UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Pavlo Tertychnyi

Low-quality Fingerprint Classification

Master's Thesis (30 ECTS)

Supervisor: Assoc. Prof. Gholamreza Anbarjafari

Tartu 2018

Low-quality Fingerprint Classification

Abstract:

Fingerprint recognition systems mainly use minutiae points information. As shown in many previous research works, fingerprint images do not always have good quality to be used by automatic fingerprint recognition systems. To tackle this challenge, in this thesis, we are focusing on very low-quality fingerprint images, which contain several well-known distortions such as dryness, wetness, physical damage, presence of dots, and blurriness. We develop an efficient, with high accuracy, deep neural network algorithm, which recognizes such low-quality fingerprints. The experimental results have been conducted on real low-quality fingerprint database, and the achieved results show the high performance and robustness of the introduced deep network technique. The VGG16 based deep network achieves the highest performance of 93% for dry and the lowest of 84% for blurred fingerprint classes.

Keywords:
Fingerprints, Biometrics, Convolutional Neural Network

CERCS:P170, Computer science, numerical analysis, systems, control

Madala Kvaliteediga Sõrmejäljepiltide Klassifitseerimine

Lühikokkuvõte:

Traditsioonilised sõrmejälgede tuvastamise süsteemid kasutavad otsuste tegemisel minutiae punktide informatsiooni. Nagu selgub paljude varasemate tööde põhjal, ei ole sõrmejälgede pildid mitte alati piisava kvaliteediga, et neid saaks kasutada automaatsetes sõrmejäljetuvastuse süsteemides. Selle takistuse ületamiseks keskendub magistritöö väga madala kvaliteediga sõrmejälgede piltide tuvastusele – sellistel piltidel on mitmed üldteada moonutused, nagu kuivus, märgus, füüsiline vigastatus, punktide olemasolu ja hägusus. Töö eesmärk on välja töötada efektiivne ja kõrge täpsusega sügaval närvivõrgul põhinev algoritm, mis tunneb sõrmejälje ära selliselt madala kvaliteediga pildilt. Eksperimentaalsed katsed sügavõppepõhise meetodiga näitavad kõrget tulemuslikkust ja robustsust, olles rakendatud praktikast kogutud madala kvaliteediga sõrmejälgede andmebaasil. VGG16 baseeruv sügavõppe närvivõrk saavutas kõrgeima tulemuslikkuse kuivade (93%) ja madalaima tulemuslikkuse häguste (84%) piltide klassifitseerimisel.

Võtmesõnad:
Sõrmejäljked, biomeetrika, CNN

CERCS:P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Acknowledgments

I would first like to thank my thesis supervisor Assoc. Prof. Gholamreza Anbarjafari (Shahab), who inspired me to write this work and who helped me a lot with finishing it.

Also, I would like to express my gratitude to my mates from GEYCE Biometrics company for they took me in their awesome family and navigated me at all aspects of my internship.

Contents

Chapter 1

Introduction

Due to its non-invasiveness, high recognition accuracy, and the use of low-cost devices, fingerprints are one of the most reliable biometric characteristics in the context of human recognition and identification. Fingerprint authentication systems deeply settled in government, business and infrastructure institutions. However, most of the capturing systems depend on the condition of the finger's surface (i.e humidity, dust, temperature, etc.), which can affect the identification accuracy.

The fingerprint identification is a widely studied problem. Nevertheless, there is a huge problem, from which all of them are suffering - low quality of fingerprints, which makes the identification process harder, less reliable and sometimes even impossible. That is why present article proposes a way to identify the problem which causes the low quality of images and potentially can help in the elimination of it. Due to the wide commercialization of fingerprint identification systems, many of the best results in the state-of-the-art are provided by private companies.

Chapter 2

Related work

2.1 Fingerprints as biometrics

Fingerprints are one of the physiological characteristics of a human for verifying and identifying. Along with them, a human has other distinctive features: face, ear print, iris and retina, palm print, vein map, voice, signature. But the fingerprints have become the most popular and widely used because of their uniqueness, good results in recognition tasks, huge databases and they don't require cumbersome equipment. People use fingerprints as a way of identification in a business sector, medical field, education, protection of information, health and life and a lot of other.

The history of fingerprint usage started at the end of 19th century with the development of bureaus for storage, verification and identification of criminal records. And nowadays boarding control police departments and private companies have databases which contain millions samples of imprints.

2.1.1 Fingerprint recognition system

Human fingerprint is a black-white image where black lines called ridges and white called valleys. Union, intersection and other combinations of ridges create characterized features of fingerprints - minutiaes. Scientists identify following minutiaes (see Fig. 2.1):

- Ridge ending - a place where the ridge line finishes.
- Ridge bifurcation - a place where the ridge divides into two new ridges.
- Lake - a valley inside a small closed boundary of ridges.
- Independent ridge - a small ridge line (bigger than an island).

- Island - a small ridge isolated by valleys.
- Spur - a small branch from the ridge.
- Crossover - an intersection of two ridges.

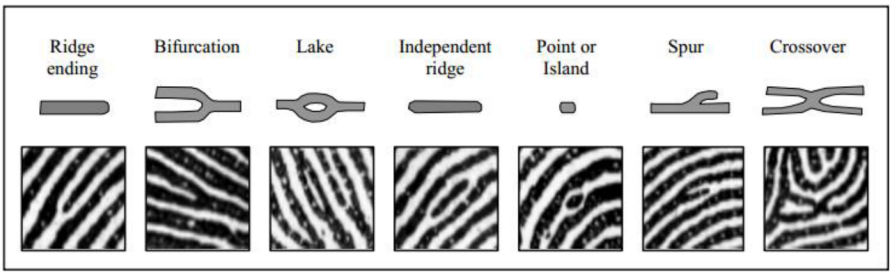


Figure 2.1: The most common minutiae types. Image is taken from [5].

All distinctive features of a fingerprint can be divided into 3 main category according to a level of detalization. See Fig. 2.2.

- Level 1 (Global features). Defines singular points and the main orientation of ridges: arch, tented arch, left loop, right loop, whorl.
- Level 2 (Local features). Defines minutiaes described above.
- Level 3 (Fine details). Defines concrete details of ridges: width, shape, contours and sweat pores.

Level 2 and level 3 are used for fingerprint matching since they represent individual unique fingerprints features. It should be clearly understandable that level 3 features can be used only with very high-resolution images because of their size.

The general schema for fingerprint recognition is as follows: fingerprint image capturing, pre-processing, feature extraction, matching. Detailed information on these stages will be present in the next sections.

2.1.2 Fingerprint image pre-processing and enhancement

Fingerprint matching can be a challenging task because of several factors: noise in the image, low scanners capacity, finger skin deformations and distortions, finger positioning on a scanner, etc. That is why the images require a lot of pre-processing to improve their quality. Usually pre-processing contains noise removing, contrast enhancement and morphological operations [75, 69]. Contrast enhancement's idea is to increase discernibility between ridges and valleys and by this improve image quality. It can be applied on the input image directly or on binary ridge image.

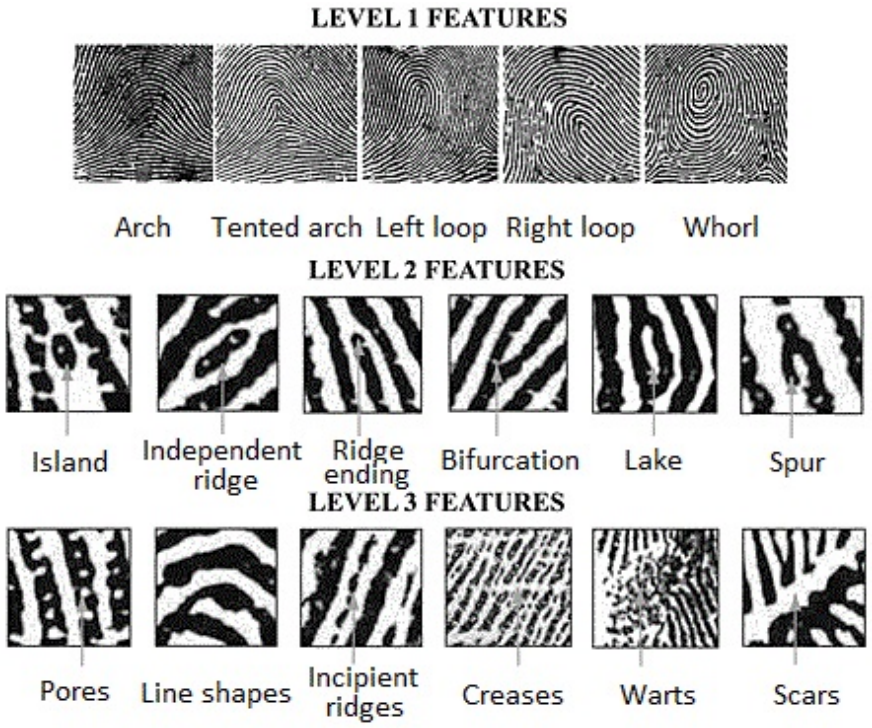


Figure 2.2: Hierarchy of feature levels. Image is taken from [22].

Histogram Equalization (HE) [89] is a global contrast enhancement method which is considered as the most commonly used because of its simplicity and effectiveness. In HE pixels values of an image are distributed uniformly which spreads the most intensities of pixels throughout all variety range. It makes the local contrast of ridges to become higher. As a part of noise influence reduction, Directional Median Filter (DMF) [80] is used to reduce noises along the ridge flow direction. Its modification called Directional Weighted Median Filter [21] was proposed three years after DMF. For a long time Gabor [33, 82] filters were used as a way to remove noise from fingerprint images which keep the structure of ridges and valleys preserved. Directional Wavelet Transform [79], which decomposes an image into blocks by applying Fourier transformation and enhances them separately, together with Gabor filters are often used as noise removal system. Short Term Fourier Transform [19] is a method which operates with non-stationary signals and analyses local ridge orientation and local ridge frequency to improve the quality of fingerprints.

As a part of image pre-processing scientists use morphological operations and most common morphological operations are present on Fig. 2.3:

- *Binarization.* Changing image from gray-scaled to black-and-white image by choosing a gray threshold. Every pixel which is above this threshold considered as white and every pixel which is below - as black. In this way image pixel intensity changes from [0..255] to [0, 1].

- *Filling of holes.* Sweat pores and noise can spoil ridges in such a way that small white dots appear inside ridges. It causes a problem in a stage of minutiae search. Each of these white dots will be counted as two bifurcations. It is important to remember that filling of holes is not used in case when 3rd Level features are used to make fingerprint match.

- *Thinning.* Thinning of an image makes ridges to be only one pixel wide. It is very useful and eases the process of finding minutiae points.

- *Dilation.* Sometimes it is useful to dilate ridges to make a fingerprint smoother.

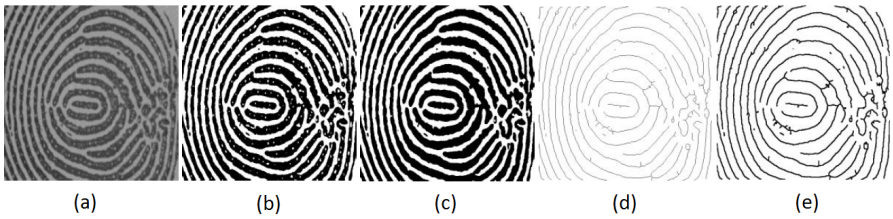


Figure 2.3: Example of morphological operations. (a) Original grayscaled image. (b) Binary image. (c) Binary image with filled holes. (d) Thinned image. (e) Dilated image. Image is taken from [69].

2.1.3 Fingerprint matching

Simply, a fingerprint matching algorithm returns a degree of similarity between two fingerprint images which is a number in a given interval (usually from 0 to 1). There are mainly two classes of fingerprint matching algorithms: minutiae based and non-minutiae based [32]. There are also hybrid methods which are a combination of them [45, 35, 84] and applied in a case when the quality of a fingerprint is not enough for matching. In turn, non-minutiae based class of algorithms can be divided into 4 categories: image based, ridge feature based, 3rd Level features based and feature-point based. But the most widely used are minutia based algorithms which are logically divided in local minutiae matching methods and global minutiae matching methods.

Non-minutia based approach

Image-based algorithms compare an input image and an image from a database to find a similarity between two of them. The weakest side of this way of matching is that it is very sensitive to alignment and non-linear deformations. Local Binary Patterns [59], Histogram of Oriented Gradient [58] and Gabor response [35, 4] are used for matching but they are also vulnerable to noise, skin conditions

and skin deformations. Ridge feature based techniques use ridge orientation and ridge frequency which describe topological information of ridge patterns to make fingerprint matching. From one side they solve a non-linear deformation problem of Image-based techniques but from another side, they have their own weakness - ridge information alone is not always enough for matching. People often use Level 3 features [32, 17] together with ridge features which add such ridge details as sweat pores and dots, ridge contours. But as it was mentioned before, to apply Level 3 features we must have images of very high resolution. Feature-point based methods usually used for object recognition and image matching but some scientists use this approach for fingerprint matching as well. For this purpose Scale Invariant Feature Transform [81] and accelerated KAZE [52] features are applied.

Minutiae-based approach

The first stage of each minutiae-based matching algorithm is a minutiae extraction. Minutiaes are presented by their spatial location coordinates and the angle of rotation. A minutiae of a given image is considered to be matched with a minutiae of an image from a database if the first falls within the tolerance box of the last. By tolerance box, we understand a permissible variation from both coordinates and direction of certain minutiae to compensate image distortions and limitations of minutiae extractors.

Since in real-life tasks the correct alignment of two matched fingerprints is left unknown, it is obvious that they will vary in some way because of pose variations, scaling and physiological aspects. That is why to reach the highest number of matched pairs of minutiaes it is crucial to make rotational alignment, scaling and bias.

As it was explained earlier, the minutia based techniques are classified as Local Minutiae Matching and Global Minutiae Matching.

- *Local Minutiae Matching.* These algorithms are taking into account local structures of minutiae. By local structures, we should understand different relationships in groups of the closest minutiaes. Such structures are invariant to global transformations of fingerprints which is undoubtedly the biggest advantage of using local matching. It also allows us to use only a part of information of a given fingerprint which is good for low-quality images and partial images which are usually not fully present in real-world tasks.

- *Global Minutiae Matching.* In opposite, these algorithms consider the set of minutiaes under the general scope. This is needed to make a proper alignment and since there are three parameters by which we should align (both coordinates and rotation) global matching may be computationally costly. Sometimes it is useful to apply so-called pre-alignment techniques

which are based mainly on singular points and orientation maps to reduce the computing costs.

Recent years minutiae-based matching algorithms tend to local matching techniques because of their invariance to distortion, ease and low computational power required. Even more, the common practice is to use a consolidation stage right after the local matching. The aim of the consolidation stage is to be sure that the local similarity of minutiae structures is supported on a global level.

There were numerous Global Minutiae Matching techniques developed for the last decades: Hough transform based [62, 48], ridge-based relative pre-alignment [34, 18], global matching of clusters of minutiae [88, 23], global minutiae matching with image correlation [77], global matching by evolutionary algorithms [73, 68, 67], Weighted global matching with adjustment of scores [37, 43], and a lot of others.

Mostly Local Minutiae Matching techniques vary in the way of computation of local similarity and in the way of final consolidation. Among local structures resemblance Nearest Neighbours [3, 78], Fixed Radius [8, 14], Texture [53, 85] and Minutiae Triplets [54, 86] approaches are the most common. And among consolidation types the most often used are Single [3, 11], Consensus [36, 15], Multiple [53, 54], Complex [28, 44], Incremental [78, 20].

2.1.4 Quality measurement systems

Fingerprint image quality assessment is one of the crucial aspects of biometric recognition systems. Quality measurement can play different roles in a scope of fingerprint recognition [26]: as a monitoring tool [41], as a control of enrolment templates (repeat the enrolment until the required fingerprint quality will not be reached), in verification and identification quality assessment, in quality-based adaptation recognition systems [24, 72] (some steps of recognition can adjust to a fingerprint quality and can perform differently depending on it). Numerous factors can affect the quality of fingerprints from global factors, such as temperature and humidity, to local factors, such as dirt, age of a participant and his/her collaboration. Typically, fingerprint quality is defined as a measurement of fingerprint's ability to extract true minutiaes and clarity of shapes of ridges and valleys.

There are three main classes of quality estimation algorithms: local features based, global features based and classifiers based.

Local features based methods

In local features based approach an input fingerprint image is divided into small rectangular blocks and each block is then classified as of low or high quality (often

it is useful to do not binary classification but multiclass classification, for example: "high", "medium", "low", "undefined"). Finally, the quality map is built to see the most "weak" regions of the fingerprint. In some approaches, each of these local blocks has a relative weight depending on how far they are to a fingerprint center - blocks closer to the center provide more reliable information.

Local Direction methods [46, 47] use local direction information to calculate several statistics of a given block, such as orientation certainty level, ridge frequency, ridge thickness, ridge-to-valley thickness ratio. The final decision of the block's quality is made upon a voting of these statistics. As a result, the final local quality map is associated to a fingerprint (see Fig. 2.4). Based on the assumption that ridge directions are changed smoothly across a fingerprint image, the average absolute difference of local orientation [13] is used to assess the image quality. To estimate clarity of local ridge-valley direction local coherence of intensity gradients are used [16].

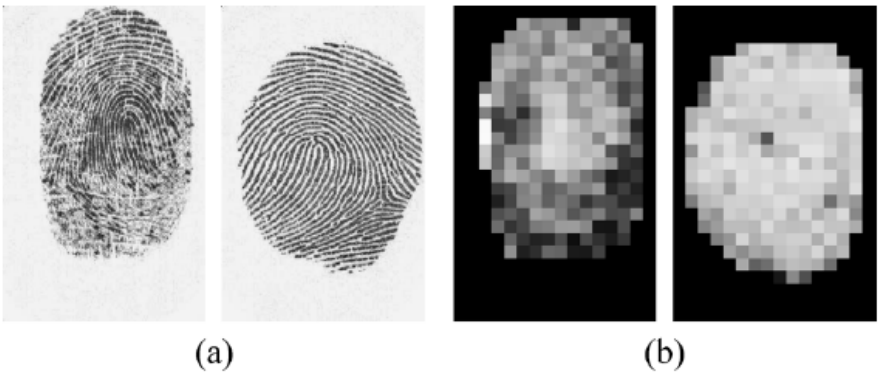


Figure 2.4: Example of a local quality map. (a) Original images of different quality. (b) Local quality maps of the correspondent images. Image is taken from [2].

Gabor filters are used as a way of detection of strong ridge direction [66]. Typically, several Gabor filters, each representing different ridge directions, are applied to local blocks. If there is noise, the response on each of these filters will be similar, otherwise, response on one filter will be more pronounced.

Pixel intensities based quality estimator [63] calculates histograms of pixels' intensities differences. If only one histogram has the maximum value exceeding the predefined threshold then the assessed block is considered to be directional, otherwise not. Pixels' intensities are also used to estimate the local contrast of ridges and valleys across the ridge flow direction inside a given local block [38].

Global features based methods

Global features based methods rely on features collected from the whole image, not as a cumulative decision taken from smaller blocks. The simplest idea is to

calculate global statistics such as ridge frequency and check its uniformity. The global structure of a fingerprint can be also estimated using 2D discrete Fourier transform.

Classifier-based methods

In classifier-based methods minutiae features are extracted from the fingerprint image and the quality of the overall image is estimated based on the quality of these features. In other words, quality is calculated as a degree of separation between two distributions for assessed fingerprint: match and non-match.

2.2 Fingerprints classification

To reduce the computational needs of fingerprint matching task scientists categorize fingerprints in advance. Thus fingerprint identification can be done using not the whole database of finger images but using a subset of it.

Among all features, only Level 1 features, the ones which describe the global direction of a ridge flow, are used for fingerprint classification. Features of Level 2 and 3 are too vary and too specific and used for fingerprint matching mostly. Therefore, fingerprints are classified into five major classes: Arch, Tented Arch, Left Loop, Right Loop, Whorl. The logical question arises - why do we need to classify fingerprints on these 5 classes? The answer is pretty simple, these classes are unevenly distributed. Arch and Tented Arch are the rarest classes and only 3.7% and 2.9% of the population have them. The rest of people have loops and whorls on their fingers with roughly equal distribution of 31.7%, 33.8% 27.9% for Right Loop, Left Loop and Whorl respectively. Thereby, in a fingerprint matching task, if we have an Arch classified fingerprint, for instance, we are no more required to compare this fingerprint with the whole database, we have to compare it only with samples of its class, which reduces the amount of work 30 times (from 100% of a database to 3.7%).

Level 1 features hold the information of the global ridge orientation (represented in an Orientation Map) and crucial points location - Singular points. By Singular Points we understand regions of a fingerprint with the highest variance, i.e. a place where ridges change their direction the most abruptly. Two types of such Singular Points can be distinguished: Core and Delta. Intuitively, cores are points where ridge flows flock into and deltas are points where ridge flows are diverging from. It is clearly seen on orientation map of a fingerprint ridge flow (see Fig. 2.5).

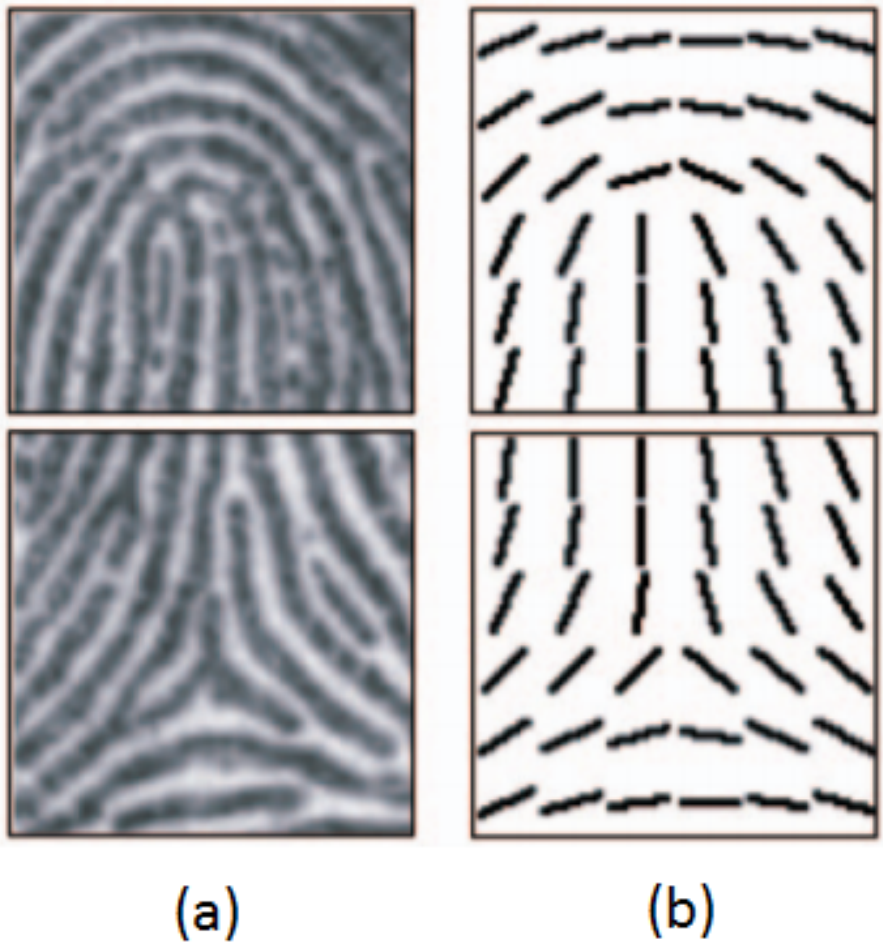


Figure 2.5: Example of a delta and a core singular points. (a) Original images. (b) Their representation in orientation map view. Image is taken from [87].

2.2.1 Types of fingerprints

As it was mentioned before, there are 5 major classes presenting Level 1 features (see Fig. 2.6).

- *Arch.* The only type of fingerprints which has no singular points. Ridges in Arch fingerprints flow from one side to another and form a small hump.

- *Tented Arch.* Has one core and one delta singular points (the delta located below the core). Ridge flow is similar to arch but more pronounced and ridges have more strong curvature.

- *Right Loop.* Has one core and one delta singular points (the delta is below and to the left of the core). At least one ridge starts on the left side, moves to a center, turn around and moves back to its start.

- *Left Loop.* Has one core and one delta singular points (the delta is below and to the right of the core). At least one ridge starts on the right side, moves to a center, turn around and moves back to its start.
- *Whorl.* Has two core and two delta singular points. One or more ridges make the complete turn around the center.

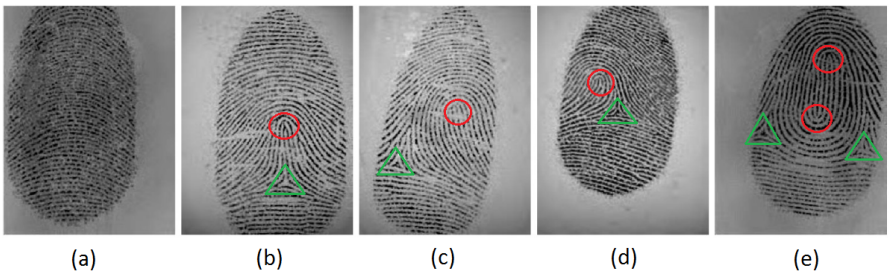


Figure 2.6: Fingerprint classes. (a) Arch. (b) Tented Arch. (c) Right Loop. (d) Left Loop. (e) Whorl. Red circles mark core singular points, green triangles mark delta singular points. Image is taken from [25].

2.2.2 Classification techniques

The earliest approaches in fingerprint classification techniques consist of finite automatons and grammars. Context-free grammars, stochastic grammars [57] and non-deterministic finite automaton [12] were used for classification. A large number of Neural Network architectures were proposed for this purpose. The very first ones were using Multilayer Perceptrons [30, 6] and Self Organizing Maps [27]. The more recent Neural Network approaches apply Convolutional Neural Network [60] which is the state-of-the-art technology in the question of image recognition. Graph matching approach is also a common way of fingerprint classification. It is based on a matching of graph-based patterns with a given image. Relational graph matching [50] and its extension solution [10] were used for the classification. Among structural techniques for the classification Decision trees [49] and Hidden Markov Models [64] can be mentioned. Nearest Neighbour algorithm is one of the most famous algorithms for fingerprint classification [9, 40] even in a scope of the last years. It is also common to use it in a combination with other techniques: Neural Networks [51] and SVM [65]. Stand-alone, SVMs are shown pretty good results [55, 56] in tackling this problem as well.

Chapter 3

Theoretical background

3.1 Random Forest

Random Forest (RF) is a supervised learning technique which shown a good performance for both classification and regression tasks. It was first proposed by Tin Kam Ho [29] in 1995, who used Random Subspace Method, and further improved by Leo Breiman [7], who combined aforementioned Random Forest with Bootstrap Aggregating approach.

The idea behind RF lies in the ensemble of decision trees which are not correlated with each other (see Fig. 3.1). As it stated in the name, "Forest" goes from multiple decision tree classifiers and "Random" goes from the way of making these trees different. To understand how the RF algorithm works we have to understand the three main aspects of it: Bootstrap Aggregating, Random Subspace Method and Decision Trees.

3.1.1 Bootstrap Aggregating

Bootstrap Aggregating or Bagging is a simple and very powerful ensemble method. The algorithm is as follows:

- Get a number of subsets of the training data (subsets with replacement).
- Train each of models in the ensemble on a separate subset of the training data.
- Calculate the average of predictions for regression task or mode for classification task.

Although Bagging is quite simple it has multiple advantages. First of all, it improves the stability and accuracy of prediction, which has to be done since the RF

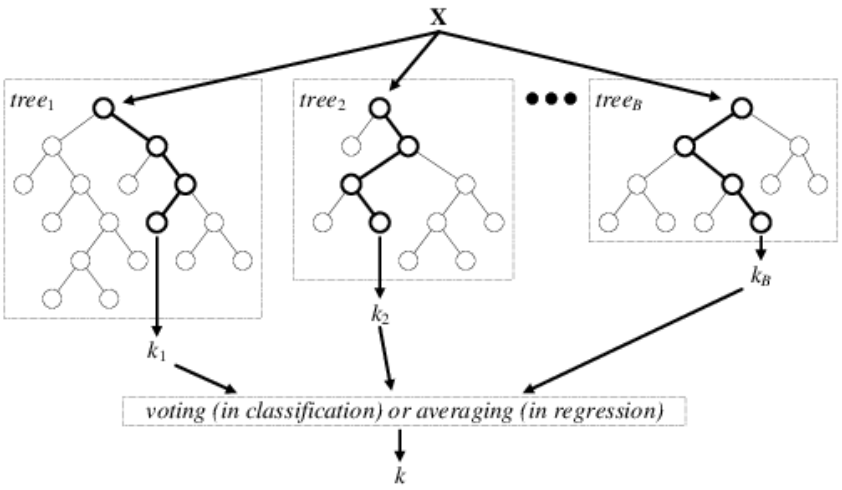


Figure 3.1: Schematic representation of Random Forest architecture. Image is taken from [76].

algorithm is random based. Second, Decision Trees are very sensitive to training data and therefore they overfit very fast if we don't do anything to prevent it, such as pruning, putting limits on the maximum tree depth and so on. Using Bagging we are less concerned about overfitting of each individual tree and trees can remain as deep and as wide as we want.

3.1.2 Random Subspace Method

Random Subspace Method, also called Feature Begging, is an ensemble method whose goal is to prevent correlation of ensemble estimators. Decision Trees decides on which feature to do a split based on a greedy algorithm which simply minimizes the error. Thus, if there are very important features, which have high distinctive abilities, they are for sure will be in the most of trees. It is a problem for ensemble techniques since estimators, in this case, have a very similar structure, which we certainly don't want. Feature Begging instead, forces each estimator to use only a subset of features to train on. It makes predictions of each tree less correlated and makes averaging of predictions more meaningful.

3.1.3 Decision Tree

Informally, Decision Tree is a representation of a rule-based classifier in the way that it is easy to grasp. In this tree, internal node represents feature and edges to children represent the value of this feature. Finally, leaves (nodes without children) correspond target classes which we want to classify.

Decision Trees are built from top to bottom by recursive partitioning of the training dataset into subsets that contain targets of only one class. Construction of a tree is about of how to split the data, or how to find a feature, so the branches are the most homogeneous. The most often ways to choose a feature to split are Information Gain and Gini Impurity.

$$I_G(p) = 1 - \sum_{i=1}^{N} p_i^2 - \text{formula for Gini Impurity}$$

$$H(T) = -\sum_{i=1}^{N} p_i \cdot log_{p_i} - \text{ formula for Entropy}$$

where $p_i,\ i \in [1..N]$ is a percentage of each class present
in the child node that results from a split in the tree

$$IG(T, A) = H(T) - \sum_{v \in values(A)} \frac{|T_v|}{|T|} H(T_v) - \text{formula for Information Gain}$$

They are very similar, as it was studied by Laura Elena Raileanu and Kilian Stoffel [61]. Summing-up their work, it only matters in 2% of cases whether you use Gini Impurity or Information Gain but Information Gain is a bit harder to calculate because of the logarithm, which can slow down the training process.

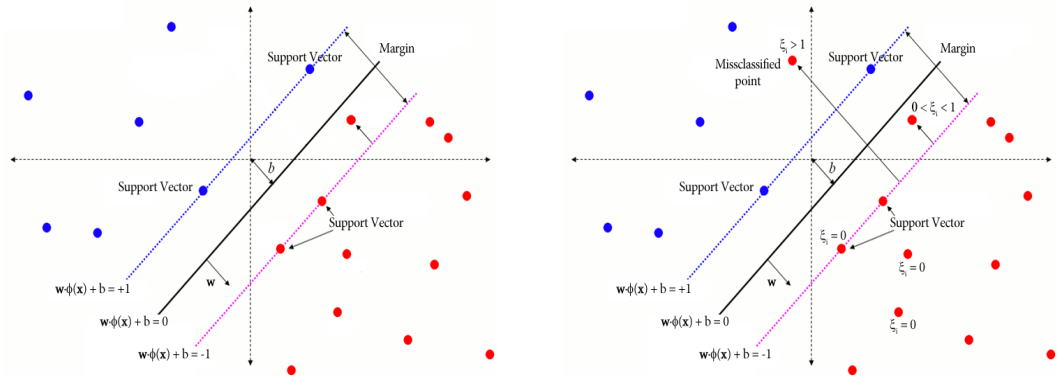
3.2 Support Vector Machines

Support Vector Machines (SVM) is a supervised method for classification and regression (mostly for classification). It was proposed by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. Then it was further improved by Vapnik in 1992 by adding a kernel trick and in 1995 by proposing of soft margin approach.

Intuitively, for given n-dimensional data points it tries to build an n-1 dimensional hyperplane which linearly separates the data. Among all (probably infinite number) hyperplanes we have to chose the best one, the one which maximizes the distance from closest data points to it. This approach called Hard Margin (see Fig. 3.2a).

Originally SVM algorithm solves binary classification tasks but it is easy to extend it to multiclass classification. In binary classification, there is only one separating hyperplane but in case of multiclass classification, we can set a hyperplane for each couple of classes. For example, in case of 3 classes, there would be 3 hyperplanes, in case of 4 classes - 6 hyperplanes and so on.

If the training data is not linear separably there are two common ways to solve this issue: Soft Margin and Kernel Trick.



(a) Hard Margin (b) Soft Margin

Figure 3.2: Schematic idea of SVM work.

3.2.1 Soft Margin

The idea behind the Soft Margin (see Fig. 3.2b) lies in relaxing conditions of the optimal hyperparameter searching. For now, it will be acceptable to a data point be in a wrong halfspace but the objective function will be penalized for this. The penalty is directly proportional to how far the outlier from the margin. In minimization task we add one more term with regularization ability. This parameter controls the trade-off between minimizing the training error and maximizing the size of margin between the halfspaces.

The main advantage of the Soft Margin SVM is that for it there is no such strict requirement for data to be completely separable. Therefore the training data can have errors as well as outliers and noise, which is more like a real-life problem. In its turn, Hard Margin will fail if it faces something of the above.

3.2.2 Kernel Trick

When it is impossible to build a hyperplane in that feature space in which the data is, we can simply change the feature space (often even to more high-dimension feature space). For example, if we couldn't build a line which separates two classes we can increase the space dimensionality and transform the data point into three-dimensional vectors, so the classes are separable by a plane (see Fig. 3.3).

The most common kernels are:

- $K(x, y) = x^T y + c$ - Linear kernel

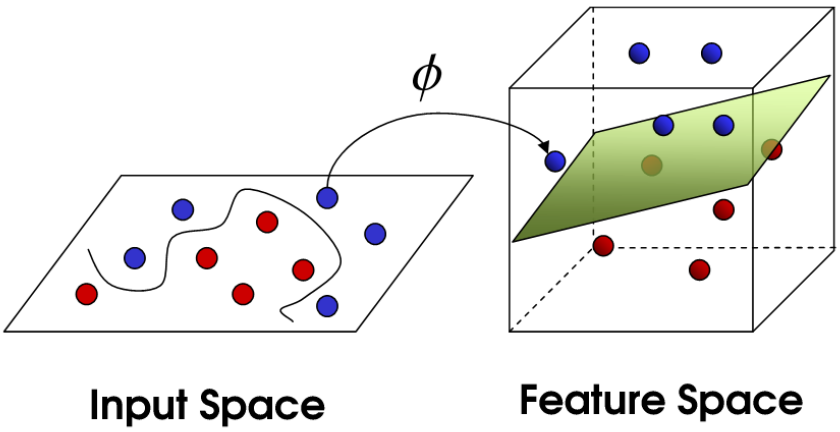


Figure 3.3: Applying of a kernel to get higher dimensionality.

- $K(x, y) = (ax^T y + c)^d$ - Polynomial kernel
- $K(x, y) = exp(-\frac{\|x-y\|^2}{2\sigma^2})$ - Radial basis functions kernel
- $K(x, y) = tanh(ax^T y + c)$ - Hyperbolic tangent (Sigmoid) kernel

3.3 Convolutional Neural Networks

Recent years almost all state-of-the-art algorithms in the field of image recognition are based on the Convolutional Neural Networks (CNN). The idea of CNNs was presented in the early 80s but there wasn't enough of computational resources to train an efficient network for image processing that times. Nowadays with a power of GPU computing, deeper theoretical investigations and more training data CNNs have become demanding.

CNN is a Feed-forward Neural Network where each neuron is responsible for a region, where regions can overlap with each other. A huge advantage of CNN is that it requires very little image pre-processing and it learns which feature to find in the image during the process of training while in other algorithms of image classification features are hand-engineered. CNN consists of the input layer, the output layer and multiple hidden layers, such as convolutional, pooling, fully-connected layers and normalization.

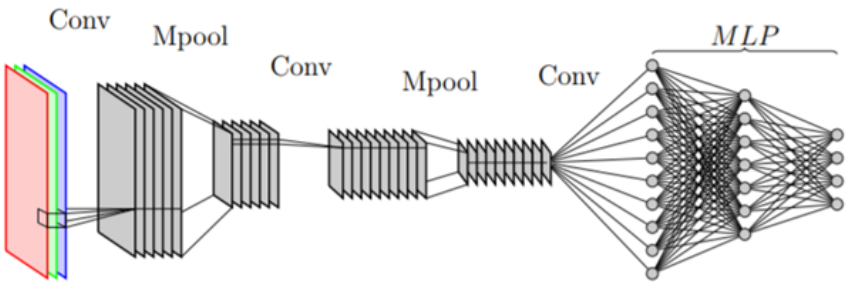


Figure 3.4: Schematic representation of a Convolutional Neural Network.

3.3.1 Architecture

Convolutional layer

Convolutional layer's parameters consist of a number of filters, simple matrices with numbers, and weights of these filters change during the process of training. In a forward pass stage, we convolve each filter through the input volume and calculate the dot product of filter matrix with the matrix of the input on every slide of the convolution operation. More detailed it can be seen on a Fig.3.5. After we've done this we receive a two-dimensional activation map which shows a response of each spatial location on a certain filter. An intuitive explanation of how convolutional layers work and why do we need them is that during a training stage filters are trained to activate if they detected some pattern. On the first layers, the patterns would be primitive, such as lines, color spots, etc. The further we go on the layers, the more complex patterns become and after simple objects, filters will learn to detect more complex details like circles, honeycombs, edges and so on. The last layers should activate when they see an entire object, a class which we want to recognize, whether it a car or a plane, a dog or a cat. See Fig.3.6. All activation maps received from each filter are then stacked into one three-dimensional output volume.

The convolutional layer is a building block of CNN and it does the most massive computational work. To be named a Convolutional Neural Network, the neural network must have at least one convolutional layer.

As we can see CNN learns to detect necessary features by itself and doesn't require any intervention in feature engineering which is undoubtedly its main advantage. We can draw an analogy between how our brain makes its image processing and how convolutional layers learn their filters. Each number in output volume (stacked activation maps of each filter) can be interpreted as activation of a neuron which monitors for some concrete small location in the input. Since regions, where we apply convolution, are overlapped (it is not necessary but in most cases yes) these neurons share parameters with their spatial neighbors. Starting from

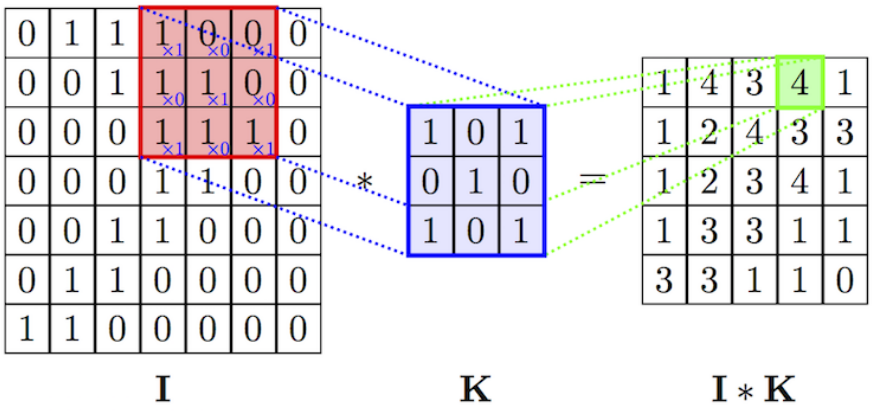


Figure 3.5: Example of the convolution operation.

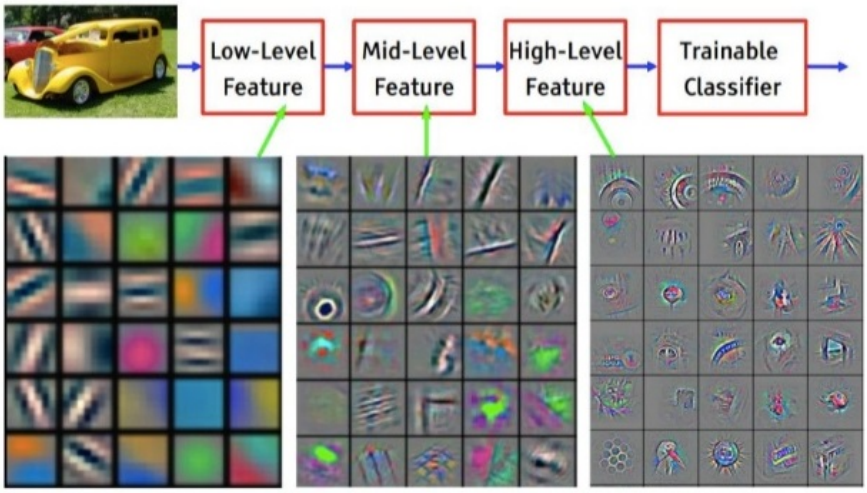


Figure 3.6: Feature visualization of convolutional net trained on ImageNet. Image is taken from [83].

here we will describe three main aspects of convolutional layers: local connectivity, spatial arrangement and parameter sharing.

Local connectivity. It is not logical, from the practical point of view, to connect a neuron to all neurons of input if we are working with high-dimensional data like images. What we are doing is connect a neuron to only a small number of neurons from the previous volume. This small number of neurons called the receptive field and it is a matrix of a filter size (obviously because during forward pass we have to make a dot product of these matrices). The connections are local along width and height but they are full along the depth of the input. This means that filters may have arbitrary length and width but must have the same depth as the input volume.

Spatial arrangement. Spatial arrangement is about how we construct an output

volume after convolution operation, meaning what would be the size of it. As it is already known, the depth of the output volume always equals to the number of filters. So neurons along the same depth dimension are monitoring the same spatial region but they are looking for different features. Width and height of the output volume are defined by a stride with which we will convolve. If stride equals to 1, we move filters by one pixel to the right and if there are some inconsistencies on borders of the input, we can do zero-padding - adding zeros on borders. Obviously, the bigger stride the smaller output is received (the most common stride is 3). Zero-padding is needed to control the size of the output volume and to reduce the number of problems dealing with how many neurons will "fit" into the input with a certain stride.

Parameter sharing. Parameter sharing trick is made up to decrease the number of parameters of the network. Parameter sharing principle is based on assumption that if one feature is learned to be detected in one place it may be useful to try to detect it in another place. For instance, if we learned to detect a wheel in some place on an image to recognize a car, it may be helpful to try to find another wheel on the image. That is why we put a constraint on neurons to use the same weights along one depth slice of an output volume. Because of it we call the weights a *filter* or sometimes a *kernel*.

This assumption is not always applicable, for example, the input images are images with faces which are centered. We need filters to learn to recognize eyes and noses to detect a face. But since each filter has one set of weights for the whole image we couldn't learn features which appear only in one location because we expect that those features may be found in different places.

Pooling layer

Adding pooling layers is one more way to reduce the number of parameters. Its responsibility is to reduce the size of the input volume. Pooling layers low down the computational cost of training and control the overfitting problem. Pooling operation is nothing else than a form of non-linear down-sampling. It applies to every depth slice separately as to two-dimensional matrix. Each input to a pooling layer is divided into non-overlapping regions and then one number represent each of these regions in the next stages. This number can be a maximum, average, L2-norm, etc. among all numbers in the region. Max-pooling is the most common because it lefts the strongest representation of an input. Usually, max-pooling layer divides an input into squares 2x2 thus after pooling the number of parameters is reduced by 75%. The depth dimension is left to be the same (see Fig. 3.7).

Some scientists are strictly against using of pooling. For instance, [70] suggest to abandon pooling layers and instead use only repeated convolutional layers with thelarger stride.

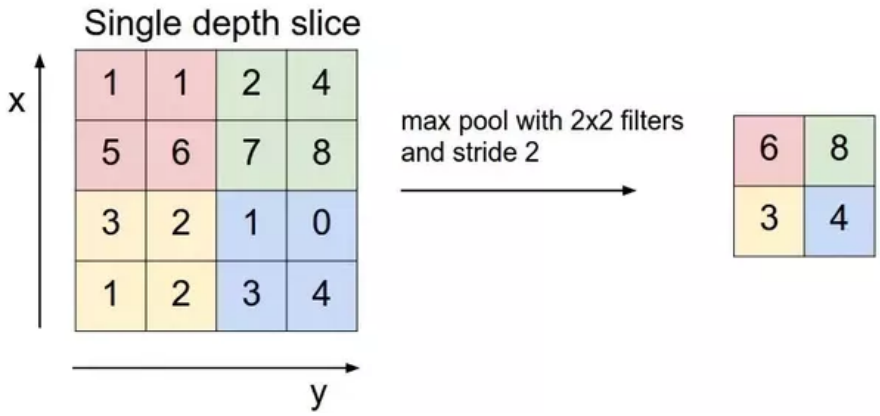


Figure 3.7: Example of max-pooling operation. Image is taken from [].

Fully-connected layer

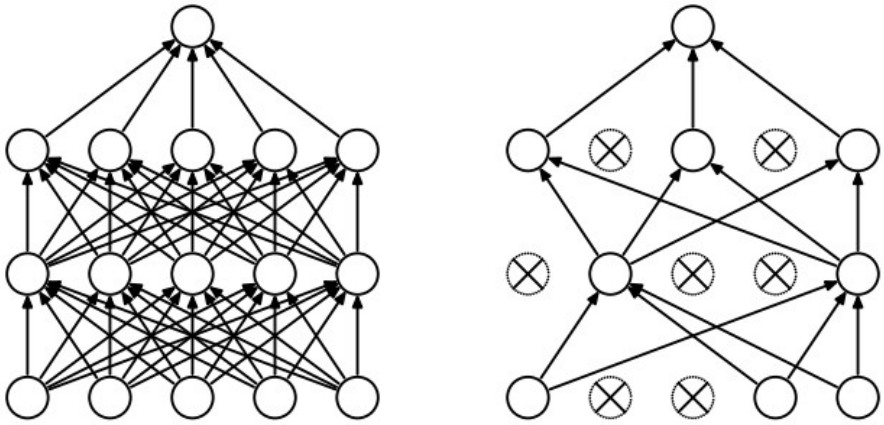
Neurons in fully-connected layers have full connections to all neurons from the previous layer as it in a regular neural network. Convolutional layers provide us high-level features and we stack a few fully-connected layers at the end of CNN to train non-linear combinations of these features. Anyway, adding fully-connected layers on top of convolutional ones is not the only solution. For example, in [74] SVM classifier was put on top of features gathered from CNN instead of a set of fully-connected layers.

Normalization layer

For all classification techniques, there is a common curse - overfitting. As it states from its name, overfitting is a situation when you are fitted to much to a training data and that is why, on a test set you receive much lower results, than it is expected. That is why scientists develop normalization techniques to put constraints on parameters in order to deal with this issue. The most commonly used nowadays are Dropout and Batch Normalization approaches:

- *Dropout.* Dropout is a regularization technique proposed by [71]. During the training, neurons are kept active with some probability p or set to zero with the probability $1 - p$. In AlexNet – the network which won ILSVRC 2012 dropout set to be 0.5 and for GoogLeNet it is set to be 0.7. The neurons which are set to zero are "dropped out", i.e. they do not contribute in a forward pass as well as in a backward pass (see Fig. 3.8). At every training epoch, (forward pass and corresponding backward) architecture of the network is different. It makes neurons to learn more strong features because they are not sure whether at the next epoch it can rely on other neurons or not. It is important to remember that during prediction we do not drop out neurons.

- *Batch normalization.* Batch normalization technique was introduced by [31] in 2015. Basically, the main idea behind the batch normalization is to normalize activations of hidden neurons by subtracting the batch mean and dividing by the batch standard deviation. By doing this we reduce internal covariance shift (the amount by which the hidden units' values shift around) and also it allows us to use higher learning rates because batch normalization takes care of that there is no activation which is very high or very low.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.8: Schematic representation of Dropout application. Image is taken from [71].

3.3.2 Activation functions

The neuron itself just calculates a weighted sum of other neurons output plus bias. Thus, output number can be infinitely big. That is why we use activation functions - to control the output value of a neuron and to control whether it will "fire" or not. That trick we took from how neurons in our brain work. But the main contribution of an activation function is that it adds non-linearity in the network. Without this feature, we could interpolate needed function by only linear functions.

ReLU

One of the most popular activation functions nowadays is the Rectifier linear Unit (ReLU). It computes the activation by formula $f(x) = max(0, x)$. Simply it puts a threshold on the output: if it's negative - put the output to zero, if it's positive - the output equals to the input. The benefit of ReLU is that it's simple, meaning that it has no complex computations of exponent and tangents, multiplications and divisions. Also is it was stated in [42] the Stochastic Gradient Descent converges

much quicker rather than with sigmoid or tanh functions. Nevertheless, ReLU has one strong disadvantage – neurons with ReLU activation are very sensitive to a large gradient flow. If there would be a large gradient flow from some units with ReLU activation, those units will be "silent" forever. But, with a proper choice of the learning rate hyperparameter, this issue is relatively rare. ReLU activation is used in hidden layers because it doesn't vanish gradients during a training process.

Softmax

The softmax activation function is used when we deal with multiclass classification and we need to have probabilities of an input to belong to a certain class. It is applicable only in the last layer of a neural network. The softmax activation squeezes the outputs of each unit to a range between 0 and 1 and it also divides each output so the total sum is equal to 1, like sum of all probabilities. The formula for the function is next:

$$\sigma(z)_j = \frac{e_j^z}{\sum_{i=1}^{N} e_i^z}$$

where z is a vector of which we want to transform into probabilities, e_j^z is a jth component of this vector.

3.3.3 Sample architecture

VGG16

VGG16 was introduced by Karen Simonyan and Andrew Zisserman who took the second place in the ILSVRC 2014. VGG stands from the name of this group – Visual Geometry Group from Oxford University. The main importance of their work lies in showing that depth of the network is a crucial aspect of the construction of a Convolutional Neural Network for image classification (having even pretty small convolution filters of size 3x3). VGG16 consists of 5 blocks of convolutional layers: the first two blocks are of 2 layers and the last three are of 3 layers. Each block ends with a max-pooling layer of size 2x2 which downsamples the input four times after the pooling operation. After the fifth convolutional block goes two fully-connected layers, each of 4096 neurons and finally one fully-connected layer of 1000 neurons – the number of classes present in ImageNet dataset. Neurons in each hidden layer are activated with ReLU and neurons in the last fully-connected layer are activated with Softmax function. The total number of parameters is almost 140 million.

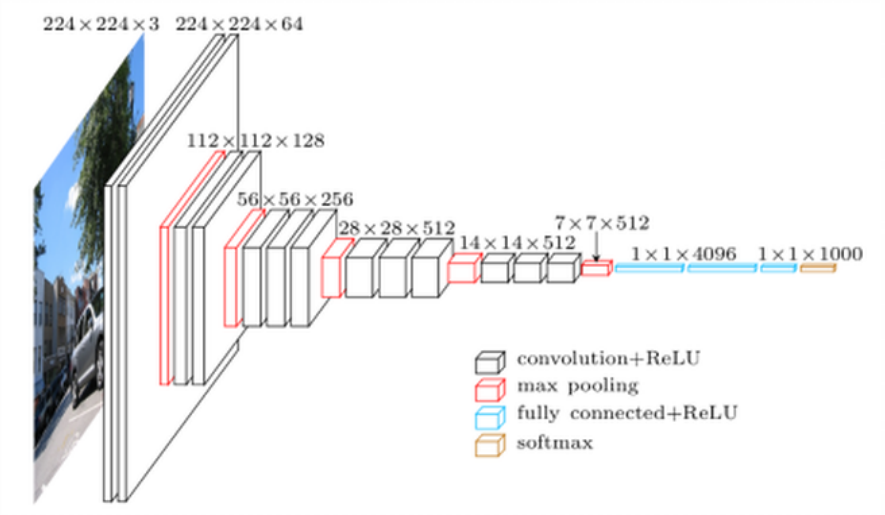


Figure 3.9: VGG16 architecture.

Filter visualization

We can see how CNN "sees" the input images by visualizing filters (see Fig. 3.10 to Fig. 3.14). To do it we can generate images which will maximize the activations of these filters.

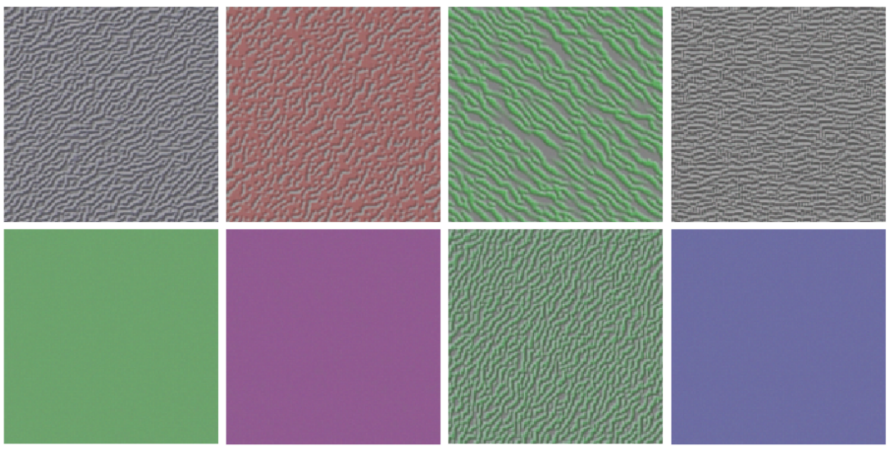


Figure 3.10: Filters of block1_conv1 layer.

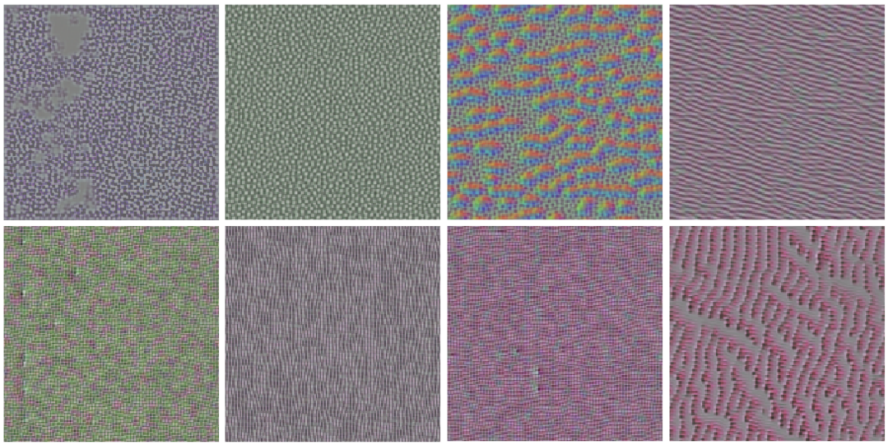


Figure 3.11: Filters of block2_conv1 layer.

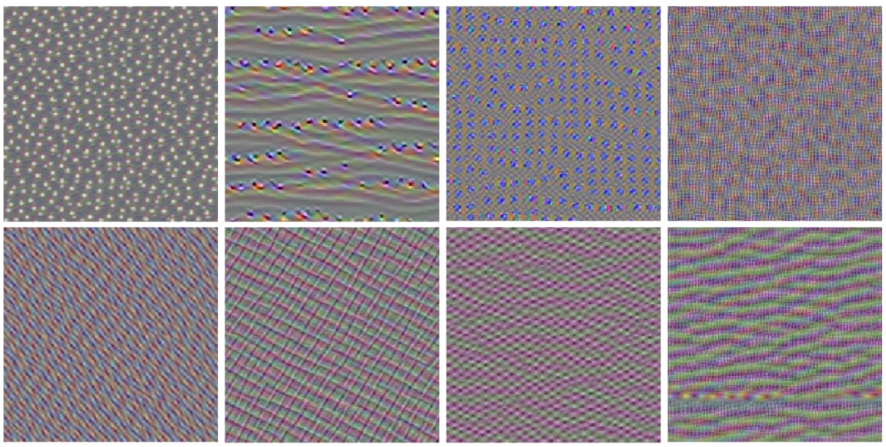


Figure 3.12: Filters of block3_conv1 layer.

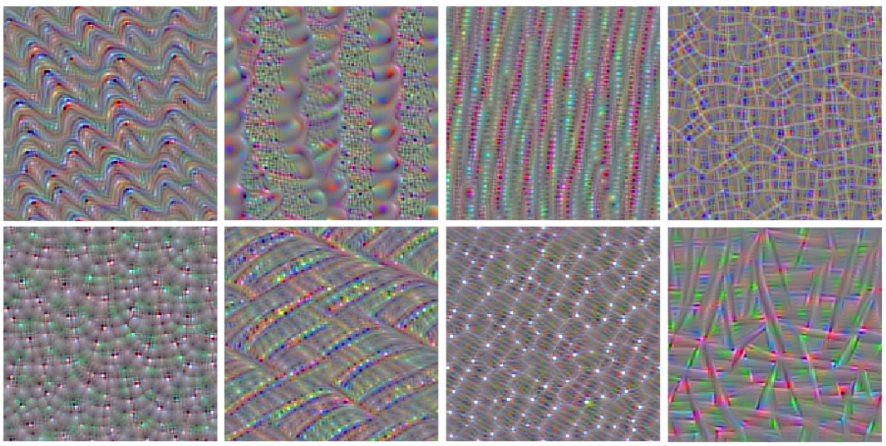


Figure 3.13: Filters of block4_conv1 layer.

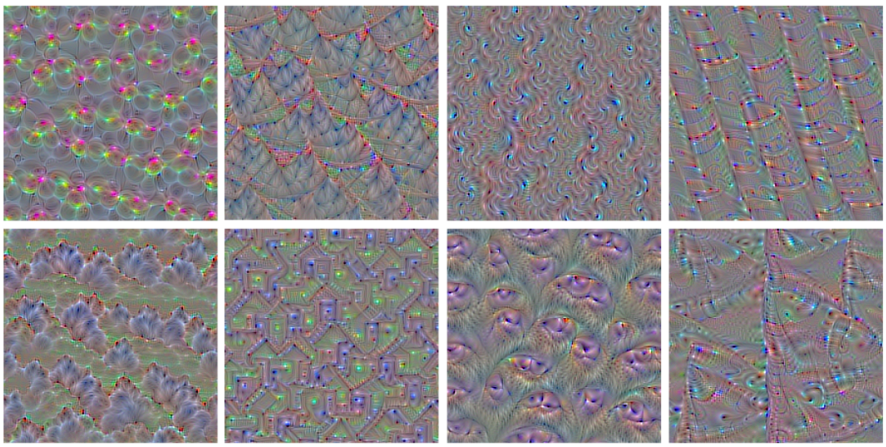


Figure 3.14: Filters of block5_conv1 layer.

As it was mentioned before, going further from layer to layer the filters are learned to detect more and more complex patterns. First and second layers show us the simple ones: colors, directions, grains. Third and fourth layers are learned to recognize more complicated structures and the final one, the fifth layer, represent details which are more looks like features - scales of fish, feathers of birds, etc.

3.4 Transfer learning for CNN

In practice, it is very hard to train a CNN from scratch for several reasons. First of all, it requires a huge database, for example, ImageNet database, the database for ILSVRC competitions, has 1.2 million of images. Also, it is computationally costly as well as time-consuming because it may take 2-3 weeks to train a neural network on ImageNet using multiple GPUs. Here the power of Transfer Learning (TL) comes. The main idea behind the TL concept is in sharing the knowledge gained by one model which solves one problem with another model which solves the similar task. In Deep Learning it is a quite common approach to use weights of the already trained network to benefit your own network. Mostly the TL is used in Natural Language Processing and Computer Vision tasks because features of these issues are logically transferable. For example, if we trained a network to recognize Spanish we can use its knowledge to recognize Portuguese because they are similar as languages, thus it would be practically wise to use features of the first model. It is not even always required for models to solve closely related problems. In terms of the Computer Vision issues, it can be a detection of cars and detection of buildings, for instance. As it was shown above features learned on early layers are general, such as lines, edges, colors, and so on. This knowledge can be useful to the CNN which aims to detect buildings since the first layers of this CNN also must recognize such simple patterns.

TL learning can be applied in several different ways:

Convolutional Networks as a fixed feature extractor. As it was mentioned previously, CNNs are often (but not always) consist of multiple convolutional layers as a core of computations and a few fully-connected layers at the end. The straightforward idea is to use features given by the last convolutional layer and to train a regular neural network using these features (or any other classifier).

Fine-tuning the CNN. The more thorough approach is to train not only the last fully-connected layers (or any other classifier) but also to fine-tune weights of all other layers of the network. The regular neural network which we stuck on top of the convolutional layers has randomly initialized weights, thus it is not practically useful to fine-tune all convolutional layers but still depends on a task which we are going to solve. Big gradients of these randomly initialized weights during the backpropagation stage can "harm" perfectly trained features which we don't want to change. Instead, we can "freeze" the early layers, meaning do not change their weights at all and just train a few last layers as well as the classifier at the end.

During the fine-tuning of the CNN there is one more thing to keep in mind. The Learning Rate should be smaller than it was during training the original ones. It comes from the expectation that CNN weights are already pretty tuned and we don't want to update them strongly and very fast.

The main factors which help us to define a strategy of applying TL in our specific case are the size of a dataset we have (small or big) and how much it is similar to the original one, the one on which the original network was trained. Here are a few common rules:

1. **The new dataset is similar to the original one and it is small.** The dataset is similar to the original so we can expect that the original network has similar not only general features but the more distinctive features too. So it is a good idea to use the fixed output of the last convolutional layer and build-up a new classifier on top of these features. Since the dataset is small it has no sense to fine-tune the last convolutional layers because the network can be easily overfitted. Intuitively, the overfitted CNN will be trained to detect not the key features but non-core ones. For example, we trained a CNN to distinguish man from women and the dataset consists of 5 men images and 5 women images. If all men in this small image set are dark-haired and all women are blondes then the network can understand that the color of hair is a distinctive feature, which is obviously not. Next, every dark-haired human will be classified as a man and every blonde human as a woman. With a dataset of several millions images this situation is nearly impossible.

2. **The new dataset is similar to the original one and it is big.** In this case, we can replace the classifier and fine-tune through the full network. Basically, we will train a completely new network but with weights initialized

with weights of another net.

3. **The new dataset is different from the original and it is small.** Since the dataset is different from the original one we must use only early features of the network from which we transfer the knowledge. The last features are too specific to the original dataset and thus using of them would be a mistake. The dataset is small, so the only thing we can do is to build a classifier on top of early features given by the original network.

4. **The new dataset is different from the original and it is big.** If we have a reasonably large set of images we can train the network from scratch. But it still will be a good idea not to initialize weights randomly but use weights of the pre-trained network as initial ones. It may accelerate the process of training.

There is one disadvantage of applying Transfer Learning. We have obvious and clear restrictions on architecture. Since we use pre-trained net we have limitations on the structure of a net. It is inconvenient to add or remove layers in the middle, only at the end.

Chapter 4

Problem definition

When dealing with massive enrollment, with a lot of people in a queue, rapidity is important. Especially when fingerprints are rejected during the verification it is valuable to know the reason, to make the subject do specific things before repeat capture and move to the next subject.

The task came from a real-life problem. The Ministry of Foreign Affairs of Spain was needed a software which will recognize the type of distortion of a given fingerprint. Having this information they can improve the fingerprint of an identified person physically. For example, the common trick is to improve a dry fingerprint by rubbing a finger on a forehead or a nose, since on these part of the body there are a lot of fat glands. The grease helps a fingerprint to increase the contact area which improves its quality. The opposite operation, removing of extra grease or sweat can improve a wet fingerprint by reducing the contact area.

The database was given by a consular post of one South America country. South America is a densely populated territory with almost half a billion of the human population. It consists of twelve sovereign states and a lot of islands. Every time someone wants to cross boundaries of one or another territory he or she has to prove his or her identity with a biometric identifier – fingerprints. Hereby, an enormous number of fingerprint scanning per day are done every day there. Institutions which faces with fingerprint authentication and identification suffering from low-quality fingerprints all the time. Considering that mostly South American countries are industrial and agricultural we understand that people there work with their hands. That is why their fingerprints are often distorted. Moreover, wet and warm climate harm the quality of skin even harder.

The only requirement they put was to make it fast, so this software could be imported into the SDK of a scanner and could work in real time. So the ideal scenario is as follows: the person comes to a consular post, put a finger on a scanner, scanner shows that its quality is low and why it is low, the police officer explains what to do with the finger and then the fingerprint gets a sufficient quality

for making an identification.

The proposed algorithm can be used not only by the government institutions but by private companies as well. For example, it can be useful for closed type manufactures where workers are obligated to verify their identity. In such factories, there are working thousands of people which makes a huge load on throughputs of verification system and since it is a factory there are probably exist some skin distortion issues.

Chapter 5

Proposed method

Convolutional neural networks shown their superiority in the world of image processing and object classification. Starting from 2012, Convolutional Neural Network started to beat all other Machine Learning techniques on ILSVRC. That is why it was decided to build a Deep Learning based classifier. To train the network from scratch we didn't have a huge amount of data so we used some workaround techniques to build-up powerful classifiers.

5.1 Model architecture and construction

Since the dataset was small we decided to apply a Transfer Learning approach and borrow some knowledge gained from another network trained to solve image classification tasks. As a basement for the classifiers, VGG16, network trained on ImageNet dataset, was taken. It has shown very good results on ILSVRC 2014 and what was the most important, it is completely available in the internet.

ImageNet dataset contains 1000 classes of different objects: cars, planes, animals, plants, and so on. It is obviously very different from fingerprint images so we chose to follow the third strategy of TL described above, that is to use only early features of the VGG16 net and train own classifier on top of these features.

The whole process of model training can be split into two parts: bottleneck feature extraction and model fine-tuning.

Bottleneck feature extraction stage.

In bottleneck feature extraction stage we passed all training samples through original VGG16 and saved outputs of 4th convolution block. After this, we got feature representation of the input images in terms of filters of 4th convolutional block. We didn't take outputs of the 5th convolutional block because they are too specific to complex objects recognition: cats, dogs, cars, etc. In our case we have

to distinguish objects with simpler shapes and structure in general and features learned in the 4th convolutional block would be perfectly enough for our case. Those features we used as inputs for a fully-connected three-layer network which were trained and which weights were saved for the next stage.

Fine-tuning stage.

In the fine-tuning stage, there were some modifications to the original VGG16 network. First of all, the last convolutional block, the 5th one, was cut off because it was trained too much to classify images from ImageNet dataset. Then we made trainable only the 4th convolutional block and frozen the first 3 convolutional blocks. On top of the resulting model, we put the fully-connected three-layer network with weights from the previous stage. Two first layers are of 256 neurons and the last one of 2 neurons – to have predictions of belonging to one or another class for the input image. After the second fully-connected layer we put a dropout layer with a probability of 0.5 to be dropped. First two layers finish with the ReLU activation function and the last one with the Softmax activation function. The learning rate is set to be 0.001 which is rather small. The learning rate is smaller than usual because weights of the top classifier are initialized randomly and weights of the convolutional layers are almost perfect and we don't want to distort them too much.

5.2 Data augmentation and preprocessing

Since we did not have enough of training images we had to apply data augmentation techniques. Data augmentation makes a random modification of an input image and provides a new image. In order to not screw up a training process, not all possible modifications can be used, for example, any pixel whitening or adding of noise are forbidden since they can change an image class. In our case we used the next random modifications (see Fig. 5.1):

- Width shift
- Height shift
- Rotation on +-10 degrees clockwise
- Horizontal flip
- Vertical flip

Data augmentation gave us 10000 of training samples per each fingerprint class instead of 2000 of the original. This extended training dataset was used in the fine-tuning stage of model training.

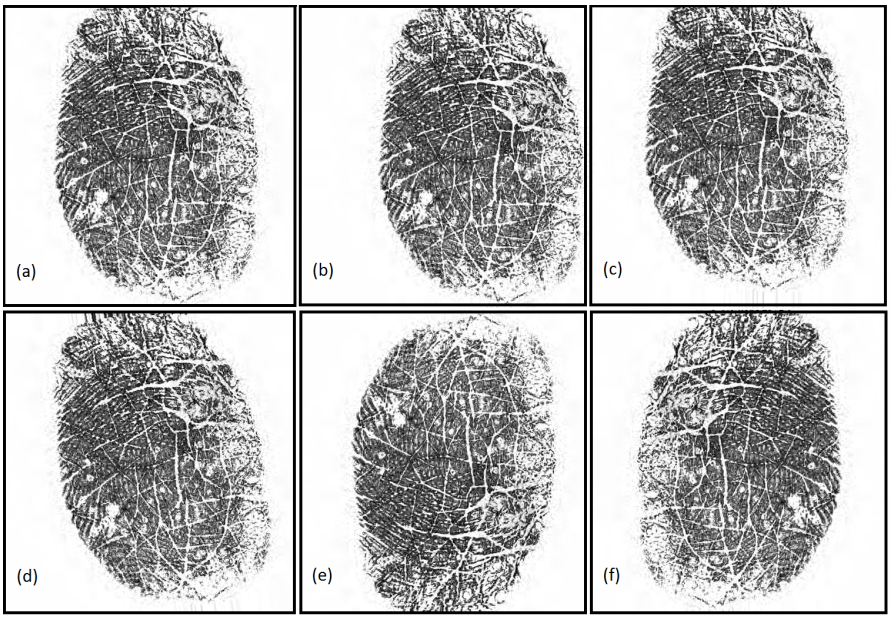


Figure 5.1: (a) Original low-quality fingerprint; (b) Width shifted low-quality fingerprint; (c) Height shifted low-quality fingerprint; (d) Rotated low-quality fingerprint; (e) Horizontally flipped low-quality fingerprint; (f) Vertically flipped low-quality fingerprint.

Also, as a part of data pre-processing, each image was resized from 512x512 to a size of 150x150 pixels and then rescaled to a range from 0 to 1. Resizing is needed in order to reduce computation cost of training and rescaling of input values is vital for training the CNN.

Chapter 6

Experimental results and discussions

6.1 Database

The database consists of 2000 of training images per each class, 1000 for images which belong to a class and 1000 which are not, the same for 200 of validation image set. There is also a test set of 100 images.

For our problem, we have 5 different classes: dry, wet, damaged, dotted and blurred type of fingerprint distortion. All images were labeled by experts from GEYCE Biometrics company [1]. Each fingerprint can belong to several classes, even for all 5, but it is a very rare situation, mostly it is 2-3 classes for a fingerprint.

Dry fingerprint (Fig. 6.1) – fingerprint which has a low contact area with a scanner surface, due to its dryness, and ridges don't fully contact with it. Potential fingerprint enhancement is to rub a fingerprint by a forehead or nose, the face areas with a huge amount of sweat and grease pores.

Wet fingerprint (Fig. 6.2) – fingerprint which has too high ability to contact with a scanner surface due to a huge amount of sweat of grease on it. In this case, not only ridges but also valleys contact the scanner surface. Potential fingerprint enhancement is to wipe out extra sweat and grease.

Damaged fingerprint (Fig. 6.3) – fingerprint which has scars, burning or inborn problem with a skin, when the skin is too thin and often chapped. There is no way to improve the fingerprint quality fast in this situation. If it is a problem of chapped skin, the fingerprint can be captured later, when wounds are healed. If the quality is low because of scars or burning, nothing can be done.

Dotted fingerprint (Fig. 6.4) – fingerprint which has clear distinguishable dots on a fingerprint image. It happens when the person is nervous or he or she has problems

with excessive sweating. These dots are situated around sweat pores of the skin and when they produce sweat during the process of scanning it is presented as black circular dots. The fingerprint is not obligated to be dry to observe this type of distortions there are examples of fingerprints which are dotted but not dry at the same time. Potential enhancement of a fingerprint is a to try to do so that identified person is not nervous.

Blurred fingerprint (Fig. 6.5) – fingerprint which has no clearly distinguishable ridges in some part of the image. This case is often when the finger wasn't fixed on a scanner surface and the image was captured while a fingerprint was slightly moved. Also, the identified person could press a finger too much so fingerprint valleys started to be seen on the image as well. Another cause of a fingerprint being blurred is when it burned, in this case, it is impossible to improve the quality instantly, it can become better with time or never. Potential enhancement is to ask an identified person to don't move and do not press too much on a scanner and retry the scanning.

Figure 6.1: Examples of dry fingerprints.

Figure 6.2: Examples of wet fingerprints.

Figure 6.3: Examples of damaged fingerprints.

Figure 6.4: Examples of dotted fingerprints.

Figure 6.5: Examples of burred fingerprints.

Some of the fingerprints have multiple issues which are making them more challenging. As it can be seen in Fig. 6.6, quality of fingerprint can be degraded by being dry, damaged and blurred or being wet and blurred. Currently, there are two problems with images:

- For some examples it is hard to classify them even for experts, for instance, a dry fingerprint can be easily misclassified as a dotted one and a blurred image can be misclassified as a wet one;
- The most of fingerprints are on the border of classes, meaning that if there is only one small piece of a fingerprint with a certain characteristic feature it is

(a) A dry, damaged and blurred fingerprint. (b) A dry and dotted fingerprint. (c) A wet and blurred fingerprint.

Figure 6.6: Examples of multiclass fingerprints

hard to say whether it should be assigned to this class or not. For example, a tiny area of a fingerprint is wet, should the whole fingerprint be classified as wet?

- Classifiers are trained to classify images only captured by Cogent CS500 scanner. Prediction of images taken by other scanners can be lower.

6.2 Results of CNN classifier

For every type of distortion, there was built one CNN with binary classification. Here are confusion matrices for dry, wet, damaged, dotted, and blurred fingerprint classes are shown in Table 6.1 to 6.5. The average time of prediction is 1.6 seconds. The overall recognition rate for all categories of low-quality classes is 89.4%. The VGG16 based deep network achieve the performance of as high as 93% for dry fingerprints. The blurred fingerprints are the most challenging low-quality sets where the correct recognition rate for that class is down to 84%. As it can be seen in the tables, the highest misclassification happens when the "not damaged" fingerprints are being classified as damaged one. This was expected as the scanned fingerprints contain discontinuity due to scanner limitations which are causing similarities to the damaged fingerprints.

Table 6.1: The confusion matrix for Dry fingerprint class, CNN classifier

		True		
		Dry	*Not Dry*	*All*
Predicted	*Dry*	61	2	63
	Not Dry	5	32	37
	All	66	34	100
Accuracy: 0.93				

Table 6.2: The confusion matrix for Wet fingerprint class, CNN classifier

		True		
		Wet	*Not Wet*	*All*
Predicted	*Wet*	31	5	36
	Not Wet	3	61	64
	All	34	66	100
Accuracy: 0.92				

Table 6.3: The confusion matrix for Damaged fingerprint class, CNN classifier

		True		
		Damaged	*Not Damaged*	*All*
Predicted	*Damaged*	42	0	42
	Not Damaged	13	45	58
	All	55	45	100
Accuracy: 0.87				

Table 6.4: The confusion matrix for Dotted fingerprint class, CNN classifier

		True		
		Dotted	*Not Dotted*	*All*
Predicted	*Dotted*	46	7	53
	Not Dotted	2	45	47
	All	48	52	100
Accuracy: 0.91				

Table 6.5: The confusion matrix for Blurred fingerprint class, CNN classifier

		True		
		Blurred	*Not Blurred*	*All*
Predicted	*Blurred*	41	9	50
	Not Blurred	7	43	50
	All	48	52	100
Accuracy: 0.84				

6.3 Results of other algorithms

To understand how good is the Convolutional Neural Networks based classifier we have to compare its performance with other common approaches of solving similar problems. As baselines two algorithms were chosen: Support Vector Machines and Random Forest classifiers. Both of them treat each pixel of the input image as a separate feature and in the same way as we pre-processed images for CNN we pre-process them in this case: resize to 150x150 pixels. Rescaling of pixels from 0 to 1 made classification results worst, so we decided not to do it, while it was crucial for CNN based classifier.

6.3.1 Results of Random Forest classifier

To make classification we trained Random Forest classifiers with the next parameters:

- The number of decision trees in the forest = 1000.
- The function to measure the quality of a split - Gini Impurity.
- The number of features to consider when looking for the best split = $\sqrt{n\_features} = 150$, where $n\_features = 150 \cdot 150 = 22500$ as a size of the image.
- Whether bootstrap samples are used when building trees = True.

As it can be seen, Random Forest can show more or less good results as for a simple classifier with average accuracy among all 5 classifiers of 83.5% (Tables 6.6 to 6.10). It happens due to the fact that it is an ensemble method of pretty simple standalone classifiers which are bad singly but can show impressive results working together. The prediction results are still worse comparably to the CNN

Table 6.6: The confusion matrix for Dry fingerprint class, RF classifier

		True		
		Dry	*Not Dry*	*All*
Predicted	*Dry*	53	1	54
	Not Dry	13	33	46
	All	66	34	100
Accuracy: 0.86				

Table 6.7: The confusion matrix for Wet fingerprint class, RF classifier

		True		
		Wet	*Not Wet*	*All*
Predicted	*Wet*	33	8	41
	Not Wet	1	58	59
	All	34	66	100
Accuracy: 0.91				

Table 6.8: The confusion matrix for Damaged fingerprint class, RF classifier

		True		
		Damaged	*Not Damaged*	*All*
Predicted	*Damaged*	41	8	49
	Not Damaged	14	37	51
	All	55	45	100
Accuracy: 0.78				

Table 6.9: The confusion matrix for Dotted fingerprint class, RF classifier

		True		
		Dotted	*Not Dotted*	*All*
Predicted	*Dotted*	41	14	55
	Not Dotted	7	38	45
	All	48	52	100
Accuracy: 0.79				

Table 6.10: The confusion matrix for Blurred fingerprint class, RF classifier

		True		
		Blurred	*Not Blurred*	*All*
Predicted	*Blurred*	41	11	52
	Not Blurred	7	41	48
	All	48	52	100
Accuracy: 0.82				

classifier especially for "hard" types of distortion: damaged, dotted and blurred. Wet an Dry fingerprints can be classified with high enough accuracy even taking into account only pixel intensity - for wet fingerprints pixels are closer to 0, to the black color, and dry fingerprints pixels are closer to 255, to the white color. But in complicate classes, the classification should be made based on more sophisticated features and here CNN based classifier shows its superiority.

6.3.2 Results of SVM classifier

To make the classification, we trained several sets of Support Vector Machines classifiers with the next parameters:

- Penalty parameter of the error term = 1.0.
- We played with four different kernels: 3rd-degree polynomial, linear, radial basis functions, sigmoid.
- Kernel coefficient $\frac{1}{n\_features} = \frac{1}{22500}$.
- Shrinking heuristic = True.

The SVM classifiers with RBF and sigmoid kernels make a constant classification, meaning they classify all fingerprints either as being belonged to a class or not belonged. It means that these kernels are not suitable to classify image data. That is why we don't present tables for them, which has no sense.

The conclusion here can be made the same - SVM is a good classifier even for such diverse data as images but it is still worse than CNN in questions of image classification. The average accuracy shown by SVM was 74% for polynomial kernel and 73.6% for the linear kernel, and the best combination of them has classification rate of 75.4% which is by far less than for CNN based classifiers (Tables 6.11 to 6.15b). As it was with Random Forest classifiers, the "simple" classes of fingerprints - wet and dry were classified with high enough accuracy due to the fact

Table 6.11: The confusion matrix for Dry fingerprint class, SVM classifier

(a) Polynomial kernel

		True		
		Dry	*Not Dry*	*All*
Predicted	*Dry*	53	6	59
	Not Dry	13	28	41
	All	66	34	100
Accuracy: 0.81				

(b) Linear kernel

		True		
		Dry	*Not Dry*	*All*
Predicted	*Dry*	54	5	59
	Not Dry	12	29	41
	All	66	34	100
Accuracy: 0.83				

Table 6.12: The confusion matrix for Wet fingerprint class, SVM classifier

(a) Polynomial kernel

		True		
		Wet	*Not Wet*	*All*
Predicted	*Wet*	29	7	36
	Not Wet	5	59	64
	All	34	66	100
Accuracy: 0.88				

(b) Linear kernel

		True		
		Wet	*Not Wet*	*All*
Predicted	*Wet*	32	8	40
	Not Wet	2	58	60
	All	34	66	100
Accuracy: 0.90				

Table 6.13: The confusion matrix for Damaged fingerprint class, SVM classifier

(a) Polynomial kernel

		True		
		Damaged	*Not Damaged*	*All*
Predicted	*Damaged*	29	17	46
	Not Damaged	26	28	54
	All	55	45	100
Accuracy: 0.57				

(b) Linear kernel

		True		
		Damaged	*Not Damaged*	*All*
Predicted	*Damaged*	20	17	37
	Not Damaged	35	28	63
	All	55	45	100
Accuracy: 0.48				

Table 6.14: The confusion matrix for Dotted fingerprint class, SVM classifier

(a) Polynomial kernel

		True		
		Dotted	*Not Dotted*	*All*
Predicted	*Dotted*	34	15	49
	Not Dotted	14	37	51
	All	48	52	100
Accuracy: 0.71				

(b) Linear kernel

		True		
		Dotted	*Not Dotted*	*All*
Predicted	*Dotted*	34	15	49
	Not Dotted	14	37	51
	All	48	52	100
Accuracy: 0.71				

Table 6.15: The confusion matrix for Blurred fingerprint class, SVM classifier

(a) Polynomial kernel

		True		
		Blurred	*Not Blurred*	*All*
Predicted	*Blurred*	35	14	49
	Not Blurred	13	38	51
	All	48	52	100
Accuracy: 0.73				

(b) Linear kernel

		True		
		Blurred	*Not Blurred*	*All*
Predicted	*Blurred*	35	11	46
	Not Blurred	13	41	54
	All	48	52	100
Accuracy: 0.76				

that such fingerprints can be easily separated by a hyperplane because of their pixels value (higher numbers for dry fingerprints and lower for wet fingerprints). But for damaged, dotted and blurred ones the situation is not so straightforward. The classification rate of SVM for them is much lower than of classification rate of CNN, especially for damaged fingerprints.

6.4 Devices and training computer characteristics

For training and testing a PC with following characteristics was used:

- *Processor.* Intel Core i7-6700HQ CPU, 2.60GHz, 4 Cores, 8 Logical Processors
- *System Type.* x64-based PC
- *Physical Memory (RAM).* 24.0 GB
- *Graphics Card.* NVIDIA GeForce GTX 960M.

Chapter 7

Conclusions

7.1 Inference

In the present work the low-quality images classifier based on Convolutional Neural Network presented. It is able to classify dry, wet, damaged, dotted and blurred classes with the average accuracy of 89.4%. Comparing to the other top rated Random Forests and Support Vector Machines classifiers CNN classifier is much more accurate (6.4% for RF and 14.5% for SVM in average). There are almost no results of similar works available in public due to the high commercialization of fingerprint-related research.

7.2 Future work

In the future, it is possible to extend the algorithm to classify more classes such as not centered, cut and small fingerprints. Also to do deeper research, it is possible to try different top classifiers trained on features received from convolutional layers of the network. It may reduce the computational cost of classification and thus decrease the training and classification performance time which is a crucial aspect of fingerprint authentication. It is also can be possible to use Local Binary CNN [39] which significantly reduces the number of parameters with approximately the same accuracy as the regular CNN.

Another obvious problem here is the lack of data. With more fingerprint examples the Neural Network can get sufficient boost in recognition abilities. So as a way of algorithm improvement we can collect more distorted fingerprint images.

References

[1] URL: http://bioidenti.com/en/home/.

[2] Fernando Alonso-Fernandez et al. "A comparative study of fingerprint image-quality estimation methods". In: *IEEE Transactions on Information Forensics and Security* 2.4 (2007), pp. 734–743.

[3] Abdallah Bengueddoudj et al. "Improving fingerprint minutiae matching using local and global structures". In: *Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on*. IEEE. 2013, pp. 279–282.

[4] Farid Benhammadi et al. "Fingerprint matching from minutiae texture maps". In: *Pattern Recognition* 40.1 (2007), pp. 189–197.

[5] Bibek Bhattarai and Naresh Kumar Giri. "FPGA Prototyping of the secured biometric based Identification system". In: (Nov. 2015).

[6] JD Bowen. "The home office automatic fingerprint pattern classification project". In: *Neural Networks for Image Processing Applications, IEE Colloquium on*. IET. 1992, pp. 1–1.

[7] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[8] Julien Bringer and Vincent Despiegel. "Binary feature vector fingerprint representation from minutiae vicinities". In: *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*. IEEE. 2010, pp. 1–6.

[9] Kai Cao et al. "Fingerprint classification by a hierarchical classifier". In: *Pattern Recognition* 46.12 (2013), pp. 3186–3197.

[10] Raffaele Cappelli et al. "Fingerprint classification by directional image partitioning". In: *IEEE Transactions on pattern analysis and machine intelligence* 21.5 (1999), pp. 402–421.

[11] JeongHee Cha et al. "Fingerprint matching based on linking information structure of minutiae". In: *International Conference on Computational Science and Its Applications*. Springer. 2004, pp. 41–48.

[12] Jeng-Horng Chang and Kuo-Chin Fan. "A new model for fingerprint classification by ridge distribution sequences". In: *Pattern Recognition* 35.6 (2002), pp. 1209–1223.

[13] Tai Pang Chen, Xudong Jiang, and Wei-Yun Yau. "Fingerprint image quality analysis". In: *Image Processing, 2004. ICIP'04. 2004 International Conference on*. Vol. 2. IEEE. 2004, pp. 1253–1256.

[14] Xinjian Chen, Jie Tian, and Xin Yang. "A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure". In: *IEEE Transactions on Image Processing* 15.3 (2006), pp. 767–776.

[15] Xinjian Chen et al. "An algorithm for distorted fingerprint matching based on local triangle feature set". In: *IEEE Transactions on information forensics and security* 1.2 (2006), pp. 169–177.

[16] Yi Chen, Sarat C Dass, and Anil K Jain. "Fingerprint quality indices for predicting authentication performance". In: *International Conference on Audio- and Video-Based Biometric Person Authentication*. Springer. 2005, pp. 160–170.

[17] Yi Chen and Anil K Jain. "Dots and incipients: extended features for partial fingerprint matching". In: *Biometrics Symposium, 2007*. IEEE. 2007, pp. 1–6.

[18] Jiangang Cheng, Jie Tian, and Hong Chen. "Fingerprint minutiae matching with orientation and ridge". In: *Biometric Authentication*. Springer, 2004, pp. 351–358.

[19] Sharat Chikkerur, Alexander N Cartwright, and Venu Govindaraju. "Fingerprint enhancement using STFT analysis". In: *Pattern recognition* 40.1 (2007), pp. 198–211.

[20] Sharat Chikkerur, Alexander N Cartwright, and Venu Govindaraju. "K-plet and coupled BFS: a graph based fingerprint representation and matching algorithm". In: *International Conference on Biometrics*. Springer. 2006, pp. 309–315.

[21] Yiqiu Dong and Shufang Xu. "A new directional weighted median filter for removal of random-valued impulse noise". In: *IEEE signal processing letters* 14.3 (2007), pp. 193–196.

[22] Mukti Dubey and Sandeep Sahu. "Fingerprint Minutiae Extraction and Orientation Detection using ROI (Region of interest) for fingerprint matching". In: *International Journal of Scientific & Engineering Research* 5.1 (2014), pp. 289–300.

[23] Kuo Chin Fan, Cheng Wen Liu, and Yuan Kai Wang. "A randomized approach with geometric constraints to fingerprint verification". In: *Pattern Recognition* 33.11 (2000), pp. 1793–1803.

[24] Julian Fierrez-Aguilar et al. "Discriminative multimodal biometric authentication based on quality measures". In: *Pattern recognition* 38.5 (2005), pp. 777–779.

[25] Mikel Galar et al. “A survey of fingerprint classification Part I: Taxonomies on feature extraction methods and learning models”. In: *Knowledge-based systems* 81 (2015), pp. 76–97.

[26] Patrick Grother and Elham Tabassi. “Performance of biometric quality measures”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.4 (2007), pp. 531–543.

[27] Ugur Halici and Güçlü Ongun. “Fingerprint classification through self-organizing feature maps modified to treat uncertainties”. In: *Proceedings of the IEEE* 84.10 (1996), pp. 1497–1512.

[28] Xiaoguang He et al. “Modeling and analysis of local comprehensive minutia relation for fingerprint matching”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.5 (2007), pp. 1204–1211.

[29] Tin Kam Ho. “Random decision forests”. In: *Document analysis and recognition, 1995., proceedings of the third international conference on*. Vol. 1. IEEE. 1995, pp. 278–282.

[30] PA Hughes and ADP Green. “The use of neural networks for fingerprint classification”. In: *Artificial Neural Networks, 1991., Second International Conference on*. IET. 1991, pp. 79–81.

[31] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[32] Anil K Jain, Yi Chen, and Meltem Demirkus. “Pores and ridges: High-resolution fingerprint matching using level 3 features”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.1 (2007), pp. 15–27.

[33] Anil K Jain and Farshid Farrokhnia. “Unsupervised texture segmentation using Gabor filters”. In: *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*. IEEE. 1990, pp. 14–19.

[34] Anil Jain, Lin Hong, and Ruud Bolle. “On-line fingerprint verification”. In: *IEEE transactions on pattern analysis and machine intelligence* 19.4 (1997), pp. 302–314.

[35] Anil Jain, Arun Ross, and Salil Prabhakar. “Fingerprint matching using minutiae and texture features”. In: *Image Processing, 2001. Proceedings. 2001 International Conference on*. Vol. 3. IEEE. 2001, pp. 282–285.

[36] Tsai-Yang Jea and Venu Govindaraju. “A minutia-based partial fingerprint recognition system”. In: *Pattern Recognition* 38.10 (2005), pp. 1672–1684.

[37] Jia Jia et al. “Fingerprint matching based on weighting method and the SVM”. In: *Neurocomputing* 70.4-6 (2007), pp. 849–858.

[38] Sungwook Joun et al. “An experimental study on measuring image quality of infant fingerprints”. In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2003, pp. 1261–1269.

[39] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. "Local binary convolutional neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. Vol. 1. 2017.

[40] Hye-Wuk Jung and Jee-Hyong Lee. "Noisy and incomplete fingerprint classification using local ridge distribution models". In: *Pattern recognition* 48.2 (2015), pp. 473–484.

[41] Teddy Ko and Rama Krishnan. "Monitoring and reporting of fingerprint image quality and match accuracy for a large user application". In: *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*. IEEE. 2004, pp. 159–164.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[43] Rajesh Kumar and BR Deva Vikram. "Fingerprint matching using multidimensional ANN". In: *Engineering Applications of Artificial Intelligence* 23.2 (2010), pp. 222–228.

[44] Dongjin Kwon et al. "Fingerprint matching method using minutiae clustering and warping". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 4. IEEE. 2006, pp. 525–528.

[45] Wonjune Lee et al. "Partial fingerprint matching using minutiae and ridge shape features for small fingerprint scanners". In: *Expert Systems with Applications* 87 (2017), pp. 183–198.

[46] Eyung Lim, Xudong Jiang, and Weiyun Yau. "Fingerprint quality and validity analysis". In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 1. IEEE. 2002, pp. I–I.

[47] Eyung Lim et al. "Fingerprint image quality analysis". In: *Image Processing, 2004. ICIP'04. 2004 International Conference on*. Vol. 2. IEEE. 2004, pp. 1241–1244.

[48] Chaoqiang Liu, Tao Xia, and Hui Li. "A hierarchical hough transform for fingerprint matching". In: *Biometric Authentication*. Springer, 2004, pp. 373–379.

[49] Manhua Liu. "Fingerprint classification based on Adaboost learning from singularity features". In: *Pattern Recognition* 43.3 (2010), pp. 1062–1070.

[50] Dario Maio and Davide Maltoni. "A structural approach to fingerprint classification". In: *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. Vol. 3. IEEE. 1996, pp. 578–585.

[51] Gian Luca Marcialis, Fabio Roli, and Paolo Frasconi. "Fingerprint classification by combination of flat and structural approaches". In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. Springer. 2001, pp. 241–246.

[52] Surbhi Mathur et al. “Methodology for partial fingerprint enrollment and authentication on mobile devices”. In: *Biometrics (ICB), 2016 International Conference on*. IEEE. 2016, pp. 1–8.

[53] Miguel Angel Medina-Pérez, Andrés Gutiérrez-Rodrıguez, and Milton Garcıa-Borroto. “Improving fingerprint matching using an orientation-based minutia descriptor”. In: *Iberoamerican Congress on Pattern Recognition*. Springer. 2009, pp. 121–128.

[54] Miguel Angel Medina-Pérez et al. “Improving fingerprint verification using minutiae triplets”. In: *Sensors* 12.3 (2012), pp. 3418–3437.

[55] Jun-Ki Min, Jin-Hyuk Hong, and Sung-Bae Cho. “Effective fingerprint classification by localized models of support vector machines”. In: *International Conference on Biometrics*. Springer. 2006, pp. 287–293.

[56] Jun-Ki Min, Jin-Hyuk Hong, and Sung-Bae Cho. “Fingerprint classification based on subclass analysis using multiple templates of support vector machines”. In: *Intelligent Data Analysis* 14.3 (2010), pp. 369–384.

[57] Bijan Moayer and King Sun Fu. “A syntactic approach to fingerprint pattern recognition”. In: *Pattern Recognition* 7.1-2 (1975), pp. 1–23.

[58] Loris Nanni and Alessandra Lumini. “Descriptors for image-based fingerprint matchers”. In: *Expert Systems with Applications* 36.10 (2009), pp. 12414–12422.

[59] Loris Nanni and Alessandra Lumini. “Local binary patterns for a hybrid fingerprint matcher”. In: *Pattern recognition* 41.11 (2008), pp. 3461–3466.

[60] Daniel Peralta et al. “On the use of convolutional neural networks for robust classification of multiple fingerprint captures”. In: *International Journal of Intelligent Systems* 33.1 (2018), pp. 213–230.

[61] Laura Elena Raileanu and Kilian Stoffel. “Theoretical comparison between the gini index and information gain criteria”. In: *Annals of Mathematics and Artificial Intelligence* 41.1 (2004), pp. 77–93.

[62] Nalini K Ratha et al. “A real-time matching system for large fingerprint databases”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.8 (1996), pp. 799–813.

[63] Nalini Ratha and Ruud Bolle. *Automatic fingerprint recognition systems*. Springer Science & Business Media, 2003.

[64] Andrew Senior. “A hidden Markov model fingerprint classifier”. In: *Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*. Vol. 1. IEEE. 1997, pp. 306–310.

[65] Shesha Shah and P Shanti Sastry. “Fingerprint classification using a feedback-based line detector”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.1 (2004), pp. 85–94.

[66] LinLin Shen, Alex Kot, and WaiMun Koo. "Quality measures of fingerprint images". In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. Springer. 2001, pp. 266–271.

[67] Weiguo Sheng et al. "A memetic fingerprint matching algorithm". In: *IEEE Transactions on Information Forensics and Security* 2.3 (2007), pp. 402–412.

[68] Weiguo Sheng et al. "Consensus fingerprint matching with genetically optimised approach". In: *pattern recognition* 42.7 (2009), pp. 1399–1407.

[69] Paramvir Singh and Lakhwinder Kaur. "Fingerprint feature extraction using morphological operations". In: *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*. IEEE. 2015, pp. 764–767.

[70] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[71] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[72] Elham Tabassi, George W Quinn, and Patrick Grother. "When to fuse two biometrics". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Workshop on Biometrics*. 2006.

[73] Xuejun Tan and Bir Bhanu. "Fingerprint matching by genetic algorithms". In: *Pattern Recognition* 39.3 (2006), pp. 465–477.

[74] Yichuan Tang. "Deep learning using linear support vector machines". In: *arXiv preprint arXiv:1306.0239* (2013).

[75] Anand V Telore. "Study of distortion detection and enhancement methods for fingerprint images". In: *Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–5.

[76] Antanas Verikas et al. "Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness". In: *Sensors* 16.4 (2016), p. 592.

[77] Dingrui Wan and Jie Zhou. "Fingerprint recognition using model-based density map". In: *IEEE Transactions on Image Processing* 15.6 (2006), pp. 1690–1696.

[78] Craig I Watson et al. "User's guide to NIST fingerprint image software 2 (NFIS2)". In: *National Institute of Standards and Technology* (2004).

[79] Zin Mar Win and Myint Myint Sein. "Texture feature based fingerprint recognition for low quality images". In: *Micro-NanoMechatronics and Human Science (MHS), 2011 International Symposium on*. IEEE. 2011, pp. 333–338.

[80] Chaohong Wu, Zhixin Shi, and Venu Govindaraju. "Fingerprint image enhancement method using directional median filter". In: *Biometric Technology for Human Identification*. Vol. 5404. International Society for Optics and Photonics. 2004, pp. 66–76.

[81] Masao Yamazaki et al. "SIFT-based algorithm for fingerprint authentication on smartphone". In: *Information and Communication Technology for Embedded Systems (IC-ICTES), 2015 6th International Conference of*. IEEE. 2015, pp. 1–5.

[82] Jianwei Yang et al. "A modified Gabor filter design method for fingerprint image enhancement". In: *Pattern Recognition Letters* 24.12 (2003), pp. 1805–1817.

[83] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[84] Fandong Zhang, Shiyuan Xin, and Jufu Feng. "Combining global and minutia deep features for partial high-resolution fingerprint matching". In: *Pattern Recognition Letters* (2017).

[85] Yangyang Zhang et al. "Fingerprint recognition based on combined features". In: *International Conference on Biometrics*. Springer. 2007, pp. 281–289.

[86] DeQun Zhao, Fei Su, and An-ni Cai. "Fingerprint registration using minutia clusters and centroid structure 1". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 4. IEEE. 2006, pp. 413–416.

[87] Jie Zhou, Fanglin Chen, and Jinwei Gu. "A novel algorithm for detecting singular points from fingerprint images". In: *IEEE transactions on pattern analysis and machine intelligence* 31.7 (2009), pp. 1239–1250.

[88] En Zhu, Jianping Yin, and Guomin Zhang. "Fingerprint matching based on global alignment of multiple reference minutiae". In: *Pattern Recognition* 38.10 (2005), pp. 1685–1694.

[89] Youlian Zhu and Cheng Huang. "An adaptive histogram equalization algorithm on the image gray level mapping". In: *Physics Procedia* 25 (2012), pp. 601–608.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Pavlo Tertychnyi**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Low-quality Fingerprint Classification

   supervised Assoc. Prof. Gholamreza Anbarjafari

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 21.05.2018