Ashok Kumar Yadav, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 11, Issue 2, (Series-V) February 2021, pp. 23-29

RESEARCH ARTICLE OPEN ACCESS

Fingerprint Authentication Using CNN for Minutiae based Authentication

Ashok Kumar Yadav, DGM ECIL and Prof. T. Srinivasulu, K U Warangal

ABSTRACT
There has been an increase in security concerns regarding fingerprint based authentication. This problem arises due to technological advancements in spoofing and hacking technologies. This has motivated the need for a more secure platform for identification. In this paper, we have used a deep Convolutional Neural Network as a pre-verification filter to filter out fake or spoof fingerprints. As deep learning allows the system to be more accurate at detecting and reducing false identification by training itself again and again with test samples, the proposed method improves the security and accuracy by multiple folds. The implementation of a novel secure fingerprint authentication platform that takes the optical image of a fingerprint as input. The given input is pre-verified using Google's pre-trained inception model for deep learning applications, and then passed through a minutia-based algorithm for user authentication. Then, the results are compared with existing models.

Keywords: Biometrics, deep learning, convolutional neural network, inception model, minutiae, fingerprint.

Date of Submission: 13-02-2021 Date of Acceptance: 27-02-2021

I. INTRODUCTION

We, human beings, can be recognised by our unique characteristics and traits. These traits can be physical ,behavioural or physiological in nature. At present, when we consider identification, we immediately associate it with biometrics. Identification, in general, is done by observing and identifying the unique characteristics of an individual. These characteristics can be features such as facial expressions, iris patterns, voice, signature , fingerprint vein and even fingerprints. We have formulated many methodologies for identifying individuals based on their unique features, but most of them are left as theories due to the lack of computational power and speed. However, in this day and age, we can convert these theories into practical uses, and can make them available for day-to-day use [10] .

Previously, identification and validation of individuals were done by information-based systems, which make use of passwords or cards for authentication. As these methods are less reliable and less secure, the stored information can be easily hacked or lost. Therefore, there has been a need for more secure, complex, and unique identifiers for user authentication. Hence, biometric traits such as such as fingerprints, palm prints, face, iris, and ECG are found to be very useful for the recognition of individuals. Among the above-mentioned biometric traits, fingerprint-based authentication is the most popular one. Because of the above-mentioned risks, people have adopted fingerprint sensors for biometric authentication in many financial transaction platforms [10].

Fingerprints have different patterns that make each individual fingerprint unique. The patterns are made by a combination of several ridges and valleys. These ridges are moulded during our time in the womb, where several factors such as friction, maternal conditions, etc. affect their final shape and structure. These patterns develop all over the human body, including the palms, soles, and even toes. As so many conditions and factors play an important role in determining the final ridge and valleys pattern, we consider fingerprint patterns to be unique to each and every individual.

Fingerprint authentication has become a norm in our day-to-day society that its vulnerability is never challenged. However, due to technology advancements, malevolent attempts which bypass security systems using fake fingerprints have been increased. Many systems today use algorithms that match the records in the database with the input

provided by the scanner for user authentication. As these methods in specific applications are not modified and updated regularly at a pace that equals or exceeds the progress made by malevolent individuals, it leaves biometric recognition at an increased risk and makes them susceptible to cyber attacks.

Today, there are many fingerprint-based systems that can authenticate at high speeds because of advanced and better algorithms. We heavily rely on man-made algorithms to provide us with great results. In this research, we are focusing mainly on fingerprint biometrics and how we can further improve its security by using enhanced verification platforms.

II. RELATED WORKS

Fingerprints are identifiers that are unique to every individual. Fingerprints can be considered as a pattern of ridges and valleys (Figure 1). Thus, fingerprint classification and verification is a pattern recognition problem. This particular type of problem is considerably tricky to solve because of the large intra-class and small inter-class differences. Optical images of fingerprints can be classified based on the details of its ridge configuration. We can break them down into three different classes, as shown below.

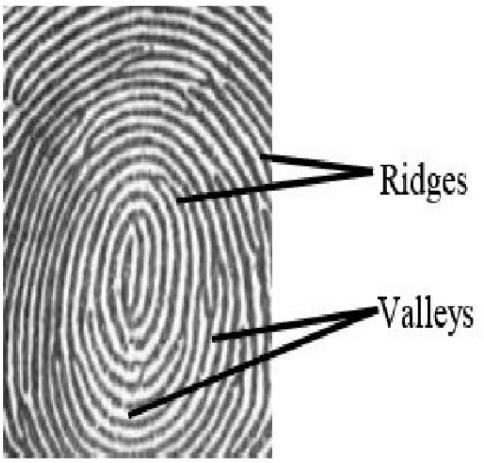

Figure 1 Ridges and Valleys in a Fingerprint.

The topmost or first-layer (also known as global) classification is based on the visual representation formed by the ridge pattern, which gives us three distinct regions named loop, whirl, and delta. In the second layer, we can drill down further to find more unique and detailed characteristics in these ridges, which are commonly referred to as minutiae. Finally, in the last layer, we can classify on the basis of all the features of the ridge pattern.

There are already many existing fingerprint biometric applications available in the market. These applications make use of feature-based learning algorithms. The features include global features such as the orientation of the optical input, ridge pattern, singular points, etc. Gabor filters [2] may be applied to highlight the selected features of the data input. These features can be extracted and classified in multiple ways depending on the methodology used. Many of these algorithms are verified for accuracy and performance against popular fingerprint databases such as NIST SD4, SD14, and SD9.

The global characteristics of fingerprints are also used for classification, in particular ridgeline flow, orientation image, singular points, and Gabor filter responses. Among them, the orientation image is the most widely used. Feature extraction can be performed in several ways, depending on the kind of feature, the quality of the image, and the accuracy that the following classification phase requires.

The goal of the classification stage is to learn a classifier based on labelled fingerprints. The algorithms proposed by various authors can be categorised as follows:

- Rule: These algorithms break down the provided fingerprint data input on the basis of the singularities, taking into account their number and positions. Here, the singularity points are found and the final classification is done based on the pattern formed by these singularities. A fine example is demonstrated in Ref. [8].
- Syntactic: These algorithms are based on a general grammar, like in the case of natural language processing applications. Here, the features extracted from the input data are stored as symbols in the database. The symbols are formed from ridgeline flows, which are then classified by a set of grammar. We can see an example of this in Ref. [13].
- Structural: These algorithms break down the fingerprint input and store it in specific data structures and formats like graphs and trees, to better understand and convert the features into a higher-level representation. An example of this can be seen in Ref.[9].
- Statistical: These algorithms are completely based on the statistical data computed. These data are generally calculated by using common statistical applications such as Bayesian decision rule, support vector machine (SVM), and K-nearest neighbour (KNN). We can see an example of statistical

approaches in SVM-based classification and improvement of fingerprint verification in Ref.[12].

- Neural network: These algorithms make use of multilayer perceptrons. The input data are taken and the extracted features are passed through the algorithm after reducing the dimensions[3]. The extracted features are generally singularities and the orientation of the image, which is used to train the perceptron to improve its performance. These networks give great classification results. A Convolutional Neural Network (CNN)-based approach for the segmentation of latent fingerprints is proposed in Ref. [19]. The CNN is trained using overlapping patches of multiple sizes to find out a fingerprint. A CNN-long short-term memory (LSTM) architecture is explained in Ref. [18] to find face anti-spoofing by combining the LSTM units with CNN. The architecture has two convolutional layers with 48 and 96 numbers of 3 × 3 filters, respectively, in each layer to learn the end-to-end features from video sequences.
- Multi-classifier: These algorithms include all the approaches that combine two or more classifiers [11].

This research is unique, as we make use of the statistical (minutiae-based) and neural network-based approaches and combine them to achieve better security and accuracy. The model proposed here is the first one of its kind, and it differs from the previous works in the following aspects:

- Our experiment proceeds through two phases – pre-verification phase and verification phase. In the pre-verification phase, good and fake fingerprints are filtered out. Then, the verification phase verifies a good fingerprint. Fingerprint images with poor quality are considered as fake fingerprints. By using the pre-verification phase, the security of the system can be increased.
- We have applied the technique of transfer learning in the pre-verification phase.
- Google's Inception-v3, which is a pre-trained deep CNN model, is used for training the pre-verification phase.
- Verification of the fingerprints is done by using Gabor filter and KNN-based methods.

III. PROPOSED METHOD

3.1 Pre-classifier CNN

3.1.1 Data Samples

The optical input that is captured from the fingerprint sensor is stored and accessed by the system in specific formats (e.g. .bmp, .jpeg, etc.). We are using these captured inputs to train our neural network. These inputs should be given extra care beforehand, as, generally, issues arise during the verification process (Figure 2).

For training to work well, we should gather at least a hundred visual samples of the fingerprints. The more we gather, the better the accuracy of the trained model. We also need to make sure that the inputs provided are good to help us in the verification and classification processes.

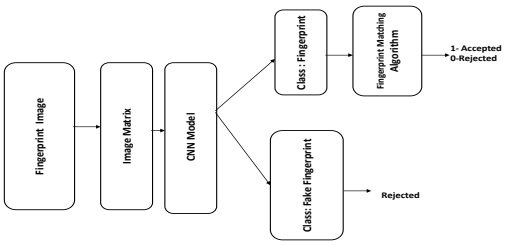

Figure 2 Proposed Block Diagram .

The input data that we have collected will not be completely used for the training process, as this can cause a chance for overfitting the neural network model. This common problem is easily resolved by dividing our data samples and using a part of them for the training process. Using a part of the data prevent the model from memorising the samples provided. We can use the remaining data inputs to check whether overfitting is occurring or not. The amount of overfitting is measured by the accuracy and precision of the trained model. We have the option to split the input data into different ratios. Generally, neural network-based applications split the input data by 1:1:8. The 80% of the split is used for the training process; the 10% of the samples is used to validate the input data during the training process; and the remaining 10% is used as the testing data input to verify the accuracy and performance of the now trained model[15].

The produced results can be further improved by applying various transformations to the input data (e.g. cropping, brightening, deforming, etc.). These transformations will help the classifier to adapt real-world scenarios to further improve the accuracy and performance.

We have used a regular optical fingerprint sensor to capture our fingerprint input. The .NET framework, which makes use of tensorflow [1], is used to develop the neural network model for the pre-classification process.

3.1.2 Inception Model

We are using a CNN for the classification of the input images (Figure 3). CNNs are neural

networks that comprise multiple layers and are generally used to classify optical data. CNNs take the optical input and process them in the form of tensors. Tensors are the representation of the data input in the form of a multi-dimensional matrix. The optical image is generally processed by applications as a two-dimensional (2D) input. Here, the model will convert the 2D input to a 4D matrix [16].

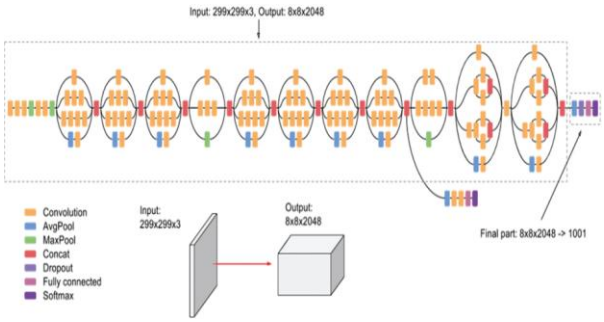

Figure 3: Basic Inception CNN Architecture.

The CNN model we are using is Google's Inception-v3 [4]. This model has already been pre-trained on all image data from ImageNet[5]. As this is a pre-trained model, the training time required for the classification process can be reduced by many folds. This model can be retrained using transfer learning, by which we can modify the model for our needs. The decrease in training time is possible because we are only training the last layer of the model to make it suitable for our application. ImageNet is a database that consists of multiple nodes, and each node comprises more than100,000 images organised by the WorldNet hierarchy.

We have used the technique of transfer learning in which the pre-trained Inception-v3 model is applied for filtering out good fingerprints. Inception-v3 uses a deep CNN architecture. In this architecture, the lower layers identify the low-level features and the higher levels identify high-level features. Thus, when we want to use this model for a different purpose, we need to train only the last layer of this model. The output layer should also be redefined. Except for the last layer, the parameters we use are the same, which the model has learned from ImageNet. The last layer of the model has 2048 filters of 1 × 1 size. We have changed this layer to a fully connected layer. This fully connected layer contains 2048 neurons. During training, the network learns the weights between the neurons and the bias of output neurons. Table 1 gives the parameters of the model.

Table 1: Parameters Used for the Proposed Model.

Parameters	Value
Number of neurons in the last layer	2048
Loss function	Cross entropy
Optimiser	Adam
Learning rate	0.001
Epochs	100
Batch size	32
Dropout	0.5
Optimiser	Adam

Google's Inception-v3 model consists of multiple convolutional kernels that vary in size. Each of these convolutional kernels identifies features for each dimension and scale. This model adopts the form of residual network that makes use of skip connections (Figure 4). Residual networks take the input data and add them to the output, thus enabling the network to predict the residual input rather than the expected output [6].

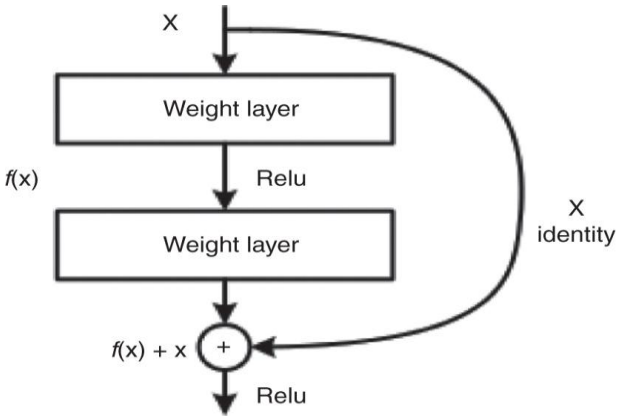

Figure 4: Skip Connection Typically Used in Residual Networks.

The inception model is trained to classify whether the provided input is a form of fingerprint or not. This model helps filter out bad inputs that have resembling features to that of a fingerprint input. Thus, this classifier is able to identify inputs with a certain level of noise. An accuracy of between 90% and 95% is obtained, although the exact value varies from run to run. This is due to the randomness in the training process.

3.2 Minutiae Verification for Authentication

3.2.1 Feature Requirements

The output of the inception model is a good fingerprint that is a distinct pattern of ridges and valleys. A ridge is defined to be a single curved

segment, whereas a valley is the area between two adjacent ridges. Minutiae points are the major features of the fingerprint and are used to match the fingerprints. These minutiae points are used to determine the uniqueness of the fingerprint. A good quality fingerprint can have 25–80 minutiae depending on the sensor and scanner used[14].

Minutiae can be defined as the points in the fingerprint where the ridges end or divide. There are several types of divisions or ends in a ridge line, which can be segregated as below:

- Ridge dots - these are extremely tiny ridges that are too short to be considered as a ridge line.
- Ridge endings – these are the end points of a ridge line.
- Ridge ponds/lakes – these are the cavities formed in between diverging ridges.
- Ridge islands – these are longer ridge dots that are found inside ridge ponds.
- Ridge bifurcations – these are points where ridges begin to diverge.
- Ridge spurs – these are the inclinations found on a ridge line.
- Ridge bridges – these are found on the point where two tiny ridges adjacent to each other join.
- Ridge crossovers – these are found at the point of overlap between two ridge lines.

Our minutiae verification algorithm takes the filtered fingerprint from the classifier as input and finds the minutiae from the fingerprint. We have extracted the ridge endings and bifurcations from the fingerprints. Then, these points are used for matching with the database.

3.2.2 Feature Extraction and Minutiae Matching

The first step for this verification approach is normalisation, which results in a better contrast of the fingerprint input. Then, this input is transformed into a grey scale representation. We will then apply several correctional adjustments to the input, such as changing the image brightness and contrast. Then, the noise distortion in the input is filtered out using a Gabor filter [2].

The Gabor filter is a linear filter that determines the output using a harmonic function, which is then multiplied using a Gaussian function. It consists of an orientation- and frequency-specific sinusoidal plane that is adjusted by a Gaussian envelope [17]. The Gabor filter provides a suitable visual representation of the input. Then, the feature map is created, which is used as the template. This template is matched in the subsequent matching step with templates of other fingerprints. The result of the matching is the matching score, which represents how good two fingerprints resemble each other.

Template matching is done by using the KNN algorithm[7]. The KNN algorithm makes use of the Euclidean distance, which can be calculated by using the following formula:

$$d(x, y)=\sqrt{(x1-y1)^2+(x2-y2)^2+\ldots+(x^n-y^n)} \quad (1)$$

The KNN algorithm assigns a class label to each minutia. Then, the *k* closest members are found by computing the Euclidean distance between the minutiae. In our experiment, the value assigned for *k* is 2. Thus, our algorithm finds two nearest minutiae for each minutia. Then, during matching, a local matching is performed to find which are the minutiae in the query fingerprint that match with the minutiae in the fingerprint templates in the database. Then, a global matching is done to find out how many minutiae of the query fingerprint match with the minutiae of the template fingerprint. Based on that, a matching score is generated, which give an indication on whether the fingerprint matches with any of the fingerprints in the database [14].

IV. RESULTS AND DISCUSSION

Currently, systems and sensors have a certain amount of false acceptance and false rejection, which is factored into during the verification process. We have used transfer learning rather than going for the traditional approach to reduce the training time.

Using tensorflow, we have achieved 94% accuracy for our model, which can be seen in Figure 5. The blue line shows the accuracy of the model on the testing set and the orange line is the accuracy of our model on the training set.

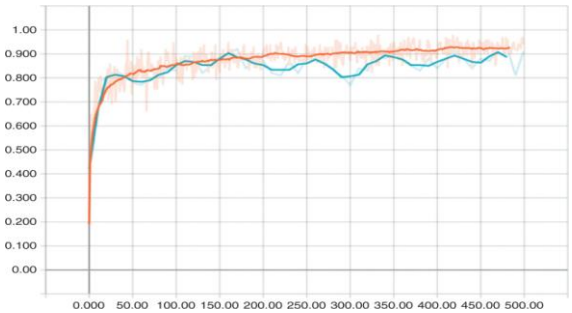

Figure 5: Training (Orange) vs. Test (Blue) Accuracy on Our Trained Model.

A fingerprint scanner that embeds a fake finger detection mechanism has to decide, for each transaction, if the current sample comes from a real

finger or from a fake one. The error in this decision should be as low as possible.

Let us assume the below:

FAR_{fd} and FAR_{iv}–the identity verification error rates.

FRR_{fd}– the proportion of real-finger transactions where the system incorrectly considered the input to come from a fake sample.

FRR_{iv} – the proportion of fake-finger transactions where the system incorrectly considered the input to come from a real finger.

For the fake-detection mechanism, the overall false rejection ratio (FRR) error can be estimated as follows (for an authorised user trying to be authenticated normally using the real enrolled finger):

$FRR=FRR_{fd}+(1-FRR_{fd})*FRR_{iv}$. (2)

The overall false acceptance ratio (FAR) error can be estimated as follows (for an attacker trying to be authenticated using a real finger, different from the enrolled one):

$FAR^{Real-Non-enrolled}=(1-FRR_{fd})*FAR_{iv}$. **(3)**.

Then, we can write

$FAR^{Fake-Non-enrolled}=FAR_{fd}*FARiv$. **(4)**

(for an attacker trying to be authenticated using a fake reproduction of a finger that is not the enrolled one) and

$FAR^{Fake-Enrolled}=FARfd*\gamma$. **(5)**

(for an attacker trying to be authenticated using a fake reproduction of the enrolled finger), where $\gamma<(1-FRRiv)\gamma<(1-FRRiv)$. If a fake fingerprint is created by using professional equipment, its quality is usually lower than the real finger it is designed to imitate, and therefore the chance that the identity verification algorithm does not match it with the user's real template is higher.

Generally, the FAR in fingerprint verification systems have a certain low value; let us assume it to be *x*.

We have found that the accuracy of fingerprint classification is around 94%. Thus, as our CNN is acting as a pre-filter, we can reduce the FAR of any system by using the multiplication rule of probability. Hence, (1−94/100)(1−94/100), which is the probability that the CNN falsely accepts a fake fingerprint is multiplied by *x*, which is the FAR of any system. Therefore, we get the following equation:

$FAR^{new}=FAR*(1-94/100)=0.06*FAR$. (6)

Let us take the experimental results from our KNN algorithm. We computed the FAR to be 0.026 and the false non-acceptance ratio (FNAR) to be 0. The resulting FAR = 0.026 can be improved as $FAR^{new}=0.026*0.06=0.00156FAR^{new}=0.026*0.06=0.00156$, thus improving the accuracy by approximately 90–95%.

Table 2 shows the comparison of the FNAR and FAR for existing methods of Fingerprint Recognition using Minutiae Score Matching (FRMSM) with the proposed method. It is observed that the FNAR for both methods is zero, and the FAR is 0.026 for the existing method and 0.00156 for the proposed method.

Table 2: Regular Minutiae Matching vs. Proposed Enhancement.

	FRMSM	**FRMSM(DL)**
FNAR	0	0
FAR	0.026	0.00156

Table3: Comparison of Various Pre-trained Deep Networks.

Model	**Accuracy range**	**Depth**	**Parameters**
NASNetMobile	0.744–0.919	–	5,326,716
VGG19	0.713–0.900	26	143,667,240
VGG16	0.713–0.901	23	138,357,544
ResNet50	0.749–0.921	168	25,636,712
Inception-v3	0.779–0.937	159	23,851,784

We have tested our model with batch sizes 4, 8, 16, and 32. The model gives good results when the batch size is 32. The network is trained for 20, 40, and 100 epochs. It is found that the model works well for 100 epochs. To avoid overfitting of the model, early stopping is also applied. Table 3 shows the comparison of various pre-trained deep network models.

V. CONCLUSION

We have done a novel experiment on fingerprint verification where the first phase is pre-

filtering of bad fingerprints and the second phase is fingerprint verification. The inception model is used for filtering out bad fingerprints. If the output of the inception model says that it is a good fingerprint, it is given to the verification module where matching of the fingerprint is performed. By combining a deep CNN pre-filter to the minutiae matching fingerprint verification algorithm, we have achieved an improved accuracy of approximately 90–95%. There are many upcoming technologies pertaining to fingerprint biometrics, such as in the case of smart phones. Qualcomm is coming up with sensors for ultrasound that can pass through solid objects to capture information from our fingertips. These technologies may have several noise factors associated with them. As noise is directly related to a high FAR, our incorporation of a pre-filter CNN will reduce noise substantially to maintain security to current authentication standards.

REFERENCES

[1]. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke and Y. Yu, X. Zheng, Tensorflow: a system for large-scale machine learning, in: OSDI, 16, pp. 265–283, 2016.

[2]. T. Barbu, Gabor filter-based face recognition technique, Proc. Roman. Acad. 11 (2010), 277–283.

[3]. J. D. Bowen, The home office automatic fingerprint pattern classification project, in: IEE Colloquium on Neural Networks for Image Processing Applications, IET, pp. 1–1, 1992.

[4]. F. Chollet, Xception: deep learning with depthwise separable convolutions, arXiv preprint (2017), 1610–02357.

[5]. J. Deng, W. Dong, R. Socher, L.- J. Li, K. Li and L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, CVPR 2009, IEEE, pp. 248–255, 2009.

[6]. K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 770–778, 2016.

[7]. M. Kaur, K-nearest neighbor classification approach for face and fingerprint at feature level fusion, Int. J. Comput. Appl. 60 (2012), 13–17.

[8]. M. Kawagoe and A. Tojo, Fingerprint pattern classification, Pattern Recogn. 17 (1984), 295–303.

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

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

[11]. D. Michelsanti, Y. Guichi, A.- D. Ene, R. Stef, K. Nasrollahi and T. B. Moeslund, Fast fingerprint classification with deep neural network, in: International Conference on Computer Vision Theory and Applications, 2018.

[12]. A. Rani and D. S. Palvee, SVM based classification and improvement of fingerprint verification, Int. J. Sci. Eng. Technol. Res 3 (2014), 879–883.

[13]. K. Rao and K. Balck, Type classification of fingerprints: a syntactic approach, IEEE Trans. Pattern Anal. Mach. Intell. 2 (1980), 223–231.

[14]. J. Ravi, K. B. Raja, K. R. Venugopal, Fingerprint recognition using minutia score matching, Int. J. Eng. Sci. Technol. 1 (2009), 35–42.

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

[16]. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015.

[17]. M. A. Turk and A. P. Pentland, Face recognition using Eigenfaces, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, Proceedings CVPR'91, IEEE, pp. 586–591, 1991.

[18]. Z. Xu, S. Li and W. Deng, Learning temporal features using LSTM-CNN architecture for face anti-spoofing, in: 3 rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, IEEE, pp. 141–145, 2015.

[19]. Y. Zhu, X. Yin, X. Jia and J. Hu, Latent fingerprint segmentation based on convolutional neural networks, in: 2017 IEEE Workshop on Information Forensics and Security (WIFS), IEEE, pp. 1–6, 2017.