BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

IMPROVEMENT OF METHODS FOR DETECTION AND CLASSIFICATION OF DAMAGES IN FINGERPRINT IMAGES

VYLEPŠENÍ METOD DETEKCE A KLASIFIKACE POŠKOZENÍ OTISKU PRSTU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
LUKÁŠ FOLTYN

SUPERVISOR
VEDOUCÍ PRÁCE
Ing. ONDŘEJ KANICH, Ph.D.

BRNO 2023

BRNO FACULTY
UNIVERSITY OF INFORMATION
OF TECHNOLOGY TECHNOLOGY

Bachelor's Thesis Assignment

148340

Institut:	Department of Intelligent Systems (UITS)
Student:	**Foltyn Lukáš**
Programme:	Information Technology
Specialization:	Information Technology
Title:	**Improvement of Methods for Detection and Classification of Damages in Fingerprint Images**
Category:	Image Processing
Academic year:	2022/23

Assignment:

1. Study the literature on fingerprint biometric recognition, the generation of synthetic fingerprints and damage artefacts in the fingerprint image. Get acquainted with different methods for detection and classification using neural networks.
2. Choose two damage artefacts (skin diseases, effects of pressure, etc.). Propose an improvement for at least three neural networks architectures, which will add detection and classification of the chosen damage artefacts.
3. Implement the proposed methods from the previous point.
4. Analyse the precision of implemented methods from previous points. Compare the results with the original methods.
5. Summarise and discuss achieved results.

Literature:

- Maltoni, D., Maio, D., Jain, A.K. and Prabhakar, S.: *Handbook of Fingerprint Recognition*. Springer, 2009, p. 512. ISBN 978-1-8488-2254-2.
- Drahanský, M.: *Hand-Based Biometrics: Methods and technology*, IET 2018, p. 430, ISBN 978-1-78561-224-4.
- Fořtová, K.: *Analysis of Convolutional Neural Networks for Detection and Classification of Damages in Fingerprint Images*, 2022. Master's thesis. FIT BUT in Brno, Brno.

Requirements for the semestral defence:

- Items 1 and 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Kanich Ondřej, Ing., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	10.5.2023
Approval date:	3.11.2022

Abstract

This study aims to improve existing methods for detecting and classifying damage in fingerprint images by leveraging previous works conducted by students at Brno University of Technology. The work is built upon three applications: line damage (scars, hairs, creases) generator, moisture generator, and application containing multiple different models for fingerprint damage detection and classification. The three best-performing models - Faster-RCNN ResNet50, Faster-RCNN ResNet101, and CenterNet ResNet101 - were selected for further improvement. The work describes the creation of a dataset using undamaged synthetic fingerprint images, with the aforementioned damages introduced artificially. Efforts to improve the prediction accuracy of the models were based on more accurate annotation of bounding boxes and adjusting the hyperparameters. While the work yielded some improvements, the results are not consistently successful across all models and damage types.

Abstrakt

Cílem této práce je vylepšit stávající metody detekce a klasifikace poškození na snímcích otisků prstů s využitím předchozích prací studentů Vysokého učení technického v Brně. Práce je postavena na třech aplikacích: generátoru čárového poškození (jizvy, vlasy, rýhy), generátoru vlhkosti a aplikaci poskytující více různých modelů pro detekci a klasifikaci poškození otisků prstů. Pro vylepšení byly vybrány tři nejpřesnější modely - Faster-RCNN ResNet50, Faster-RCNN ResNet101 a CenterNet ResNet101. Práce popisuje vytvoření datové sady pomocí nepoškozených syntetických snímků otisků prstů s výše uvedenými uměle zavedenými poškozeními. Snaha o zlepšení přesnosti predikce modelů byla založena na přesnějším anotovaní ohraničujících boxů a úpravě hyperparametrů. Přestože práce přinesla určitá zlepšení, výsledky nejsou konzistentně úspěšné u všech modelů a typů poškození.

Keywords

fingerprints, synthetic fingerprints, damaged fingerprint images, convolutional neural networks, detection, classification

Klíčová slova

otisky prstů, syntetické otisky prstů, poškozené snímky otisků prstů, konvoluční neuronové sítě, detekce, klasifikace

Reference

FOLTYN, Lukáš. *Improvement of Methods for Detection and Classification of Damages in Fingerprint Images.* Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ondřej Kanich, Ph.D.

Rozšířený abstrakt

Tato bakalářská práce se zaměřuje na zlepšení výkonnosti vybraných modelů pro detekci a klasifikaci čárových poškození a vlhkosti na snímcích otisků prstů. Jsou zde zkoumány tři detekční modely: Faster R-CNN ResNet50, Faster R-CNN ResNet101 a CenterNet ResNet101, které jsou trénovány s různými konfiguracemi (kombinace hyperparametrů a datových sad), aby se následně u zvolené konfigurace mohl posoudit její pozitivní či negativní vliv na příslušný model během trénovacího procesu.

V první fázi je vytvořena umělá datová sada, která využívá aplikace pro generování čárových poškození a vlhkosti ve snímcích otisků prstů. Pro trénink modelů je nezbytné snímky anotovat ohraničujícími boxy, což vedlo k potřebě právě tyto dva zmíněné generátory o anotace ohraničujících boxů rozšířit. Pro čárová poškození jsou implementovány dva různé přístupy: první zahrnuje všechna poškození do jednoho velkého boxu, zatímco druhý přístup se snaží vykreslit boxy tak, aby pokrývaly minimum zdravé plochy, čehož je dosaženo pužitím velkým množstvím malých boxů přesně vyplňující jednotlivé čáry. V případě generátoru vlhkosti, který již základní anotace poskytoval, byl implementován druhý přístup, který se pokoušel některé nadbytečné boxy odstranit a jiné zpřesnit. Všechny popisované metody jsou poté využity v rámci experimentů.

Experimenty se skládají ze tří částí: 1. použití původních parametrů vybraných modelů, 2. úprava hyperparametrů a 3. změna datových sad s odlišnými metodami anotace ohraničujících boxů. Hlavní metrikou pro porovnání výkonnosti modelů je průměrná správně detekovaná a klasifikovaná plocha, s vyhodnocením dalších metrik, jako je průměrná nedetekovaná plocha, průměrná nadbytečně detekovaná plocha a průměrné skóre správně detekované a klasifikované oblasti.

Výsledky ukázaly, že úpravy hyperparametrů měly minimální vliv na modely Faster R-CNN, zatímco u modelu CenterNet došlo k výraznému zlepšení, tedy k navýšení přesnoti o 5 %. Ve třetí sadě experimentů, která používala jiné přístupy k anotaci ohraničujících boxů, oba modely Faster R-CNN vykazovaly výrazný pokles celkové přesnosti. Nejvíce byl však ovlivněn model CenterNet, jehož pokles byl více drastický. Tento pokles v přesnosti byl připisován již zmiňovanému druhému přístupu anotace ohraničujících boxů u čárového poškození. Modely pravděpodobně považovaly každý malý ohraničující box za samostatný celek, což je ve výsledku odstínilo od vnímaní čárového poškození v rámci globálního kontextu.

Naproti tomu, u vlhkosti druhý přístup anotování přinesl pozitivní výsledky u modelů Faster R-CNN ResNet50 a ResNet101. Pouze u modelu CenterNet ResNet101 nedošlo k žádnému zlepšení, kvůli silnému vlivu snímků s čárovým poškozením. Výsledný natrénovaný model byl tedy prakticky téměř nepoužitelný.

Výsledkem práce je tedy 9 natrénovaných modelů, z nichž byl nejlepší model Faster R-CNN ResNet101 natrénovaný v 5. experimentu, který dosáhl hodnoty 90,646 % u metriky správně detekované a klasifikované plochy. Závěrem lze konstatovat, že tato práce poskytuje pohled na výkonnost různých modelů hlubokého učení pro detekci a klasifikaci poškození otisků prstů. Zdůrazňuje význam pečlivého výběru architektury modelu, nastavení jednotlivých hyperparametrů a metod anotace pro dosažení optimálních výsledků.

Improvement of Methods for Detection and Classification of Damages in Fingerprint Images

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Ondřej Kanich, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

....
Lukáš Foltyn
May 8, 2023

Contents

Chapter 1

Introduction

Fingerprint biometrics has become one of the most widely used personal identification methods due to its reliability and accuracy. The uniqueness of fingerprints and their stability over time make them an ideal biometric identifier for various applications, including law enforcement, access control, and mobile device authentication. However, skin diseases and many external factors can impact the accuracy and reliability of fingerprint recognition systems. They can alter the patterns and ridges on the fingers, making it difficult for the sensors to capture a clear and accurate fingerprint image. Therefore, automated detection and classification of such damaged fingerprint images could contribute to an improvement of biometric systems.

In the past, skin disease detection and classification were typically done through manual inspection by dermatologists and other medical professionals. This process involved visually examining skin lesions, rashes, and other abnormalities. Then the experts using their experience, made a diagnosis. While this approach can be beneficial, it is time-consuming and can be subject to human error and bias.

In addition to manual inspection, various computer vision and machine learning techniques have been used for skin disease detection and classification in the past. These include techniques such as feature extraction and pattern recognition, which involve analyzing specific features and characteristics of skin lesions to identify specific diseases. However, these techniques often require extensive feature engineering and may not be able to capture the full range of variations and complexities in skin diseases.

More recently, deep learning techniques such as convolutional neural networks (CNNs) have become tremendously helpful in automating the detection and classification of specific patterns in images, in the case of this work, skin diseases. By analyzing large datasets of fingerprint images, CNNs can learn to recognize patterns and features indicative of specific skin diseases or other damages of various types, enabling them to detect and classify a wide range of dermatological conditions accurately. This work aims to explore the performance of the top three CNNs presented in Katerina Fortova's master thesis [16] on fingerprint images affected by moisture and line damage.

This thesis is organized into five chapters, with the initial chapters gradually explaining the necessary background for the practical portion of the work. The subsequent chapters discuss the implementation process and the results obtained. Chapter 2 establishes a fundamental understanding of human skin anatomy, the components that form a fingerprint, and the various types of potential damage artifacts that can affect fingerprints. It also examines the technologies utilized for capturing fingerprint images and the possibility of synthetic fingerprint generation. Additionally, this chapter addresses the matching process of images in biometric systems. Chapter 3 delves into the topic of neural networks, starting with basic concepts such as artificial neurons, multi-layer perceptrons, activation functions, and loss functions. The chapter then explains how these components function collectively in the neural network training process, known as backpropagation. Convolutional neural networks, their architectures, and object detection approaches are outlined at the end of the chapter. Chapter 4 covers the generation of the synthetic dataset and the essential preprocessing steps, along with the selected CNN architectures and their respective results. The final Chapter 5 summarizes the findings and a few thoughts on how the work could be improved.

Chapter 2

Fingerprint Biometrics

This chapter delves into fingerprint biometrics, an effective and widely used identification method. It starts with an analysis of skin structure and its most relevant parts. The subsequent section highlights the unique patterns of fingerprints and their individuality, contributing to their reliability in biometric systems.

Next, the examination focuses on the challenges of damaged fingerprint images due to factors like moisture and line damage and their influence on fingerprint recognition. The chapter also describes a range of fingerprint sensor technologies, including optical, capacitive, and ultrasound sensors, followed by an overview of the fingerprint recognition process and techniques employed in fingerprint matching. The chapter concludes with a discussion on synthetic fingerprint generation for research and development purposes.

2.1 Skin Structure

Understanding the basic structure of human skin is crucial for comprehending fingerprints. The skin is often considered the largest organ of the body, accounting for approximately 16 % of body weight. It performs several vital functions, such as protection against physical or chemical damage, prevention of excessive water loss from the body, and thermoregulation. It is divided into three layers (Figure 2.1): the outermost layer, the **epidermis**; the middle layer, the **dermis**; and the innermost layer, the **subcutaneous tissue**. [29]

2.1.1 Epidermis

The epidermis is the thinnest component of the skin, consisting of a stratified, squamous epithelium layer containing at least four different cell types: keratinocytes, melanocytes, Langerhans cells, and Merkel cells [14]. Since no blood vessels exist in the epidermis, the layer continually dies off and renews [29]. Epidermis cells also contain a protein that is difficult to dissolve in water, making the skin effectively impermeable [15].

2.1.2 Dermis

The dermis is a connective tissue layer, 15-40 times thicker than the epidermis, with significantly lower energy requirements. It consists of two distinct layers:

- **Reticular dermis**: the bottom layer of the dermis, contains blood vessels, glands, hair follicles, lymphatics, nerves, and fat cells, and is surrounded by collagen and elastin fibers [45].

- **Papillary dermis**: the thin upper layer of the dermis, consisting of collagen fibers, fibroblast cells, fat cells, blood vessels, nerve fibers, and touch receptors [45]. The papillary layer forms irregular projections known as dermal papillae, interlacing with the epidermal ridges [19].

2.1.3 Subcutaneous Tissue

The subcutaneous tissue, composed of subcutaneous fat and other cell types, primarily functions in thermoregulation and protecting underlying organs, muscles, and bones from physical damage by covering them with stored fat. [48]

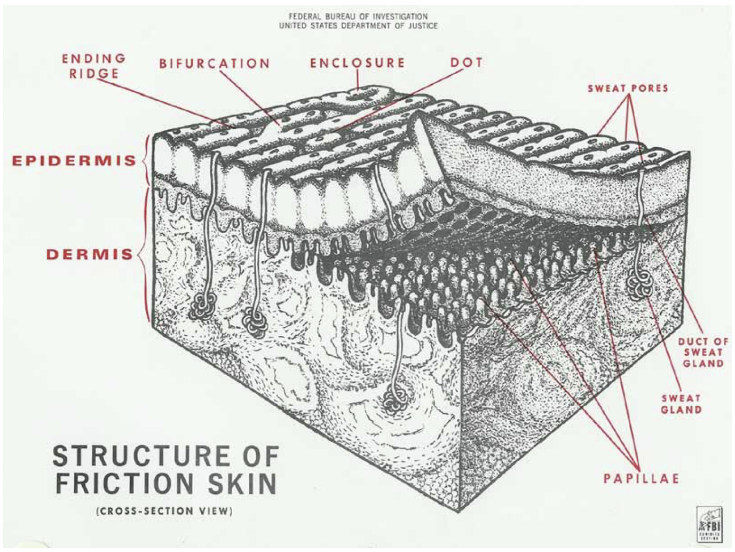

Figure 2.1: Structure of the human skin [11].

2.2 Individuality of a Fingerpint

Fingerprints are unique, individual characteristics present on the fingers and thumbs of humans. They form during fetal development and remain unchanged throughout a person's life. No two fingerprints are identical, as demonstrated by numerous historical findings and research studies. [13]

Fingerprints consist of a series of ridges and valleys on the skin's surface as displayed in Figure 2.2. These ridges, sometimes called friction ridges, are formed by the dermal papillae mentioned earlier in Section 2.1.2, which are small protruberances on the skin containing sweat glands and blood vessels. The patterns formed by the ridges and valleys are what make each fingerprint unique. [30]

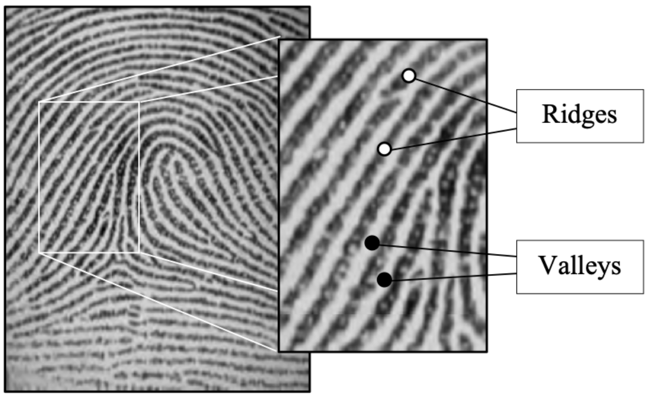

Figure 2.2: Fingerprint's ridges and valleys [30].

The features that can be extracted when analyzing a fingerprint pattern can be divided into three levels based on different scales [30]:

- **Level 1**: The overall ridge flow pattern is analyzed at this first global level. This includes the ridge direction as well as the number of ridges. Then, there are special regions called *singular points*, which are assumed to have distinctive shapes. These regions are usually classified into three types: *loop*, *delta* (or *arche*) and *whorl* (Figure 2.3) . [30] [14]

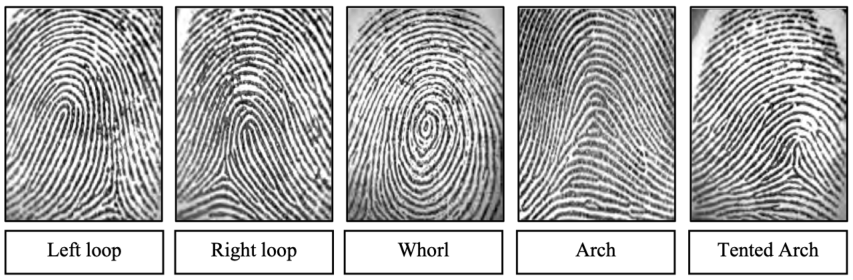

Figure 2.3: Singular point types [30].

- **Level 2**: The characteristics of ridges and valleys called *minutiae* can be found at this local level. Many different local ridge characteristics can be extracted, but most of them appear very rarely, depending on the fingerprint's quality. The two most prominent ones are the *ridge endings* and *ridge bifurcations* (Figure 2.4) . Minutiae are the most commonly used features in automatic fingerprint matching. Finding a center point called the *core* is also beneficial if possible. The core is defined as the center of the northmost loop-type singularity. It can be used for pre-aligning the fingerprint images when trying to match them. [30]

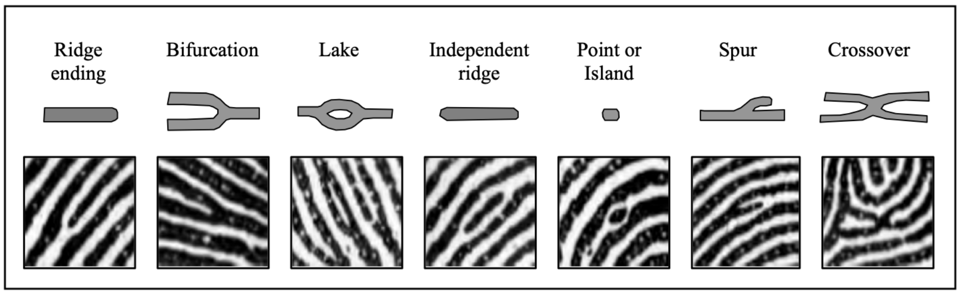

Figure 2.4: Most common minutiae types [30].

- **Level 3**: At the very-fine level, intra-ridge details can be detected. This includes a ridge width, shape, curvature, contours, or other permanent details such as incipient ridges. There are also *sweat pores*, whose positions and shapes are considered to be very distinctive. However, to extract these features, a high-resolution fingerprint image of good quality is required, which is rarely the case in the real world. [30]

2.3 Skin Diseases and their Impact on a Fingerprint

The skin is a very complex organ that is exposed to various external factors every single day. These factors can cause many skin diseases, which can significantly impact the fingerprint depending on the severity of the skin disease itself. If the disease or other damaged artifact, e.g., a cut or burn wound, affects only the surface of the skin, the fingerprint pattern, when recovered, remains the same. However, if the structure of the ridges in the epidermis and the underlying dermis is destroyed, the ridges will not grow into the same pattern as before. The diseases can be divided into three groups based on their impact on the skin. [14]

- Diseases causing **histopathological changes of the epidermis and dermis** can cause problems for all kinds of fingerprint scanners since they can influence the color and internal structure of the skin. Diseases such as fingertip eczema, pyoderma, or Raynaud's phenomenon can be found in this group. [14]
- Diseases causing **skin discoloration** could cause problems for optical scanners or scanners that use skin color as a part of their antispoof detection. Typical representatives: pitted keratolysis, carotenosis, or xanthomas. [14]
- Diseases causing **histopathological changes at the junction of the epidermis and dermis**, which means potential changes in the structure underneath the skin at the junction between dermis and epidermis. This could cause some trouble for ultrasonic scanners since that is where they acquire fingerprint images. The most common representatives are hand eczema, verruca vulgaris (warts), psoriasis, or epidermolysis bullosa. [14]

2.3.1 Moisture

Moisture is not perceived as damage one can have on a fingerprint. However, it plays a crucial role in the quality of fingerprint images as it can heavily degrade the clarity of captured ridge patterns. When skin is optimally hydrated, the frictional and adhesive properties of the finger's surface allow for better contact with the capturing surface, resulting in a more defined and high-quality image. However, excess moisture can lead to smudging and distortion of the fingerprint image, making it difficult to identify unique features. On the other hand, dehydrated skin may not transfer enough sweat and oils to produce a clear fingerprint. As a result, the fingerprint image may appear faint, with broken or missing ridges, making it challenging to analyze and match against other prints. Examples of fingerprint with different moisture levels can be seen in Figure 2.5. [32]

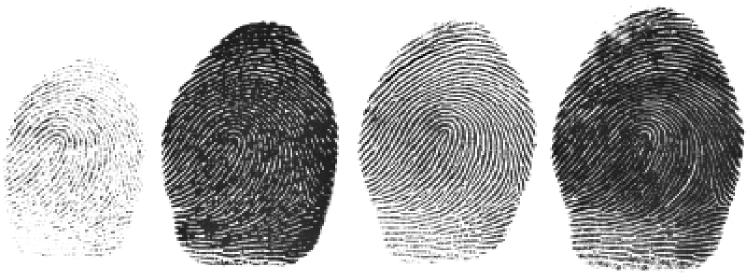
Figure 2.5: Fingerprint samples acquired from the same finger with skin subjectively perceived as (left to right): dry (62.5 %), wet (99.9 %), normally moist (84.6 %), and normally moist (84.6 %) with finger strongly pressed against the sensor [32].

2.3.2 Line Damage

In this work, the term „line damage artifact“ refers to three distinct types: **creases**, **scars**, and **hairs**. Hairs, similarly to moisture, do not cause direct damage to the fingerprint itself but are an external factor that can degrade the final fingerprint image (fallen hair between the sensor and the fingerprint). According to Vanessa Joriova's work [25], hair damage appears as a thin grey line with a slight white padding around it (Figure 2.6). The padding is created by the fingerprint wrapping around the hair and not touching the sensor properly.

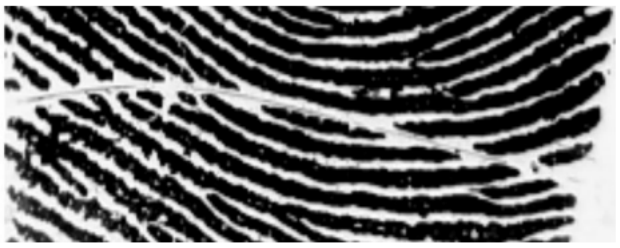
Figure 2.6: Fingerprint with a hair artifact [25].

Creases are irregular stripes that cross ridges and valleys in the fingerprints (Figure 2.7). They are a common occurrence among elderly individuals. Aging, manual work, accidents, and other factors cause them. Without enhancing the final image, creases can cause problems for minutia extraction algorithms as they can lead to the detection of a spurious minutia or the omission of some minutia. [49]

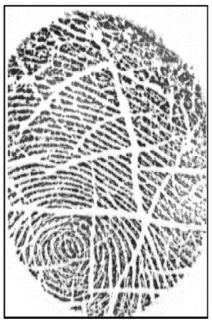
Figure 2.7: Creases in a fingerprint [49].

Scars are a more severe type of line damage that can sometimes be mistaken for creases due to their size and shape. Scars may have specific features such as black outlines or patches inside (Figure 2.8). However, scars are considered more permanent than creases, mainly if the wound damages the epidermis layer. Following images show examples of generated scars on a fingerprint. [25]

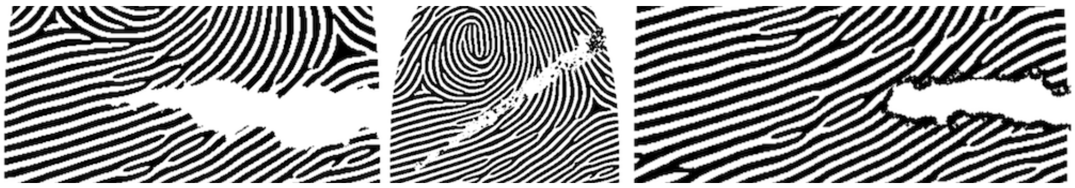
Figure 2.8: Generated scars (from left to right): normal, patches, outline.

2.4 Fingerprint Acquisition

In today's world, fingerprint images can be acquired through various methods, typically classified into two main categories: *offline* and *live-scan* [30]. The offline image acquisition process involves pressing or rolling an ink-smeared fingertip onto paper, which is then digitized. *Latent* fingerprints, which play a significant role in forensic applications, can be found at crime scenes and lifted from surfaces using specific chemical techniques. These latent fingerprints also fall under the offline category. [30]

Live-scan fingerprint acquisition is now more convenient and has become the preferred method [30]. In contrast, live-scan images are captured by placing the finger on an electronic fingerprint sensor and scanning it in real time. The technologies utilized for these sensors will be discussed later in this chapter.

Figure 2.9 illustrates the primary function executed by the fingerprint scanner. Initially, the sensor captures the ridge pattern of the positioned fingertip. Generally, the acquired signal is in analog form, though exceptions may occur. Subsequently, the analog signal undergoes processing via an A/D (Analog to Digital) converter and afterward can be transferred to a computer through the interface module in the digital form. [30]

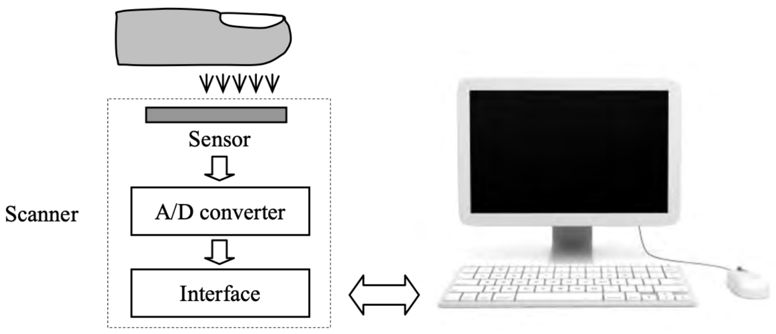

Figure 2.9: Block diagram of a fingerprint scanner [30].

The two most commonly used sensor technologies today are *optical* and *capacitive*, along with the more advanced *ultrasound* sensor [6][36]. Additionally, there are other technologies available for fingerprint acquisition, including *thermal*, *pressure*, *electro-optical*, *e-field*, and *MEMS* [14].

2.4.1 Optical Sensor

The optical sensor operates on the principle of light reflection demonstrated in Figure 2.10. Typically, a light source, such as an LED, illuminates the protective glass at an angle that reflects all the light to the camera (usually a CCD or CMOS sensor) when no finger touches the surface. However, when a finger is placed on the surface, the glass makes contact with the ridges, disrupting the reflection and causing little to no light to be reflected. As a result, the ridges in the final fingerprint image appear dark, while the valleys appear bright. The height difference between ridges and valleys is significant, making this sensor a reliable anti-spoofing mechanism. [30][26]

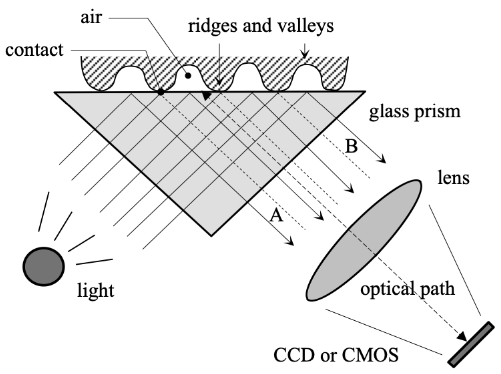

Figure 2.10: Principle of the optical sensor [30].

2.4.2 Capacitive Sensor

A capacitive sensor consists of a grid of micro-capacitor plates embedded within a chip, covered by a non-conductive silicon dioxide layer where the finger is placed. The capacitor's discharge is based on the distance between the finger's surface and the capacitor plate (Figure 2.11). The closer the skin, the more the capacitor discharges, resulting in a distinction between the ridges and valleys. Similar to optical sensors, capacitive sensors are not easily deceived by flat images of fingerprints, providing a reliable anti-spoofing mechanism. [30][14][6]

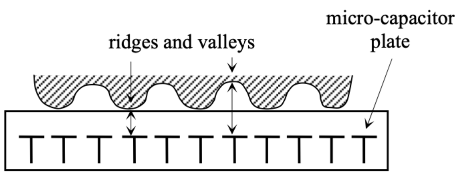

Figure 2.11: Principle of the capacitive sensor [30].

2.4.3 Ultrasound Sensor

Ultrasound sensors emit ultrasound pulses and measure the returning signals after they bounce off the surface (Figure 2.12) . Due to the ultrasound waves' ability to penetrate beneath the skin's surface, the resulting ridge pattern image is captured from a deeper skin layer. Consequently, this method demonstrates resistance to potential damage or impurities found on the skin's surface, providing a more reliable and secure fingerprint recognition. [30][14][6]

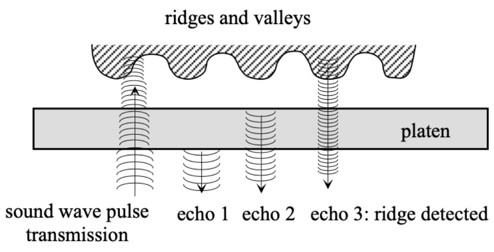

Figure 2.12: Principle of the capacitive sensor [30].

2.5 Fingerprint Recognition

Before an individual's fingerprint can be recognized, they must undergo an enrollment or registration process. This process takes longer, as the acquired image and its template creation must be of high quality [14]. The template is then stored in a data storage system, from which the biometric system later retrieves it. When designing a biometric system, it is crucial to determine how an individual will be recognized. Based on the application's requirements, the system can function as either a verification or an identification system. The primary difference between these systems is the number of comparisons required when matching user-obtained fingerprints [30].

A verification system serves to confirm the identity claimed by the user. It performs a one-to-one comparison between the currently captured biometric data and the previously enrolled template. At the end of the process, the user is either granted or denied access. [30]

Conversely, an identification system searches a database of biometric templates to find a match and identify the user. In this case, one-to-many comparisons must be conducted. For larger systems, a pre-selection process can be implemented to eliminate some templates and perform matching against a smaller subset of the database. In the identification system, the user does not claim any identity, as it is determined by the system itself if a template match is found. [30]

2.6 Fingerprint Matching Process

The matching process consists of a series of steps and algorithms applied to the captured input fingerprint, enabling it to be compared with another fingerprint. The returned results can be either a binary value indicating a match (success/failure) or a percentage score representing the similarity between the compared fingerprints. Generally, the process involves the following steps:

- **Fingerprint image capture**: This step can be accomplished using any of the sensor technologies mentioned in Section 2.4. However, it is essential to consider the fingerprint image quality, as it can significantly impact the final result. A liveness check is also crucial to prevent attempts at system spoofing. [14]

- **Fingerprint Processing**: The primary objective of this step is to extract minutiae from the fingerprint image accurately. While this is relatively straightforward when the input is of high quality, it is not always the case. Fingerprint images can often be less than ideal due to factors like skin conditions, sensor noise, and other elements [24]. In such instances, enhancement techniques must be employed.

  Examples of these enhancements include histogram equalization for improving image contrast [18] and the 2D Gabor filter, which is frequently associated with the computation of the orientation field. The orientation field serves as a map of the fingerprint image, indicating the direction of ridges. Additional steps, such as binarization and thinning, are often performed before the minutiae extraction itself. During binarization, each pixel in the image is assigned a new value (1 or 0) based on the intensity of surrounding pixels compared to a given local threshold. Thinning, on the other hand, ensures ridges are only 1-pixel wide [14].

  Minutiae extraction techniques can be broadly classified into those that work on binarized images and those that work on grayscale images [5]. For each identified minutia, the x and y coordinates, minutia type (ending, bifurcation), and gradient (ridge orientation) are stored [14]. All extracted minutiae form the final biometric template, which is subsequently used for matching.

  During these various processing stages, artificial noise may be introduced into the image, resulting in the detection of false minutiae. In such situations, post-processing techniques can be employed [18]. Suppose the fingerprint image contains missing or damaged ridges. In that case, interpolation techniques can be utilized to fill in the missing data [30], or cubic Bezier curves can be used to restore ridges [44].

  The overall process is illustrated in Figure 2.13.

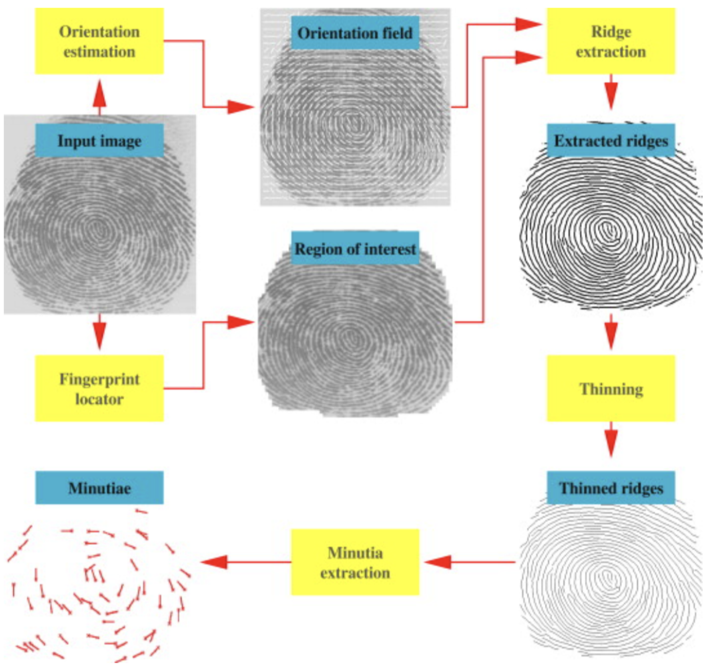

Figure 2.13: Feature extraction process [24].

- **Final Comparison**: As discussed earlier, the extracted features form the biometric template. When a system acquires a new fingerprint for verification or identification purposes, the same features are extracted and compared to a stored template. However, matching techniques are not limited to minutiae-based methods. Alternative approaches include correlation-based methods, analysis of papillary line properties, and even examination of 3D fingerprint characteristics. [14]

2.7 Synthetic Fingerprint Generation

With the widespread use of fingerprints in today's security systems, there is a continuous effort to develop and refine fingerprint recognition algorithms. However, obtaining sizable datasets of real fingerprint images can be costly, time-consuming, and challenging to share due to personal data protection concerns. To accurately assess the effectiveness of an algorithm, it needs to be thoroughly tested and evaluated using large fingerprint datasets. [30]

This is where synthetic fingerprint generation (SFinGe) comes into play, reversing some of the steps outlined in Section 2.6. A fingerprint area, orientation image, and frequency image are generated independently and then used as inputs for the ridge generation process. The resulting binary ridge pattern is rendered, incorporating additional fingerprint-specific noise (Figure 2.14). [30]

Suppose multiple impressions of the same synthetic fingerprint need to be created, a *master fingerprint* is initially generated. This master fingerprint serves as a template, which can then produce impressions by applying displacement, rotation, distortion, skin condition, and noise. [30]

When generating damaged fingerprints affected by skin diseases, a master fingerprint is first created. The skin condition can be incorporated using a specially designed algorithm, through an in-depth study of the specific skin disease is necessary. Alternatively, the damage caused by the disease can be extracted from a real fingerprint image and mapped onto the master fingerprint. [27]

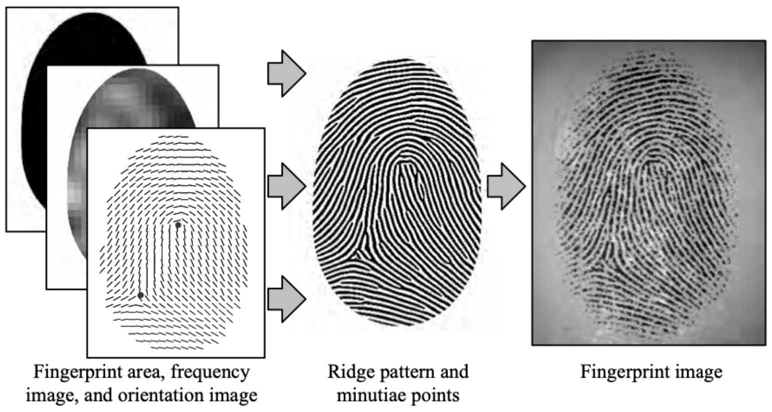
Fingerprint area, frequency image, and orientation image
Ridge pattern and minutiae points
Fingerprint image

Figure 2.14: Synthetic fingerprint generation process [30].

Chapter 3

Neural Networks

Neural networks provide a range of powerful techniques for solving problems in fields like pattern recognition or data analysis. While it may seem to many that neural networks are a relatively new concept, the opposite is true. They were first introduced back in 1943. Since then, they have been slowly evolving and improving. However, it was not until the 1990s that they started gaining real popularity. The main reasons for that are the massive quantity of data available to train the neural networks, the tremendous increase in computing power, and the improvements in the algorithms themselves [22]. In this chapter, the similarity between human neurons and artificial ones will be briefly discussed. Then, the essential building blocks of neural networks will be described. The end of the chapter provides an overview of Convolutional Neural Network architectures and touches on the topic of possible approaches in object detection.

3.1 Biological Neuron

Neurons are the fundamental units of the human brain and nervous system. They are divided into three main parts: *dendrities*, an *axon*, and *soma* (cell body) as displayed in Figure 3.1. Both the axon and dendrites have a tree-like structure. The axon branches are called synapses and are connected to other neurons' dendrites (sometimes directly to the cell body). In that way, neurons can communicate by sending short electrical signals through axons. When another neuron receives a sufficient number of signals in a short period from other neurons, it fires its own signal. Even though the individual neuron may seem to behave relatively simply, the complexity lies in the whole network containing billions of neurons, where each neuron is typically connected to thousands of others. [22] [47]

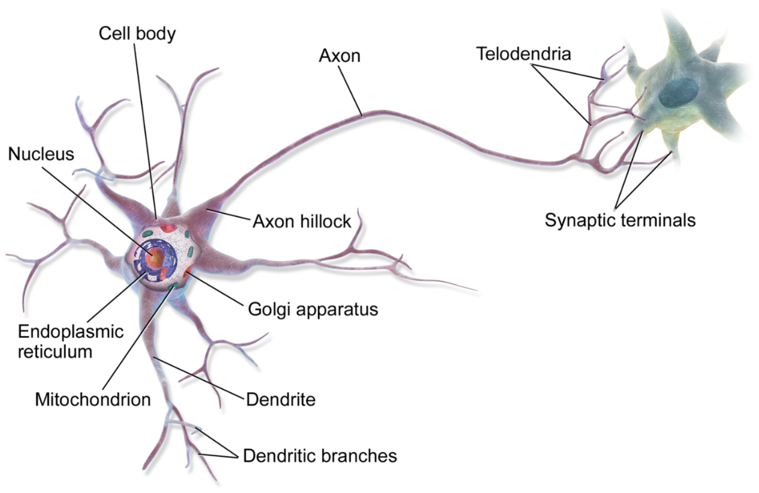

Figure 3.1: Biological neuron structure [22].

3.2 Artificial Neuron

An artificial neuron is an essential building block of every artificial neural network. It has a set of inputs, where each input is assigned a weight for multiplying the input values. There is usually also one extra input called bias. The inputs can be compared to the dendrites of a biological neuron. In the body (soma) of the artificial neuron, the weighted inputs and bias are summed together and run through an *activation function*. The obtained value from the activation function is the neuron (axon) output, which is then passed to other neurons in the network, if there are any. Otherwise, it is considered to be a final result. No matter how many inputs the neuron has, the output is always a single value. A simple schema of an artificial neuron is shown in Figure 3.2. [4] [22]

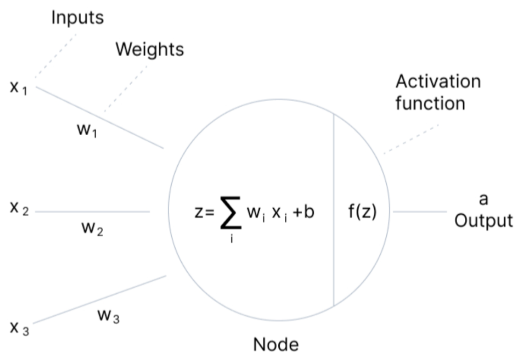

Figure 3.2: Artificial neuron schema [4].

3.3 Perceptron

The perceptron, illustrated in Figure 3.3, is one of the simplest and most well-known artificial neural network architectures. It is a single-layer neural network containing only linear threshold units (LTUs). LTU is a type of artificial neuron whose output is always binary. Such output can be achieved by using the Heavside step activation function. [22]

$$heavside(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{3.1}$$

The perceptron is fed one training instance at a time, which results in predictions. For every neuron that produced a wrong prediction, the connection weights from the inputs that would have contributed to the correct prediction are reinforced. Therefore, the perceptron can classify instances simultaneously into multiple binary classes based on the number of used LTUs. [22]

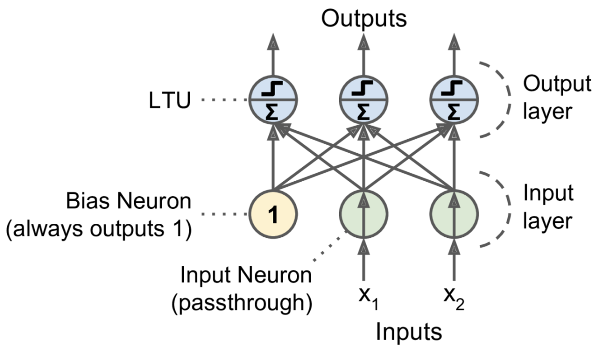

Figure 3.3: Perceptron [22].

3.4 Multi-layer Perceptron

Since it has been shown that perceptrons have many weaknesses, in particular the fact that they are incapable of solving some trivial problems (e.g., the exclusive OR (XOR) classification problem), the multi-layer perceptron (MLP) was introduced (Figure 3.4). It is a feedforward neural network consisting of an input layer, one or more hidden layers, and the output layer. All the layers are fully connected, which means that the output of each neuron is connected to the input of all neurons in the following layer. The hidden layers are responsible for learning the complex patterns in the data. The MLP is trained using the backpropagation algorithm described in Section 3.7. However, it is essential to note that for the backpropagation to work correctly, the Heavside step activation function had to be replaced by one of many different but suiting activation functions. Some examples of such functions are described in the next section. [22]

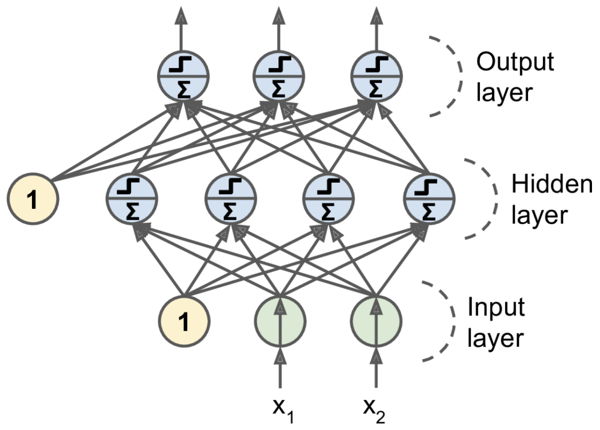

Figure 3.4: Multi-Layer Perceptron [22].

3.5 Activation Function

When designing a neural network, deciding what activation functions to use for each layer is crucial as it can significantly influence performance and prediction accuracy. The activation function transforms an input signal into an output signal, fed as an input to the next layer in the stack. The neural network would be equivalent to a linear regression model if no activation functions were present. Thus using them introduces a non-linearity and allows the creation of complex prediction models. An essential feature of an activation function must be **differentiability** so that backpropagation can be implemented. The most common activation functions are the sigmoid function, the hyperbolic tangent function, the rectified linear unit (ReLU), and the softmax function, some of them being shown in Figure 3.5 together with their derivatives. [42]

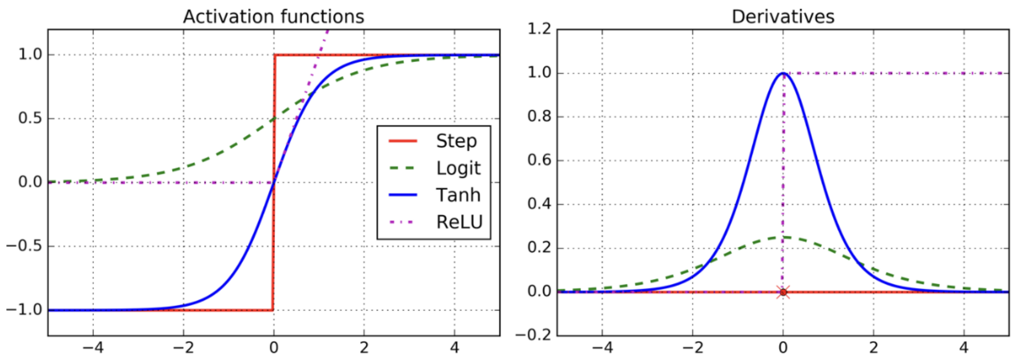

Figure 3.5: Activation functions and their derivatives [22].

3.5.1 Sigmoid Function

Sigmoid is a simple activation function centered around 0.5 that outputs values from the interval (0,1) [42]. Due to its properties, if the sigmoid function is used in multiple layers, a vanishing gradient problem can occur. This problem is caused by the fact that the derivatives of the sigmoid function are close to zero for most input values, which results in the gradient being minimal. Thus, the weights and biases corrections are barely propagated to the initial layers of the neural network [46]. However, the function can be useful for models where the outputs are considered to be predicted probabilities since the value is always between 0 and 1 [41]. The sigmoid function is defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

3.5.2 Hyperbolic Tangent Function

The hyperbolic tangent activation function is very similar to the sigmoid function, but its range differs by going from -1 to 1 instead of (0,1). In hidden layers, tanh is almost always preferred over the sigmoid since its gradient is steeper; therefore, the neural network training should converge quicker. Tanh can also be used in the output layer for mapping the output values as strongly negative, neutral, or strongly positive. Unfortunately, tanh suffers from the same vanishing gradient problem as the sigmoid function [42][4]. The formula for tanh is:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.3}$$

3.5.3 Rectified Linear Unit (ReLU)

The ReLU is the most popular activation function used in hidden layers of neural networks. It outputs the values from interval [0, infinity). Since for the negative input values, the result is zero; the neuron does not get activated, thus taking less computational time than sigmoid or tanh. On the other hand, the gradient always being zero for any negative value can create dead neurons that are never activated as their weights and biases do not get updated. This problem can be solved using Leaky ReLU, which returns a small negative number instead of zero [21]. The ReLU is defined as:

$$ReLU(x) = max(0, x) \tag{3.4}$$

3.5.4 Softmax Function

The softmax activation function is used exclusively in the output layer of the neural network for classification problems. It transforms the raw outputs of the neural network into a vector of probabilities, essentially a probability distribution over the possible classes [35]. The softmax function is defined as:

$$softmax(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.5}$$

3.6 Loss Function

Loss function plays a vital role in the training of the neural network. It is used to measure how well the model is performing and how much it needs to be improved by computing the distance between the current output of the algorithm and the expected output. It can be categorized into two groups: **regression loss functions** and **classification loss functions**. The most common regression loss functions are the mean squared error (MSE) and the mean absolute error (MAE). The most common classification loss functions are the cross-entropy loss and the hinge loss. [34]

- Mean Squared Error (MSE) - is the most common loss function used for regression problems. It is calculated as the average of the squared differences between the predicted and the actual values [41].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

- Cross-entropy loss - is one of the most commonly used loss functions for classification problems. It is closely connected to the softmax activation function returning a vector of probabilities. Each predicted class probability is compared to the actual output of 0 or 1, and a loss is calculated, penalizing the probability based on the distance from the expected value. Thanks to its logarithmic nature, the penalty is large for differences close to 1 and small for differences approaching 0 [28]. The overall cross-entropy loss can be calculated with the following formula:

$$CE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(\hat{y}_{ij}), where\ m\ is\ number\ of\ classes \tag{3.7}$$

3.7 Backpropagation

Backpropagation is an algorithm used to train neural networks. The algorithm's name comes from the error being propagated backward through the network from the output layer to the input layer. The error is calculated using the mentioned loss function $L(w)$. The goal is to minimize the error by adjusting the weights and biases of the neural network or, in other words, to find the local minimum of the loss function. In searching for the local minimum, the gradient of the loss function with respect to each weight and bias has to be calculated. The gradient could be defined in the following way:

$$\nabla L = \left(\frac{\partial L}{\partial w_1}; \frac{\partial L}{\partial w_2}; \ldots; \frac{\partial L}{\partial w_n}\right) \tag{3.8}$$

The weights and biases are then modified, going in the opposite direction of the gradient. If the partial derivative is negative, the weight is increased. If the partial derivative is positive, the weight is decreased [20][39]. A simple illustration can be seen in the Figure 3.6.

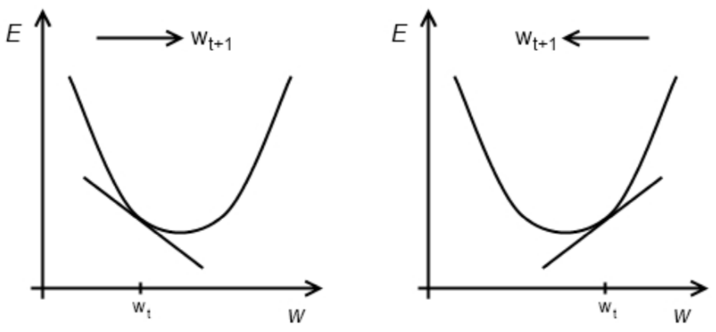

Figure 3.6: A simple example of finding a local minimum of the loss function [20].

The process is repeated until the error is minimized or the maximum number of iterations is reached. When it comes to the step length, it is based on the learning rate η and the steepness of the gradient itself. All partial derivatives can be calculated using the chain rule since the whole neural network is a composition of integration and activation functions [20]. The overall searching process of the local minimum is called **gradient descent**. Three types of gradient descent differ in the amount of data used to compute the gradient of the loss function. Depending on a chosen gradient descent type, there is a trade-off between the model's accuracy and the training computational time [39].

3.7.1 Batch Gradient Descent

Batch gradient descent computes the gradient of the loss function using the entire training set. The formula for updating each of the parameters is following:

$$w_i = w_i - \eta \frac{\partial}{\partial w_i} L(w) \tag{3.9}$$

Even though it is the most accurate method, the computational time could be extremely high since the gradient has to be computed in every single iteration. It also does not allow for online learning, which means adding new data to the training set on the fly. In this case, the whole neural network would have to be retrained. [39]

3.7.2 Stochastic Gradient Descent

The introduction of the Stochastic gradient descent reduced the high computational time of the Batch Gradient Descent. It computes the gradient of the loss function using only one randomly picked training sample at a time.

$$w_i = w_i - \eta \frac{\partial}{\partial w_i} L_j(w) \tag{3.10}$$

However, using just one training sample at a time can lead to parameter updates having a higher variance, which causes the loss function to fluctuate more on each iteration, thus making it harder to converge to the local minimum. To overcome this potential problem, the learning rate is decreased over time. The Stochastic Gradient Descent can be used for online learning. [39][7]

3.7.3 Mini-batch Gradient Descent

The Mini-batch Gradient Descent combines the advantages of the Batch Gradient Descent and the Stochastic Gradient Descent. It performs an update for every mini-batch of n training samples.

$$w_i = w_i - \eta \frac{\partial}{\partial w_i} L_{(j:j+n)}(w) \tag{3.11}$$

Using fewer training samples than BGD speeds up the training process. On the other hand, using more training examples than SGD reduces the variance of the parameter updates, contributing to a smoother convergence. The mini-batch size depends on each application's needs. [39]

3.8 Convolutional Neural Networks

The Convolutional Neural Network (CNN) is a particular type of neural network mostly used when working with images. It can solve tasks like image classification, pattern recognition, or image segmentation. Nevertheless, CNN is not limited to images only, as it is also used in voice recognition. The CNNs follow the same principles as traditional Artificial Neural Networks (ANNs). However, they add new layers that deal with the dimensionality reduction of the input data while preserving the essential features. This is a massive advantage over the traditional ANNs as they would struggle with the computational complexity required to process the high-dimensional data. They would also lose the spatial information as they flattened the input data to a vector. The CNNs generally consist of one or more convolutional layers, pooling layers, and finally, a fully connected layer [33][3]. An example of such architecture can be seen in the Figure 3.7.

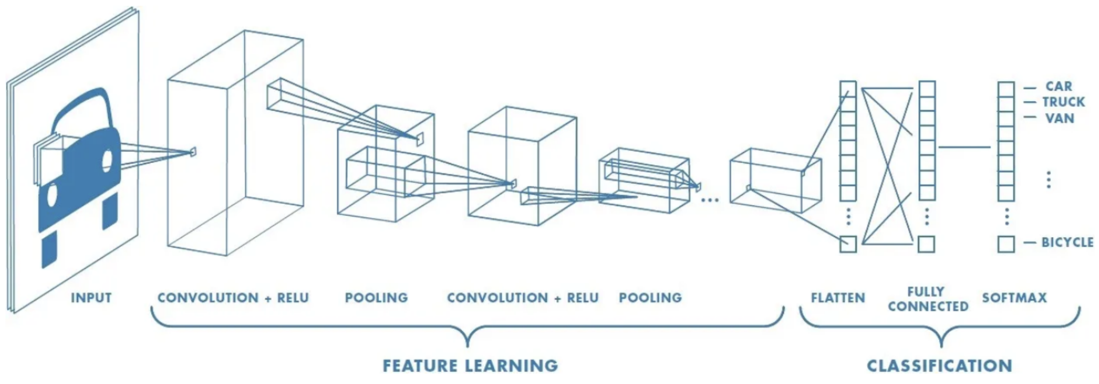

Figure 3.7: A general example of CNN architecture [40].

The convolutional layer plays a vital role in how CNNs operate. It consists of a set of learnable kernels, sometimes called convolutional filters. The kernels are usually relatively small square matrices following the depth of the input image (e.g., the depth of the RGB image is 3). Each kernel is applied by sliding over the input image with a given stride and computing the dot product between the kernel and the currently pooled part of the image (Figure 3.8). The result is then stored in a 2D feature map. The main objective of the convolutional layer is to extract spatial features from the input image. Typically, the first convolutional layer extracts quite simple features such as edges. However, by stacking multiple convolutional layers, the CNN can learn more complex features. [33][3][40]

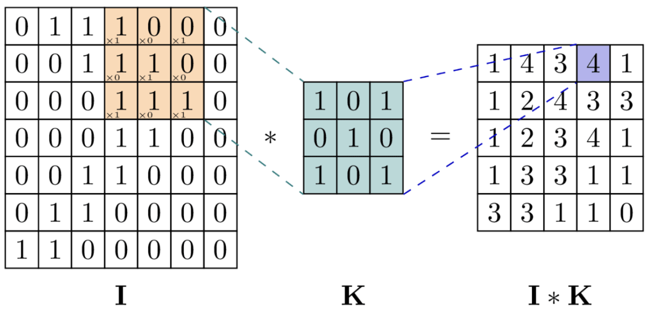

Figure 3.8: Image convolution [37].

Usually, every convolutional layer is followed by a pooling layer, whose job is to reduce the feature map's dimensionality and thus decrease the computational complexity. In most CNNs, the pooling layers use a kernel size of 2×2 and stride of 2. This scales the feature map down to 25 % of its original size as it maps 4 pixels into one. The standard pooling layer is the max pooling layer, which simply takes the maximum value from the portion of the image covered by the kernel. [33][3][40]

3.9 CNN Architectures

Over the past decade, many CNN architectures have been presented with various modifications. While modifications to the architectures, such as structural reformulation, regularization, and parameter optimization, have played a role in improving performance, the most significant advancements have come from processing unit reorganization and the development of novel blocks. Processing unit reorganization refers to the optimization of hardware for CNN computations. This includes using specialized hardware, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), to accelerate the computation of CNNs. These advancements have allowed for the processing of larger datasets and deeper architectures. Novel blocks refer to the building blocks of CNN architectures, which have been improved to better capture essential features in the input data. The most significant development has been using deeper architectures, allowing for better feature representation. In the following section ResNet model will be described as it is used later for experiments. However, many widely used models exist, such as VGGNet, DensNet, and EfficientNet. They all differ in their input size, depth, and robustness. Understanding these architectures' features can help engineers choose the most suitable one for their task. [2]

3.9.1 ResNet

ResNet (Residual Network) was first introduced in 2015 by four Microsoft researchers. The idea is described in their paper „Deep Residual Learning for Image Recognition" [23]. The objective was to design a deep neural network that would not suffer from the vanishing gradient compared to other CNN architectures that started encountering such difficulties as more layers were added. ResNet uses residual blocks (Figure 3.9) to build the network. The residual block contains one or more convolutional layers alongside a skip connection that adds the input of the residual block (in other words, output from the previous layers) to the output features created by the residual block. The skip connection allows setting up an alternate shortcut for the gradient to pass through. This enables the network to learn the residual mappings, i.e., the difference between the input and output of the convolutional layers. Doing so shows that the network converges much more easily despite having hundreds of layers. Resnet comes in many variants, including ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The number of layers in the network comes as a part of the model's name, e.g., ResNet-50 has 50 layers [2][8].

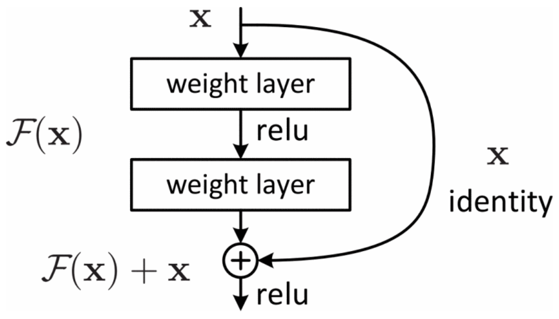

Figure 3.9: Residual block [23].

3.10 Object Detection with CNNs

Object detection is a fundamental task in computer vision that focuses on identifying and locating objects within an image or video frame. It plays a crucial role in various applications, including autonomous vehicles, robotics, video surveillance, and image recognition systems. Over the years, object detection has evolved significantly, and Convolutional Neural Networks (CNNs) have become the backbone of modern object detection methods. These deep learning models have demonstrated superior performance compared to traditional machine learning techniques. Object detection models can generally be categorized into two types [38] [31]:

- **Single-stage networks** are designed for speed and efficiency. These models directly predict object locations and class probabilities in a single forward pass without requiring a separate region proposal step. Popular single-stage networks include YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector).
- **Two-stage networks** divide the object detection task into two stages: a region proposal step and a classification and bounding box regression. This approach allows for more accurate localization of objects at the cost of increased complexity and computation time. Examples of two-stage networks include R-CNN (Region-based Convolutional Neural Networks) and its variants, such as Fast R-CNN and Faster R-CNN.

3.10.1 Faster R-CNN

Faster R-CNN is a deep convolutional network designed for object detection, which evolved from its predecessors, R-CNN and Fast R-CNN. R-CNN is the first attempt at an object detection model that uses a pre-trained CNN to extract the features from generated region proposals made by a Selective search algorithm. However, the R-CNN suffers from slow processing and high storage requirements. Fast R-CNN improved upon R-CNN by implementing the ROI (region of interest) Pooling layer, enabling shared computations across region proposals, which increased speed and reduced storage needs. However, Fast R-CNN still relies on the time-consuming Selective Search algorithm. Faster R-CNN addresses this limitation by introducing the Region Proposal Network (RPN), a fully convolutional network that generates region proposals. The final architecture of Faster R-CNN comprises two main modules: the RPN, which generates region proposals, and Fast R-CNN, which detects objects within these proposed regions. The process is illustrated in Figure 3.10. [17]

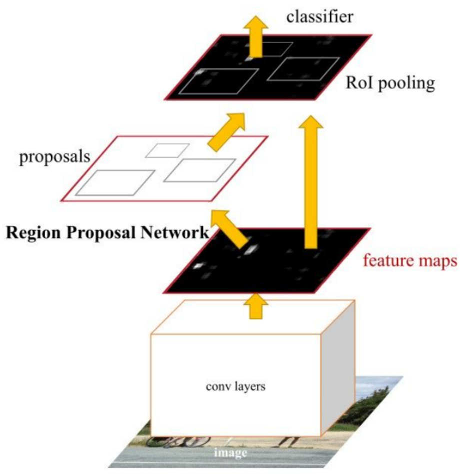

Figure 3.10: Faster R-CNN architecture [17].

3.10.2 CenterNet

CenterNet is an object detection architecture (Figure 3.11) , first published in 2019, that does not rely on anchors (predefined bounding boxes of specific size and shape), typically used in other detection models. Anchor-based models generate many incorrect predictions, which have to be filtered out by the NMS (Non-Maximum Suppression). Instead, CenterNet uses a more efficient method to eliminate irrelevant predictions, speeding up the detection process. The model generates a confidence heatmap highlighting objects' centers in an image. CenterNet can discard unnecessary predictions without needing computationally intensive NMS techniques by focusing on object centers. In addition to predicting a confidence heatmap that shows object centers, CenterNet also has a second part, similar to other detectors, that estimates the size (width and height) and position (x- and y-coordinates) of the bounding boxes relative to the object centers determined by the heatmap. By working together, these two components enable accurate and efficient object detection without the need for anchor boxes. [1]

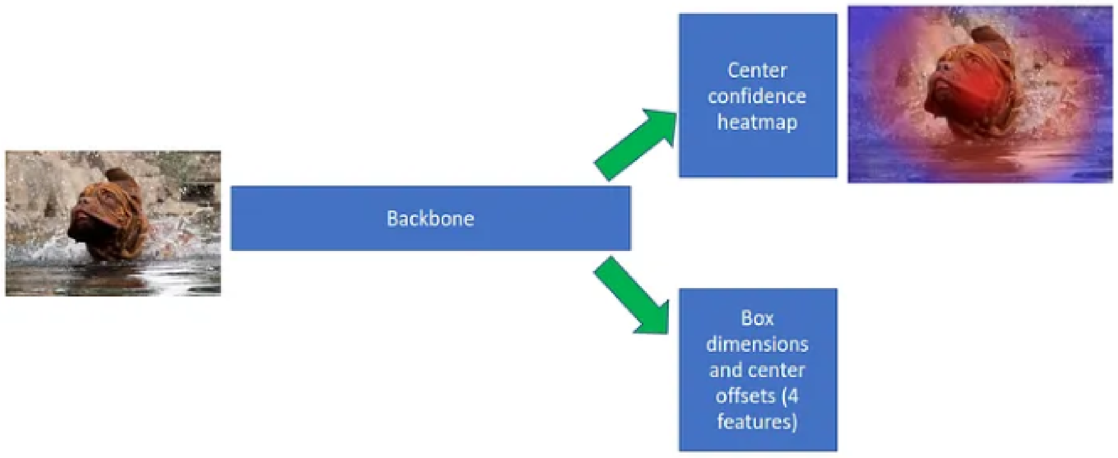
Center confidence heatmap
Backbone
Box dimensions and center offsets (4 features)

Figure 3.11: CenterNet architecture [1].

Chapter 4

Solution Proposal and Implementation

This chapter presents a solution for detecting and classifying line damage and moisture effects in fingerprint images, outlining the implementation process. Initially, the prerequisites for training the convolutional neural networks are discussed, which involve several crucial steps. These include acquiring a dataset of healthy synthetic master prints, introducing damage to those master prints, and generating annotation files containing essential information about the damage, such as the bounding boxes and the respective damage class.

Subsequently, the chapter delves into training the three most performant combinations of CNN architectures and object detection approaches, as identified in Katerina Fortova's work [16]. Each model will undergo four training iterations, with variations in the dataset, hyperparameters, or both. This iterative approach aims to progressively enhance the models' performance and observe the effects of the individual adjustments.

4.1 Fingerprint Images Dataset

For the purpose of this work, a generated dataset of 3000 synthetic master prints was provided by the STRaDe research group from the Brno University of Technology. To augment the number of samples, the dataset was expanded by applying a horizontal flip to each image and introducing a minor rotation of up to 10 degrees in either direction with a 50 % likelihood for increased variation. Given that this work addresses two types of damage, the final dataset of 6000 images was partitioned into three distinct groups. The first group consists of 2900 images dedicated to moisture generation, while the second group, also containing 2900 images, is reserved for line damage generation. The residual 200 undamaged images are retained for subsequent testing. The distribution of these groups is illustrated in the Table 4.1.

Damage type	**Synthetic fingerprints**
Moisture	2900
Line damage	2900
Healthy	200
Total	6000

Table 4.1: Dataset split for damage generation.

The line damage generation is made possible by a python console application developed by Vanessa Joriova [25]. This application is capable of generating three previously mentioned types of line damage as described in Section 2.3.2: **hairs**, **creases**, and **scars**. Numerous configuration options are available for damage generation, such as length, width, intensity for creases, and patch or outline parameters for scars. The ultimate configuration applied to each sample is determined randomly, ensuring equal probability for each damage type.

A C++/Qt application created by Veronika Svoradova [43] is utilized for moisture generation. To enable execution by an external script for automated generation, the application was modified into a console application. Similar to the line damage generator, moisture intensity is randomly assigned for each sample.

Further, the damaged samples were divided into training, validation, and test sets in a 70:15:15 ratio (Table 4.2). The test set additionally incorporates 200 healthy samples. Two distinct datasets will be employed throughout the experiments to explore different approaches for generating bounding boxes, as explained in the following section.

Damage type	**Training set**	**Validation set**	**Test set**
Line damage	2030	435	435
Moisture	2030	435	435
Healthy	0	0	200
Total	4060	870	1070

Table 4.2: Dataset split resulting in training, validation, and test set.

4.2 Bounding Boxes

Bounding boxes represents a crucial part of training models to detect objects within images. While the line damage generator application did not initially provide bounding boxes, it did save the coordinates of the pixels where the damage was introduced. Consequently, the application was modified to make use of this information and generate bounding boxes accordingly.

The first and more straightforward approach to creating bounding boxes involved taking the damaged pixels and rendering them on a black canvas in white. Then, a *bitwise_and* operation was performed between the damaged canvas and the binary mask to eliminate damaged pixels outside the fingerprint (Figure 4.1). In the final step, the bounding box was created by finding the minimum and maximum coordinates of the remaining damaged pixels. This approach is illustrated in Figure 4.2.

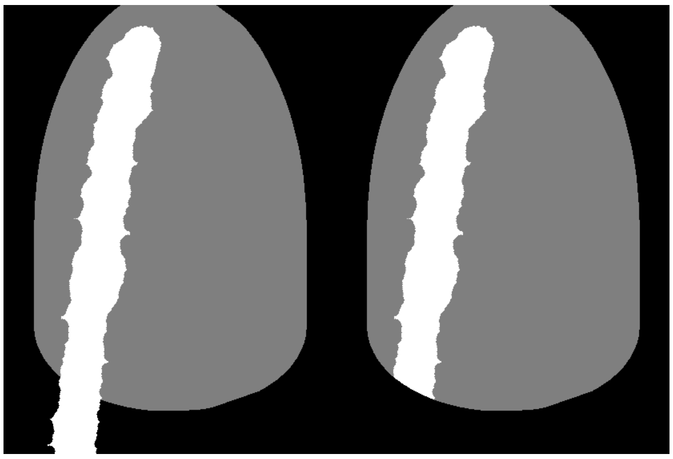
Figure 4.1: All damaged pixels (left) and damaged pixels inside the fingerprint (right).

The previously mentioned approach, while simplistic, raises concerns regarding its efficiency in handling specific scenarios, such as when a lengthy hair is generated diagonally across the fingerprint or when multiple small creases are dispersed throughout the fingerprint area. In such cases, the resulting bounding box could be disproportionally large, encompassing nearly the entire fingerprint. This excessive size could negatively impact the model's performance, diminishing its capacity to accurately detect and classify the relevant damage features within the fingerprint images.

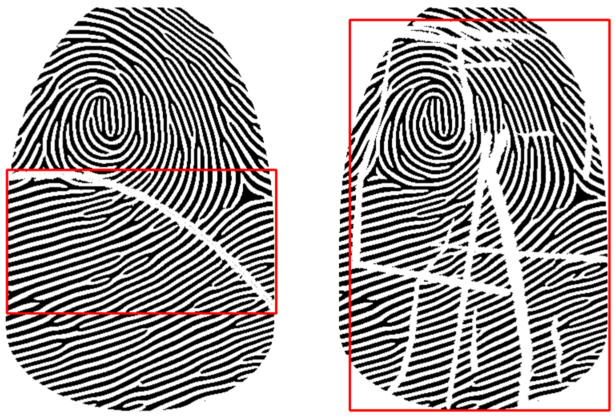
Figure 4.2: Bounding boxes for hair damage (left) and creases (right).

To address this potential issue, a second approach was developed. Initially, damaged pixels within the fingerprint were identified and clustered based on their connectivity to other damaged pixels. In essence, each cluster represents a group of damaged pixels surrounded by an undamaged area. These clusters were then filtered according to the ratio of damaged to undamaged area within their bounding boxes. A threshold of 0.5 was established, implying that a cluster's bounding box must contain at least 50 % damaged pixels. Accepted clusters were then removed from the damaged canvas.

The remaining clusters that were not immediately accepted underwent further processing using a separate algorithm. This algorithm iteratively examines lines from the top to the bottom of the bounding box, encompassing all remaining damage. For each line, the left and right extremes are identified. The left extreme is defined as the first damaged coordinate at the transition between undamaged and damaged pixels. In contrast, the right extreme is the last damaged coordinate at the transition between damaged and undamaged pixels. After examining a fixed number of lines, in this case, 12, the extremes are divided into groups. This step is primarily employed to handle creases, as multiple creases may appear on the same line. In contrast, there is typically only one group for hairs or scars since the damage is considered as a whole.

Using the extremes from each group and the processed lines, the final bounding box is created by calculating the `min_x`, `max_x`, `min_y`, and `max_y` coordinates. The algorithm continues until the bottom of the bounding box (encompassing all the damage) is reached. The outcome of this approach is illustrated in Figure 4.3 and can be compared to the Figure 4.2.

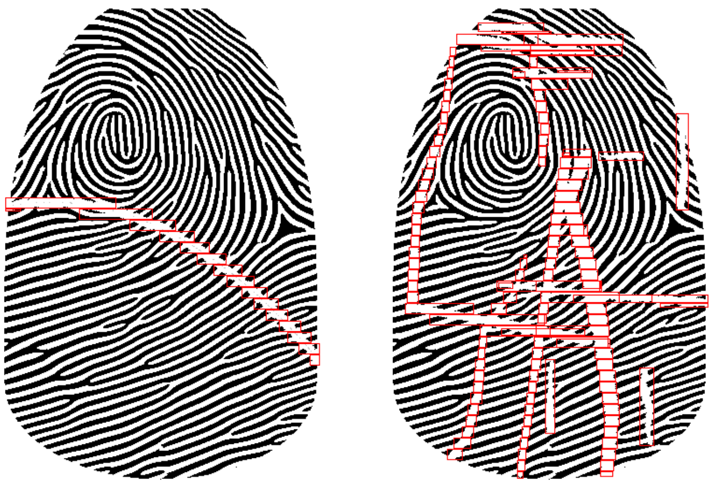

Figure 4.3: Result of a different approach for finding bounding boxes: hair damage (left) and creases (right).

Conversely, the moisture generator application supplied a data structure comprising all bounding boxes for the generated damage. However, visualizing these bounding boxes made it evident that they were not always accurate. Some bounding boxes were depicted outside the fingerprint, while others heavily overlapped, sharing the same area (Figure 4.4). Although the latter issue might not pose significant problems, efforts were made to reduce the number of overlapping bounding boxes to improve the overall accuracy and efficiency of the model training process.

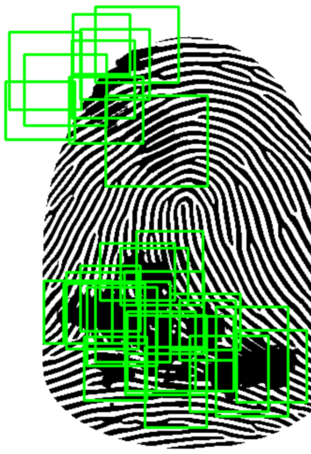
Figure 4.4: Original moisture bounding boxes.

The process of modifying moisture bounding boxes was more straightforward compared to that of line damage. The first step involved cropping any bounding boxes that extended beyond the fingerprint's bounding box. The second step removed all bounding boxes whose area remained predominantly outside the fingerprint, with a removal threshold set at approximately 70 %. The final step consisted of performing ten iterations of merging bounding boxes with at least 70 % intersection in their areas. The number of iterations was determined through experimentation, as additional iterations did not contribute to further merging.

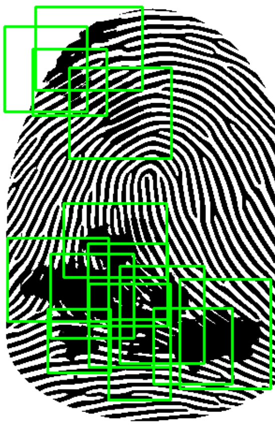
Figure 4.5: Modified moisture bounding boxes.

Throughout the process of handling bounding boxes in both the Python line damage generator and the C++ moisture generator, the OpenCV computer vision library was employed as a valuable tool to facilitate many of the mentioned tasks. [9]

4.3 Data Annotations

In order to utilize the models from Katerina Fortova's work [16], it is necessary to adhere to a specific annotation format. These annotations are stored in XML files, with each file containing information about its corresponding image in the `<file>` tag, the image dimensions in the `<size>` tag, and a list of bounding boxes in the `<object>` tag. Each bounding box is defined by its coordinates `<xmin>`, `<ymin>`, `<xmax>`, and `<ymax>` and a `<name>` tag that specifies the type of damage enclosed within the bounding box. The following listing presents an example of an XML file containing annotations for a single moisture image:

```
<annotation>
  <filename>moisture_SG_1689_2.png</filename>
  <size>
    <width>416</width>
    <height>560</height>
  </size>
  <object>
    <name>moisture</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>95</xmin>
      <ymin>144</ymin>
      <xmax>134</xmax>
      <ymax>185</ymax>
    </bndbox>
  </object>
  <object>
    <name>moisture</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>147</xmin>
      <ymin>196</ymin>
      <xmax>223</xmax>
      <ymax>279</ymax>
    </bndbox>
  </object>
</annotation>
```

To simplify the process and given that the generators were written in different programming languages, the generators were designed to output the coordinates of bounding boxes to the standard output. The script executing these generators processed this output, generating the final XML file.

4.4 Used Models

As previously mentioned, the models utilized in this work are the three best-performing combinations of backbone CNNs for feature extraction and object detection approaches. These combinations were selected based on their performance on a dataset containing real and synthetic fingerprints in [16] master thesis. The list is as follows:

- Faster R-CNN ResNet50 V1 640×640
- Faster R-CNN ResNet101 V1 640×640
- CenterNet Resnet101 V1 FPN 512×512

4.4.1 Model Configuration

Each model has its own configuration file called `pipeline.config`, which stores all the information about the model, such as architecture, hyperparameters, and data input settings. The configuration files may differ from model to model due to their unique architectures. However, some core parameters are identical for all of them. Almost all parameters adjusted in attempts to improve the results are shared between the models. The only exception is the `use_dropout` along with the `keep_dropout_probability` parameter, which are not present in the CenterNet approach.

- `batch_size`: determines the number of training samples processed at once during a single pass through the model, in other words, the number of samples used for one update of the network's weights. Larger batch sizes can lead to a smoother convergence but may require more memory, especially if the model is large. A smaller batch size can produce noisier gradient estimates, but it can also help to avoid local minima, potentially leading to better solutions.
- `num_steps`: the overall number of training steps to perform.
- `learning_rate`: all models trained in this work use cosine learning rate decay. This method is slowly decreasing the learning rate with increased steps performed [12]. The following parameters define the configuration of the decay:
  - `total_steps`: specifies the total number of steps over which the cosine decay, even using the warmup phase, is performed. In the case of this work, it is always equal to the `num_steps` parameter.
  - `learning_rate_base`: is the initial learning rate value used by the cosine decay learning rate. The learning rate will gradually decrease from this value over the training steps.
  - `warmup_learning_rate`: is the initial learning rate used during the warmup phase of training. It is lower than the base learning rate, allowing the model to gradually adapt to the training data before larger updates are applied.
  - `warmup_steps`: specifies the number of steps the model takes during the warmup phase. During this period, the learning rate linearly increases from the warmup learning rate to the base learning rate.

- `use_dropout`: a boolean parameter indicating whether a dropout should be used during training. Dropout is a regularization technique that helps prevent overfitting by randomly setting a portion of the neurons to zero during the training process. If a neuron is dropped, other neurons have to step in handle the correct feature representation. This can lead to a network capable of better generalization [10].

- `keep_dropout_probability`: represents a probability that a neuron will be kept during training. In this work, the probability is set to 0.75, meaning that each neuron has a 75 % chance of being kept during training.

4.4.2 Model Evaluation Metrics

The master thesis [16] provides various evaluation metrics to assess the model's performance. For the purpose of this work, the following metrics were chosen:

- **Average correctly detected and classified area**: This metric represents the percentage size of the correctly detected and classified area relative to all annotated bounding boxes. The implementation involves calculating the intersection of all detected and annotated bounding box pixels for correct classification. If the metric's value is 100 %, then the model would also correctly predict all annotated bounding boxes. Since fingerprints without damage have no annotations in their annotation files and do not contain any damaged bounding boxes, a 0 % result is obtained if any bounding box prediction is made. In contrast, a 100 % result is achieved if no bounding box is predicted.

- **Average not detected annotated area**: This metric measures the percentage size of the annotated area within bounding boxes that the trained model did not detect during evaluation.

- **Average extra detected area**: This metric indicates the percentage of pixels in predicted bounding boxes that were detected in addition to all annotated bounding boxes. The implementation calculates the difference between the pixels of the predicted bounding boxes and the pixels of the annotated bounding boxes. For example, if the value of this metric was 15 %, then 15 % of the area of all predicted bounding boxes had not been annotated.

- **Average detection score of a correctly detected and classified area**: When a trained model predicts a bounding box, it assigns a detection score alongside the prediction. The detection score signifies the confidence level of the trained model regarding the classification of a bounding box in a given class. A detection score is obtained for each correctly detected and classified pixel. However, a given pixel may be part of multiple predicted bounding boxes with different detection scores, in which case only the highest detection score of all possible occurrences is considered. Subsequently, all correctly detected and classified pixels and their detection scores are summed, and the average is calculated.

4.5 Experiments

This final section of the chapter delves into the experiments conducted with the models referenced in Section 4.4. The subsections are primarily centered around experimenting with a single architecture, employing various configuration parameters or datasets, specifically regarding the annotation's bounding box types. Each experiment is assigned a unique ID, which is used in tables and figures for identification purposes. Every experiment undergoes a new training process, followed by testing and evaluation. Within each experiment, a table outlining the configuration parameters for each model is presented, alongside a table that evaluates the models based on the metrics listed in Section 4.4.2. Due to the lengthy names of metrics, tables use the following abbreviations:

- AVG correct D&C - Average correctly detected and classified area
- AVG not detected - Average not detected annotated area
- AVG extra detected - Average extra detected area
- AVG detection score D&C - Average detection score of a correctly detected and classified area

It is important to mention that two distinct datasets for training are utilized throughout the experiments. The first dataset comprises fingerprints damaged by the methods depicted in Figures 4.2 and 4.4, whereas the second dataset employs approaches illustrated in Figures 4.3 and 4.5.

During the evaluation phase of the experiments, the model assesses each image within the test dataset. For every analyzed image, the predicted bounding boxes are drawn and marked with their associated damage type and detection score. Sample images illustrating these predictions can be found in Appendix B.

4.5.1 Faster R-CNN ResNet50 V1 640×640 Experiments

This section explores experiments with Faster R-CNN combined with a ResNet50. The first experiment was configured identically to the master thesis [16]. In the second experiment, several parameters were modified, including adding 5 000 training steps, reducing the base and warmup learning rates, and introducing a 25 % chance of dropping a neuron during training. Attempts were made to increase the batch size to values like 16 or 32; however, this was impossible due to GPU memory limitations. Therefore, the batch size was increased only by two samples. Both experiments were trained on the first dataset, and their results did not significantly differ. Nevertheless, as the first experiment took less time to train, the final experiment using the second dataset was configured in the same manner as the first experiment. Individual configurations are presented in Table 4.3.

ID	lr_base	num_steps	warmup_lr	warmup_steps	batch_size	dropout	dataset
1	0.04	25 000	0.013333	2 000	2	-	first
2	0.002	30 000	0.0002	3 000	4	0.25	first
3	0.04	25 000	0.013333	2 000	2	-	second

Table 4.3: Faster R-CNN ResNet50 V1 640×640 training configurations.

The outcomes of the experiments can be found in Tables 4.4, 4.5, and 4.6. Among the three experiments, the first one had the best results, achieving an average correctly detected and classified area of 89.277 %. The second experiment followed closely with 87.522 %. Although it demonstrated improved moisture accuracy, it misclassified 10.5 % of healthy fingerprints as damaged. The final experiment, which was trained on the second dataset, ended up with the lowest total accuracy among the three. Its line damage accuracy was significantly worse, falling short by 20 %. However, the moisture detection performance was increased quite a bit, with an added accuracy of approximately 6-7 % compared to the other two experiments. It is important to highlight that when comparing the outcomes of the first and third experiments, which differ solely in the datasets used, the confidence in predictions on the first dataset was higher, even for moisture detection, where the accuracy was lower.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	92.621	81.004	100.0	89.277
AVG not detected	7.379	18.996	-	10.723
AVG extra detected	3.895	3.939	-	3.185
AVG detection score D&C	96.223	92.678	-	76.796

Table 4.4: Metrics results for ID 1.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	92.443	81.693	89.5	87.522
AVG not detected	7.557	18.307	-	12.478
AVG extra detected	4.618	4.408	-	3.67
AVG detection score D&C	96.689	90.995	-	76.302

Table 4.5: Metrics results for ID 2.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	72.183	87.902	98.5	83.493
AVG not detected	27.817	12.098	-	16.507
AVG extra detected	21.574	5.7	-	11.088
AVG detection score D&C	70.368	89.079	-	64.822

Table 4.6: Metrics results for ID 3.

4.5.2 Faster R-CNN ResNet101 V1 640×640 Experiments

This section is experimenting with the combination of Faster R-CNN and ResNet101. All the experiments have the same configuration as ResNet50 in the previous section. While there was a slight improvement in the second experiment, the third one, which employs the second dataset, utilized the configuration from the first experiment once more, primarily due to time complexity considerations (Table 4.7).

ID	**lr_base**	**num_steps**	**warmup_lr**	**warmup_steps**	**batch_size**	**dropout**	**dataset**
4	0.04	25 000	0.013333	2 000	2	-	first
5	0.002	30 000	0.0002	3 000	4	0.25	first
6	0.04	25 000	0.013333	2 000	2	-	second

Table 4.7: Faster R-CNN ResNet101 V1 640×640 training configurations.

The results of these experiments can be found in Tables 4.8, 4.9, and 4.10. In this case, using ResNet101, the modified parameters in the second experiment outperformed the original configuration, although it resulted in a minor difference of 0.5 % and a total accuracy of 90.646 %. The third experiment, which utilized numerous small bounding boxes for line damage annotation, created an even larger gap in the model's performance compared to ResNet50. The line damage accuracy dropped to 70.596 %, a 25 % decrease compared to the first two experiments that used a simpler bounding box approach.

When comparing the best results of ResNet50 and ResNet101, the latter surpassed the former by roughly 3 % in line damage accuracy, leading ResNet101 to achieve a slightly higher overall accuracy. Furthermore, ResNet101 demonstrated greater confidence in its predictions than ResNet50, reaching almost 100 % confidence for the line damage class.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	95.555	80.637	99.0	90.134
AVG not detected	4.445	19.363	-	9.866
AVG extra detected	3.506	4.205	-	3.135
AVG detection score D&C	99.66	94.331	-	78.865

Table 4.8: Metrics results for ID 4.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	95.566	81.884	99.0	90.646
AVG not detected	4.434	18.116	-	9.354
AVG extra detected	3.884	4.385	-	3.361
AVG detection score D&C	99.301	92.486	-	77.969

Table 4.9: Metrics results for ID 5.

Metric	**Line damage**	**Moisture**	**Healthy**	**Total**
AVG correct D&C	70.596	88.145	100.0	83.227
AVG not detected	29.404	11.855	-	16.773
AVG extra detected	20.74	5.781	-	10.782
AVG detection score D&C	70.986	90.225	-	65.539

Table 4.10: Metrics results for ID 6.

4.5.3 CenterNet Resnet101 V1 FPN 512×512 Experiments

In the final experiments, CenterNet and ResNet101 were combined. As this model uses an input image size of 512 × 512 pixels, the number of steps was set to 60 000, anticipating a faster processing time. This is over twice the amount of steps used for the Faster-RCNN models. In the first experiment, the original configuration was left unchanged. The second experiment attempted to increase the batch size to a number greater than 4, given the lower image resolution. However, GPU limitations were exceeded once again. As a result, the second experiment increased the batch size from 2 to 4 and decreased the learning rate from 0.003 to 0.0005 and the warmup learning rate from 0.0003 to 0.00005. This change led to a significant improvement in prediction accuracy. Consequently, the third experiment, trained on the second dataset, utilized the configuration from the second experiment. The final confugurations are also displayed in Table 4.11.

ID	lr_base	num_steps	warmup_lr	warmup_steps	batch_size	dataset
7	0.003	60 000	0.0003	4 000	2	first
8	0.0005	60 000	0.00005	4 000	4	first
9	0.0005	60 000	0.00005	4 000	4	second

Table 4.11: CenterNet Resnet101 V1 FPN 512×512 training configurations.

The final experiments are displayed in the Tables 4.12, 4.13, and 4.14: The outcomes of the first experiment are rather disappointing compared to the initial experiments of Faster-RCNN models, with the total accuracy lagging behind by approximately 5 %. However, adjusting the parameters in the second experiment led to a significant improvement, adding over 5 % accuracy to line damage and 6 % accuracy to moisture detection, resulting in a total accuracy of 89.414 %. The third experiment, trained on the second dataset, did not work well with the CenterNet object detection approach. While the performance of Faster-RCNN models dropped by 7 %, here it dropped by a full 36 % compared to the best experiment using CenterNet. This is a very significant drop in performance, making the network unusable.

Metric	Line damage	Moisture	Healthy	Total
AVG correct D&C	88.313	74.006	100.0	84.681
AVG not detected	11.687	25.994	-	15.319
AVG extra detected	3.276	4.492	-	3.158
AVG detection score D&C	78.235	64.493	-	58.025

Table 4.12: Metrics results for ID 7.

Metric	Line damage	Moisture	Healthy	Total
AVG correct D&C	93.878	80.082	100.0	89.414
AVG not detected	6.122	19.918	-	10.586
AVG extra detected	2.311	4.047	-	2.585
AVG detection score D&C	91.064	71.59	-	66.126

Table 4.13: Metrics results for ID 8.

Metric	Line damage	Moisture	Healthy	Total
AVG Correct D&C	19.602	65.669	100.0	53.358
AVG not detected	80.398	34.331	-	46.642
AVG extra detected	8.262	4.703	-	5.271
AVG detection score D&C	41.109	64.853	-	43.078

Table 4.14: Metrics results for ID 9.

4.5.4 Summary of Achieved Results

This final section of the chapter presents a comprehensive summary of the results obtained from all experiments conducted. Three detection models, namely Faster-RCNN ResNet50, Faster-RCNN ResNet101, and CenterNet ResNet101, were each trained three times with varying parameters or datasets.

The primary objective was to enhance the performance of each model compared to the initial experiment, which utilized original parameters taken from [16]. The second experiment aimed to augment performance by modifying the hyperparameters of the trained models, whereas the third experiment sought to further boost performance by adopting a more sophisticated approach for annotating bounding boxes.

The average correctly detected and classified area serves as the most informative metric and is thus employed to draw final conclusions. The Figure 4.6 illustrates the comparisons between the first and second experiments. Alterations in hyperparameters for the Faster-RCNN models exhibited negligible impact. Utilizing ResNet50 as a backbone resulted in a minor performance decline while employing ResNet101 led to a slight performance increase. Conversely, the CenterNet model demonstrated a significant performance improvement, achieving the greatest enhancement among the three models.

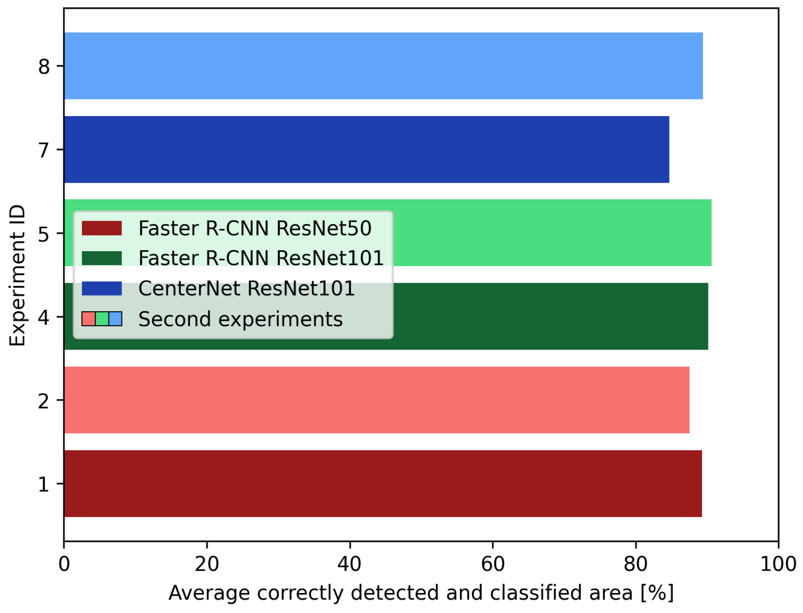

Figure 4.6: Performance of the first and second experiments.

For the third experiment, the hyperparameters were selected based on superior performance in either the first or second experiments. The sole distinction was the method employed for annotating bounding boxes. However, the outcomes deviated significantly from initial expectations. Both Faster-RCNN models displayed a noticeable decline in total accuracy, and the CenterNet model experienced a drastic performance drop. These results are depicted in Figure 4.7, where the abovementioned experiments can be seen under IDs 3, 6, and 9.

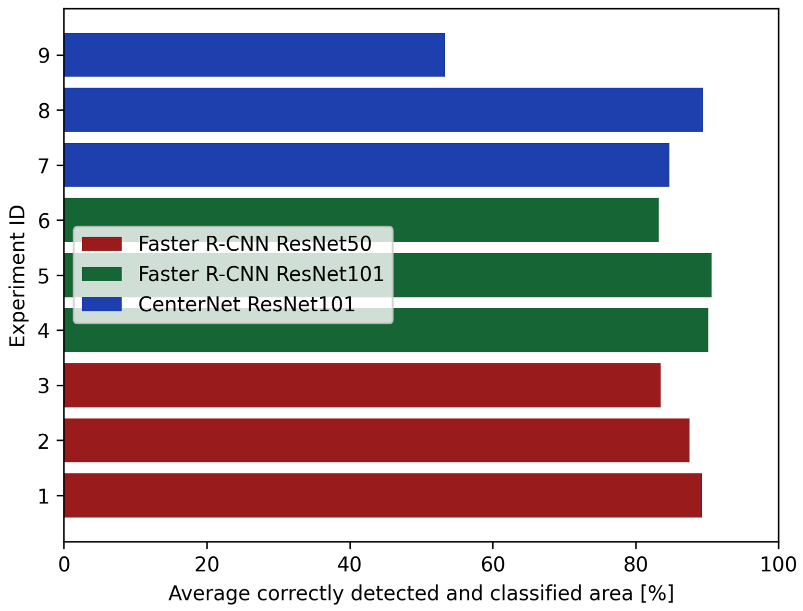

Figure 4.7: Average correctly detected and classified area for each experiment.

The suboptimal performance of third experiments is attributed to the approach adopted for annotating line damage, as illustrated in Figure 4.8. This method employed numerous small bounding boxes to cover line damage, minimizing the extent of the healthy area being annotated. The issue likely stems from the object detection models treating each bounding box as a separate object, thereby failing to learn the global pattern of line damage.

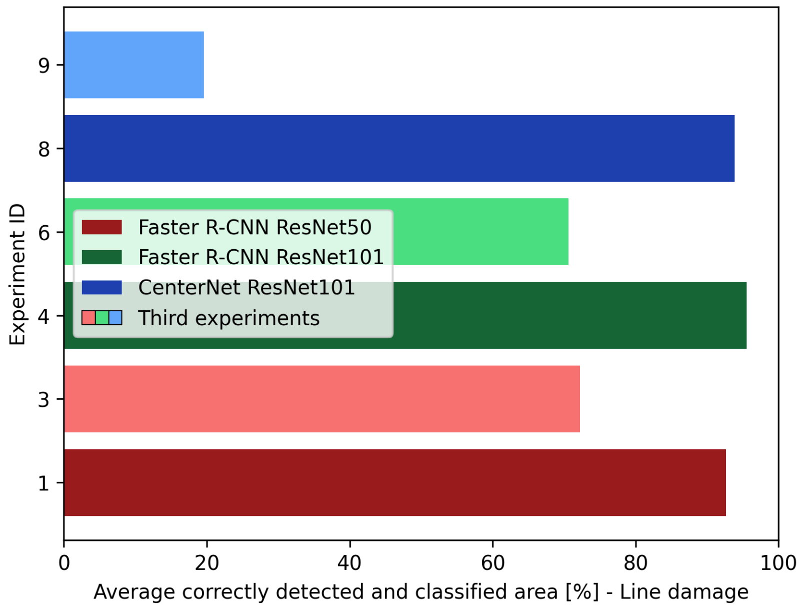

Figure 4.8: Comparison of third experiments for line damage detection.

While the endeavor to enhance annotations for line damage detection proved unsuccessful, the attempt to refine moisture bounding boxes yielded positive results. The performance of Faster-RCNN ResNet50 and ResNet101 models improved, as evidenced by Figure 4.9.

The CenterNet model did not benefit from the altered moisture annotation approach, likely due to the heavy influence of line damage fingerprint images.

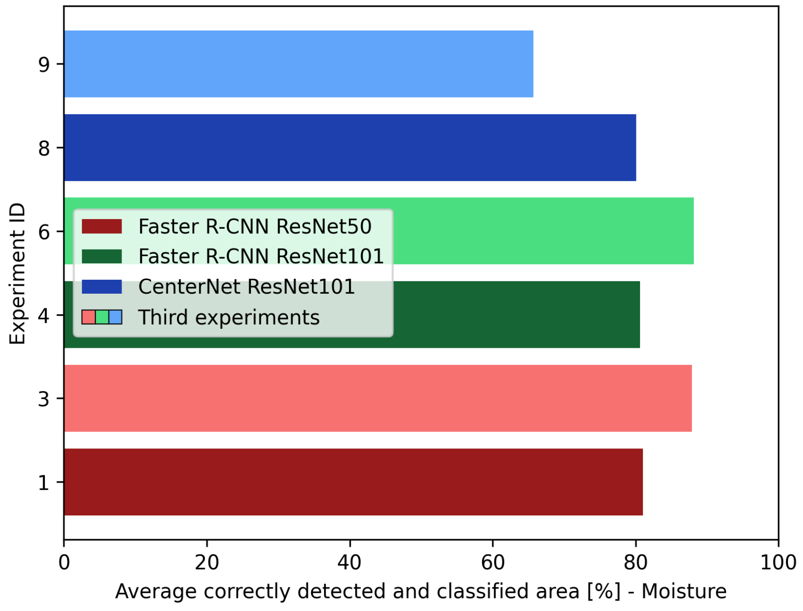

Figure 4.9: Comparison of third experiments for moisture detection.

Chapter 5

Conclusion

The primary goal of this bachelor thesis was to investigate three existing object detection models and develop enhancements for the task of detecting and classifying line damage and moisture in fingerprint images. The Faster-RCNN ResNet50, Faster-RCNN ResNet101, and CenterNet ResNet101 models were chosen for experimentation in order to obtain the most optimal results. The experiments focused on improving each model's performance by adjusting hyperparameters and exploring alternative methods of annotating bounding boxes, which were incorporated into applications capable of generating line damage and moisture.

Although some models, such as CenterNet ResNet101, demonstrated substantial performance improvements upon hyperparameter adjustment, the attempt to refine annotations for line damage detection was unsuccessful. This was attributed to the object detection models treating each bounding box as a separate entity, which prevented the models from learning the global pattern of line damage. In contrast, refining moisture bounding boxes enhanced performance for Faster-RCNN ResNet50 and ResNet101 models. Moreover, the Faster R-CNN with ResNet101 model delivered the best overall performance, achieving a total accuracy of 90.646 % and the highest confidence when assigning a bounding box to a class, with an average of 78.865 %.

To further improve the performance of the models, a comprehensive revision of the bounding box annotation system would be recommended. Instead of employing small bounding boxes to cover line damage, a single bounding box could encompass the entire line damage area, utilizing a rotated bounding box for diagonal lines. In the case of creases, each crease would have its own bounding box. Additional performance improvements could be realized through further hyperparameter experimentation, particularly with batch size and learning rate, as all models experienced convergence difficulties. However, in the case of batch size, such experimentation would necessitate more powerful GPUs with increased memory capacity or multi-GPU training.

Alternatively, considering a different approach, such as segmentation or pixel-wise classification, may be more suitable for this task. Assigning a class label to each pixel in the image rather than detecting distinct objects would enable the model to identify damaged fingerprint regions without requiring bounding boxes. U-Net and DeepLab are popular architectures for similar purposes and may be worth exploring.

Bibliography

[1] Almog, U. *CenterNet*. 2021. Available at: https://towardsdatascience.com/centernet-explained-a7386f368962.

[2] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al Dujaili, A., Duan, Y. et al. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*. Springer. 2021, vol. 8, p. 1–74. DOI: https://doi.org/10.1186/s40537-021-00444-8.

[3] Amini, A. *MIT 6.S191: Convolutional Neural Networks*. 2022. Available at: https://www.youtube.com/watch?v=uapdILWYTzE.

[4] Baheti, P. *Activation Functions in Neural Networks*. Available at: https://www.v7labs.com/blog/neural-networks-activation-functions.

[5] Bansal, R., Sehgal, P. and Bedi, P. Minutiae extraction from fingerprint images-a review. *ArXiv preprint arXiv:1201.1422*. 2011.

[6] Bentley, P. *How does a smartphone 'read' my fingerprint?* Available at: https://www.sciencefocus.com/science/smartphone-fingerprint-scanners/.

[7] Bento, C. *Stochastic Gradient Descent explained in real life*. 2021. Available at: https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32.

[8] Boesch, G. *Deep Residual Networks (ResNet, ResNet50)*. 2023. Available at: https://viso.ai/deep-learning/resnet-residual-neural-network/.

[9] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

[10] Brownlee, J. *Dropout Regularization in Deep Learning Models with Keras*. 2022. Available at: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/.

[11] Cooney, L. *Introduction to Fingerprint Evidence*. Available at: https://projects.nfstc.org/ipes/presentations/Cooney-prints.pdf.

[12] Correa, S. *Cosine Learning rate decay*. 2019. Available at: https://scorrea92.medium.com/cosine-learning-rate-decay-e8b50aa455b.

[13] Drahanský, M. *Biometrické systémy - Studijní opora*. Available at: https://www.fit.vutbr.cz/study/courses/BIO/private/BIO_Studijni_opora.pdf.

[14] Drahanský, M. *Hand-Based Biometrics: Methods and Technology*. The Institution of Engineering and Technology, 2018. 1–430 p. ISBN 978-1-78561-224-4. Available at: https://www.fit.vut.cz/research/publication/11556.

[15] Dylevský, P. *Kožní ústrojí - Anatomie kůže*. Available at: https://vos.palestra.cz/skripta/anatomie/12a1.htm.

[16] Fořtová, K. *Analýza konvolučních neuronových sítí pro detekci a klasifikaci poškození otisku prstu*. Brno, CZ, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/25006/.

[17] Gad, A. F. *Faster R-CNN Explained for Object Detection Tasks*. 2021. Available at: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/.

[18] Greenberg, S., Aladjem, M., Kogan, D. and Dimitrov, I. Fingerprint image enhancement using filtering techniques. In:. February 2000, vol. 15, p. 322–325 vol.3. DOI: https://doi.org/10.1109/ICPR.2000.903550. ISBN 0-7695-0750-6.

[19] Grujičić, R. *Papillary layer of dermis*. Available at: https://www.kenhub.com/en/library/anatomy/papillary-layer-of-dermis.

[20] Günther, F. and Fritsch, S. Neuralnet: training of neural networks. *R J.* 2010, vol. 2, no. 1, p. 30.

[21] Gupta, D. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* 2020. Available at: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[22] Géron, A. *Neural networks and deep learning*. O'Reilly Media, Inc., 2018. Available at: https://learning.oreilly.com/library/view/neural-networks-and/9781492037354/.

[23] He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, p. 770–778. DOI: https://doi.org/10.1109/CVPR.2016.90.

[24] Jain, A. and Pankanti, S. Chapter 23 - Fingerprint Recognition. In: Bovik, A., ed. *The Essential Guide to Image Processing*. Boston: Academic Press, 2009, p. 649–676. DOI: https://doi.org/10.1016/B978-0-12-374457-9.00023-8. ISBN 978-0-12-374457-9. Available at: https://www.sciencedirect.com/science/article/pii/B9780123744579000238.

[25] Jóriová, V. *Generování čárových poškození do syntetického otisku prstu*. Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/25005/.

[26] Jun, W. J. *How Does an Optical Fingerprint Sensor Work?* Available at: https://www.youtube.com/watch?v=CLdrbn8XYIw.

[27] Kanich, O., Košťák, D. and Drahanský, M. Psoriasis Damage Simulation into Synthetic Fingerprint. In: *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*. 2019, p. 1–4. ISSN 1617-5468.

[28] Koech, K. E. *Cross-Entropy Loss Function.* 2020. Available at: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[29] Kolarsick, P. A., Kolarsick, M. A. and Goodwin, C. Anatomy and physiology of the skin. *Journal of the Dermatology Nurses' Association.* LWW. 2011, vol. 3, no. 4, p. 203–213. DOI: https://doi.org/10.1097/JDN.0b013e3182274a98.

[30] Maltoni, D., Maio, D., Jain, A. K. and Prabhakar, S. *Handbook of fingerprint recognition.* Springer Science & Business Media, 2009.

[31] MathWorks. *What Is Object Detection?* Available at: https://www.mathworks.com/discovery/object-detection.html.

[32] Olsen, M. A., Dusio, M. and Busch, C. Fingerprint skin moisture impact on biometric performance. In: IEEE. *3rd International Workshop on Biometrics and Forensics (IWBF 2015).* 2015, p. 1–6. DOI: https://doi.org/10.1109/IWBF.2015.7110223.

[33] O'Shea, K. and Nash, R. An introduction to convolutional neural networks. *ArXiv preprint arXiv:1511.08458.* 2015. DOI: https://doi.org/10.48550/arXiv.1511.08458.

[34] Pere, C. *What are Loss Functions?* 2020. Available at: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904.

[35] Priya, B. *Softmax Activation Function: Everything You Need to Know.* Available at: https://www.pinecone.io/learn/softmax-activation/.

[36] Ramteke, M. *Fingerprint Sensor – Working & Its Applications.* Available at: https://www.semiconductorforu.com/fingerprint-sensor-working-its-applications/.

[37] Riebesell, J. *Convolution Operator.* Available at: https://tikz.net/conv2d/.

[38] Rizzoli, A. *The Ultimate Guide to Object Detection.* Available at: https://www.v7labs.com/blog/object-detection-guide.

[39] Ruder, S. An overview of gradient descent optimization algorithms. *CoRR.* 2016, abs/1609.04747. DOI: https://doi.org/10.48550/arXiv.1609.04747. Available at: http://arxiv.org/abs/1609.04747.

[40] Saha, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* 2018. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[41] Sharma, S. *Activation Functions in Neural Networks.* 2017. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[42] Sharma, S., Sharma, S. and Athaiya, A. Activation functions in neural networks. *Towards Data Sci.* 2017, vol. 6, no. 12, p. 310–316. ISSN 2455-2143.

[43] Svoradová, V. *Pokročilé generování projevů poškození do syntetických otisků prstů.* Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/24014/.

[44] Tu, Y., Yao, Z., Xu, J., Liu, Y. and Zhang, Z. Fingerprint restoration using cubic Bezier curve. *BMC bioinformatics*. Springer. 2020, vol. 21, no. 21, p. 1–19. DOI: https://doi.org/10.1186/s12859-020-03857-z.

[45] Venus, M., Waterman, J. and McNab, I. Basic physiology of the skin. *Surgery (Oxford)*. Elsevier. 2010, vol. 28, no. 10, p. 469–472. DOI: https://doi.org/10.1016/j.mpsur.2010.07.011.

[46] Wang, C.-F. *The Vanishing Gradient Problem*. 2019. Available at: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484.

[47] Woodruff, D. A. *What is a neuron?* Queensland Brain Institute. Available at: https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron.

[48] Yolanda Smith, B. *What is Subcutaneous Tissue?* Available at: https://www.news-medical.net/health/What-is-Subcutaneous-Tissue.aspx.

[49] Zhou, J., Chen, F., Wu, N. and Wu, C. Crease detection from fingerprint images and its applications in elderly people. *Pattern Recognition*. Elsevier. 2009, vol. 42, no. 5, p. 896–906. DOI: https://doi.org/10.1016/j.patcog.2008.09.011.

Appendix A

Contents of the Included Storage Media

This appendix serves to list the contents of the attached storage media. Due to its size, the contents are in compressed form. The decompressed archive has the following structure:

- `experiments/` - Contains nine trained models for detecting and classifying line damage and moisture in fingerprint images.
- `LineDamageGenerator/` - Source codes of the line damage generator.
- `master-prints/` - Contains 3000 generated master prints used to generate damaged fingerprints.
- `Masters-Thesis/` - Scripts for downloading, training, and testing the models.
- `MoistureGenerator/` - Source codes of the moisture generator.
- `text/` - Source codes of the thesis in LaTeX format.
- `create_dataset.py` - Script for creating the dataset of damaged fingerprints.
- `README.md` - File containing instructions for working with the attached files.
- `xfolty17.pdf` - Final version of the thesis in PDF format.

Appendix B

Examples of Fingerprint Damage Detection and Classification

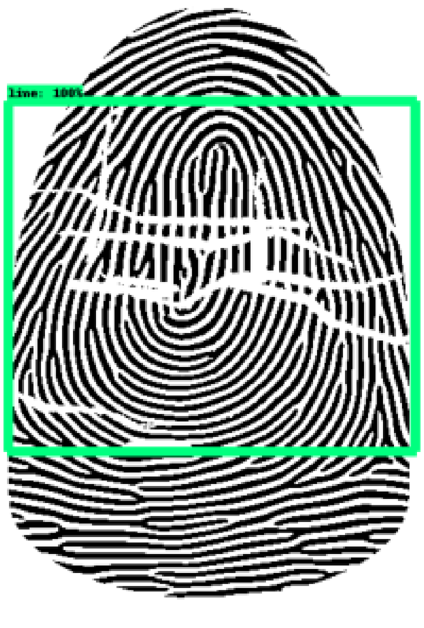

Figure B.1: Creases

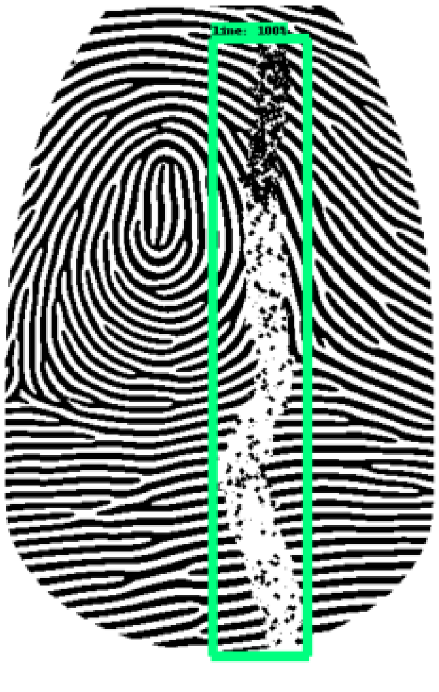

Figure B.2: Scar with patches

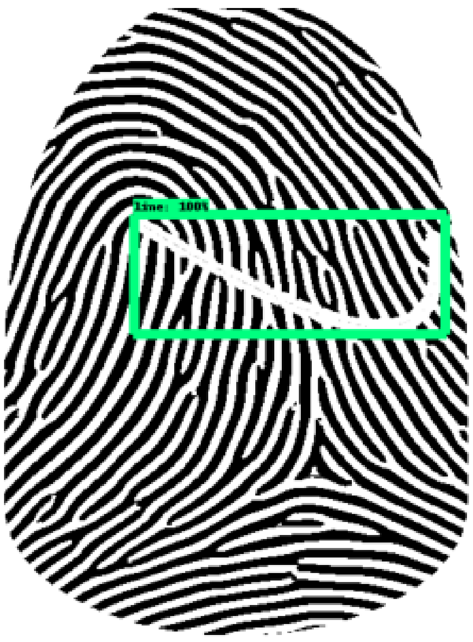

Figure B.3: Hair

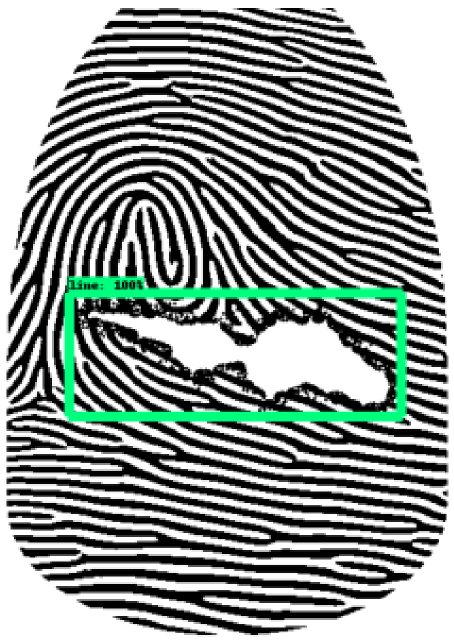

Figure B.4: Scar with outline

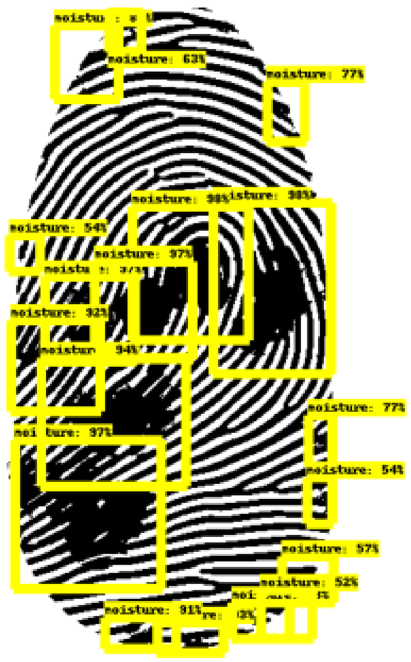

Figure B.5: Minor moisture influence

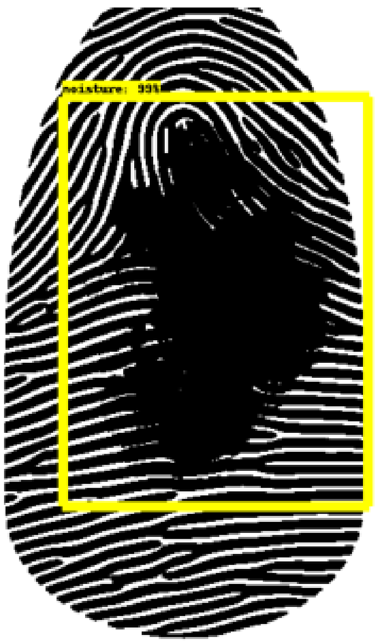

Figure B.6: Greater moisture influence