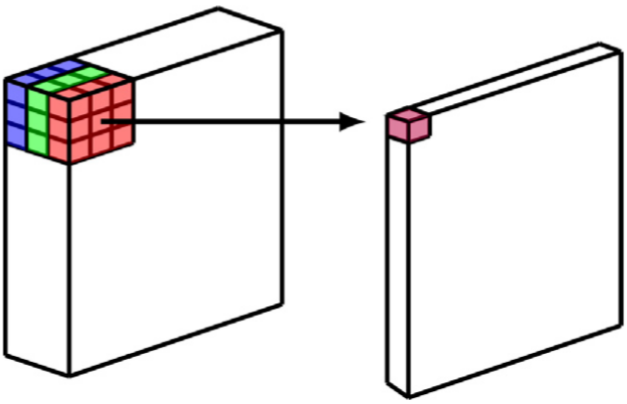


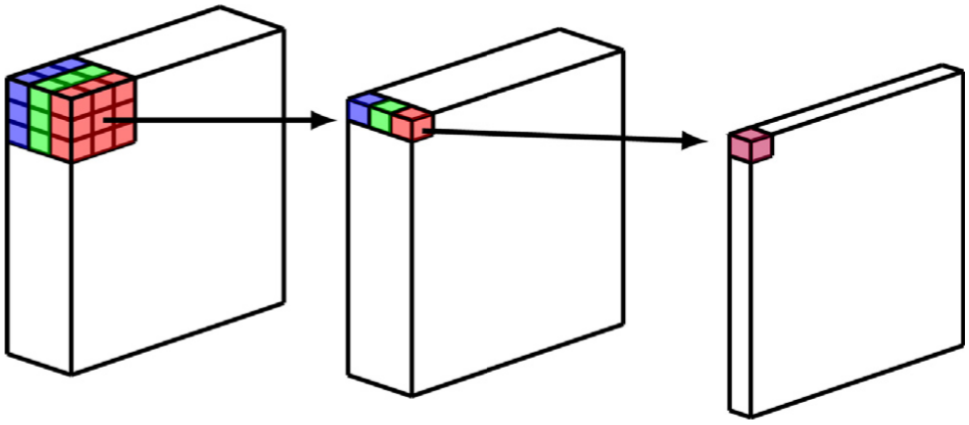
Figure 7.3: Residual block with shortcut connection; He et al. (2016)

Input			
F	**D**	**K**	**# Params**
7	3	64	9'408
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	128	73'728
1	2	128	256
3	128	128	147'456
1	2	128	256
1	64	128	8'192
1	2	128	256
3	128	128	147'456
1	2	128	256
3	128	128	147'456
1	2	128	256

Figure 7.4: First layer and first two districts of the original ResNet18 architecture



(a) Conventional Convolutional Neural Network



(b) Depthwise Separable Convolutional Neural Network

Figure 7.5: Conventional and depthwise separable convolutions; Kamal et al. (2019)

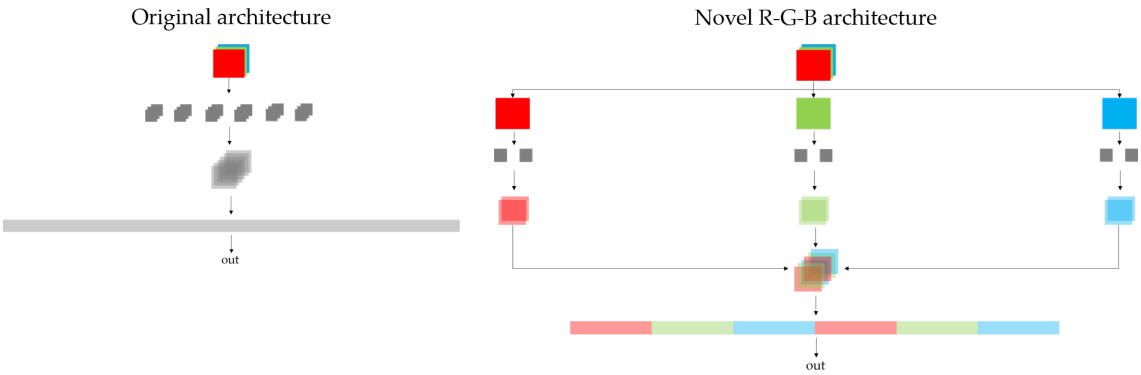


Figure 7.6: Original architecture to process RGB images (left) vs. novel R-G-B architecture to process RGB images (right)

Input

F	D	K	# Params	F	D	K	# Params	F	D	K	# Params
5	1	2	50	5	1	2	50	5	1	2	50
1		2	2	1	0	2	2	1	0	2	2
5	2	16	800	5	2	16	800	5	2	16	800
1		16	16	1	0	16	16	1	0	16	16
5	16	40	16'000	5	16	40	16'000	5	16	40	16'000
1		40	40	1	0	40	40	1	0	40	40

fully-connected
10'164

fully-connected
850

Total parameter count
61'738

Output

Figure 7.7: R-G-B-LeNet architecture

Algorithm 1: LeNet

Input: Batch of Images
Output: Batch of Predictions
Data: Training Data

```
def init():
    initialise architecture

def forward(batch of images):
    propagate batch of images through the network
    return batch of predictions
```

(a) Original LeNet

Algorithm 2: R-G-B-LeNet

Input: Batch of Images
Output: Batch of Predictions
Data: Training Data

```
def init():
    initialise architecture for each channel

def forward(batch of images):
    split(batch of images)
    temp = []
    propagate each channel through its architecture
    append each propagated channel to temp
    concatenate(temp)
    propagate batch of images through the fully-connected layers
    return batch of predictions

def split(batch of images):
    transform batch of images from [batch_size, 3 channels, height, width] into a list with
      [batch_size, 1 channel, height, width] for each channel
    return list

def concatenate(temp):
    transform temp into batch of images with [batch_size, in_features, height, width]
      where in_features is equal to the number of incoming channels of the subsequent
      fully-connected layer
    return batch of images
```

(b) R-G-B-LeNet

Figure 7.8: Original and R-G-B-LeNet pseudocode implementation

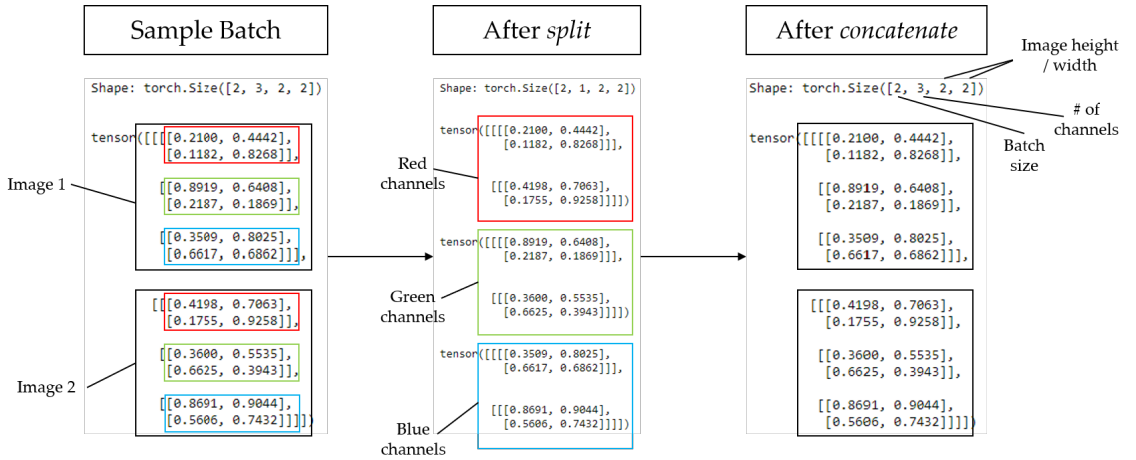


Figure 7.9: Sample R-G-B processing of a batch of images

Input

F	D	K	# Params	F	D	K	# Params	F	D	K	# Params
7	1	21	1'029	7	1	21	1'029	7	1	21	1'029
1	2	21	42	1	2	21	42	1	2	21	42
3	21	63	11'907	3	21	63	11'907	3	21	63	11'907
1	2	63	126	1	2	63	126	1	2	63	126
3	63	21	11'907	3	63	21	11'907	3	63	21	11'907
1	2	21	42	1	2	21	42	1	2	21	42
3	21	63	11'907	3	21	63	11'907	3	21	63	11'907
1	2	63	126	1	2	63	126	1	2	63	126
3	63	21	11'907	3	63	21	11'907	3	63	21	11'907
1	2	21	42	1	2	21	42	1	2	21	42
3	21	126	23'814	3	21	126	23'814	3	21	126	23'814
1	2	126	252	1	2	126	252	1	2	126	252
3	126	42	47'628	3	126	42	47'628	3	126	42	47'628
1	2	42	84	1	2	42	84	1	2	42	84
1	21	42	882	1	21	42	882	1	21	42	882
1	2	42	84	1	2	42	84	1	2	42	84
3	42	126	47'628	3	42	126	47'628	3	42	126	47'628
1	2	126	252	1	2	126	252	1	2	126	252
3	126	42	47'628	3	126	42	47'628	3	126	42	47'628
1	2	42	84	1	2	42	84	1	2	42	84

Figure 7.10: First layer, first two districts, and *Crosshair-* (red) and *Channelblocks* (green) of the R-G-B-ResNet18 architecture visualised including parameter count

Chapter 8

Experimental Setup

In this chapter, we describe the setup of our experiments. We conduct a total of 19 experiments, where each experiment consists of three building blocks: The network, the dataset, and the image chromaticity. We give an overview over all the experiments, describe how we selected the hyper-parameters, how we trained and evaluated the network performance, and finally, how we tested the performance on a test set.

8.1 Overview

- Experiment 1: LeNet on *CIFAR-10* with RGB images
- Experiment 2: LeNet on *CIFAR-10* with greyscale images
- Experiment 3: R-G-B-LeNet on *CIFAR-10* with RGB images
- Experiment 4: ResNet18 on *CIFAR-10* with RGB images
- Experiment 5: ResNet18 on *CIFAR-10* with greyscale images
- Experiment 6: R-G-B-ResNet18 on *CIFAR-10* with RGB images
- Experiment 7: ResNet18 on *ImageNet* with RGB images
- Experiment 8: ResNet18 on *ImageNet* with greyscale images
- Experiment 9: R-G-B-ResNet18 on *ImageNet* with RGB images
- Experiment 10: ResNet34 on *ImageNet* with RGB images
- Experiment 11: ResNet34 on *ImageNet* with greyscale images
- Experiment 12: R-G-B-ResNet34 on *ImageNet* with RGB images
- Experiment 13: ResNet34 pretrained on *ImageNet* on *VGGFace2* with RGB images
- Experiment 14: ResNet34 pretrained on *ImageNet* on *VGGFace2* with greyscale images
- Experiment 15: R-G-B-ResNet34 pretrained on *ImageNet* on *VGGFace2* with RGB images
- Experiment 16: ResNet34 pretrained on *VGGFace2* on *Labeled Faces in the Wild* with RGB images

- Experiment 17: ResNet34 pretrained on *VGGFace2* on *Labeled Faces in the Wild* with greyscale images
- Experiment 18: R-G-B-ResNet34 pretrained on *VGGFace2* on *Labeled Faces in the Wild* with RGB images
- Experiment 19: ResNet34 pretrained on *VGGFace2* on RGB images on *Labeled Faces in the Wild* with greyscale images

8.2 Hyperparameter Selection

According to Bengio, the definition of a **hyperparameter** for a learning algorithm A is defined as: "[...] a variable to be set prior to the actual application of A to the data, one that is not directly selected by the learning algorithm itself." (2012, p. 8). However, choosing the right set of hyperparameters requires years of experience (Smith, 2018) and there is no *one size fits all* approach. There are various techniques and recommendations to ease the selection of hyperparameters. However, most of these techniques are guided towards training state-of-the-art classifiers, which would be beyond the scope of this thesis. Therefore, we chose to adopt a different approach to select the optimal set of hyperparameters by applying a **grid search** on *CIFAR-10* with LeNet. We chose this particular dataset and network configuration to establish a baseline within reasonable training time and overall computational cost. We then reused the resulting optimal set of hyperparameters for all subsequent experiments and compared the results, provided they were based on the same dataset. It is worth mentioning that we use the term optimal relative to our study, since we focused on conducting multiple comparable experiments. In total, we had three hyperparameters to configure. The learning rate, the batch size for the training set, and the optimiser. For our grid search, we chose three learning rates (0.1, 0.01, 0.001), four batch sizes (16, 32, 64, 128), and two optimisers (Adam and SGD). These values also are common in the literature (Bengio, 2012; Ruder, 2016) and allowed us to compare a total of 24 different sets of hyperparameters $(3 \times 4 \times 2)$. We defined each set of hyperparameters as a **run** and trained the network for 100 epochs for each run. We chose the number of epochs to be 100 as Lecun et al. (1998) reached convergence in the original LeNet paper after 10 to 12 epochs on the *MNIST* dataset. Since we used *CIFAR-10* which is more complicated to learn, we argued that our implementation of LeNet will take more epochs to reach a satisfactory level of predictive power. In the end, we selected the three hyperparameters resulting from the best run (highest accuracy on the validation set) out of the 24 we started with. Henceforth, we set the learning rate to 0.01, the batch size for the training set to 64, and used SGD as our optimiser. Additionally, we used the same pseudorandom number generator for all our experiments to make them deterministic. Thus, rerunning the experiments yields exactly the same results since e.g. the weights of a network are always initialised with the same numbers. The complete list of all the hyperparameters and settings to reproduce each experiment can be found in the appendix, Table A.1.

8.3 Network Training and Validation

Our training and validation procedure follows the proposed guidelines described in Goodfellow et al. (2016). Therefore, we divided each epoch in a training and a validation phase. During the training phase, the network processed batches of images, calculated the cross-entropy loss, applied backpropagation to calculate the gradients, and adjusted the weights using SGD to minimise the cross-entropy loss for each batch. In other words, during the training phase, the networks learned on the training data. We shuffled the training data for each training phase to avoid

providing the data in a meaningful order, since not shuffling the data could bias the optimisation algorithm (Ruder, 2016). After the entire training data was processed, the validation phase started in which the network with its adjusted weights was applied on the validation data. During the validation phase, the network again processed batches of images and calculated the loss for each batch but did not apply backpropagation and did not adjust its weights. The idea behind this procedure was to estimate the network's ability to generalise on the validation data after the network learned on the entire training data. That is, estimating whether the patterns it learned were specific to the data it was trained on (Prechelt, 1998). Whereas the network used the loss to differentiate and optimise (Howard and Gugger, 2020), we used the validation accuracy (the number of correctly classified images divided by the total number of images) as the measure to estimate the generalisation, i.e. the performance of our networks (Goodfellow et al., 2016; Smith, 2018). The validation accuracy is relatively intuitive and easy to understand from a human perspective. We used the validation accuracy to select the optimal set of hyperparameters and to compare the performance of our subsequent experiments. For all our experiments conducted on *CIFAR-10* and *ImageNet* after the grid search, we additionally applied **early stopping** to terminate the training process once the performance dropped under a certain threshold. Early stopping can be used to avoid overfitting and decrease computational costs. The idea is to apply a stopping criterion on the validation set which, once broken, terminates the training process (Bengio, 2012; Prechelt, 1998). However, choosing the stopping criterion involves a trade-off between training time and performance. Since we focused on conducting multiple comparable experiments, we also focused on the training time. This allowed us to construct several experiments with different network architectures and datasets. As described in Prechelt (1998), focusing on the training time allows to use a fast stopping criterion. The study suggested to terminate the training process after the performance measure did not improve over five consecutive epochs. Since we used relatively complex network architectures and datasets, we chose to use ten epochs for the experiments conducted on *CIFAR-10* and *ImageNet*. That is, we terminated the entire process once the validation accuracy did not increase over ten consecutive epochs. For our experiments on *VGGFace2*, we did not apply early stopping due to time limitations but ran each experiment for 20 epochs. This allowed us to complete and compare the *VGGFace2* experiments within the time period of the thesis (we estimated the R-G-B experiment on *VGGFace2* to take about 30 days to complete, which would have been beyond the scope of this thesis). However, to enhance the performance on *VGGFace2* data, we used the networks pretrained on *ImageNet*. This allowed us to reach a relatively high performance despite only training the networks for 20 epochs. A similar approach is shown in Günther et al. (2017), where a pretrained network on *ImageNet* reached convergence around three times faster yielding comparable results compared to a network trained from scratch.

8.4 Network Testing

We tested the networks we trained on *VGGFace2* on *Labeled Faces in the Wild* to estimate their generalisability. Therefore, we used ***View 2***, a protocol provided by the *Labeled Faces in the Wild* database and specifically developed for performance reporting. The protocol consists of ten data splits where performance should be reported for each split alongside the mean and standard deviation over the ten splits (Huang et al., 2008). We then used *Bob* (Anjos et al., 2012, 2017; Günther et al., 2012, 2016) to normalise the images, extract a feature vector of size 256 for each image, calculate an overall representation (model) for each person, associate each model with a set of probes and calculate the distance-cosine for each comparison, decide whether the probe and the model belong to the same person, and report the overall performance as the accuracy on each split. Finally, we calculated the mean and standard deviation over the ten data splits. In addition to testing networks pretrained and applied on the same image chromaticities, we tested

the network's performance when pretrained on RGB images and applied on greyscale (3-channel pseudo-colour) images. This allowed us to further investigate how filters learned on colour images behave when applied on greyscale images. In medical studies for example, images often contain only a single channel and datasets are relatively small and scarce. Therefore, practitioners rely on pretrained networks and fine-tune the networks according to their needs. However, pretrained networks are often trained on RGB images, which forces practitioners to comply with the shape and structure of the original data (Talo, 2019; Xie and Richmond, 2018).

Chapter 9

Results

In this chapter, we report the results obtained on each dataset. For *CIFAR-10*, *ImageNet*, and *VGGFace2*, we illustrate the validation accuracy as a function of the number of epochs. Additionally, we show the highest validation accuracy and the corresponding number of epochs required to reach it. We use the highest validation accuracy to compare the performances of all the experiments. For the testing of the pretrained *VGGFace2* networks on *Labeled Faces in the Wild*, we show the Receiver Operating Characteristic (ROC) curve for each experiment to visualise their performance. Furthermore, we report the accuracy for each of the ten splits of *View 2*, alongside the mean accuracy and the standard deviation over the ten splits. We use the mean accuracy to compare the performances of all the experiments conducted on *Labeled Faces in the Wild*.

9.1 CIFAR-10

We evaluated two preexisting network architectures, LeNet and ResNet18, on RGB as well as on greyscale images to investigate on the influence of colour on image classification. Additionally, we transformed both networks into our novel R-G-B architecture to process the RGB colour channels separately. Thus, we could compare the impact of colour processing within CNNs. Figure 9.1 and Table 9.1 illustrate the results obtained from the total of six experiments conducted on *CIFAR-10*. Whereas Figure 9.1 illustrates the validation accuracy as a function of the number of epochs, Table 9.1 shows the highest validation accuracy obtained with each experiment and the corresponding number of epochs required. Regarding LeNet, we observe from Figure 9.1 that the three experiments all reach relatively different accuracies after the first epoch. The RGB experiment reaches the highest accuracy (42.84%) after the first epoch and results in the second highest accuracy (62.43%) in the end. The greyscale experiment reaches the second highest accuracy (35.99%) after the first epoch and results in the lowest accuracy (62.01%) in the end. The R-G-B experiment reaches the lowest accuracy (31.74%) after the first epoch and results in the highest accuracy (62.94%) in the end. Furthermore, Figure 9.1 illustrates that the three experiments perform with marginal differences. The three accuracy-curves start to overlap at around epoch 10 and remain overlapping until the end. This is also reflected in Table 9.1 with the best (R-G-B) and worst (greyscale) performing experiments having a difference of $62.94\% - 62.01\% = 0.93\%$. Additionally, Table 9.1 shows that the number of epochs required to reach the highest accuracy is almost identical with the RGB experiment requiring the least amount of epochs (15) and the greyscale experiment requiring the most (20). Regarding ResNet18, Figure 9.1 shows that the three experiments all reach similar accuracies after the first epoch. The RGB experiment reaches the highest accuracy (55.28%) after the first epoch and results in the highest accuracy (75.87%) in the end. The greyscale experiment reaches the lowest accuracy (52.35%) after the first epoch and results in

the second highest accuracy (73.13%) in the end. The R-G-B experiment reaches the second highest accuracy (54.49%) after the first epoch and results in the lowest accuracy (71.30%) in the end. Additionally, Figure 9.1 shows that the three experiments perform with relatively high margins. The three accuracy-curves stop to overlap at around epoch 5 and become separated more clearly. This is also reflected in Table 9.1 with the best (RGB) and worst (R-G-B) performing experiments having a difference of $75.87\% - 71.30\% = 4.57\%$. Table 9.1 also shows that the number of epochs required to reach the highest accuracy is relatively different between the three experiments with the RGB experiment requiring the most amount of epochs (43) and the greyscale experiment requiring the least (22). Overall, the R-G-B experiment performs the best with LeNet and the worst with ResNet18. The RGB experiment performs better than the greyscale experiment with both, LeNet and ResNet18.

Experiment	**Network**	
	LeNet	**ResNet18**
RGB	62.43 / 15	75.87 / 43
Greyscale	62.01 / 20	73.13 / 22
R-G-B	62.94 / 16	71.30 / 30

Table 9.1: Results on *CIFAR-10*: Validation accuracy (%) / Number of epochs

9.2 ImageNet

We evaluated two preexisting network architectures, ResNet18 and ResNet34, on RGB as well as on greyscale images to investigate on the influence of colour on image classification. Additionally, we transformed both networks into our novel R-G-B architecture to process the RGB colour channels separately. Thus, we could compare the impact of colour processing within CNNs. Figure 9.2 and Table 9.2 illustrate the results obtained from the total of six experiments conducted on *ImageNet*. Whereas Figure 9.2 illustrates the validation accuracy as a function of the number of epochs, Table 9.2 shows the highest validation accuracy obtained with each experiment and the corresponding number of epochs required. Regarding ResNet18, Figure 9.2 shows that the three experiments all reach similar accuracies after the first epoch. The RGB experiment reaches the highest accuracy (12.71%) after the first epoch and results in the highest accuracy (57.50%) in the end. The greyscale experiment reaches the second highest accuracy (11.32%) after the first epoch and results in the second highest accuracy (54.53%) in the end. The R-G-B experiment reaches the lowest accuracy (9.16%) after the first epoch and results in the lowest accuracy (50.92%) in the end. Additionally, Figure 9.2 shows that the three experiments perform with relatively high margins. The three accuracy-curves stop to overlap at around epoch 5 and become separated more clearly. This is also reflected in Table 9.2 with the best (RGB) and worst (R-G-B) performing experiments having a difference of $57.50\% - 50.92\% = 6.58\%$. Table 9.2 also shows that the number of epochs required to reach the highest accuracy is relatively different between the three experiments with the RGB experiment requiring the most amount of epochs (107) and the greyscale experiment requiring the least (74). Regarding ResNet34, Figure 9.2 shows a similar pattern to the experiments conducted with ResNet18. The RGB experiment reaches the highest accuracy (13.27%) after the first epoch and results in the highest accuracy (60.24%) in the end.

The greyscale experiment reaches the second highest accuracy (9.62%) after the first epoch and results in the second highest accuracy (57.76%) in the end. The R-G-B experiment reaches the lowest accuracy (8.62%) after the first epoch and results in the lowest accuracy (54.38%) in the end. Figure 9.2 also shows that the three experiments perform with relatively high margins. The three accuracy-curves stop to overlap at around epoch 5 and become separated more clearly. This is also reflected in Table 9.2 with the best (RGB) and worst (R-G-B) performing experiments having a difference of $60.24\% - 54.38\% = 5.86\%$. Table 9.2 also shows that the number of epochs required to reach the highest accuracy is comparable between the three experiments with the RGB and greyscale experiment requiring the most amount of epochs (98) and the R-G-B experiment requiring the least (90). Overall, the RGB experiment performs the best, followed by the greyscale experiment, and finally the R-G-B experiment with both, ResNet18 and ResNet34.

Experiment	**Network**	
	ResNet18	**ResNet34**
RGB	57.50 / 107	60.24 / 98
Greyscale	54.53 / 74	57.76 / 98
R-G-B	50.92 / 89	54.38 / 90

Table 9.2: Results on *ImageNet*: Validation accuracy (%) / Number of epochs

9.3 VGGFace2

We evaluated one preexisting network architecture, ResNet34, on RGB as well as on greyscale images to investigate on the influence of colour on face recognition. Additionally, we transformed the network into our novel R-G-B architecture to process the RGB colour channels separately. Thus, we could compare the impact of colour processing within CNNs. We pretrained the networks on *ImageNet* and trained them for an additional 20 epochs on *VGGFace2*. Figure 9.3 and Table 9.3 illustrate the results obtained from the total of three experiments conducted on *VGGFace2*. Whereas Figure 9.3 illustrates the validation accuracy as a function of the number of epochs, Table 9.3 shows the highest validation accuracy obtained with each experiment after 20 epochs. Figure 9.3 shows that the three experiments all reach similar accuracies after the first epoch. The RGB experiment reaches the highest accuracy (83.22%) after the first epoch and results in the highest accuracy (93.62%) in the end. The greyscale experiment reaches the second highest accuracy (82.57%) after the first epoch and results in the second highest accuracy (93.34%) in the end. The R-G-B experiment reaches the lowest accuracy (82.46%) after the first epoch and results in the lowest accuracy (92.78%) in the end. Furthermore, Figure 9.3 illustrates that the three experiments performed with marginal differences. The three accuracy-curves overlap at the beginning and remain close until the end. This is also reflected in Table 9.3 with the best (RGB) and worst (R-G-B) performing experiments having a difference of $93.62\% - 92.78\% = 0.84\%$. Overall, the RGB experiment performs the best, followed by the greyscale experiment, and finally the R-G-B experiment.

Experiment	Network
	ResNet34
RGB	93.62
Greyscale	93.34
R-G-B	92.78

Table 9.3: Results on *VGGFace2*: Validation accuracy (%) after 20 epochs

9.4 Labeled Faces in the Wild

To test the performance of the networks pretrained on *VGGFace2*, we evaluated them on *Labeled Faces in the Wild*. Additionally, we tested the network's performance when pretrained on RGB images and applied on greyscale (3-channel pseudo-colour) images. This allowed us to further investigate how filters learned on colour images behave when applied on greyscale images. Figure 9.4 and Table 9.4 illustrate the results obtained from the total of four experiments conducted on *Labeled Faces in the Wild*. Whereas Figure 9.4 illustrates the Receiver Operating Characteristic (ROC) curves, Table 9.4 shows the accuracy for each fold and the corresponding mean and standard deviation over the ten folds. Figure 9.4 shows that the four experiments perform with marginal differences. The RGB experiment performs the best, followed by the greyscale experiment, the R-G-B experiment, and finally the RGB on greyscale experiment. This is also reflected in Table 9.4 with the RGB experiment having the highest mean accuracy (96.26%) μ over the ten folds, followed by the greyscale experiment (96.23%), the R-G-B experiment (95.49%), and finally the RGB on greyscale experiment (95.33%). The standard deviation σ shows a slightly different pattern with the RGB experiment having the lowest standard deviation (0.92), followed by the greyscale experiment (1.11), the RGB on greyscale experiment (1.22), and finally the R-G-B experiment (1.48). Overall, the RGB experiment performs the best, followed by the greyscale experiment, the R-G-B experiment, and finally the RGB on greyscale experiment.

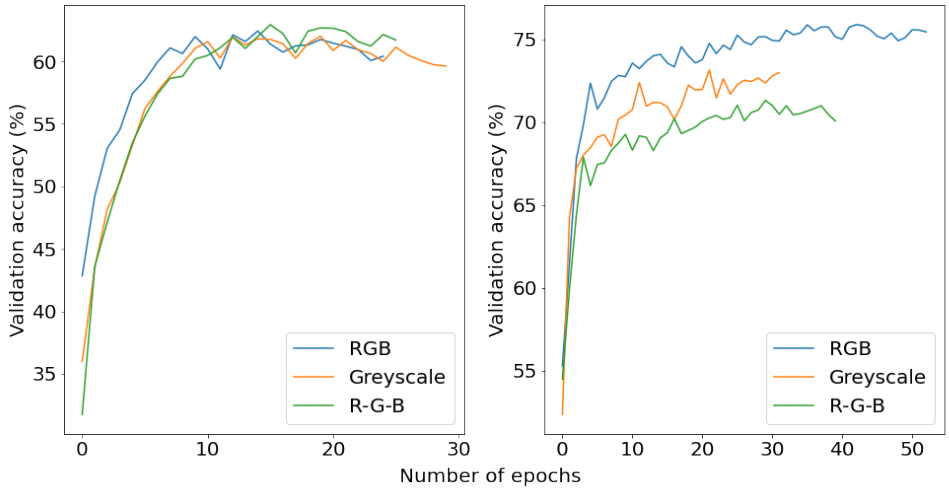


Figure 9.1: Results on *CIFAR-10*: Validation accuracy (%) as a function of the number of epochs for LeNet (left) and ResNet18 (right)

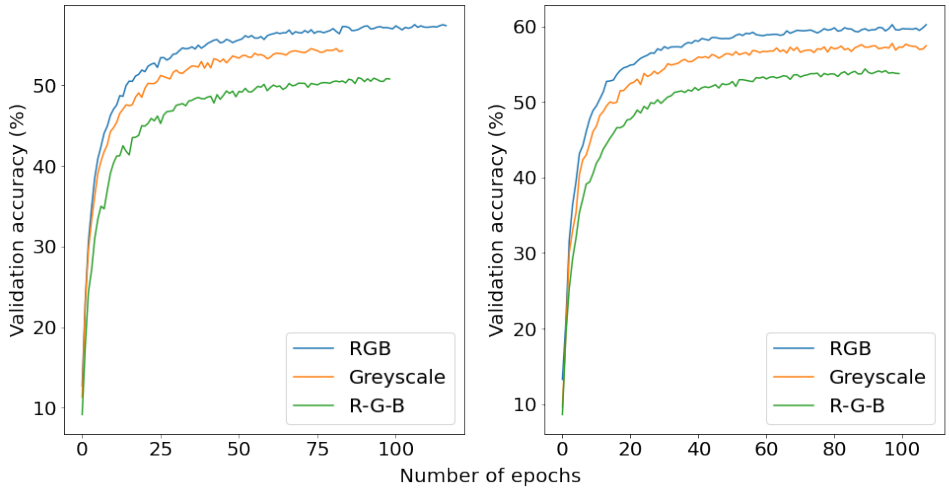


Figure 9.2: Results on *ImageNet*: Validation accuracy (%) as a function of the number of epochs for ResNet18 (left) and ResNet34 (right)

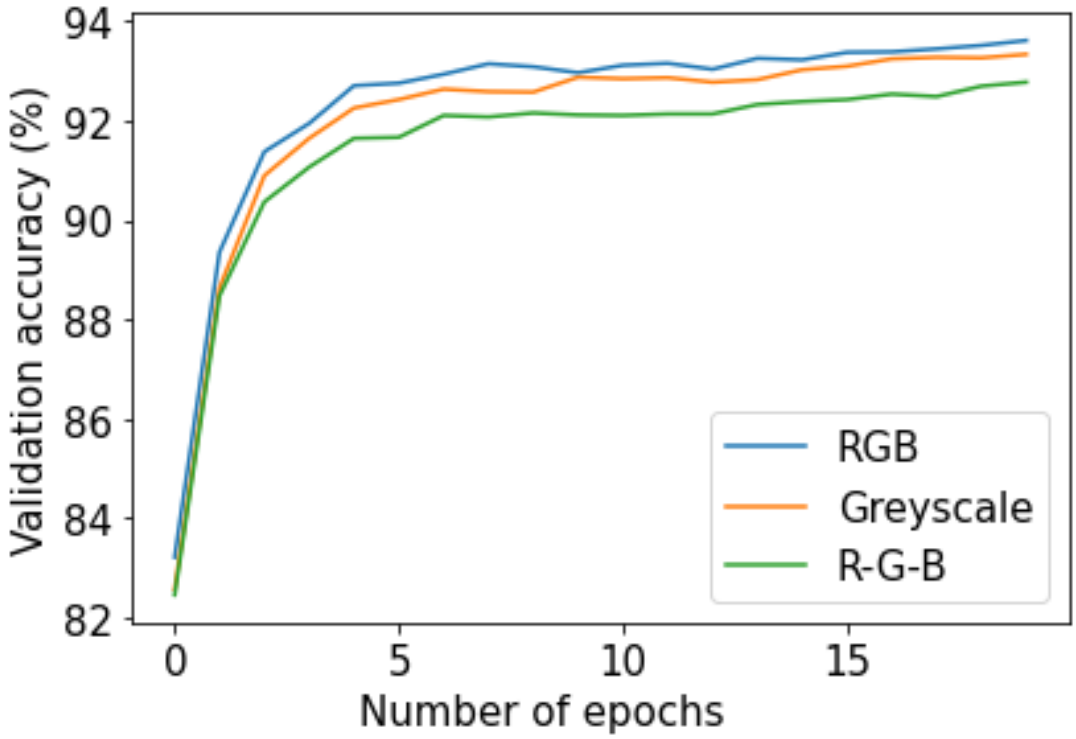


Figure 9.3: Results on *VGGFace2*: Validation accuracy (%) as a function of the number of epochs for ResNet34

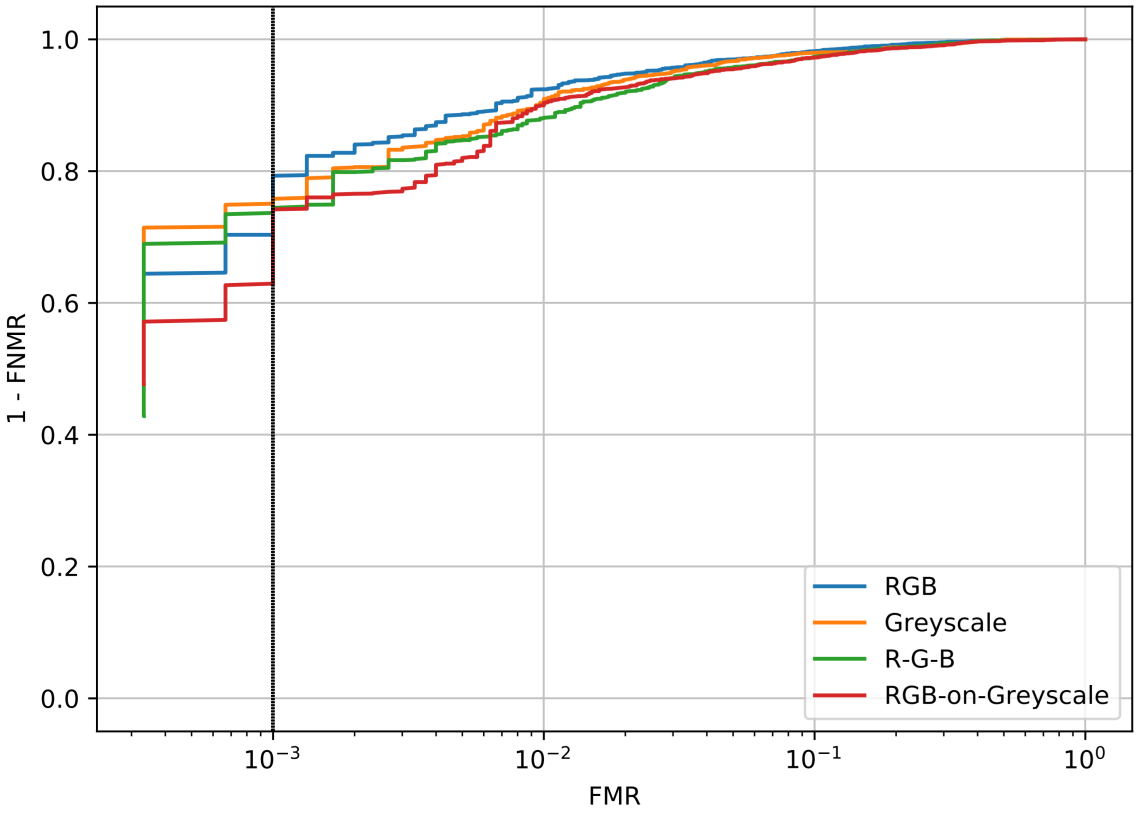


Figure 9.4: Results on *Labeled Faces in the Wild*: ROC curves for ResNet34

Experiment	Fold										μ	σ
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10		
RGB	97.00	96.00	96.33	95.67	95.00	97.00	95.33	95.67	98.33	96.33	96.26	0.92
Greyscale	96.00	96.67	95.33	95.67	94.67	96.00	96.33	95.67	99.00	97.00	96.23	1.11
R-G-B	96.33	97.00	94.00	94.00	92.33	96.33	96.00	96.00	97.33	95.67	95.49	1.48
RGB on Greyscale	95.33	95.33	93.33	94.33	95.00	95.67	95.33	94.67	98.33	96.00	95.33	1.22

Table 9.4: Results on *Labeled Faces in the Wild*: Accuracy (%) per fold and mean accuracy μ and standard deviation σ per experiment

Chapter 10

Discussion

In this thesis, we compared the performance of Convolutional Neural Networks using multiple datasets, network architectures, and image chromaticities to investigate on the influence of colour on image classification and face recognition. More specifically, we trained and evaluated LeNet, ResNet18, and ResNet34 on RGB and greyscale images from *CIFAR-10*, *ImageNet*, *VGGFace2*, and *Labeled Faces in the Wild* data. To extend our analysis beyond the application of pre-existing network architectures, we transformed each network into its R-G-B counterpart, a novel CNN architecture that learns convolutional filters on the colour channels of an RGB image separately and fuses those layers later in the network architecture. First, we address the question whether RGB images improve image classification and face recognition performance compared to greyscale images. Second, we discuss the performance of our novel R-G-B approach and how its performance relates to the influence of colour on image classification and face recognition. Third, we summarise our findings to generalise the influence of colour on image classification and face recognition. Finally, we outline the thesis' limitations and provide recommendations for future work.

On the performance of RGB and greyscale images: Throughout all of our experiments, we find that networks trained on RGB images outperform networks trained on greyscale images. This finding indicates that colour positively influences the performance of Convolutional Neural Networks on both, image classification and face recognition, a finding supported by Arandjelović (2012), Buhrmester et al. (2019), Clark et al. (2019), Jones and Abbott (2006), and Rajapakse et al. (2004). Interestingly, our results contradict the claims of Bui et al. (2016), Sachin et al. (2017), and Xie and Richmond (2018). They showed that classifiers trained on greyscale images led to a higher accuracy at lower computational costs compared to classifiers trained on RGB images. Our results obtained with ResNet18 on *CIFAR-10* and ResNet18 on *ImageNet* showcase that networks trained on greyscale images require significantly less epochs to train compared to networks trained on RGB images. This suggests that greyscale images require lower computational costs. The exceptions being the results obtained with LeNet on *CIFAR-10*, where the greyscale experiment required the most amount of epochs, and ResNet34 on *ImageNet*, where the greyscale experiment required the same amount of epochs as the RGB experiment. Therefore, regarding the influence of colour on image classification, our results suggest that there is a trade-off between accuracy and computational costs, where RGB images favour the former and greyscale images the latter.

On the performance of our novel R-G-B approach: Our results show that our novel R-G-B approach is applicable. However, it yields a worse performance compared to regular architectures applied on either RGB or greyscale images. Interestingly, the R-G-B experiment performed the best with LeNet on *CIFAR-10*. However, we argue that this is a result of the relatively high fluctuation in accuracy and that the results might look different when training the networks on *CIFAR-10* with a less constricting early stopping rule. According to Engilberge et al. (2017) and Rafegas and Vanrell (2018), Convolutional Neural Networks possess colour-sensitive units which are distributed

throughout the entirety of the network architecture. However, both studies show that shallower layers are more colour-sensitive whereas deeper layers are more class-specific. This is interesting as it suggests that the stage at which the layers in our R-G-B approach are fused, has an impact on the performance of the network. Regarding our approach (fusing the layers towards the end of the network), we argue that while this approach allows the network to learn on red, green, and blue channels separately, it removes colour-coded information representing non-primary colours such as orange. Rafegas and Vanrell (2018) illustrated the hue distribution of *ImageNet* which is biased towards the colours orange and blue. They argued that this might be a result of the presence of brownish animals, human skin tones, and sky backgrounds. In such a case, our R-G-B approach is not able to learn from one of the most frequent colours in the data because the colour orange is not represented within the network. Interestingly, our R-G-B approach performed better compared to the RGB on greyscale experiment on *Labeled Faces in the Wild*. Pretraining on RGB images and then applying the pretrained network on greyscale (3-channel pseudo-colour) images is a common procedure in e.g. medical sciences because medical data is relatively scarce and often available as greyscale only. Thus, practitioners use pretrained networks and fine-tune them according to their needs (Talo, 2019; Xie and Richmond, 2018). This finding shows that while our R-G-B approach is outperformed by networks trained on RGB or greyscale images, it is capable of yielding comparable results obtained with established procedures.

On the influence of colour on image classification and face recognition: Our experiments show that there are differences in performance depending on the image chromaticity and the colour processing within the network architecture. Networks trained on RGB images outperform networks trained on greyscale images and both outperform networks trained with our novel R-G-B approach. Our R-G-B approach outperforms networks trained on RGB images that are applied to greyscale images. Overall, our results are very consistent throughout all the experiments and neither one of the experiments shows significantly worse results compared to the others. We want to highlight this finding, because we hope it aspires future research to further investigate on the influence of colour on image classification and face recognition.

On the thesis' limitations and recommendations for future work: To provide future researchers with a starting point, we would like to mention the thesis' limitations and provide recommendations on how to tackle them. First, we chose to use the luma transformation to transform the RGB images into their greyscale counterparts because of its availability in PyTorch and its wide usage in image processing software (Kanan and Cottrell, 2012). However, there are many alternatives on how to transform RGB images into greyscale images. It would be interesting to see how the transformations impact the overall results and whether similar findings to the ones presented in Kanan and Cottrell (2012) could be reported. Additionally, Shih and Liu (2005) as cited in Lu et al. (2018) showed that the component image R of an RGB image outperformed component images from other colour spaces in face recognition. Thus, comparing the performance of the component image R with the performance of greyscale converted images could be interesting as both images are represented by a single channel. Second, all of our experiments are based on a single run and are not averaged over multiple runs. Additionally, we used a pseudorandom number generator (PRNG) to initialise the weights of our networks and make the results deterministic in nature. While this is convenient for testing and reproducing results, the usage of a PRNG has been shown to impact the results (Bird et al., 2020). Thus, to make our results more robust, it would be beneficial to run each experiment multiple times with different PRNGs. Third, to conduct a high number of experiments within the given time period of the thesis, we decided to apply an early stopping rule, terminating the training once the validation accuracy did not improve over ten consecutive epochs. Additionally, we used our networks pretrained on *ImageNet* and trained them for an additional 20 epochs on *VGGFace2* instead of training the networks on *VGGFace2* from scratch. While these decisions allowed us to complete our experiments within reasonable time, it potentially biases the results. Therefore, we suggest to focus on a subset of our experiments and

rerun them with less constraints. For example, training the networks on *VGGFace2* from scratch until they reach convergence. Fourth, we proposed a novel approach of processing RGB images within a CNN. However, there are various ways our approach can be altered. The layers could be fused together earlier in the network architecture and then further propagated as in a traditional network. Thus, the network could learn on the colour channels separately as well as learn non-primary colour representations. Additionally, our R-G-B approach could be applied to other network architectures and/or with other hyperparameters to further investigate its performance.

We believe that the total of 19 experiments conducted using four datasets, six network architectures, and two image chromaticities help to further understand the influence of colour on image classification and face recognition. Additionally, the thesis provides explanations and illustrations of core concepts with regard to Deep Learning with CNNs, further contributing to the ease at which future research can build upon our work. We are convinced that while our work is not exhaustive, it is a step into the right direction to better understand the influence of colour on image classification and face recognition.

Chapter 11

Conclusion

We investigated on the influence of colour on image classification and face recognition by comparing the performance of Convolutional Neural Networks using multiple datasets, network architectures, and image chromaticities. We conducted a total of 19 experiments spanning over four datasets, six network architectures, and two image chromaticities. Additionally, we implemented a novel architecture, R-G-B, that learns convolutional filters on the colour channels of an RGB image separately and fuses those layers later in the network architecture, allowing us to extend our analysis beyond the application of pre-existing network architectures. Throughout all of our experiments, we found networks trained on RGB images performing marginally better compared to networks trained on greyscale images, and both performing better compared to R-G-B networks. This finding suggests that colour positively influences the performance of Convolutional Neural Networks on both, image classification and face recognition. Furthermore, our results show that learning convolutional filters on the colour channels separately and fusing those layers towards the end of the network architecture does not improve the performance of Convolutional Neural Networks. Based on our observations, we argue that fusing the layers earlier in the network architecture might yield different results and that more work is required to quantify colour processing within Convolutional Neural Networks. We suggest various improvements to build upon our work and further investigate on the influence of colour on image classification and face recognition.

Appendix A

Attachments

Link to GitLab: https://gitlab.ifi.uzh.ch/aiml/theses/master-rueegge

Figure A.1: Sample images from one person from the normalised *VGGFace2* dataset

Input

F	D	K	# Params
7	3	64	9'408
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	64	36'864
1	2	64	128
3	64	128	73'728
1	2	128	256
3	128	128	147'456
1	2	128	256
1	64	128	8'192
1	2	128	256
3	128	128	147'456
1	2	128	256
3	128	128	147'456
1	2	128	256
3	128	256	294'912
1	2	256	512
3	256	256	589'824
1	2	256	512
1	128	256	32'768
1	2	256	512
3	256	256	589'824
1	2	256	512
3	256	256	589'824
1	2	256	512
3	256	512	1'179'648
1	2	512	1'024
3	512	512	2'359'296
1	2	512	1'024
1	256	512	131'072
1	2	512	1'024
3	512	512	2'359'296
1	2	512	1'024
3	512	512	2'359'296
1	2	512	1'024
fully-connected			5'130

Total parameter count
11'181'642

Output

Figure A.2: Original ResNet18 architecture including parameter count

Input

F	D	K	# Params	F	D	K	# Params	F	D	K	# Params
7	1	21	1'029	7	1	21	1'029	7	1	21	1'029
1	2	21	42	1	2	21	42	1	2	21	42
3	21	63	11'907	3	21	63	11'907	3	21	63	11'907
1	2	63	126	1	2	63	126	1	2	63	126
3	63	21	11'907	3	63	21	11'907	3	63	21	11'907
1	2	21	42	1	2	21	42	1	2	21	42
3	21	63	11'907	3	21	63	11'907	3	21	63	11'907
1	2	63	126	1	2	63	126	1	2	63	126
3	63	21	11'907	3	63	21	11'907	3	63	21	11'907
1	2	21	42	1	2	21	42	1	2	21	42
3	21	126	23'814	3	21	126	23'814	3	21	126	23'814
1	2	126	252	1	2	126	252	1	2	126	252
3	126	42	47'628	3	126	42	47'628	3	126	42	47'628
1	2	42	84	1	2	42	84	1	2	42	84
1	21	42	882	1	21	42	882	1	21	42	882
1	2	42	84	1	2	42	84	1	2	42	84
3	42	126	47'628	3	42	126	47'628	3	42	126	47'628
1	2	126	252	1	2	126	252	1	2	126	252
3	126	42	47'628	3	126	42	47'628	3	126	42	47'628
1	2	42	84	1	2	42	84	1	2	42	84
3	42	255	96'390	3	42	255	96'390	3	42	255	96'390
1	2	255	510	1	2	255	510	1	2	255	510
3	255	85	195'075	3	255	85	195'075	3	255	85	195'075
1	2	85	170	1	2	85	170	1	2	85	170
1	42	85	3'570	1	42	85	3'570	1	42	85	3'570
1	2	85	170	1	2	85	170	1	2	85	170
3	85	255	195'075	3	85	255	195'075	3	85	255	195'075
1	2	255	510	1	2	255	510	1	2	255	510
3	255	85	195'075	3	255	85	195'075	3	255	85	195'075
1	2	85	170	1	2	85	170	1	2	85	170
3	85	510	390'150	3	85	510	390'150	3	85	510	390'150
1	2	510	1'020	1	2	510	1'020	1	2	510	1'020
3	510	170	780'300	3	510	170	780'300	3	510	170	780'300
1	2	170	340	1	2	170	340	1	2	170	340
1	85	170	14'450	1	85	170	14'450	1	85	170	14'450
1	2	170	340	1	2	170	340	1	2	170	340
3	170	510	780'300	3	170	510	780'300	3	170	510	780'300
1	2	510	1'020	1	2	510	1'020	1	2	510	1'020
3	510	170	780'300	3	510	170	780'300	3	510	170	780'300
1	2	170	340	1	2	170	340	1	2	170	340

fully-connected
5'110

Total parameter count
10'963'048

Output

Figure A.3: R-G-B-ResNet18 architecture inlcuding parameter count

Experiment	Network	Data	Chromaticity	Seed	LR	BS	Shuffle	#W	#GPUs	Optim	ES
#1	LeNet	CIFAR-10	RGB	42	0.01	64	True	1	1	SGD	10 epochs
#2	LeNet	CIFAR-10	Greyscale	42	0.01	64	True	1	1	SGD	10 epochs
#3	R-G-B-LeNet	CIFAR-10	RGB	42	0.01	64	True	1	1	SGD	10 epochs
#4	ResNet18	CIFAR-10	RGB	42	0.01	64	True	1	1	SGD	10 epochs
#5	ResNet18	CIFAR-10	Greyscale	42	0.01	64	True	1	1	SGD	10 epochs
#6	R-G-B-ResNet18	CIFAR-10	RGB	42	0.01	64	True	1	1	SGD	10 epochs
#7	ResNet18	ImageNet	RGB	42	0.01	64	True	8	4	SGD	10 epochs
#8	ResNet18	ImageNet	Greyscale	42	0.01	64	True	8	2	SGD	10 epochs
#9	R-G-B-ResNet18	ImageNet	RGB	42	0.01	64	True	8	2	SGD	10 epochs
#10	ResNet34	ImageNet	RGB	42	0.01	64	True	8	2	SGD	10 epochs
#11	ResNet34	ImageNet	Greyscale	42	0.01	64	True	8	2	SGD	10 epochs
#12	R-G-B-ResNet34	ImageNet	RGB	42	0.01	64	True	8	2	SGD	10 epochs
#13	ResNet34	VGGFace2	RGB	42	0.01	64	True	8	2	SGD	-
#14	ResNet34	VGGFace2	Greyscale	42	0.01	64	True	8	2	SGD	-
#15	R-G-B-ResNet34	VGGFace2	RGB	42	0.01	64	True	8	2	SGD	-
#16	ResNet34	LFW	RGB	-	-	1	-	-	-	-	-
#17	ResNet34	LFW	Greyscale	-	-	1	-	-	-	-	-
#18	R-G-B-ResNet34	LFW	RGB	-	-	1	-	-	-	-	-
#19	ResNet34	LFW	RGB on Greyscale	-	-	1	-	-	-	-	-

Table A.1: Configuration for each experiment: *LR* represents the learning rate, *BS* the batch size, *#W* the number of workers used to load the data, *Optim* the optimiser, and *ES* the early stopping rule.

Bibliography

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6.

Albon, C. (2018). *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*. " O'Reilly Media, Inc.".

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., Awwal, A. A. S., and Asari, V. K. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *arXiv:1803.01164 [cs]*. arXiv: 1803.01164.

Anjos, A., El-Shafey, L., Wallace, R., Günther, M., McCool, C., and Marcel, S. (2012). Bob: a free signal processing and machine learning toolbox for researchers. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1449–1452.

Anjos, A., Günther, M., de Freitas Pereira, T., Korshunov, P., Mohammadi, A., and Marcel, S. (2017). Continuously reproducing toolchains in pattern recognition and machine learning experiments.

Arandjelović, O. (2012). Colour invariants under a non-linear photometric camera model and their application to face recognition from video. *Pattern Recognition*, 45(7):2499–2509. Publisher: Elsevier.

Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W.-D., and McWilliams, B. (2017). The shattered gradients problem: If resnets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

Bianco, S., Cusano, C., Napoletano, P., and Schettini, R. (2017). Improving CNN-based texture classification by color balancing. *Journal of Imaging*, 3(3):33. Publisher: Multidisciplinary Digital Publishing Institute.

Bird, J. J., Ekárt, A., and Faria, D. R. (2020). On the effects of pseudorandom and quantum-random number generators in soft computing. *Soft Computing*, 24(12):9243–9256. Publisher: Springer.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bo, L., Ren, X., and Fox, D. (2011). Depth kernel descriptors for object recognition. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 821–826. IEEE.

Bo, L., Ren, X., and Fox, D. (2013). Unsupervised feature learning for RGB-D based object recognition. In *Experimental robotics*, pages 387–402. Springer.

Buhrmester, V., Münch, D., Bulatov, D., and Arens, M. (2019). Evaluating the Impact of Color Information in Deep Neural Networks. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 302–316. Springer.

Bui, H. M., Lech, M., Cheng, E., Neville, K., and Burnett, I. S. (2016). Using grayscale images for object recognition with convolutional-recursive neural network. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 321–325.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE.

Chen, T., Xu, R., He, Y., and Wang, X. (2015). A Gloss Composition and Context Clustering Based Distributed Word Sense Representation Model. *Entropy*, 17(9):6007–6024. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

Chokkadi, S. and Bhandary, A. (2019). A study on various state of the art of the art face recognition system using deep learning techniques. *arXiv preprint arXiv:1911.08426*.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Chollet, F. (2018). *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG.

Clark, A., Simpson, J., and Hall, J. (2019). Comparing CNN Inputs for Terrain Classification using Simulation. In *2019 First International Conference on Transdisciplinary AI (TransAI)*, pages 43–47. IEEE.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Dutt, A., Pellerin, D., and Quénot, G. (2020). Coupled ensembles of neural networks. *Neurocomputing*, 396:346–357. Publisher: Elsevier.

Ebner, M. (2007). *Color constancy*, volume 7. John Wiley & Sons.

Engilberge, M., Collins, E., and Süsstrunk, S. (2017). Color representation in deep neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2786–2790. IEEE.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130.

Gastaldi, X. (2017). Shake-shake regularization. *arXiv preprint arXiv:1705.07485*.

Gonzales, R. and Woods, R. (2008). *Digital Image Processing. 3rd*. Upper Saddle River: Pearson.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR.

Grauman, K. and Leibe, B. (2011). Visual object recognition. *Synthesis lectures on artificial intelligence and machine learning*, 5(2):1–181. Publisher: Morgan & Claypool Publishers.

Günther, M., El Shafey, L., and Marcel, S. (2016). Face recognition in challenging environments: An experimental and reproducible research survey. In *Face recognition across the imaging spectrum*, pages 247–280. Springer.

Günther, M., Rozsa, A., and Boult, T. E. (2017). AFFACT: Alignment-free facial attribute classification technique. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 90–99. IEEE.

Günther, M., Wallace, R., and Marcel, S. (2012). An open source framework for standardized comparisons of face recognition algorithms. In *European Conference on Computer Vision*, pages 547–556. Springer.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Howard, J. and Gugger, S. (2020). *Deep Learning for Coders With Fastai and PyTorch: AI Applications Without a PhD*. O'Reilly Media, Inc.: Sevastopol, CA, USA.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., and Wu, Y. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in neural information processing systems*, pages 103–112.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106. Publisher: Wiley-Blackwell.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jain, A. K. and Li, S. Z. (2011). *Handbook of face recognition*, volume 1. Springer.

Jones, C. and Abbott, A. L. (2006). Color face recognition by hypercomplex gabor analysis. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 6–pp. IEEE.

Jones, M. J. and Rehg, J. M. (2002). Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96. Publisher: Springer.

Kamal, K. C., Yin, Z., Wu, M., and Wu, Z. (2019). Depthwise separable convolution architectures for plant disease classification. *Computers and Electronics in Agriculture*, 165:104948. Publisher: Elsevier.

Kanade, T. (1974). Picture processing system by computer complex and recognition of human faces. Publisher: Kyoto University.

Kanan, C. and Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition? *PloS one*, 7(1):e29740. Publisher: Public Library of Science.

Krizhevsky, A. and Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view RGB-D object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824. ISSN: 1050-4729.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113. Publisher: IEEE.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. Number: 7553 Publisher: Nature Publishing Group.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Conference Name: Proceedings of the IEEE.

LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104 Vol.2. ISSN: 1063-6919.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Liu, Z. and Liu, C. (2008). A hybrid color and frequency features method for face recognition. *IEEE transactions on image processing*, 17(10):1975–1980. Publisher: IEEE.

Liu, Z., Yang, J., and Liu, C. (2010). Extracting multiple features in the CID color space for face recognition. *IEEE Transactions on image Processing*, 19(9):2502–2509. Publisher: IEEE.

Lu, Z., Jiang, X., and Kot, A. (2018). Color space construction by optimizing luminance and chrominance components for face recognition. *Pattern Recognition*, 83:456–468. Publisher: Elsevier.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41. Publisher: ACM New York, NY, USA.

Monti, R. P., Tootoonian, S., and Cao, R. (2018). Avoiding degradation in deep feed-forward networks by phasing out skip-connections. In *International Conference on Artificial Neural Networks*, pages 447–456. Springer.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.

Naik, Y. (2014). Detailed survey of different face recognition approaches. *International Journal of Computer Science and Mobile Computing*, 3(5):1306–1313.

Namatēvs, I. (2017). Deep convolutional neural networks: Structure, feature extraction and training. *Information Technology and Management Science*, 20(1):40–47. Publisher: Sciendo.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA.

Oloyede, M. O., Hancke, G. P., and Myburgh, H. C. (2020). A review on face recognition systems: recent approaches and challenges. *Multimedia Tools and Applications*, 79(37):27891–27922. Publisher: Springer.

Pal, K. K. and Sudeep, K. S. (2016). Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1778–1781. IEEE.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. Publisher: British Machine Vision Association.

Perrott, C. G. and Hamey, L. G. (1991). Object recognition, a survey of the literature. Publisher: Citeseer.

Phong, N. H. and Ribeiro, B. (2020). Rethinking Recurrent Neural Networks and other Improvements for Image Classification. *arXiv preprint arXiv:2007.15161*.

Polyak, S. (1957). The vertebrate visual system. Publisher: University of Chicago Press Chicago.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Rafegas, I. and Vanrell, M. (2018). Color encoding in biologically-inspired convolutional neural networks. *Vision research*, 151:7–17. Publisher: Elsevier.

Rajapakse, M., Tan, J., and Rajapakse, J. (2004). Color channel encoding with NMF for face recognition. In *2004 International Conference on Image Processing, 2004. ICIP'04.*, volume 3, pages 2007–2010. IEEE.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Ren, J. S., Wang, W., Wang, J., and Liao, S. (2012). An unsupervised feature learning approach to improve automatic incident detection. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 172–177. IEEE.

Roberts, L. G. (1963). *Machine perception of three-dimensional solids*. PhD Thesis, Massachusetts Institute of Technology.

Rossion, B. and Pourtois, G. (2004). Revisiting Snodgrass and Vanderwart's object pictorial set: The role of surface detail in basic-level object recognition. *Perception*, 33(2):217–236. Publisher: SAGE Publications Sage UK: London, England.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., and Bernstein, M. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252. Publisher: Springer.

Sachin, R., Sowmya, V., Govind, D., and Soman, K. P. (2017). Dependency of various color and intensity planes on CNN based image classification. In *International symposium on signal processing and intelligent recognition systems*, pages 167–177. Springer.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Shih, P. and Liu, C. (2006). Improving the face recognition grand challenge baseline performance using color configurations across color spaces. In *2006 International Conference on Image Processing*, pages 1001–1004. IEEE.

Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for image classification. *Ph. D. thesis*. Publisher: Citeseer.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3. Issue: 2003.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Socher, R., Huval, B., Bath, B., Manning, C. D., and Ng, A. Y. (2012). Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems*, pages 656–664.

Steward, J. M. and Cole, B. L. (1989). What do color vision defectives say about everyday tasks? *Optometry and vision science: official publication of the American Academy of Optometry*, 66(5):288–295.

Syafeeza, A. R., Khalil-Hani, M., Liew, S. S., and Bakhteri, R. (2014). Convolutional neural network for face recognition with pose and illumination variation. *International Journal of Engineering & Technology*, 6(1):0975–4024.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708.

Talo, M. (2019). Automated classification of histopathology images using transfer learning. *Artificial intelligence in medicine*, 101:101743. Publisher: Elsevier.

Tanaka, J. W. and Presnell, L. M. (1999). Color diagnosticity in object recognition. *Perception & Psychophysics*, 61(6):1140–1153. Publisher: Springer.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970. Publisher: IEEE.

Torres, L., Reutter, J.-Y., and Lorente, L. (1999). The importance of the color information in face recognition. In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 3, pages 627–631. IEEE.

Unnikrishnan, A., Sowmya, V., and Soman, K. P. (2018). Deep AlexNet with Reduced Number of Trainable Parameters for Satellite Image Classification. *Procedia computer science*, 143:931–938. Publisher: Elsevier.

Wistuba, M., Rawat, A., and Pedapati, T. (2019). A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*.

Wu, C. W. (2018). ProdSumNet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions. *arXiv preprint arXiv:1809.02209*.

Wu, J. (2020). Convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, page 35.

Wurm, L. H., Legge, G. E., Isenberg, L. M., and Luebker, A. (1993). Color improves object recognition in normal and low vision. *Journal of Experimental Psychology: Human perception and performance*, 19(4):899. Publisher: American Psychological Association.

Xie, Y. and Richmond, D. (2018). Pre-training on grayscale ImageNet improves medical image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

Yamada, Y., Iwamura, M., and Kise, K. (2018). Shakedrop regularization.

Yang, J. and Liu, C. (2008). Color image discriminant models and algorithms for face recognition. *IEEE Transactions on Neural Networks*, 19(12):2088–2098. Publisher: IEEE.

Yip, A. W. and Sinha, P. (2002). Contribution of color to face recognition. *Perception*, 31(8):995–1003. Publisher: SAGE Publications Sage UK: London, England.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503. Publisher: IEEE.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.