DEEP LEARNING IN FINGERPRINT ANALYSIS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Pelin İRTEM

July 2020
İZMİR

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, *Nesli Erdoğmuş* for her invaluable support and guidance throughout this thesis. I am grateful for her patience, encouragement, motivation, and positive approach.

I would like to thank my husband, *Emre İrtem* for his endless support, motivation, and love.

I would also like to thank my grandparents for making my childhood enjoyable and full of good memories.

Finally, I would like to express my infinite gratitude to my mother, *Cahide Şenkula*, and my father, *Zafer Şenkula* for their love, endless support, and sacrifices. This thesis is dedicated to them.

This thesis is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) under the grant 217E092.

ABSTRACT

DEEP LEARNING IN FINGERPRINT ANALYSIS

Fingerprints are one of the most widely used personal identification traits. They play a crucial role in forensics because they are considered to be unique to each person. For many years, the identification of individuals had been carried out by human operators. However, with technological developments, automated fingerprint recognition systems have arisen, and the growth in the population has increased the importance of their robustness.

On the other hand, deep learning has led to many impressive developments in the area of computer vision. Fingerprint analysis is indeed in the scope of image processing and computer vision; however, the usage of deep learning in fingerprint analysis is rather limited. This study focuses on using deep learning techniques on two different stages of the automated fingerprint recognition pipeline: Fingerprint classification and fingerprint minutiae extraction. Deep learning systems are developed for those two selected stages and analysed with respect to several aspects such as dataset size and different network architectures.

ÖZET

PARMAK İZİ ANALİZİNDE DERİN ÖĞRENME

Parmak izleri, en yaygın kullanılan kişisel kimlik saptama özelliklerinden biridir. Kişiye özgü oldukları için, adli vakalarda önemli bir rol oynarlar. Uzun yıllar boyunca, kimlik tespit işlemleri insan operatörler tarafından gerçekleştirilmiştir. Ancak, teknolojik gelişmelerle birlikte, otomatik parmak izi tanıma sistemleri ortaya çıkmış ve nüfustaki artış, bu sistemlerin sağlamlıklarının önemini artırmıştır.

Diğer yandan, derin öğrenme, bilgisayarla görü alanında birçok etkileyici gelişmeye yol açmıştır. Parmak izi analizi aslında görüntü işleme ve bilgisayarla görü kapsamındadır; ancak, parmak izi analizinde derin öğrenmenin kullanımı oldukça sınırlıdır. Bu çalışma, otomatik parmak izi tanıma sürecinin iki farklı aşamasında derin öğrenme tekniklerinin kullanılmasına odaklanmaktadır: Parmak izlerinin sınıflandırması ve parmak izi özellik noktalarının çıkarılması. Seçilen iki aşama için derin öğrenme sistemleri geliştirilmiş ve bu sistemler veri kümesi boyutu, farklı ağ mimarileri gibi çeşitli yönlere göre analiz edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES viii

CHAPTER 1. INTRODUCTION 1
1.1 Outline of Thesis 3

CHAPTER 2. BACKGROUND 4
2.1 Fingerprint Classification 6
2.2 Fingerprint Matching and Minutiae Extraction 7
2.3 Advantages of Deep Learning 8

CHAPTER 3. RELATED WORK 10
3.1 Fingerprint Classification 10
3.2 Minutiae Extraction 16

CHAPTER 4. FINGERPRINT CLASSIFICATION 19
4.1 Dataset 19
4.2 Model Training 20
4.2.1 Model Architectures 21
4.2.1.1 VGG 21
4.2.1.2 ResNet 22
4.2.2 Transfer Learning 24
4.2.3 Dataset Sizes 26
4.2.4 Summary 26
4.3 Experimental Results 27
4.3.1 Results for Different Model Types 27
4.3.2 Results for Different Transfer Learning Setups 28
4.3.3 Results for Different Dataset Sizes 29
4.4 Summary 30

CHAPTER 5. MINUTIAE EXTRACTION 34
5.1 Dataset 34
5.2 Model Training 34
5.3 Experimental Results 37
5.4 Summary 46

CHAPTER 6. CONCLUSION AND FUTURE WORK 48
6.1 Conclusion 48
6.2 Future Work 49

REFERENCES 50

LIST OF FIGURES

Figure **Page**

Figure 1.1 Fingerprint recognition pipeline. 2
Figure 2.1 Level 1 Features: Singular Points and Fingerprint Classes. 5
Figure 2.2 Level 2 Features: Minutiae Points. 5
Figure 2.3 Level 3 Features: Sweat Pores. 5
Figure 3.1 Some features of fingerprints. 10
Figure 3.2 The algorihtm diagram of Karu and Jain [4]. 12
Figure 3.3 Example of strings created by [12]. 13
Figure 3.4 The steps of methodology used by Maio and Maltoni [14]. 13
Figure 3.5 The steps of methodology used by Capelli et al. [15]. 14
Figure 4.1 Configurations of VGG networks [35]. 22
Figure 4.2 A Residual block [36]. 23
Figure 4.3 Configurations of ResNet architectures. 24
Figure 4.4 Accuracies of models with different architectures. 28
Figure 4.5 Accuracies of models with transfer learning methodologies. . . . 29
Figure 4.6 Accuracies of models trained with different dataset sizes. 30
Figure 4.7 Accuracy results of all experiments with 5 classes. 31
Figure 4.8 Accuracy results of all experiments with 4 classes. 32
Figure 5.1 Custom network architecture used for minutiae extraction. 36
Figure 5.2 Patch examples. 38
Figure 5.3 Different patch sizes and patch extraction approaches. 40
Figure 5.4 Effect of morphological operations. 41
Figure 5.5 Prediction process on an example image. 42
Figure 5.6 Low quality image and patch examples. 44

LIST OF TABLES

Table		Page
Table 4.1	Transfer Learning scenarios in terms of dataset similarity and sizes.	25
Table 4.2	Confusion matrix for five-class classification of fine-tuned VGG19 model trained with 2000 samples.	33
Table 4.3	Confusion matrix for four-class classification of fine-tuned VGG19 model trained with 2000 samples.	33
Table 4.4	Comparison between the accuracies of the proposed method and the literature for fingerprint classification.	33
Table 5.1	Performances of the 8 models.	43
Table 5.2	Comparison between the proposed method and [31].	43
Table 5.3	Performance results of the custom network architectures according to the quality of the images.	45
Table 5.4	Performances of the data augmentation applied and not applied models.	46

CHAPTER 1

INTRODUCTION

Identification of people is a requirement for both personal and property security. Naturally, people recognize and identify each other by some distinctive characteristics such as face and gait. These and many other cues are also used in automated systems to identify people and they are called biometric traits.

Biometric traits are reliable identifiers because they cannot be easily lost, stolen, or forgotten like knowledge or possession-based authentication mechanisms [1]. There are many kinds of biometric traits, and in general, they are grouped into two categories: physical and behavioural traits. While physical traits include fingerprint, face, and iris; behavioural traits include gait, signature, and writing behaviour.

In order to use a trait as a means of identification, it has to have several properties. Uniqueness, permanence, universality, and collectability are some of them. Fingerprints have all the necessary properties to be used for reliable and high-scale identification. They are unique to each person in the world, even identical twins have different fingerprints. Unless some certain damage is taken, they are permanent and do not change much as people get older. Except for some anomalies and disabilities, everyone has fingers and thus fingerprints. Ridges and valleys on the fingers can be easily captured by using various imaging techniques and sensors. Due to the sweat and the oil surface on the finger skin, they are left on the surfaces and can even be collected from where they make physical contact.

Due to those properties of the fingerprints mentioned above, they have earned a crucial role in various areas of usage, primarily in forensics and security. Before automated systems, fingerprint experts were matching the fingerprints collected from persons to be identified or found on the crime scenes with the ones in the database manually. However, with the increasingly large and mobile populations, the manual work required for matching has also increased and even become impossible. As a result, with the guidance of the approach that the experts use, automated fingerprint recognition systems are developed. Nowadays, almost every forensics and law enforcement agency uses automated fingerprint identification systems(AFIS). Besides forensics, fingerprints are widely accepted in other areas such as biometric passports, phone locks, attendance and entrance systems, etc.

In fact, automated fingerprint systems have been actively used for decades now. However, more robust and accurate systems are always needed because fingerprint recog-

nition is still a challenging pattern recognition problem with high intra-class variability and high inter-class similarity [1].

Deep learning is proven to perform well on many pattern recognition and computer vision tasks. Since fingerprint analysis is a sub-domain of computer vision and it is actually a pattern recognition problem, deep learning is a good candidate for improving robustness and accuracy in fingerprint analysis, while spending less or even no effort on the intermediate tasks such as feature extraction and image enhancement. Traditionally, various machine learning approaches have already been adopted for different stages of fingerprint recognition. However, those require extra pre-processing effort prior to using machine learning algorithms for the task at hand. For instance, for fingerprint classification and minutiae extraction, orientation fields should be extracted and only then they are fed to a machine learning algorithm for better performance. In contrast, when deep learning is adopted, the network is expected to automatically learn the important features from the data and use them to make necessary decisions.

As mentioned previously, fingerprint recognition is a very challenging task because of high intraclass variance and interclass similarity. In order to cope with this challenging problem, it is broken down into several steps such as fingerprint classification, orientation extraction, minutiae extraction, and fingerprint matching(Figure 1.1). This study focuses on fingerprint classification and minutiae extraction since they are the two most fundamental steps for accurate matching. Without manual feature extraction and preprocessing, deep learning models are developed for these two tasks and different architectures and different dataset sizes are used for training. The objective of this study is to observe the effect of dataset sizes and different deep learning architectures on the success rates of the models and in the meantime, to achieve state-of-the-art results without human-engineered feature extraction and preprocessing steps, and integrating domain knowledge.

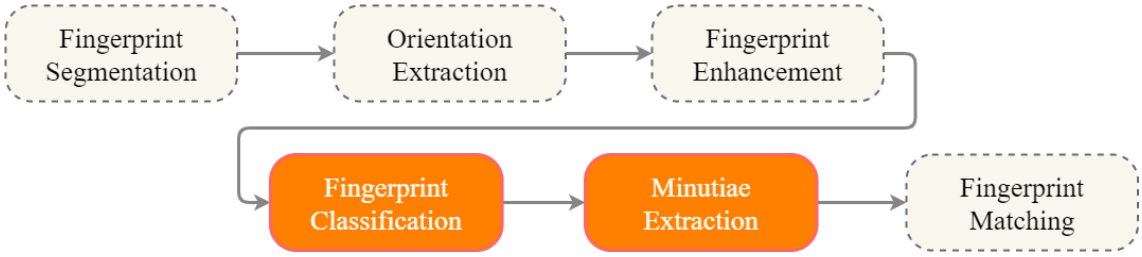


Figure 1.1: Fingerprint recognition pipeline. This study focuses on the orange parts: fingerprint classification and minutiae extraction.

1.1 Outline of Thesis

This thesis is organized as follows: Chapter 1 gives an introduction to the area of fingerprint analysis, and the scope of this thesis. Chapter 2 provides background information about the fingerprint structure, and the fingerprint analysis process. In Chapter 3, literature is reviewed in detail in terms of fingerprint classification and minutiae extraction. Chapter 4 provides details of the study on fingerprint classification with explanations of the dataset, model architectures, experiments, and their results. Chapter 5 gives the details of our approach for minutiae extraction, the conducted experiments, and their results. Finally, Chapter 6 concludes the thesis and provides direction of future research.

CHAPTER 2

BACKGROUND

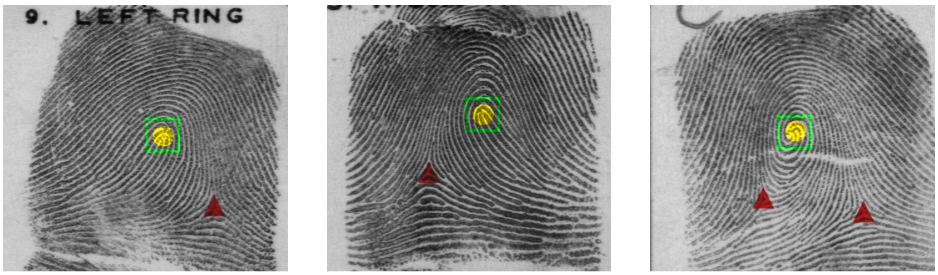
The skin at the fingertips has a wavy, indented structure. It contains protruded lines with different curvatures and dented areas between those lines which are called **ridges** and **valleys**, respectively. Fingerprints are the imprints of those ridges and valleys at the fingertip and they transmit that on surfaces. The form of the ridges and valleys, and the pattern they create cause fingerprints to have different and distinctive details. In general, those details are grouped into three levels at different scales [1]:

- **Level 1** features are global features that the ridge lines shape. These features are called singular points and there are two types of singular points: **delta** and **loop**. As seen in Figure 2.1, loop is the inflection point of the parallel ridge lines and delta is the triangular area where different ridge lines meet. In addition, the upper-most singular point is called the **core**. The shape and location of these are not unique to each fingerprint, rather they help to describe coarse level shapes which are useful for grouping fingerprints.

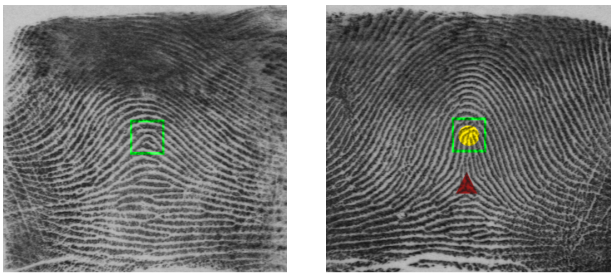
- **Level 2** features are local features where the ridge line characteristics change at a finer level. These features are called **minutiae points**, and they provide the fingerprints the bulk of their distinctiveness. There are many types of minutiae points however the two most important and mostly used are **ridge ending** and **bifurcation**. A ridge ending is as the name implicates where a ridge ends, and a bifurcation is where ridge lines split into two ridges like a fork. Those two types of minutiae points are illustrated in Figure 2.2.

- **Level 3** features are observed at the finest level, and they mainly consist of very tiny details such as sweat pores (Figure 2.3). These features also make a high contribution to the uniqueness of the fingerprints. Although their distinctiveness is very high, they may be misleading because they are so small and hence very difficult to locate accurately. The image resolution should be very high to use these features as a proper means of identification.

These characteristics of fingerprints are used in various steps of fingerprint recognition which are previously mentioned in Chapter 1. For instance, for fingerprint classification, Level 1 features and for fingerprint matching mostly Level 2 and Level 3 features are used.

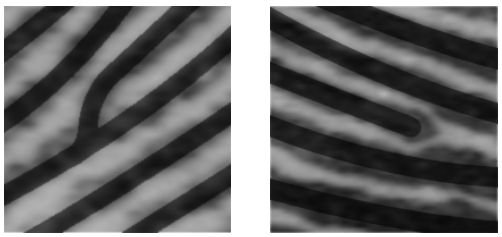


(a) Left Loop (b) Right Loop (c) Whorl



(d) Arch (e) Tented Arch

Figure 2.1: Level 1 Features: Singular Points and Fingerprint Classes. The colors and shapes indicate singular points: red triangle for delta, yellow circle for loop. Green squares show the cores. Note that Arch class has no singular points, thus its core is where the curvature is highest.



(a) Bifurcation (b) Ridge ending

Figure 2.2: Level 2 Features: Minutiae Points.

Figure 2.3: Level 3 Features: Sweat Pores as an example of Level 3 Features.

2.1 Fingerprint Classification

For automated fingerprint identification, most of the time, a large number of comparisons is needed where the query fingerprint is compared with all existing fingerprints in the database. Because fingerprints are used mostly in forensics, the number of fingerprints in the database is counted by millions most of the time. Reducing the search space is critical for both reducing search time and increasing the matching accuracy. Fingerprint classification has an important role in automated fingerprint identification systems since it aims at reducing the search space by assigning predefined classes to fingerprints.

According to Henry classification system [2], there are 5 classes of fingerprints:

- Arch
- Tented Arch
- Left Loop
- Right Loop
- Whorl

These classes are determined by the singular points in the fingerprint. According to the location and amount of the singular points, and the ridge flow orientation, fingerprint classes can be assigned. In detail, the Arch class has no singular points. It has a smoother curvature and nearly parallel ridge lines that are a little bit curved around the center of the fingerprint. Tented Arch class is similar to Arch class visually, but its curvature is higher and it has one delta and one loop. The delta is located directly below the loop and is vertically aligned with it. Left Loop and Right Loop also have one delta and one loop however they differ from each other and the Tented Arch class in the location of deltas and loops, and in the curvature orientation around the loops. In Left Loop class, delta is located below and to the left of the loop and the orientation of the curvature around the loop is towards right. Vice versa, in the Right Loop, delta is located below and to the right of the loop and the curvature is towards left. Whorl class, unlike other classes, may have two deltas and two loops. Additionally, the ridge curvatures make a 360° tour around the center. In Figure 2.1 an example for each fingerprint class is also shown with the corresponding singular points.

Once the fingerprint class is determined, search for the matching fingerprint may be conducted in a smaller database that only includes the fingerprints that belong to the assigned class. Smaller database favours less search time and increased matching accuracy. Moreover, in some cases, the search space gets very small because of the natural distribution of the classes among people. Only 3.7% of the people have Arch and 2.9%

have Tented Arch class while the majority have Left Loop, Right Loop and Whorl(33.8%, 31.7%, and 27.9% respectively) [3]. For instance, if the query fingerprint is determined as a member of the Arch or the Tented Arch class, the search space will be much smaller compared to the initial one.

To sum up, fingerprint classification is a coarse level matching. Since the features that it relies on are global and not distinctive, it cannot be directly used for accurate matching. However, it is useful for determining which fingerprints can be eliminated from the dataset(pruning) due to having different classes. Thus, it helps to reduce the search space.

2.2 Fingerprint Matching and Minutiae Extraction

Fingerprint matching is the core stage of fingerprint recognition. The main objective is to match the input fingerprint with the enrolled fingerprints in the database and detect or confirm the identity of the person. However, this is a challenging task since fingerprints have large intraclass variance, i.e. same fingers may create very different impressions. This is mainly due to he following are the main reasons [1]:

- **Displacement:** Fingerprints may be located in different places in the paper/sensor. Even though some sensors guide strict placement into a specific area, very small displacements may cause large differences pixel-wise.
- **Rotation:** Similar to displacement, fingerprints may be rotated while transferring to paper or obtaining via sensor.
- **Partial Overlap:** Because of displacement, rotation, or other factors, some parts of the fingerprint may not be visible. When the obtained impressions are not containing the fingerprint fully, it may cause fingerprints to look very differently.
- **Non-linear Distortion:** Because of the 3D shape and elasticity(rubber-like structure)of fingers, fingerprints may be distorted when pressed against a 2D surface like paper or sensor.
- **Variable pressure:** When pressing the finger to the sensor or paper, the pressure may vary. This causes variations in the fingerprint impressions.
- **Changing skin conditions:** While capturing the fingerprint via ink and paper or a sensor, the finger skin may be wet, dry, or oily. These factors affect the ridge line thickness in the impressions. For instance, wet skin's impressions look darker.

- **Noise:** While obtaining the fingerprint via sensors, there may be internal noise caused by the sensor. Also, the paper might be noisy or while taking the photo of the paper or scanning it, some noise might be introduced to the fingerprint.
- **Feature extraction errors:** Prior to fingerprint matching, some features are extracted such as ridge orientation field, singular points, and minutiae points or some preprocessing is applied to the fingerprint. During those steps, different errors and noise may be introduced that hinders the matching accuracy.

In order to get accurate matching, many different approaches are developed and they are mainly grouped into 3 categories [1]: correlation-based matching, minutiae-based matching, and non-minutiae feature-based matching. In correlation-based matching, fingerprints are aligned and compared according to their pixel correspondences. In minutiae-based matching, minutiae points are extracted and matched with the templates that are extracted during the enrolment. In non-minutiae feature-based matching, the features other than minutiae are used for matching because minutiae extraction may be erroneous especially in low-quality fingerprints. In this category, the main idea is basically to use the features of the ridge pattern. However, ridge pattern distinctiveness is not very high.

Most widely accepted method for matching is minutiae-based matching [1]. Thus, obtaining an accurate list of minutiae points has become crucial. However, this task is not easy, especially in low-quality images. Even though the actual minutiae points can be extracted correctly, many spurious minutia points are usually also detected or algorithms even fail to detect all of the existing minutia points correctly. In order to increase the accuracy of minutiae extraction, some preprocessing and enhancement steps are required. For instance, in many minutiae extraction algorithms, binarized images are used instead of grayscale fingerprint images. Sometimes, the intensity values are normalized according to predefined parameters. Some filtering techniques are used to increase the contrast of the invisible parts on the fingerprint. In short, there are many different enhancement techniques used to increase the quality of the fingerprint image and thus, to increase the accuracy of minutiae extraction. Once the minutiae points are accurately extracted, the matching step gives better results.

2.3 Advantages of Deep Learning

Deep learning is famous for achieving great performances in the computer vision domain with less or no effort spent on manual feature extraction or image enhancement. in its traditional methodology, the fingerprint analysis contains many applications of manual feature extraction and image enhancement, and these algorithms mostly rely on domain

knowledge. Deep learning is expected to reduce that need in different stages of the fingerprint recognition pipeline with its strong representation ability.

Moreover, the intraclass variance of fingerprints also poses a big problem for traditional methods. However, deep learning methods are able to learn the image variations due to various factors such as rotation, translation, and distortions from the training data. For this reason, it is expected that deep learning can overcome the generalization problems without any need of manual optimization.

CHAPTER 3

RELATED WORK

3.1 Fingerprint Classification

Many approaches have been developed to classify fingerprints and in most of them, the fingerprint classification is divided into two main tasks: feature extraction and classification. The features mostly used for fingerprint classification are the global features(Level 1 features). The majority of the methods focus on ridge-line flow, orientation image, singular points, and Gabor filter responses (Figure 3.1). Ridge-line flow is the global trace of the ridges, they give intuition about the orientation and curvature of the ridges. Orientation image is a representation of the local ridge flow. Most of the time, they are calculated block-wise on the fingerprint image and called as the orientation map(OM). Singular points are, as mentioned before, the areas where a large change in the curvature occurs(the loop, in some researches referred as the core [4, 5, 6], and delta). Finally, the Gabor filter responses are obtained by applying Gabor Filters to the fingerprint image or orientation image. They are good at detecting ridges and valleys.

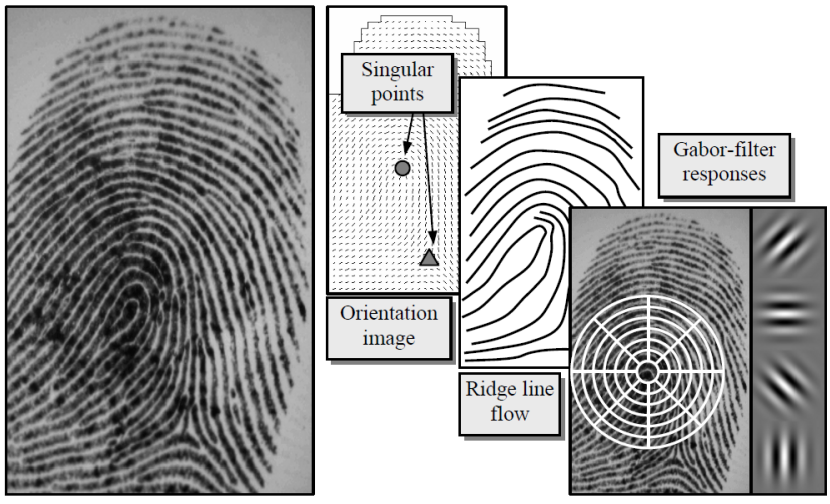


Figure 3.1: Some features of fingerprints that are used for fingerprint classification [1].

Classification methodologies are coarsely classified into six categories according to [1]: rule-based approaches, syntactic approaches, structural approaches, statistical approaches, neural network-based approaches, and multi-classifier based approaches.

Rule-based approaches are based on singular points. Because singular points differ in number and location among fingerprint classes, the classes can be assigned according to their relative positioning. As mentioned in Section 2.1, the Arch class has no singular points; the Tented Arch, the Left Loop, and the Right Loop have one delta and one loop; the Whorl has two loops and two deltas. The delta locations in Tented Arch, Left Loop, and Right Loop also differ. In [7], Kawagoe and Tojo use Poincare index for detecting singular points. Using singular points and the rules given above, first, a coarse classification is obtained and next, a fine classification is applied it by tracing the ridge line flow. However, using singular points may introduce some errors, especially on noisy images. In [4], Karu and Jain adopt an iterative smoothing approach in order to reduce the noise. After computing the orientation image, they apply smoothing, detect singular points, and classify fingerprints. The algorihtm is shown in Figure 3.2. They also use Poincare index as in [7] in order to determine the singularity type of a point (delta, core, or no singularity) in the orientation image by looking at the changes in the direction angles in a closed curve around that point. If the sum of orientation angles, while iterating counter clock-wise around the point, make 180°, the point is stated as a core. If the sum is -180°, the point is stated as a delta, and if the sum of the angles is 0 it means there is no singular point. In order to determine the classes according to the given rules, they use the number and the location of the singular points. For instance, to discriminate Arch from other classes they use the number of singular points i.e Arch has no singular points. In order to discriminate Tented Arch, Left Loop, or Right Loop, they connect delta and core with a line and look at its intersection with the orientation lines. In a more recent research [8], a fuzzy rule-based classification system is proposed which considers the uncertainty in classifying Tented Arch, Left Loop, and Right Loop. In that study, the authors also propose a coherence method in order to correctly classify the fingerprints that have singular points but could not be classified accurately because the singular points are not detected by the algorithm. In rule-based approaches, the focus is more on correctly detecting singular points rather than assigning the classes. If the singular point detection algorithm makes errors, the classification fails. For singular point detection, there are other methods than Poincare index such as complex filter [5, 9] and zero-pole model [10]. However, singular point detection is very error-prone especially in noisy and partially imperceptible fingerprints. Thus, rule-based methods can easily fail to detect the correct fingerprint classes.

In syntactic approaches, the patterns are represented with symbols and a grammar over the symbols. Using the grammar, fingerprint classes are assigned. In [11], Moayer and Fu used a symbol set obtained from orientation image patches. With a class of context-free grammars, they describe fingerprint patterns and this allows classification. In

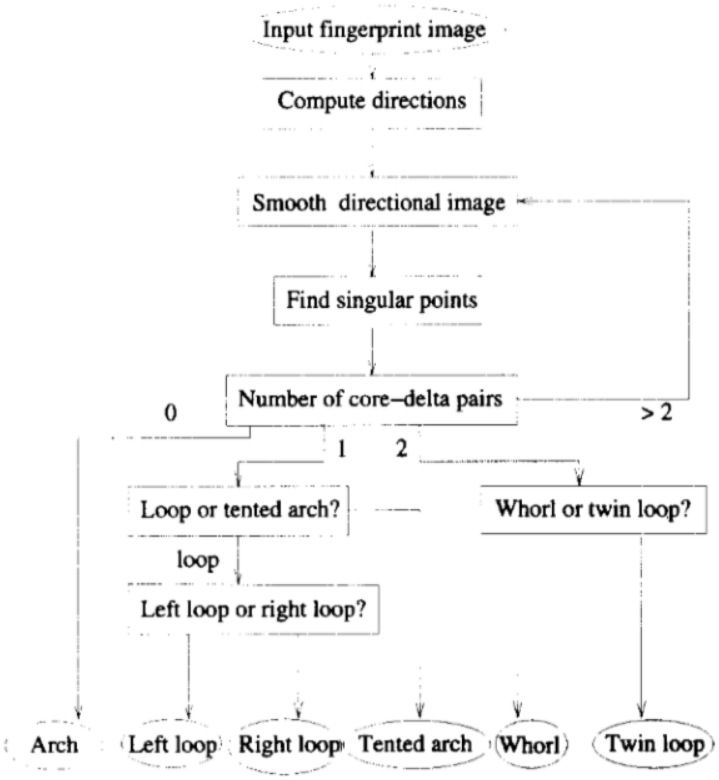


Figure 3.2: The algorihtm diagram of Karu and Jain [4].

the approach which Rao and Balck use in [12], a string is generated from the direction changes in the ridge line flow. Then using a context-free grammar, classes are assigned according to the generated strings. Examples of strings are shown in Figure 3.3. However, syntactic approaches require complex grammars, thus there are not many publications using this method anymore. In a relatively recent research, Chang and Fan [13] used ridge patterns to determine the classes. They argue that there are 10 basic ridge patterns and using ridge shapes and the ridge distribution sequence, those patterns can be detected and used in classification.

Structural approaches are based on higher-level representations of features such as graphs or trees. In [14], Maio and Maltoni propose an approach that segments orientation image into similarly oriented partitions and constructs a relational graph by making each region a node and connecting those nodes. Using inexact graph matching techniques, the graph and the model graphs of the classes are compared. In Figure 3.4, the steps of the algorithm are shown. Capelli et al. [15] use template-based graph matching instead of relational graphs. In order to create homogeneous regions as in [14], they extract a set of dynamic masks obtained from fingerprint classes and they use them to guide the segmentation. Those dynamic masks are used to calculate a cost function and the cost function is used for classification. Their approach can handle translation and rotation

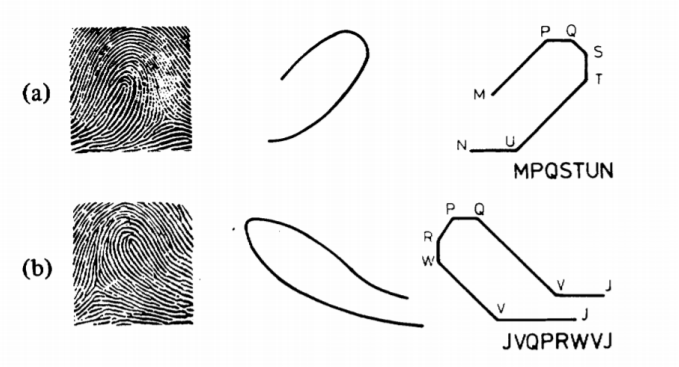


Figure 3.3: Example of strings created by [12].

variances and it works even if some of the singular points are missing. The algorithm is shown in 3.5.

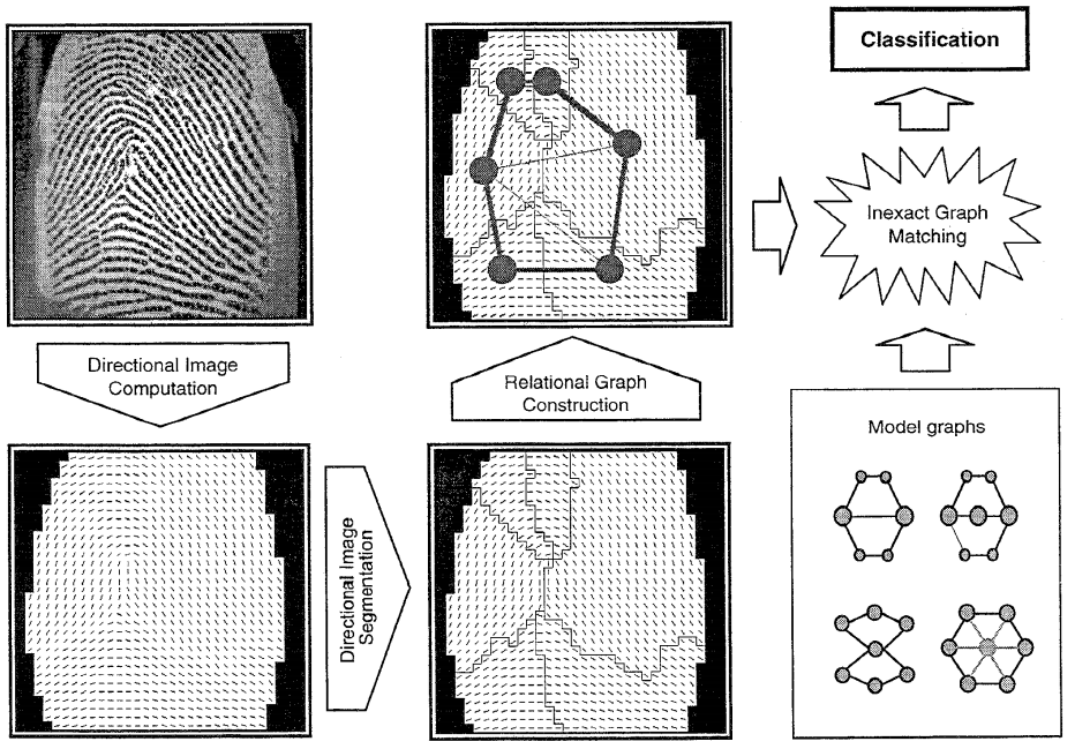


Figure 3.4: The steps of methodology used by Maio and Maltoni [14].

In [5], Lui proposes a method that uses adaboosted decision trees for fingerprint classification. His approach is based on singular points that are detected at different scales using complex filters. Then, a feature vector is created with different specifications of singular points such as relative location, direction, and certainty for each scale. For

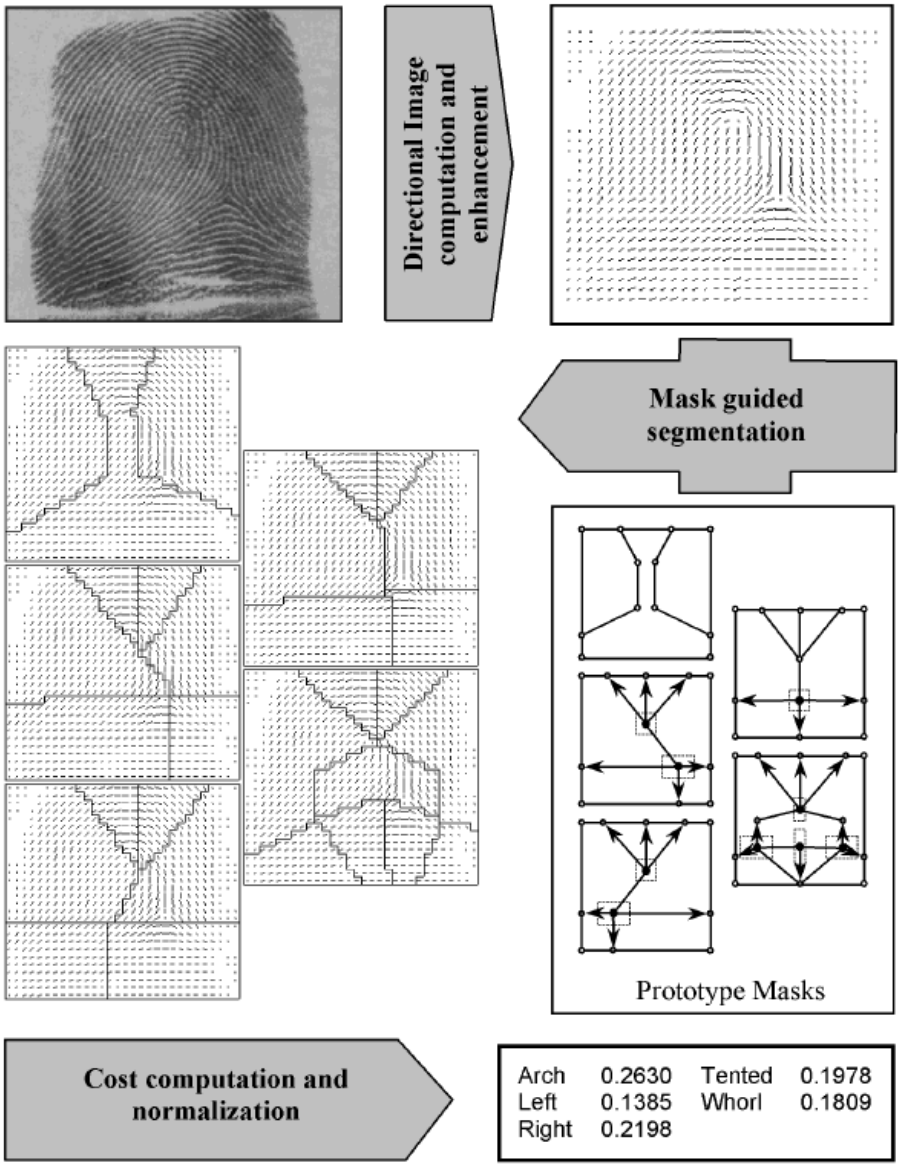


Figure 3.5: The steps of methodology used by Capelli et al. [15].

determining classes, adaboosted decision trees are used.

In statistical approaches, feature vectors are extracted from the images and used in statistical classifiers such as SVM or k-nearest neighbour(k-NN) algorithm. In [16], Luo et al. used k-NN to classify the feature vectors obtained by using Curvelet transform (CT) and gray-level cooccurrence matrix (GLCM). They achieved 94.6% accuracy for five-class classification problem and 96.8% accuracy for four-class classification problem with 1.8% rejection. In [17], Yao, Frasconi, and Pontil propose to use SVM for fingerprint classification. They use FingerCode [18] as the feature vector and train different SVMs for classification experiments.

In multi-classifier approaches, in order to improve performances, multiple classi-

fiers are combined. In [18], Jain, Prabhakar, and Hong propose FingerCode which is a 192-dimensional representation of fingerprints derived from local ridge structures. Using the FingerCode the fingerprints are classified with a two-stage classifier. They use k-NN in the first stage and multiple neural networks in the second stage. In [6], Cao, Pang, Liang, and Tian use combinations of rule-based, k-NN, and SVM classifier hierarchically. Their algorithm consists of five stages: First, they detect Arch classes using complex filters. Then, they detect Whorl classes using core points and ridge line flow classifier. In the third stage, using feature vector obtained by complex filter responses and the orientation map, k-NN classifier detects the top two probable classes. Using the ridge line classifier they distinguish Loops from other classes except for Whorl class in the fourth stage. Lastly, they use SVM to determine the final class. They achieved 95.9% accuracy for five-class classification and 97.2% accuracy for four-class classification.

In neural network-based approaches, earlier applications consist of using multilayer perceptrons and small fast forward neural networks. With the emerge of deep learning, a plethora of studies that adopt deep neural networks are published. Wang, Han, Wu, and Guo [19] use orientation field as the input for a deep neural network and use stacked sparse autoencoders for the classification of fingerprints. In [20], Conic Radon Transform is used as a feature extractor. CRT is applied on the fingerprint images and the original images are used as inputs for the CNN. They achieved 96.5% accuracy on 4-class classification. In [21], Tang, Li, Liu, and Feng combined domain knowledge and a small network architecture. They integrated orientation fields into their network(FClassNet) which contains three modules: first is the backbone network whose hidden layers represent texture, orientation and quality characteristics of the fingerprints, second is for extracting the orientation field and segmentation using the orientation and quality layers from the first module, and the third is to perform a hybrid classification which combines the 4-dimensional feature vector obtained by a convolutional network which uses orientation field and singular points that are extracted using the orientation field. They performed only 4-class classification and achieved 95.1% accuracy on NIST SD4 database by using the whole dataset for testing. In [22], Listyalina and Mustiadi propose an approach that uses transfer learning with GoogLeNet which is pretrained on the ImageNet database. They freeze the first 10 layers of the network and trained the rest with the adoption of the last layer for fingerprint classification task. They achieved 94.7% and 96.2% accuracy results on 5-class and 4-class classification, respectively. In [23], Michelsanti et al. also apply transfer learning on fingerprint classification but without any frozen layers. They fine-tune two VGG networks which are pretrained on ImageNet. They perform only four-class classification and the best accuracy they report is 95.05%.

3.2 Minutiae Extraction

Many different approaches have been developed for minutiae extraction for many years. These will be coarsely analysed into two groups as traditional methods and deep learning-based methods. Traditional methods mostly extract features manually which require domain knowledge. Additionally, most methods perform on preprocessed(binarized and skeletonized) fingerprint images.

In [24], Farina et al. propose an approach that uses binarized and then skeletonized fingerprint images. In their method, they clean ridge bridges using a novel method in which ridge positions are used instead of directional maps. Then, the reliability of the extracted minutiae points is assessed for matching. Their algorithm performs the following steps for the extraction of minutiae points: Pixel codification where minutiae classification is performed and unclassified configurations are removed, pre-filtering where spurious minutiae points are removed as much as possible, skeleton enhancement where ridge breaks are repaired and bridges, spurs, and short ridges are eliminated, minutiae invalidation where close minutiae points, bridges and spurs are invalidated, and topological validation where island removal, bifurcation and endpoint validation is performed according to neighbouring ridge layout.

In [25], Zhao and Tang use a three-step algorithm which are preprocessing, minutiae extraction and post-processing. In the preprocessing step, before thinning the binarized fingerprint image, they eliminate misconnections and isolated regions such as islands, holes, and dots. They apply morphological operations on the binarized image so that spurious bridges and spurs do not appear on the skeletonized image. Then, they detect and fill the small holes and remove small dots from the binary image. In minutiae extraction step, they use Rutovitz Crossing Number concept. They state that the skeleton should be one pixel wide, however, this is not always achieved. They detect those pixels(called "bug pixels" by the authors) and eliminate them without disturbing the skeleton connectivity at fork regions. At the end of this process, they obtain minutiae points. The elimination of spurious minutiae points is performed at the post-processing step. They use the duality property of the minutiae points which means if there is a ridge ending in the fingerprint image, there is a bifurcation in the valley structure. (In other words, if there is a ridge ending in the positive image, there is a bifurcation in the negative image.) They observe that for spurious minutiae points, there is a bridge structure. They define a bridge structure and its dual break as an H-point and eliminate those H-points.

In [26], Jiang, Yau, and Ser propose an algorithm that traces gray-level ridges by using piece-wise linear lines of different lengths. In some selected points, they smooth the fingerprint image with an adaptive-oriented smoothing filter. They detect minutiae points while tracing the ridges and forming skeleton. In the skeleton image, each ridge

gets a number and the ridge numbers are associated with minutiae points. This helps in the post-processing step to eliminate spurious minutiae points according to spatial, structural, and ridge relationships.

With the emergence of deep neural networks, researchers shifted their focus to using deep learning on minutiae extraction. In [27], Jiang et al. propose using deep convolutional neural networks with a patch-based approach. They train two sequential networks: the first one is JudgeNet which detects candidate minutiae patches and the second one is LocateNet which detects the precise location of the minutiae points in the candidate patches detected by JudgeNet. They use 45x45 and 63x63 sized patches with minutiae and non-minutiae labels determined according to center 27x27 pixels. In LocateNet, the center 27x27 pixels area is divided into 9x9 pixels areas. The location label is assigned as one of the 9x9 areas that contain the minutiae point. They use Chinese criminal investigation fingerprint database for their experiments and they achieve 94.59% precision, 91.63% recall, and 93.08% F1 score.

Darlow and Rosman [28], propose a similar architecture to [27], but instead of using two networks, they use one network(MeNet) for classifying the patch center as minutiae and non-minutiae and by post-processing, they get the precise locations of the minutiae points. They obtain the input for MeNet by sliding 30x30 window on the full-size fingerprint images. They get a probability map over the full-size image by using MeNet and use median filter to smooth the output probability map. After applying iterative thresholding to the probability map, they obtain final minutiae points and locations. They test their results on combinations of FVC databases(FVC2000, FVC2002, and FVC2004). They label the fingerprint images in those databases using an automated process that they constructed with several commercial minutiae extractors. They combine the results of each extractor, post-process the combined results using morphological operations and thresholding and label the responses of this process as positive class and the background black regions as negative classes. For minutiae extraction performance of MeNet, they sample the FVC databases that they combined, perform their minutiae extraction test on it and achieve 14.2% miss rate.

In [29], Tang, Gao, Feng, and Liu proposed an approach that combines the domain knowledge and the representative power of deep learning to obtain better results on latent fingerprints. They use the traditional pipeline of minutiae extraction such as orientation estimation, segmentation, enhancement, and minutiae extraction as convolutional kernels with fixed weights. They expand these layers with additional convolutional layers with released weights so that the network also learns the background details. They call their architecture FingerNet and apply Non-maximum suppression(NMS) as a post-processing step. They perform their tests using FVC2004 and NIST SD27(latent) databases. However, their results strongly depend on the quality of the enhancement and segmentation.

In [30], Nguyen, Cao, and Cain also combine the domain knowledge and the deep learning approach with two networks: CoarseNet and FineNet. They use CoarseNet to extract the orientation image, enhanced image, and segmentation map and to obtain candidate minutiae patches. After, they use the FineNet to detect more reliable minutiae points in the candidate patches. They extract 45x45 sized patches using CoarseNet and resize them to 160x160 to train the FineNet to decide if the 10x10 center area of the 45x45 sized patch contains minutiae point or not. They also evaluate their results on FVC2004 and NIST SD27. They obtain 85.9% precision, 84.8% recall on FVC2004 and 71.2% precision, 75.7% recall on NIST SD27 databases.

Recently, Zhou et al. [31] propose a two-stage algorithm with two networks that share a fully convolutional network. In the first stage, the fingerprint image is divided into small cells that may contain only one minutiae and using the shared feature map generated from the shared convolutional network, the candidate patches are selected. Each cell gets a probability of being minutiae and higher scored cells are selected as candidate patches. The network finds the coarse location of the minutiae point in the patch, thus the patch is not necessarily the center of the cell. In the second network, by using the shared feature map, they obtain the probability, the location, and the direction of the minutiae point which is at the center of the patch. They perform their tests on FVC2002 and FVC2004 databases. They achieve 87.90% precision, recall and F1 score on FVC2002 DB1-A database.

For minutiae extraction, unfortunately, there is not a common benchmark dataset as we have for fingerprint classification. The only dataset that was available with ground truth minutiae points is FVC2002. Therefore, the results of this study are only comparable with [31]. It is a difficult process to obtain a database with manually marked minutiae points. This problem is also stated by many researchers and some authors either come up with some solutions such as extracting ground truth with several commercial minutiae extractors or used obtained small databases like this study.

CHAPTER 4

FINGERPRINT CLASSIFICATION

4.1 Dataset

For fingerprint classification, one of the most important benchmark databases is National Institute of Standards and Technology's Special Database 4(NIST SD4) [32]. It contains 4000 fingerprint images taken from 2000 fingers with 2 impressions(named as F for first and S for second). The fingerprint images are the scanned versions of rolled fingerprint impressions which are imprinted with ink on cards. Each fingerprint image is 512x512 pixel and labelled with at least one of the following 5 classes: Arch(A), Tented Arch(T), Left Loop(L), Right Loop(R), and Whorl(W). In the database, there are 400 fingerprint images of each class.

Moreover, 350 fingerprints(17.5% of the database) are labelled with two classes due to the fact that they are difficult to differentiate even for the fingerprint experts. In the literature, there are two different approaches for those two-class fingerprint images. First one is eliminating the ambiguous fingerprints and using only the ones that have one class as in [33]. Second approach is using only one label for the algorithm development stage(generally for training machine learning systems) and using both labels as correct in the testing stage, like [4, 5, 6, 18, 23]. In this study, the second approach is adopted since it is more common in the literature. The first label is used for training and, in the testing stage, the prediction is assumed to be correct if the predicted class is one of the two classes.

In order to partition the data, for training and testing, the common approach in the literature is dividing the dataset into two parts in such a way that the impressions of the same fingers will not be in train and test partitions separately. Otherwise, it would introduce bias by training with one and testing with another impression of the same finger. In this study, the mentioned approach is followed for getting comparable results with the literature. The database is divided into two parts in a way that training set contains both impressions of the first 1000 fingers i.e 2000 images(F1-1000 and S1-1000), test set contains impressions of the remaining 1000 fingers(F1001-F2000 and S1001-S2000).

For performance calculation, there are mainly two metrics in the literature which basically address the same intuition: accuracy and error rate. In addition, some authors also shared their results using confusion matrices. In this study, the main performance calculation metric is chosen as accuracy. Additionally, confusion matrices of the best performing models are also shared.

Lastly, because Tented Arch and Arch class are very similar to each other, algorithms sometimes fail to discriminate them. In many publications, the Tented Arch class is merged with Arch class and four-class classification(A, L, R, T) is performed. In this study, in order to have comparable with the literature, four-class classification is also performed.

4.2 Model Training

In fingerprint classification, deep learning is a good candidate to achieve accurate results because fingerprints consist of typical patterns and the problem is a classical multi-class image classification problem. Earlier methodologies of fingerprint classification mostly rely on feature extraction and therefore, the classification accuracy highly depends on the quality of the extracted features. In this study, the main aim is to train deep learning systems which do not require feature extraction step for accurate classification and instead extracts important abstract features internally. Additionally, with the use of deep learning, the need for preprocessing and image enhancement will be eliminated compared to the traditional approach.

In this study, in order to observe the effect of model architectures and achieve state-of-the-art results, different models are trained for fingerprint classification. PyTorch Framework [34] is used in training of the models and there are many model architectures that PyTorch supplies to developers. The experiments are conducted with two of them: VGG [35], and ResNet [36].

In the model training stage, the effect of transfer learning, a popular technique used in deep learning especially when the dataset size for training is small, is also analysed. The main idea behind transfer learning is using the knowledge stored in the weights of the previously trained models for the problems where less data is available. Although the dataset size of NIST SD4 is not that insufficient for classification, the effect of injecting knowledge from other datasets is observed in this study.

Additionally, the models are trained with different numbers of training images in order to observe the effect of training set size. Because deep learning systems are data-hungry, the expectation is that more data yields better performance results.

The details of the model architectures and experiments are explained in the following sections.

4.2.1 Model Architectures

In this study using VGG, and ResNet architectures, 3 different models are trained on NIST SD4 dataset: VGG16, VGG19, and ResNet18. The performance results of these models on NIST SD4 database are shared in Section 4.3.

4.2.1.1 VGG

In [35], Simonyan and Zisserman propose a new Convolutional Neural Network(CNN) architecture named as VGG. Mainly, what they propose is simplicity in filter sizes. Instead of using large convolution filters such as 7x7, 3x3 filters are used, but the depth of the network is increased up to 19 layers. They argue that this approach reduces the number of parameters compared to using less but larger filters and increases discrimination power by introducing more non-linear layers. They prepared 6 configurations with different layer depths but with the same generic design. The configurations are shown in Figure 4.1.

In spite of their powerful structure, there are some drawbacks of VGG networks. Although it is stated that the number of parameters is less than the previous network architectures, still the amount is not sufficiently small. There are 138 million parameters in configuration D(16 layers) and 144 million parameters in configuration E(19 layers). The networks take up more than 500 MB storage space. Additionally, because they are deep i.e. the network has many layers, and have many parameters, training these networks may take a very long time. However, they can achieve good results in many tasks thus, they are still preferable.

In this study, adopted configurations of VGG architecture are VGG16 and VGG19 (configuration D and E respectively) with batch normalization layers.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.1: Configurations of VGG networks [35].

4.2.1.2 ResNet

In [36], He et al. argued that theoretically, when the layer size increases, the model accuracy should also increase. However, in practice, they observed that when the layers get deeper, the accuracy degrades. When identity networks are added to a simple network, normally it is expected that the accuracy should be the same with the simple network because there is no effect of identity networks, they simply output what the input is. However, when the layers get deeper, the accuracy decreases compared to the simple network. In order to overcome this problem, authors hypothesized that instead of learning underlying mapping of directly stacked layers($H(x)$), it is easier to learn the residual mapping($F(x)$). According to that hypothesis, identity shortcut connections are added to

the network and several layers are skipped. With that model instead of optimizing $H(x)$ function, $F(x) = H(x) - x$ is used and thus residual mapping is fitted as $F(x) + x$. (Figure 4.2). With this approach, very deep networks (for instance 152 layers) become trainable.

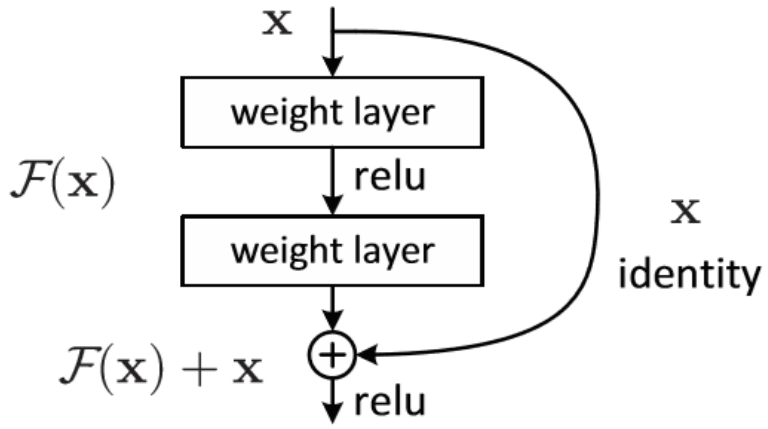


Figure 4.2: A Residual block [36].

The constructed architecture is similar to VGG and mostly 3x3 convolutional filters are used. When the architecture is compared to plain networks with the same layer size, it is shown that the residual network performs better. However, when shallower versions of the networks are used, residual networks achieve similar results, on the other hand, they take less time to train. The proposed configurations are shown in Figure 4.3 as it is in [36].

The advantage of ResNet architecture is the ability to train very deep networks while keeping the number of parameters very low. For example, ResNet50 has about 23 million parameters which are very low compared to VGG networks even though VGG networks are shallower. Thus, the memory they take is much less than VGG networks, e.g ResNet18 takes up only about 45 MB storage space. Due to these reasons, it is much faster to train ResNet networks. Because they are faster in training and capable of training much more layers with high accuracy results, they are preferred to be used for many different tasks.

In this study, ResNet18 architecture is selected because the depth of that network would be enough for the fingerprint classification problem and the training would take a shorter time.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.3: Configurations of ResNet architectures. Brackets indicate that they are residual blocks and the number beside them indicate the number of blocks used [36].

4.2.2 Transfer Learning

Deep learning systems can achieve good results when they are trained with large amounts of data. However, it is not always easy to obtain sufficient data for many tasks. Transfer learning aims at reducing the need for huge amounts of data. The main idea is transferring the knowledge obtained while solving one problem to another. This approach is feasible thanks to the structure of neural networks. Neural networks learn basic general information such as edges and corners in the first layers and towards last layers, they learn more problem-specific information. For many problems, the basic information that is required is the same i.e. the network should first learn to detect edges, etc. For similar tasks with similar datasets, even more information can be shared and the previously trained network is suitable for reuse with more layers. If the previous task/dataset is not very similar to the task at hand, transfer learning is still applicable if there is a sufficiently large dataset for a smaller training. Moreover, even if there is enough data(e.g millions of images) to train the network from scratch, applying transfer learning may reduce the time required for training. Since it may take weeks to train a large network with a huge amount of data, transferring the common knowledge allows the researchers to spend more time on problem-specific knowledge.

There are two widely used techniques to apply transfer learning: **Fine-Tuning** the pretrained network and using the pretrained network as a **Feature Extractor**.

- **Feature Extractor:** Using the pretrained network as a feature extractor means only the last layer of the pretrained network is removed. The weights of remaining layers are frozen in order not to get affected from the new training process and this frozen

part serves as feature extractor. The output of the remaining, frozen network is used as a feature vector. In order to adapt the network to the new problem, the only remaining task is training a new fully connected layer on top of the network to replace the last fully connected layer or training another linear classifier such as SVM.

- **Fine-Tuning:** Fine-tuning is generally applied in two ways: partially training some layers of the network or fully training all layers. Because more general features are learned in the earlier layers, freezing weights of those layers and training several latter layers may be preferred. However, if there is enough data for training, the network may be fully trained with the initialization of a previously trained network's weights.

The choice of using a network as a feature extractor or fine-tuning the network depends on some concerns. In Table 4.1, transfer learning scenarios in terms of dataset similarities and dataset sizes are explained.

Table 4.1: Transfer Learning scenarios in terms of dataset similarity and sizes.

	Similar Datasets	**Different Datasets**
Small Dataset Size	Using the pretrained network as a feature extractor might be suitable due to the similarity of the datasets. It is expected that the network previously learned the characteristics of the dataset and that information can be successfully shared.	This is the most difficult case where both feature extractor and fine-tuning approaches might not be beneficial. However, a feature extractor can be obtained from the network by removing not only the last layer but also the earlier layers and a linear classifier can be trained using the extracted features.
Large Dataset Size	Fine-tuning the network is preferable because there is enough data to train some layers and introduce more variance related to the specific characteristics of the new dataset.	Fine-tuning the network is preferable with training many or, maybe, all of the layers. The data may even be enough for training the network from scratch, but initialization of the weights with a pretrained model's weights can be beneficial.

The models pretrained on the ImageNet [37] database are used in many cases when applying transfer learning. ImageNet project is a very important milestone and aims to provide a large database for researches in the computer vision field. It contains about 14 million images of approximately 22000 object categories. ImageNet project organizes a competition every year, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [38] which is one of the largest and most popular competitions on object recognition

and image classification. In ILSVRC, a subset of ImageNet dataset is used which has approximately 1.2 million training images, 50000 validation images, and 100000 testing images and, 1000 object categories [35, 36]. Many famous Convolutional Neural Network architectures are actually developed in the scope of this competition.

In this study, with the three aforementioned deep neural network models(VGG16, VGG19, and ResNet18), two techniques of transfer learning are applied using the networks previously trained on the ImageNet dataset. In the fine-tuning case, all layers of the network are trained with initialization of the pretrained network's weights. This is because the fingerprint data is quite different compared to data in ImageNet. In the feature extractor case, only the last fully connected layers of each network are removed and replaced with new, randomly initialized fully connected layers. The weights of the convolutional parts of the networks are frozen and only the last newly added fully connected layer is trained for performing classification. In this approach, none of the layers are fine-tuned and the convolutional parts of the networks act as a feature extractor.

4.2.3 Dataset Sizes

Deep learning systems are data-hungry systems. In order to train deep and accurate networks, the dataset size should be sufficiently large. Although "sufficiently large" differs from case to case(Sometimes millions of data points are needed.), generally more data means better performance results and better generalization abilities because more complex model architectures can be trained if there is enough data and, in this way, the variations of the data can be captured in more detail.

In this study, in order to observe the effect of dataset sizes, the models are trained using different number of training samples. The expectation is that increasing the dataset size also increases model performances. However, there might be a maximum point in the model performances that the models reach and after that point increasing the amount of data would not make improvements. This situation is also tried to be observed in this study.

4.2.4 Summary

In this study, several training setups are implemented. Three model types are trained with two transfer learning techniques and without any applying transfer learning.

(For the sake of completeness, not pretrained models will also be mentioned in the transfer learning title). Additionally, the training subset of NIST SD4 is split further into 5 different partitions of different sizes and all the previously mentioned models with different transfer learning setups are trained on these datasets.

In detail, in the experiments {Model Types x Transfer Learning Setups x Dataset Sizes} models i.e. 3 model types (VGG16, VGG19 and ResNet18) x 3 transfer learning setups (Fine-tuning, Feature extractor and not pretrained) x 5 dataset sizes (125,250,500,1000 and 2000) = 45 models are trained. These steps are also repeated for the the four-class version of the problem. Thus, in total, 90 models are trained for the experiments.

For performance calculation, all tests are performed on NIST SD4 test dataset, as mentioned in Section 4.1.

4.3 Experimental Results

In this section, performances of the models are reported in terms of accuracies, and confusion matrices. Results are shared under separate sections for each experiment context (Model Types, Transfer Learning, and Dataset Sizes) in terms of maximum performances they can achieve. In the summary section, the results are interpreted all together, and a more detailed analysis is performed in terms of the fingerprint classification domain.

4.3.1 Results for Different Model Types

Models are trained with VGG16, VGG19, and ResNet18 architectures in this study. Accuracy results of the models are shown in Figure 4.4. These are the highest accuracies that the models achieve in different setups which are explained before in Section 4.2. The aim of this section is to compare the maximum performances that the models can achieve regardless of transfer learning types and dataset sizes. In this context, the best performing model is, not surprisingly, VGG19 due to its complexity. VGG16 follows VGG19 and the worst performance belongs to ResNet18. However, the accuracy of ResNet18 is still remarkably close to VGG models. The results show that both architectures are really powerful. As expected, training of ResNet18 took much less time compared to VGG networks yet, it achieved similar results to VGG networks.

These results show that using deep learning for fingerprint classification is practical and yields to highly accurate results. Without the need of any feature engineering, and

any complicated preprocessing (except for normalization using the mean and standard deviation of the dataset), deep learning systems achieve high results that is comparable with the literature in both four-class and five-class classification.

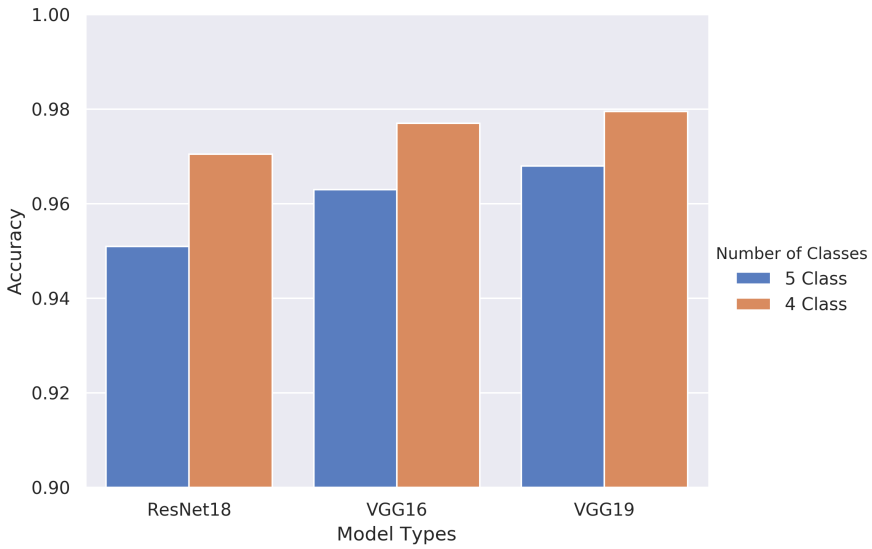


Figure 4.4: Accuracies of models with different architectures.

4.3.2 Results for Different Transfer Learning Setups

All models are trained using two types of transfer learning and once with no pretraining networks. As in the model types comparison, the most accurate models are chosen to compare transfer learning setups because the intention is to see the maximum accuracy point that they can achieve. The accuracy results are shown in Figure 4.5. In this setup, fine-tuned models take the lead which is not surprising. Although the fingerprint dataset is not very similar to the ImageNet dataset, initializing the networks with the weights of the pretrained networks contributes to the performances as it kind of introduces more data to the networks. However, again because the dataset is different from ImageNet data, in the feature extractor seyup, the networks cannot achieve good results. The information learnt in the previous network seems not enough to describe the fingerprint data.

As a result, it is observed that if the networks are trained with sufficient amount of problem-specific data, transfer learning can achieve very good and even state-of-the-art results for fingerprint classification.

Additionally, another observation obtained while training the networks is that the

convergence of the models with fine-tuning are easier. Not pretrained networks sometimes fail to converge and training process is required to be restarted especially for VGG networks. However, this is not the case for fine-tuned networks since they are already well initialized.

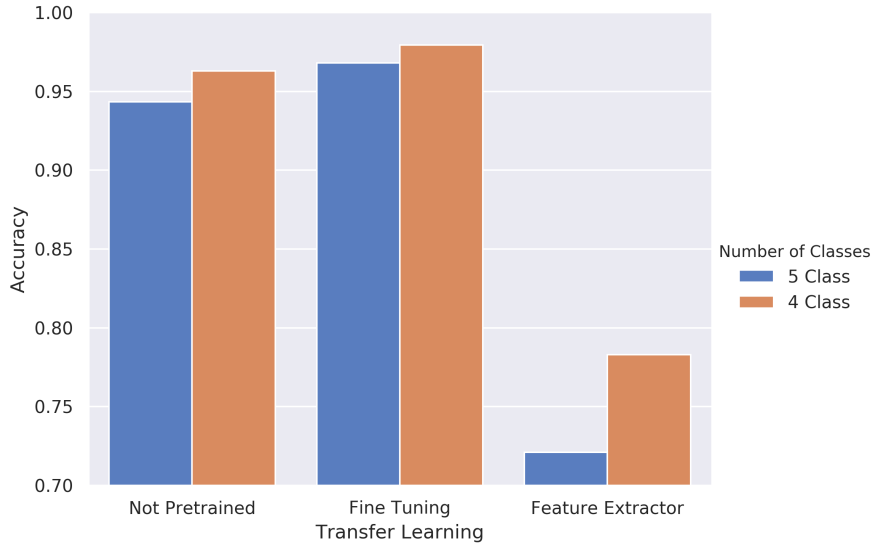


Figure 4.5: Accuracies of models with transfer learning methodologies.

4.3.3 Results for Different Dataset Sizes

In this experiment, all models are trained with 5 different training set sizes: 125, 250, 500, 1000, and 2000. The test dataset size remains the same i.e. 2000 image. The proportion between classes are also kept the same in differently sized training sets in order to have the same uniform distribution among the classes and not to affect accuracies with class imbalance. The expectation is that with the growth in dataset size, the models perform better. The results are shown in Figure 4.6. The figure shows the maximum accuracies achieved by the models trained with mentioned dataset sizes in various experiments, they may be again interpreted as they are the highest accuracies that those dataset sizes can achieve in the experimental setups designed for this study.

The results show that, as expected, the increase in dataset sizes improves model performances. Therefore, it may be stated that with larger number of training samples the deep learning models can perform better. However, accuracies begin to saturate after a point. The accuracy increase between the models with 1000 dataset size and 2000 dataset size is lower than the increase between the models with 125 dataset size and 250 dataset size. With this observation, it may be stated that even though the model performances

improve with more training samples, it is very probable that there is a maximum point that the improvement will stop. However, with 2000 data, it is not possible to state the amount of data where that maximum point definitely is. Nevertheless, in both four-class and five-class classification, the improvement in accuracies is remarkable with increasing dataset sizes.

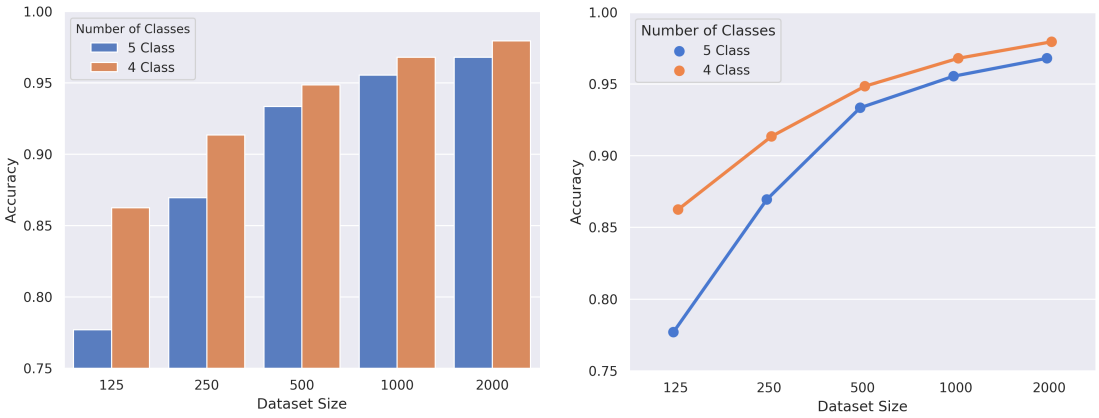


Figure 4.6: Accuracies of models trained with different dataset sizes.

4.4 Summary

The results for all experiments conducted in the scope of this thesis for fingerprint classification are shown in Figure 4.7 (for five-class classification) and Figure 4.8 (for four-class classification) as accuracies calculated at every 5 epochs. It can be clearly seen that as the dataset size increases, in all cases, model performances improve. In a similar fashion, transfer Learning via fine-tuning achieves the highest results compared to other techniques, and the feature extractor method fails in classification of fingerprints (because the datasets are not similar). Additionally, for the networks that are not pretrained, training dataset size is very important. They cannot learn the underlying structure of the data when the number of training samples is too small. However, with the help of transfer learning, the accuracies increase significantly in especially VGG models. According to the results, ResNet model is able to extract more information with fewer data without overfitting. This is mainly due to the identity block architecture. The VGG models generally require more data. VGG19 model performs worst when the dataset is too small (such as 125), this may be mainly because the architecture is deeper and more complex, and with less data it is more prone to overfitting and hence cannot generalize well. However, when enough data is supplied, it gives the best performance among all other setups. Basically, fine-tuning

the network, kind of supplies more data by transferring knowledge to the networks, so VGG19 reaches the maximum performance with fine-tuning.

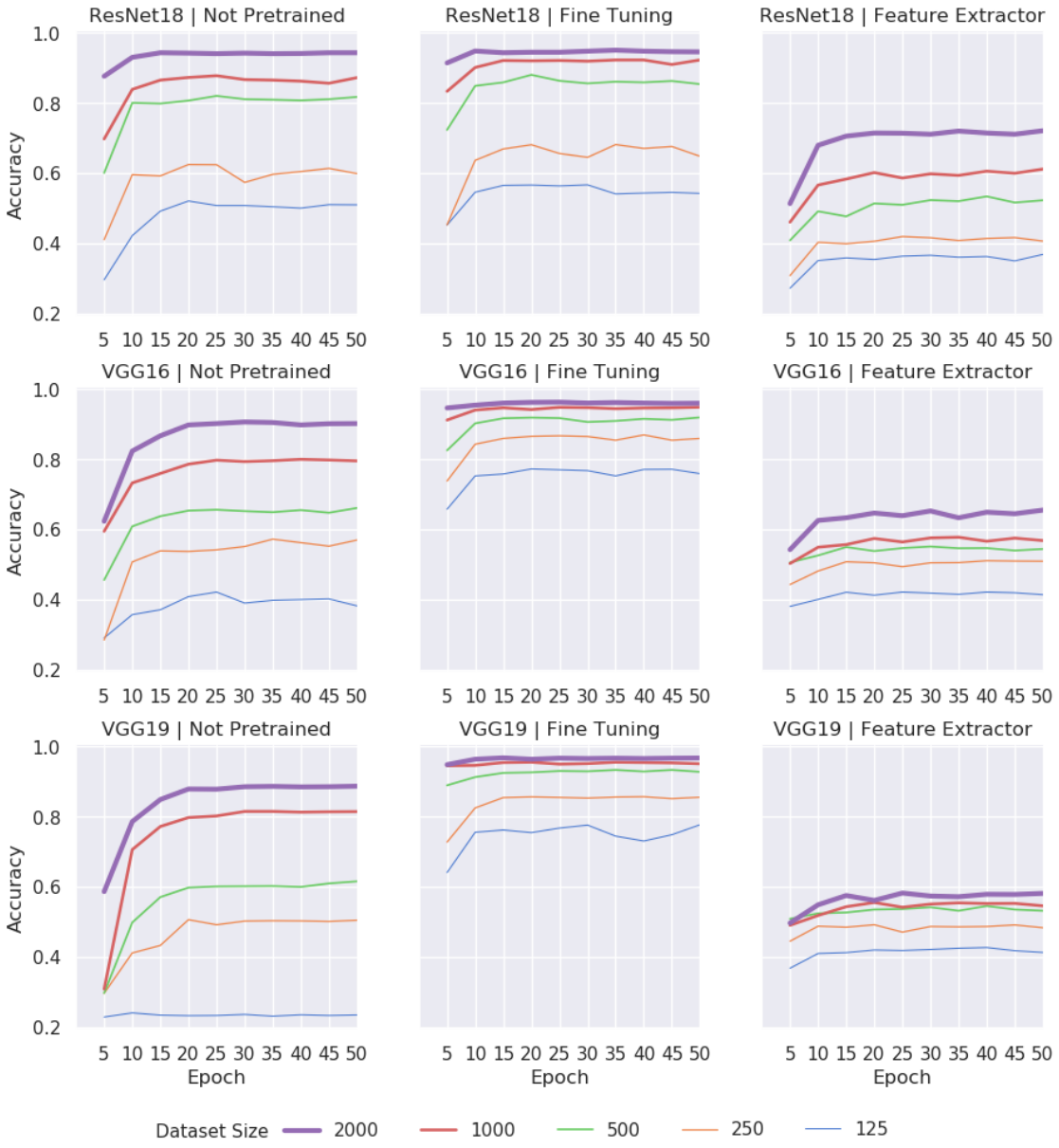


Figure 4.7: Accuracy results of all experiments with 5 classes. Test set accuracies are calculated at every 5 epoch.

In fingerprint classification, confusion matrices are also a common way to show the system performances. The confusion matrices obtained by the best performing models for five-class and four-class classification are shown in Table 4.2 and Table 4.3, respectively. The aim of this analysis is mainly to be comparable with the literature. Best performing models in both four-class and five-class classification are fine-tuned VGG19 networks

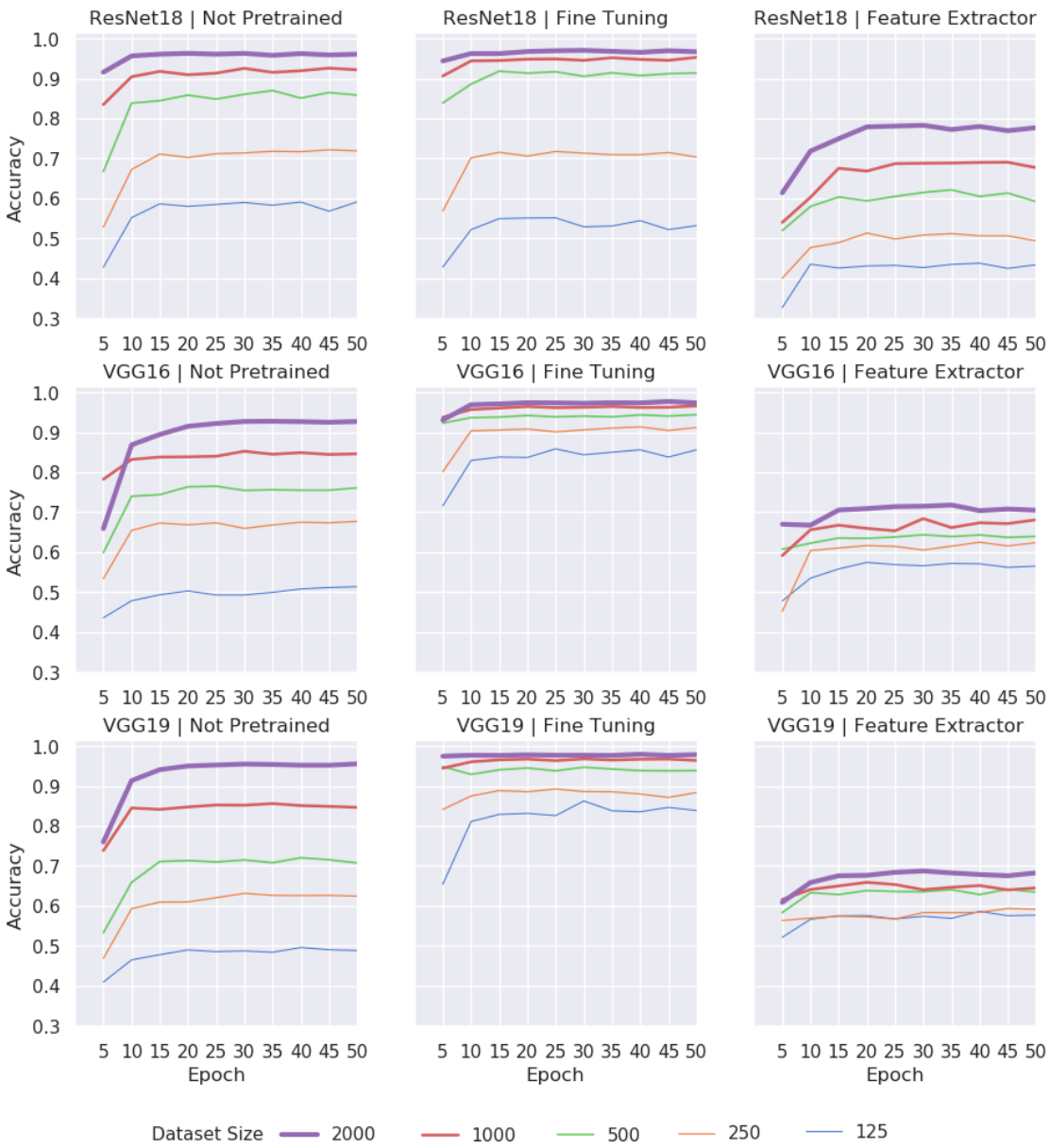


Figure 4.8: Accuracy results of all experiments with 4 classes. Test set accuracies are calculated at every 5 epoch.

trained with of 2000 samples. In the confusion matrices, the rows do not sum up to 400 because of the multiple labelled images mentioned in Section 4.1. The predictions for the images that have two classes are assumed as correct if the predicted label is one of the two labels. In five-class classification, the model shows the worst performance in the Tented Arch class with 93.54% accuracy. It mostly predicts the Tented Arch class as the Arch class as expected. For the Arch class, the only mistake that the model makes is to confuse 5 images for the Tented Arch class. When the Tented Arch and Arch classes are combined, this confusion is removed and the accuracy for both classes increases to 99.13%. Additionally, this model also seems to confuse Tented Arch and Loop classes

Table 4.2: Confusion matrix for five-class classification of fine-tuned VGG19 model trained with 2000 samples.

	Predicted label					
True Label 96.8%	**A**	**L**	**R**	**T**	**W**	**Accuracy**
A	428	0	0	5	0	98.84%
L	1	383	0	9	1	97.20%
R	2	0	380	18	0	95%
T	17	4	3	348	0	93.54%
W	1	2	1	0	397	99%

Table 4.3: Confusion matrix for four-class classification of fine-tuned VGG19 model trained with 2000 samples.

	Predicted label				
True Label 97.95%	**A**	**L**	**R**	**W**	**Accuracy**
A	801	4	3	0	99.13%
L	11	387	0	0	97.23%
R	17	1	376	0	95.43%
W	3	1	1	395	98.75%

in five-class classification. In four-class classification, merging the two classes obviously shifts the confusion of Tented Arch and Loop classes to Arch and Loop classes. The main reason for this problem might be the rotation of fingerprint images. The only difference between Tented Arch and Loop classes is the delta locations, in other words, the curvature directions. If the delta points are close to the center, the model may find it difficult to differentiate between the Loops and Tented Arch classes.

The results show that the performances outperform many systems proposed in the literature such as [21, 22, 23] and show that with minimal effort, training deep learning systems can achieve very good results in fingerprint classification. In Table 4.4, comparison of the results between the proposed method and the literature is shown.

Table 4.4: Comparison between the accuracies of the proposed method and the literature for fingerprint classification.

Method	**5 class**	**4 class**
Tang et.al [23]	-	95.1%
Listyalina and Mustiadi [24]	94.7%	96.2%
Michelsanti et.al. [25]	-	95.05%
Proposed method	**96.8%**	**97.95%**

CHAPTER 5

MINUTIAE EXTRACTION

5.1 Dataset

Fingerprint Verification Challenge is a large international competition in the fingerprint research area which was first held in 2000. The aim is to create a benchmark for the fingerprint verification task and evaluate the recent advances in the academy and industry. In this study, the database published for FVC2002 competition is used for our minutiae extraction study. The FVC2002 database is composed of four different databases, two of them are obtained using different optical sensors (DB1, DB2), one is obtained using a capacitive sensor (DB3), and the last one is synthetically generated (DB4). These databases are split into two parts Set A and Set B, for parameter tuning and performance evaluation, respectively. In this study,DB1-A database is used because the hand-labelled ground-truth minutiae data is available for that database. The FVC2002 DB1-A database contains 8 impressions of 100 fingers, in total 800 fingerprint images of size 388x374 pixel .

Training and test split is performed in accordance with the common approach adopted in the machine learning community: 80% for training, 20% testing. While splitting the dataset, it is made sure that training and test sets do not share the different impressions of the same finger. In other words, the split process is based on fingers, not impressions.

5.2 Model Training

In the minutiae extraction phase, generally, patch-based systems are adopted in the literature. We develop a similar in this study. However, in many studies, there is a preprocessing step prior to minutiae extraction. In traditional methods, the fingerprint image is enhanced and in deep learning methods, the features that require domain knowledge is used. (There may be more than one network to detect minutiae points and their precise

locations.) The aim of this study is to eliminate these preliminary steps and to analyse if the networks would still be able to detect minutiae points accurately with no pre-processing using a single minutiae extractor network. Additionally, the effects of patch sizes, network architectures, and simple data augmentation on the accuracies are also observed.

In order to do this, again a classification system is adopted. Minutiae and non-minutia patches are extracted from fingerprint images for training and query patches from test set are classified into classes as they are minutia or not.

The adopted methodology can be coarsely described as following:

1. FVC2002 DB1-A dataset is split into training and test sets.

2. Training and test patches are extracted from both sets.

3. Using extracted minutia patches, which contain minutiae, and non-minutiae patches, which do not contain minutiae, classifier networks are trained. The performance of the networks is measured on test patches to analyse the performance of the patch-based binary classification networks.

4. After the training step is completed, the network is tested over full-sized test images in a sliding window fashion in order to extract the true performance of the model in terms of the minutia extraction problem.

Because there are two types of minutiae points, normally this problem requires a three-class classifier with bifurcation, ridge ending, and non-minutiae classes. However, the main aim of the minutiae extraction stage is to detect minutiae points regardless of their type and many matching algorithms do not discriminate between the types of the minutiae points. Thus, minutiae classes are merged into one class and classification is performed with 2 classes which are "minutiae" and "non-minutiae".

In the patch extraction part, minutiae patches are extracted with the minutiae point at the center of the patch. Equal number of non-minutiae patches are extracted randomly in two different ways. First approach is the naive one, it is guaranteed that the patch will not contain any minutia points at all. However, this approach might be problematic because there are minutiae points which are very close to each other in fingerprints. When the randomly taken patches are filtered not to contain any minutiae points, it may cause loss of many informative patches, especially in larger patch sizes. In the second approach, non-minutiae patches are extracted so that 10x10 square at the center of the image does not contain any minutiae points. With this approach, it is intended to overcome the problem mentioned in the first approach. However, this might also be problematic because the network may extract irrelevant information from the minutiae points which are in the non-minutiae patches. For both methods for non-minutiae patch extraction, the number of flat regions that contain all white areas are also limited so that they do not suppress the number of patches that contain ridge information. Additionally, two different patch sizes

are adopted in order to analyse the effect of the patch sizes: 30x30 and 50x50. For both patch sizes, the two patch extraction approaches explained above are adopted.

Because the patch-based binary classification problem is not very complex and the patch sizes are small, a complex network architecture such as VGG is not required. For this reason, two types of networks are used in this study: A custom small network designed for this task and ResNet18. The custom network architecture is shown in Figure 5.1.

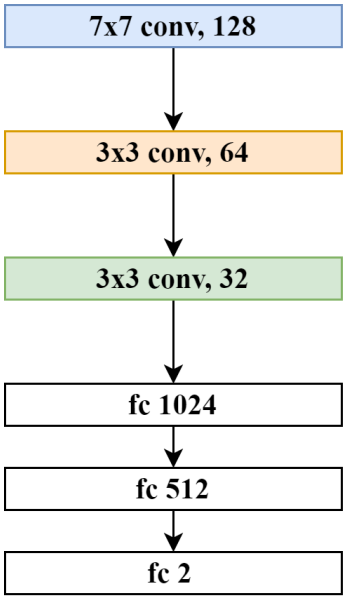


Figure 5.1: Custom network architecture used for minutiae extraction.

Additionally, the effect of simple data augmentation is observed by applying random horizontal and vertical flip to the patches. Data augmentation introduces more variability to the network, thus its logic is similar to introducing more data. However, because it would take too long to train all the training setups with and without data augmentation, unlike to fingerprint classification experiments, it is not applied for all model training setups. The effect is observed in both patch sizes and patch extraction approaches only with the custom network.

In total, 12 networks are trained for minutiae extraction experiments: 8 models with data augmentation (2 Patch sizes x 2 Patch extraction approaches x 2 Model types) and 4 models without data augmentation (2 Patch sizes x 2 Patch extraction approaches x 1 Model Type).

The predictions over the whole test images are made using a sliding window approach. A window with the same size as the patches is slid over the test images pixel by pixel and the class of each window is predicted by the model with a probability. With this approach, all pixels are scanned if they contain minutiae or not.

After the predictions are completed on whole test images, response images are constructed using prediction probabilities. However as a drawback of pixel by pixel sliding window approach, the network may detect too many spurious minutiae points because there are too many pixels that are not minutiae. The response images are post-processed in order to get more accurate results and get rid of spurious responses. As post-processing, hard thresholds on the probabilities, morphological operations, and Non-maximum Suppression(NMS) are applied to the response images. The thresholded response image is firstly opened and then skeletonized. The aim of using these morphological operations is to get rid of tiny response regions that result in spurious minutiae points and to merge close responses that gather around genuine minutiae points and get a more confident response. After applying morphological operations, NMS is applied to the image. NMS is commonly used for post-processing in object detection. The aim is to reduce the number of responses by applying intersection over union strategy. This strategy eliminates the prediction boxes that have lower probabilities if the intersection of the boxes divided by their union is larger than a threshold. After applying NMS, the remaining boxes are taken as the resultant predictions of the system. The precise location of the minutiae points is taken as the center of the prediction boxes. True positive(TP), false positive(FP) and false negative(FN) predictions are calculated using the boxes in a way that minutiae points in the remaining boxes are calculated as TP, boxes that do not contain minutiae points as FP, and minutiae points that are not enclosed by any boxes as FN. Because the predictions are made pixel by pixel, true negatives(TN) are not calculated since the amount of correct non-minutiae predictions are huge. In these calculations, it is guaranteed that if there are multiple minutiae points in prediction boxes, they are not counted in true positives more than once.

The performances are evaluated in terms of precision, recall, and f-measure using the calculated TP, FP, and FN values. Accuracy is not preferred since it is not a proper measurement in the constructed methodology. Due to the pixel-wise sliding window approach, predicted non-minutiae and minutiae points are highly imbalanced. Accuracy would be positively affected by the number of true negative predictions thus using it might be misleading.

5.3 Experimental Results

The experimental setup is constructed with 12 models, as mentioned previously: 50x50, and 30x30 patch sizes, two non-minutiae patch extraction approaches(no minutiae at all, and no minutiae in 10x10 center window) with two model types(the custom network

and ResNet18) and applying data augmentation or not (only in the custom network). Examples of patches used in training are shown in Figure 5.2.

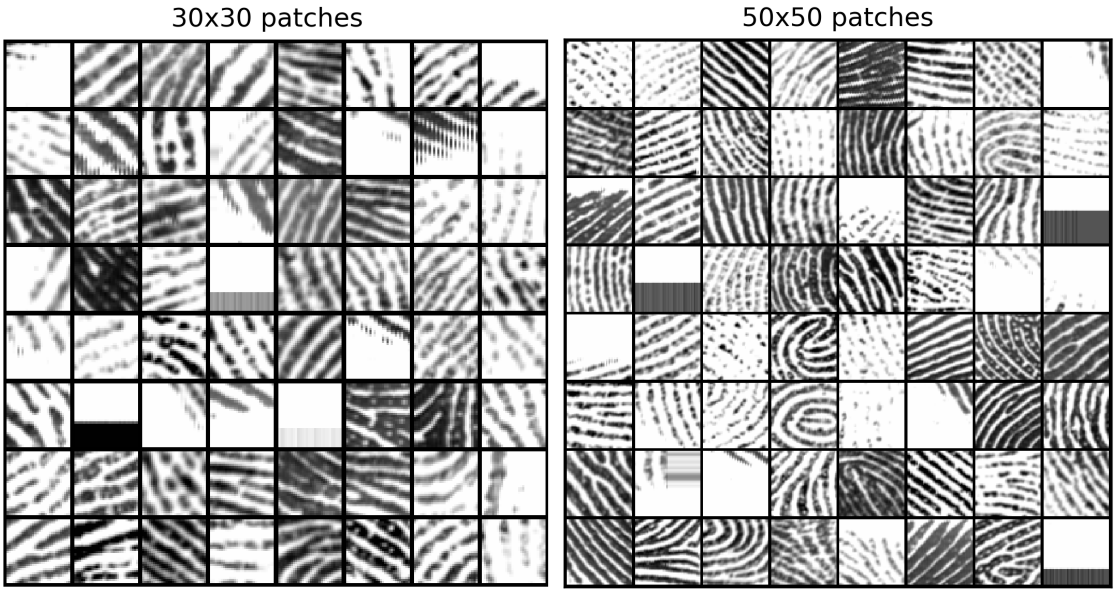
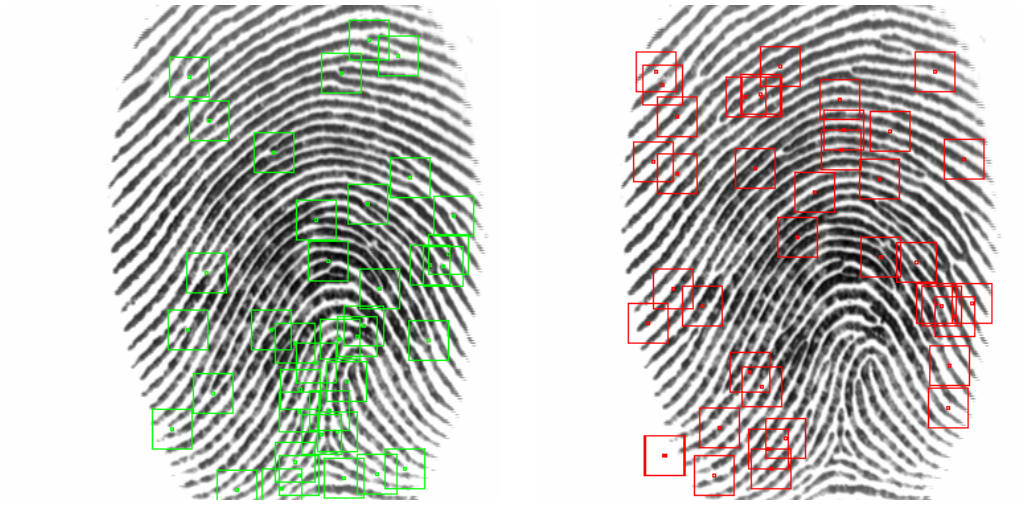
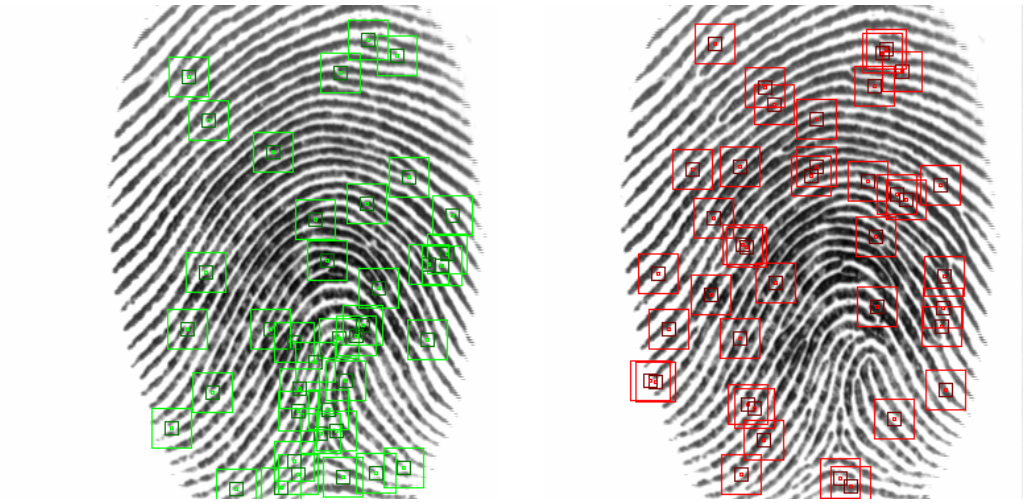


Figure 5.2: Patch examples of 30x30 and 50x50, respectively.

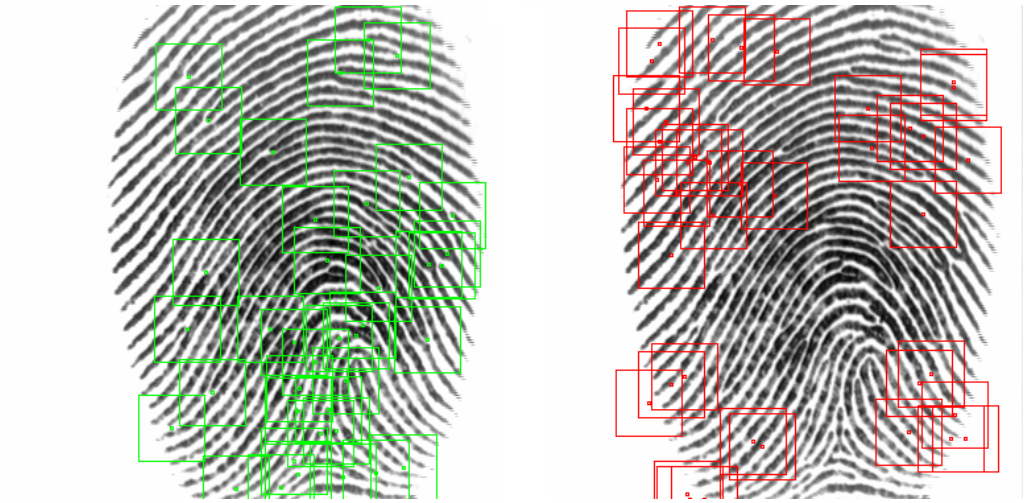
When the first approach (no minutiae at all) is adopted in 50x50 patch size, it is observed that the patches are generally taken from the sides and areas in the fingerprint images with very few minutiae points. More informative parts, i.e. core areas of the images and the parts where the minutiae points are close and frequent, are not sampled. The center core areas of the fingerprints carry important information because ridges are more intense at those parts and the curvature changes are also high. Due to this fact, more minutiae points are observed in those areas. Eliminating them in training might cause the network to fail on those parts in testing because in the training step the network does not learn that kind of information. In 50x50 patch size and the first approach shown in 5.3(c), it can be seen that non-minutiae patches are not taken from the center of the fingerprint image where many minutiae points are observed. However, by applying the second approach (no minutiae in 10x10 center window), also shown in Figure 5.3(d), this problem is alleviated. The samples are more uniformly distributed over the full-size image. Additionally, even though the second approach is also helpful for a more uniform patch extraction for 30x30 sized patches, the first approach does not seem to cause that much problem as in 50x50 patch size (Figure 5.3(a,b)).



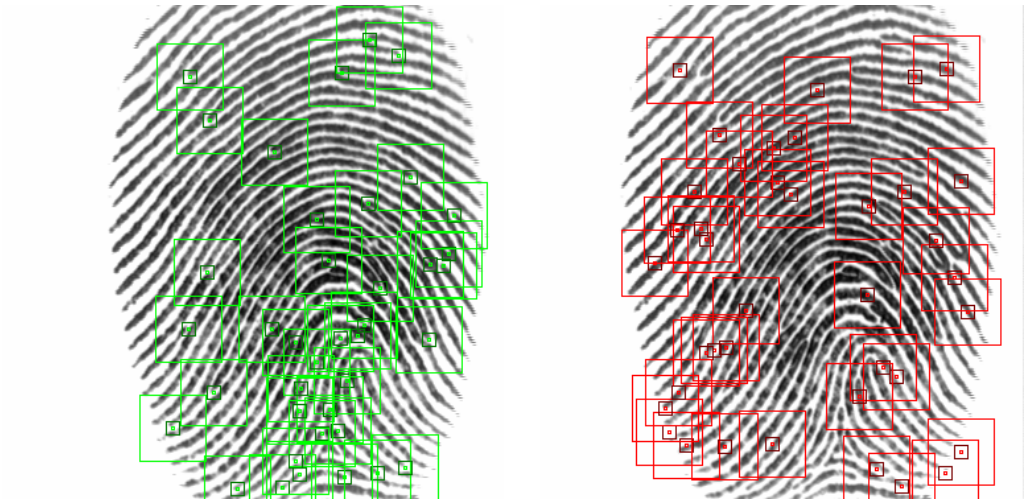
(a) 30x30 patches with non-minutiae patches that do not contain any minutiae points(first approach).



(b) 30x30 patches with non-minutiae patches that do not contain minutiae points at 10x10 center square(second approach).



(c) 50x50 patches with non-minutiae patches that do not contain any minutiae points(first approach).



(d) 50x50 patches with non-minutiae patches that do not contain minutiae points at 10x10 center square(second approach).

Figure 5.3: Different patch sizes and patch extraction approaches applied on an example image. Green squares indicate minutiae patches, red squares indicate non-minutiae patches.

For post-processing, it is observed that applying only NMS is not sufficient to eliminate spurious minutiae points. Applying morphological operations prior to NMS, eliminates more spurious minutiae points and helps the system make more precise predictions. The effect of applying morphological operations before NMS is shown in Figure 5.4 with images where both operations are applied and only NMS applied to the thresholded response image. It is observed that especially for low-quality images, even though small values are used for the intersection over union threshold, the number of remaining prediction boxes is still too many. Additionally, applying a hard threshold to the response images before both morphological operations and NMS helps to reduce the number of weak responses by preserving only very confident predictions for further processing. In all training setups, same post-processing steps are applied in a way that: probability threshold is 0.9, structuring element for opening is a disk with a radius of 2, NMS threshold for intersection over union is 0.5 and the prediction boxes are 20x20 pixels. An example prediction process in which the response image is post-processed using probability threshold, morphological operations, and NMS is shown in Figure 5.5.

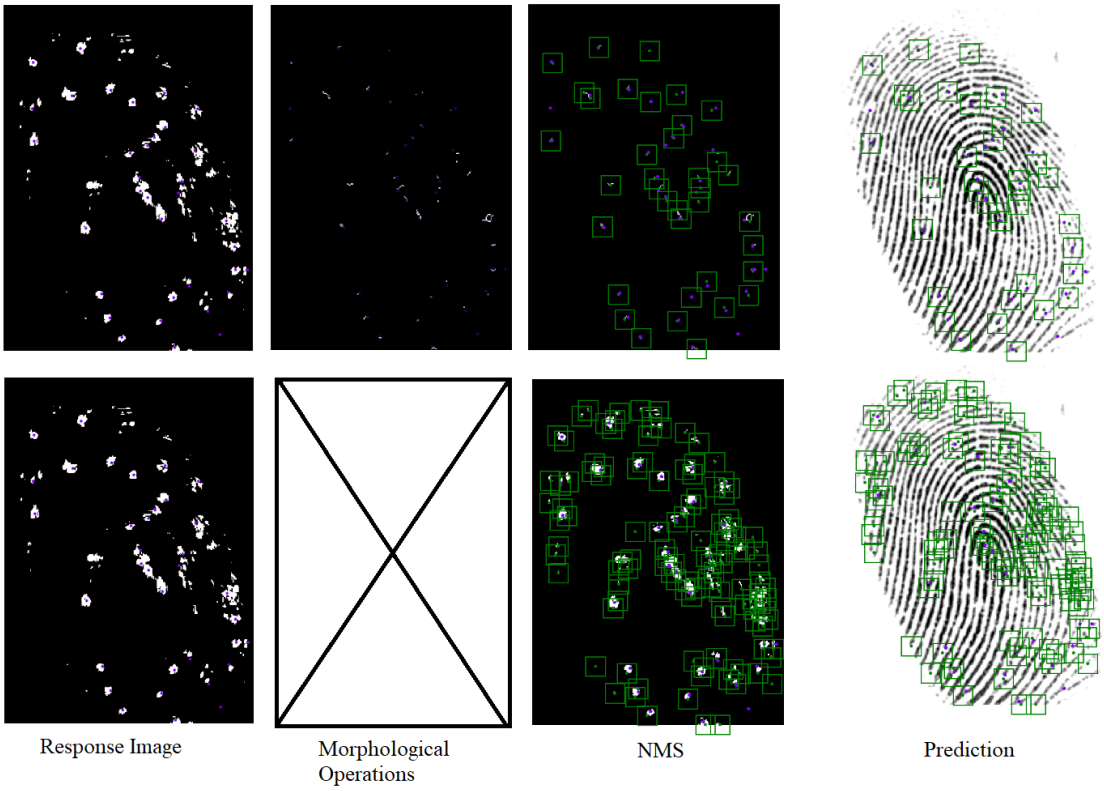


Figure 5.4: Effect of morphological operations. The first row shows morphological operations applied on the thresholded response image, the second row shows only NMS applied on the same image.

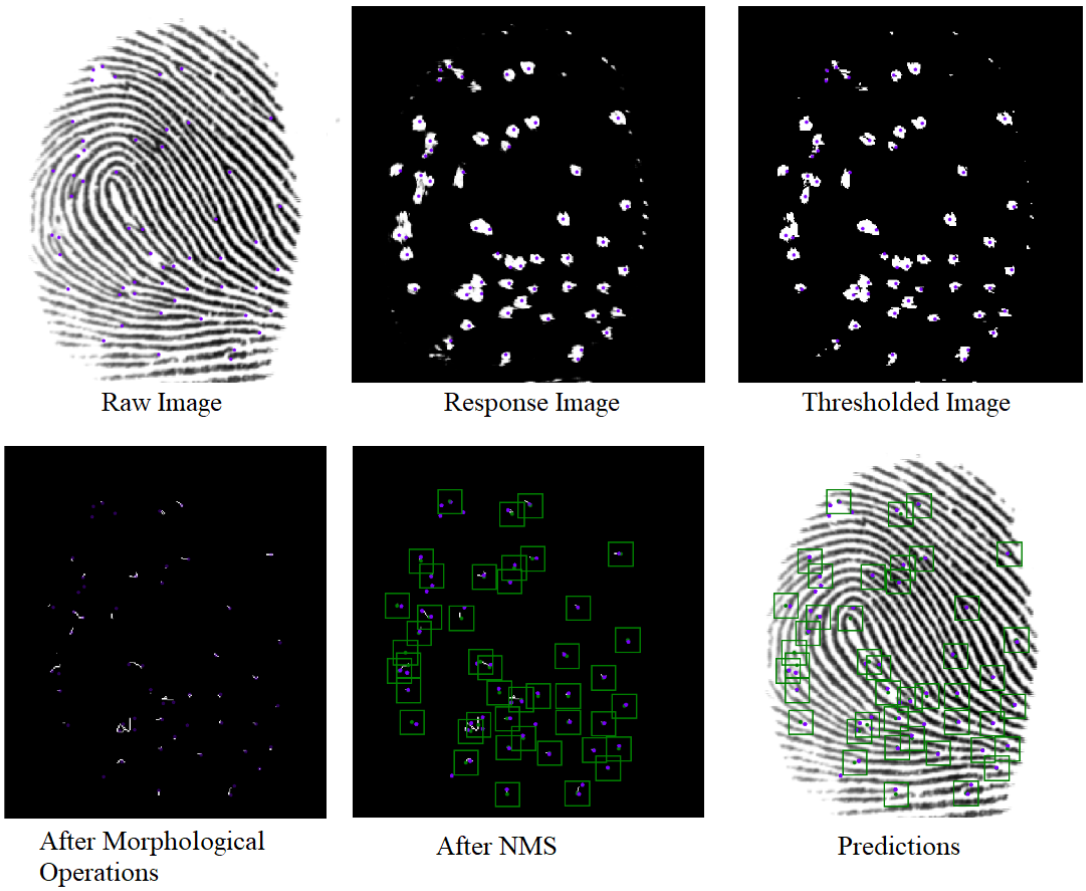


Figure 5.5: Prediction process on an example image. Purple dots show the ground truth minutiae points.

Performances of the 8 models trained for different patch sizes, patch extraction approaches, and model types are shown in Table 5.1. (In this table, data augmentation is applied for all models .) The results show that the custom small network performs better in all setups compared to the ResNet18 architecture. This supports the idea of not using very deep architectures because the problem is not that complex and patch sizes are small. A small network with 3 convolutional and 3 fully connected layers are capable of capturing the underlying structure of the data better. Additionally, 30x30 patch sized models with the first non-minutiae patch extraction approach(no minutiae at all) are observed to perform better than all other setups with both custom and ResNet18 architectures. They even achieve the best results with 82.62% precision, 92.24% recall, and 87.18% F1 score with the custom network architecture which is very close to [31] (Table 5.2). Because the patch size is small, the strict discrimination between minutiae and non-minutiae patches might have affected the network positively. When the 30x30 and 50x50 patches are compared, it is observed that 50x50 patch size has remarkably worse performance than 30x30 patch size in both model architectures. The main reason for this might be extracting

Table 5.1: Performances of the 8 models. C: custom network, RN18: ResNet18. 30x30 (1) and 50x50 (1) indicate the performance of the model which the patches are extracted with the first approach(not minutiae points at all). 30x30 (2) and 50x50 (2) indicate the second patch extraction approach (no minutiae in 10x10 center window).

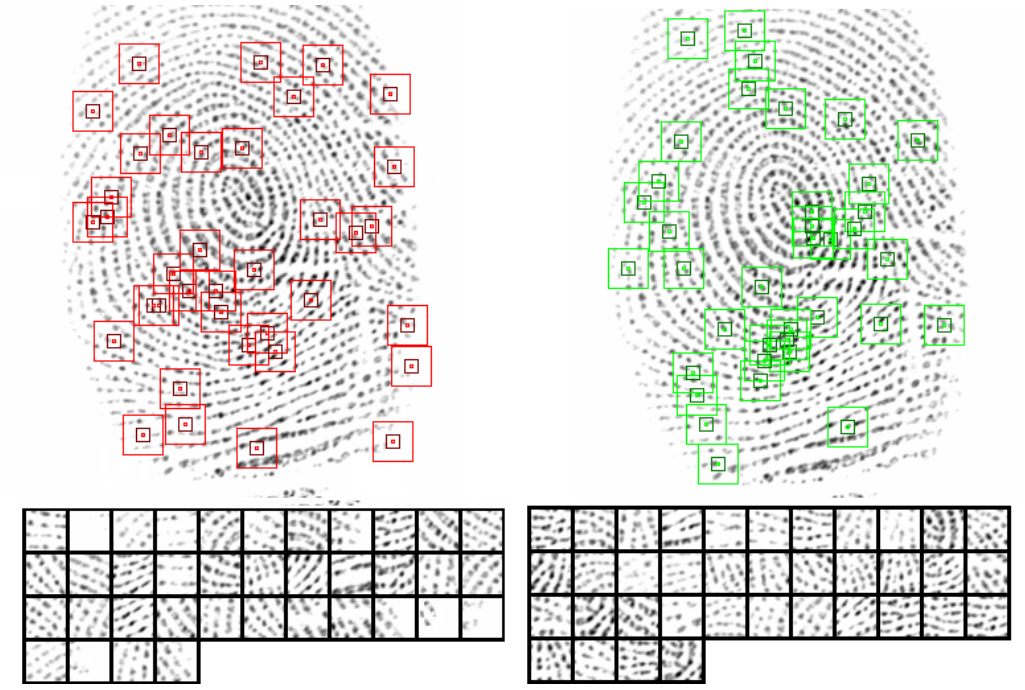
	30x30 (1)		**30x30 (2)**		**50x50 (1)**		**50x50 (2)**	
	C	**RN18**	**C**	**RN18**	**C**	**RN18**	**C**	**RN18**
TP	5800	5912	5674	5718	6222	6131	5847	3461
FP	1220	2552	1565	2684	3299	5307	3288	1850
FN	488	376	620	576	488	579	398	2784
Precision (%)	82.62	69.85	78.38	68.06	65.35	53.60	64.01	65.17
Recall (%)	92.24	94.02	90.15	90.85	92.73	91.37	93.63	55.42
F1 Score (%)	87.17	80.15	83.85	77.82	76.67	67.57	76.03	59.90

Table 5.2: Comparison between the proposed method and [31].

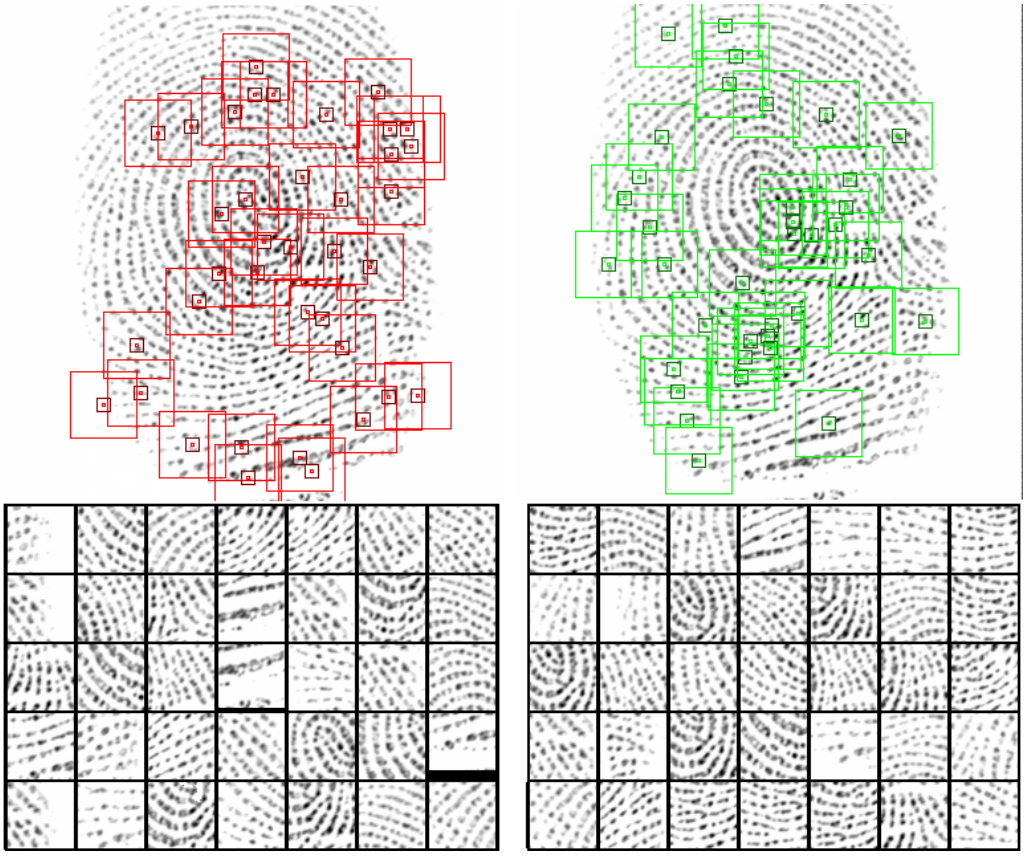
	Precision(%)	**Recall**(%)	**F1 Score**(%)
Zhou et al.[33]	87.9	87.9	87.9
Proposed method	**82.62**	**92.24**	**87.17**

insufficiently informative patches, especially in the first approach. However, it is seen that applying the second approach is also not helpful to overcome this problem for both patch sizes. The reason might be that the network gets confused when it sees minutiae points in non-minutiae patches. However, the results indicate that 30x30 patch sized custom network with the first patch extraction approach which strictly discriminates minutiae and non-minutiae patches, performs sufficiently good for a patch-based approach which does not include any preprocessing and feature extraction step.

Another observation is that, when the image qualities are low, patches might contain misleading information. This is mainly caused by the fact that low-quality images contain many disconnected ridge lines which are normally not minutiae. However, when small patches are taken from the image, they may look like ridge endings. This confuses the network during both training and testing. In training, the network sees non-minutiae patches which are very similar to minutiae patches. At test time, the network detects many spurious minutiae points on the disconnected ridges. In many patches, the network actually makes sensible predictions because even humans may fail to classify those patches correctly. Because all in all, the whole sight of the image is lost due to the patch-based approach, and those patches look like minutiae points. Patch examples of some low-quality images are shown in Figure 5.6.



(a) 30x30 patches.



(b) 50x50 patches.

Figure 5.6: Low quality image and patch examples. Red squares on the left images show extracted non-minutiae patches, green squares on the right images show minutiae patches of the same fingerprint image.

Due to the fact that low-quality images may cause problems, the performances of the models are also analysed with respect to the image qualities using the custom network. Precision, recall, and F1 Score metrics are shown in Table 5.3 for each quality level: high, medium, and low. It is observed that the models perform very well on high-quality images. The results are very satisfactory in especially in high-quality 30x30 patches with the first approach (88.09% precision, 93.90% recall, and 90.91% F1 Score). However, as expected, in medium and low-quality images, the model performances decrease especially in terms of precision. Although most of the genuine minutiae points are correctly detected, the spurious minutiae points suppress the performance. These results show the importance of preprocessing prior to training especially in low-quality images.

Table 5.3: Performance results of the custom network architectures according to the quality of the images.

(a) 30x30 (1)

	Precision(%)	**Recall**(%)	**F1 Score**(%)
High	88.09	93.90	90.91
Medium	80.63	92.38	86.10
Low	73.53	87.75	80.02
Overall	82.62	92.24	87.17

(b) 30x30 (2)

	Precision(%)	**Recall**(%)	**F1 Score**(%)
High	86.62	93.90	90.11
Medium	80.22	90.35	84.98
Low	67.32	85.18	75.20
Overall	78.38	90.15	83.85

(c) 50x50 (1)

	Precision(%)	**Recall**(%)	**F1 Score**(%)
High	77.16	92.88	84.30
Medium	63.72	93.51	75.79
Low	48.34	90.20	62.95
Overall	65.35	92.73	76.67

(d) 50x50 (2)

	Precision(%)	**Recall**(%)	**F1 Score**(%)
High	72.44	94.40	81.97
Medium	63.99	93.30	75.91
Low	50.55	92.57	65.39
Overall	64.01	93.63	76.03

In addition to the above analysis, the effect of simple data augmentation is also observed again using the custom network. The results with and without data augmentation are shown in Table 5.4. By applying data augmentation to the same sized data, the performances are boosted remarkably. In all cases, F1 Scores are boosted up to nearly 10%. This result can be interpreted also as the effect of dataset size because increasing the dataset size and using data augmentation favours the same goal: introducing more variability to the model in order to increase its generalization ability to perform well on unseen data.

Table 5.4: Performances of the data augmentation applied and not applied models. 30: 30x30 and 50:50x50 patch sizes. (1) indicates the first non-minutiae patch extraction approach. (2) indicates the second patch extraction approach.

	No data augmentation				**Data augmentation**			
	30(1)	**30(2)**	**50(1)**	**50(2)**	**30(1)**	**30(2)**	**50(1)**	**50(2)**
TP	5474	5441	5853	5441	5800	5674	6222	5847
FP	1917	2469	5326	4777	1220	1565	3299	3288
FN	814	853	857	804	488	620	488	398
Precision(%)	74.06	68.79	52.36	53.25	82.62	78.38	65.35	64.01
Recall(%)	87.05	86.45	87.23	87.13	92.24	90.15	92.73	93.63
F1 Score(%)	80.04	76.61	65.44	66.1	87.17	83.85	76.67	76.03

All in all, all the networks perform well in terms of recall metric which means that they can all detect genuine minutiae points accurately(except for one model). Where they differentiate is the number of spurious minutiae points that they detect. However, the models show promising results although no preprocessing and quality enhancement steps are applied. Increasing the amount of non-minutiae patches in training or an alternative approach to the pixel by pixel sliding window approach in testing may help to decrease the number of spurious minutiae points detected.

5.4 Summary

To sum up, in this study, a patch-based approach is adopted for minutiae extraction and models are trained using different patch sizes. The main aim is again to eliminate the preprocessing and feature extraction steps and trying to achieve accurate results using deep learning. The trained networks are tested over whole test images for performance

calculation in pixel by pixel sliding window fashion. By applying different post-processing techniques, the performances of the minutiae extraction models are enhanced and systems that perform very well on high-quality images and perform satisfactorily on medium and low-quality images are obtained. However, the importance of preprocessing, in low-quality images for getting highly accurate results in overall performance is also observed.

The results show that the performances of the systems are very high in terms of recall, but need improvements in precision because they detect too many spurious minutiae points. However, in general, without any preprocessing and feature extraction steps, the models achieve promising and even state-of-the-art results.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this study, deep learning systems are developed for fingerprint classification and minutiae extraction. The main aim is to analyse performances of deep learning systems under different conditions. Deep learning reduces the need for manual work and domain knowledge required before training fingerprint systems. Analysis are done on the effects of different model architectures, dataset sizes, transfer learning setups and simple data augmentation.

For fingerprint classification, different deep learning architectures are deployed to analyse the effect of model complexity. Additionally, transfer learning approach is adopted and different types of transfer learning are deployed to observe performance differences for all models. The models are trained with different dataset sizes in order to observe the impact of number of training samples. The results showed that deep learning is a good method for fingerprint classification and the models can achieve state-of-the-art results. In addition, even the dataset from which the knowledge is transferred is different, fine-tuned networks help to increase model performances. The experiment for different dataset sizes showed that when the dataset sizes increase the model performances also increase. As a result of all experiments, the best performing model for fingerprint classification is showed up as a fine-tuned VGG19 network with maximum number of training samples both for four-class and five-class classification. However, the results also show that ResNet18 networks can also achieve good and even better results in smaller dataset sizes. The performance results are shared in terms of accuracy and confusion matrices for fingerprint classification. The highest accuracies for 5-class and 4-class models are observed to be 96.8% and 97.95%, respectively.

For minutiae extraction, a patch-based classification approach is adopted. For different patch sizes, a small custom neural network architecture and ResNet18 are used in model training. Additionally, different approaches are adopted for patch extraction. The results are shared in terms of precision, recall and F1 Score for 2 different patch sizes(30x30 and 50x50), two different non-minutiae patch extraction approaches(non-

minutiae patch does not contain any minutiae points at all or the 10x10 square around the center points does not contain any minutiae points) and two model architectures(custom, simple architecture and ResNet18). The custom architecture is observed to perform better on minutiae extraction problem compared to ResNet18. Additionally, because the dataset has different quality images, the results are also discussed with respect to the qualities of the fingerprint images using the custom network architecture with different patch sizes and patch extraction approaches. The performances of the networks are very good in high-quality images. However, when the quality decreases, the network makes more mistakes by introducing spurious minutiae points. The best performance achieved by the models is 82.62% precision, 92.24% recall, and 87.17% F1 Score. The results show that without the need for binarizing or thinning, which is a common and essential step in many studies, the models are able to detect minutiae points with acceptable and promising performance.

All in all, for both fingerprint classification and minutiae extraction, it is observed that deep learning is a good methodology. Experiment results show that with the adoption of deep learning, very satisfactory results can be achieved. These results are also promising for applying deep learning for other steps of the fingerprint recognition pipeline.

6.2 Future Work

In minutiae extraction, extracting minutiae angles are also important for accurate matching. In order to find the angles, another network can be developed or the developed networks can be broadened to cover also the angle estimation. In the developed system, the number of minutiae and non-minutiae patches are equal, however they are highly imbalanced in whole images. The ratio of the patch data size can be changed to mimic the imbalanced ratio. Additionally, for different quality images, different systems can be trained with different patch sizes and then ensembled. For example, larger patch sizes can be used for low-quality images to gain insight into the underlying ridge continuity.

Last but not least, this study shows that deep learning is useful and practical for fingerprint analysis, thus it can be used in other stages of the fingerprint recognition pipeline such as orientation extraction, or end to end minutiae detection and matching.

REFERENCES

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

[2] E. R. Henry, *Classification and uses of finger prints*. HM Stationery Office, 1905.

[3] C. L. Wilson, G. T. Candela, and C. I. Watson, "Neural network fingerprint classification," *Journal of Artificial Neural Networks*, vol. 1, no. 2, pp. 203–228, 1994.

[4] K. Karu and A. K. Jain, "Fingerprint classification," *Pattern recognition*, vol. 29, no. 3, pp. 389–404, 1996.

[5] M. Liu, "Fingerprint classification based on adaboost learning from singularity features," *Pattern Recognition*, vol. 43, no. 3, pp. 1062–1070, 2010.

[6] K. Cao, L. Pang, J. Liang, and J. Tian, "Fingerprint classification by a hierarchical classifier," *Pattern Recognition*, vol. 46, no. 12, pp. 3186–3197, 2013.

[7] M. Kawagoe and A. Tojo, "Fingerprint pattern classification," *Pattern recognition*, vol. 17, no. 3, pp. 295–303, 1984.

[8] S. C. Chua1a, E. K. Wong, and A. W. C. Tan, "A fuzzy rule-based fingerprint image classification," *International Journal of Applied Engineering Research*, vol. 11, no. 13, pp. 7920–7925, 2016.

[9] K. Nilsson and J. Bigun, "Localization of corresponding points in fingerprints by complex filtering," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2135–2144, 2003.

[10] L. Fan, S. Wang, H. Wang, and T. Guo, "Singular points detection based on zero-pole model in fingerprint images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 929–940, 2008.

[11] B. Moayer and K. S. Fu, "A syntactic approach to fingerprint pattern recognition," *Pattern recognition*, vol. 7, no. 1-2, pp. 1–23, 1975.

[12] K. Rao and K. Balck, "Type classification of fingerprints: A syntactic approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 3, pp. 223–231, 1980.

[13] J.-H. Chang and K.-C. Fan, "A new model for fingerprint classification by ridge distribution sequences," *Pattern Recognition*, vol. 35, no. 6, pp. 1209–1223, 2002.

[14] D. Maio and D. Maltoni, "A structural approach to fingerprint classification," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 3, pp. 578–585, IEEE, 1996.

[15] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, "Fingerprint classification by directional image partitioning," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 402–421, 1999.

[16] J. Luo, D. Song, C. Xiu, S. Geng, and T. Dong, "Fingerprint classification combining curvelet transform and gray-level cooccurrence matrix," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[17] Y. Yao, P. Frasconi, and M. Pontil, "Fingerprint classification with combinations of support vector machines," in *International conference on audio-and video-based biometric person authentication*, pp. 253–258, Springer, 2001.

[18] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 4, pp. 348–359, 1999.

[19] R. Wang, C. Han, Y. Wu, and T. Guo, "Fingerprint classification based on depth neural network," *arXiv preprint arXiv:1409.5188*, 2014.

[20] D. El Hamdi, I. Elouedi, A. Fathallah, M. K. Nguyuen, and A. Hamouda, "Combining fingerprints and their radon transform as input to deep learning for a fingerprint classification task," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1448–1453, IEEE, 2018.

[21] Y. Tang, R. Li, Y. Liu, and J. Feng, "Fclassnet: a fingerprint classification network integrated with the domain knowledge," *Science China Information Sciences*, vol. 62, no. 12, p. 229102, 2019.

[22] L. Listyalina and I. Mustiadi, "Accurate and low-cost fingerprint classification via transfer learning," in *2019 5th International Conference on Science in Information Technology (ICSITech)*, pp. 27–32, IEEE, 2019.

[23] D. Michelsanti, A.-D. Ene, Y. Guichi, R. Stef, K. Nasrollahi, and T. B. Moeslund, "Fast fingerprint classification with deep neural networks.," in *VISIGRAPP (5: VISAPP)*, pp. 202–209, 2017.

[24] A. Farina, Z. M. Kovacs-Vajna, and A. Leone, "Fingerprint minutiae extraction from skeletonized binary images," *Pattern recognition*, vol. 32, no. 5, pp. 877–889, 1999.

[25] F. Zhao and X. Tang, "Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction," *Pattern Recognition*, vol. 40, no. 4, pp. 1270–1281, 2007.

[26] X. Jiang, W.-Y. Yau, and W. Ser, "Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge," *Pattern recognition*, vol. 34, no. 5, pp. 999–1013, 2001.

[27] L. Jiang, T. Zhao, C. Bai, A. Yong, and M. Wu, "A direct fingerprint minutiae extraction approach based on convolutional neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 571–578, IEEE, 2016.

[28] L. N. Darlow and B. Rosman, "Fingerprint minutiae extraction using deep learning," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 22–30, IEEE, 2017.

[29] Y. Tang, F. Gao, J. Feng, and Y. Liu, "Fingernet: An unified deep network for fingerprint minutiae extraction," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 108–116, IEEE, 2017.

[30] D.-L. Nguyen, K. Cao, and A. K. Jain, "Robust minutiae extractor: Integrating deep networks and fingerprint domain knowledge," in *2018 International Conference on Biometrics (ICB)*, pp. 9–16, IEEE, 2018.

[31] B. Zhou, C. Han, Y. Liu, T. Guo, and J. Qin, "Fast minutiae extractor using neural network," *Pattern Recognition*, vol. 103, p. 107273, 2020.

[32] C. I. Watson and C. L. Wilson, "Nist special database 4," *Fingerprint Database, National Institute of Standards and Technology*, vol. 17, no. 77, p. 5, 1992.

[33] M. Galar, J. Derrac, D. Peralta, I. Triguero, D. Paternain, C. Lopez-Molina, S. García, J. M. Benítez, M. Pagola, E. Barrenechea, *et al.*, "A survey of fingerprint classification part ii: experimental analysis and ensemble proposal," *Knowledge-Based Systems*, vol. 81, pp. 98–116, 2015.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.