Nexo Revista Científica
Vol. 36, No. 06, pp. 906-925/Diciembre 2023

ISSN-E 1995-9516
Universidad Nacional de Ingeniería
http://revistas.uni.edu.ni/index.php/Nexo
https://doi.org/10.5377/nexo.v36i06.17447

Key generation method from fingerprint image based on deep convolutional neural network model

Método de generación de claves a partir de imagen de huellas dactilares basado en un modelo de red neural convolucional profunda

Mithak Ibrahim Hashem [1, 2], Kadhim Hasen Kuban[2]

[1] Department of Software, College of Information Technology, University of Babylon, Hilla, Iraq.
[2] Department of Computer Science, College of Education for Pure Sciences, University of Thi Qar, Nasiriyah, 64001, Iraq.
mithaki.sw.hdr@student.uobabylon.edu.iq

(recibido/received: 28-agosto-2023; aceptado/accepted: 15-noviembre-2023)

ABSTRACT

Biometrics effect our live. Security applications employ biometrics. Biometric encryption is growing. Encryption requires biometric key creation. Long, random, and unexpected is the key. Information and communication security research emphasizes long, strong encryption keys. The proposed system uses fingerprint biometrics to generate a long, random biometric encryption key for symmetric encryption. Pre-processing removed noise from donor fingerprint images in the dataset. The program then trains an updateable Tuned VGG-16 convolutional neural network model and tests it on fingerprint images to learn fundamental fingerprint properties. The convolutional neural netwoprk CNN model retains the final weights for the second model to extract encryption key features. Transfer learning built a second convolutional neural network model to retrieve features without relearning. Keeping vector mean for processing. The last step generates an encryption key based on each person's vector of unique biometric features can be used for symmetric encryption algorithms to encrypt personal documents on the personal PC or personal cloud. Our CNN based method uses biometrics to recognize people and create safe and trustworthy encryption keys with over 99% accuracy in testing. Our 98%-accurate deep ANN classifier exceeds the support vector machine and random forest classifiers.

Keywords: Biometrics; Fingerprint; CNN Model; Transfer Learning; Key Generation.

RESUMEN

La biometría afecta nuestra vida. Las aplicaciones de seguridad emplean biometría. El cifrado biométrico está creciendo. El cifrado requiere la creación de una clave biométrica. Largo, aleatorio e inesperado es la clave. La investigación sobre seguridad de la información y las comunicaciones hace hincapié en claves de cifrado largas y sólidas. El sistema propuesto utiliza biometría de huellas dactilares para generar una clave de cifrado biométrica aleatoria larga para el cifrado simétrico. El preprocesamiento eliminó el ruido de las imágenes de huellas dactilares de los donantes en el conjunto de datos. Luego, el programa entrena un modelo de red neuronal convolucional Tuned VGG-16 actualizable y lo prueba en imágenes de huellas dactilares para conocer las propiedades fundamentales de las huellas dactilares. El modelo CNN de red neuronal convolucional conserva los pesos finales para que el segundo modelo extraiga las características

clave de cifrado. El aprendizaje por transferencia construyó un segundo modelo de red neuronal convolucional para recuperar características sin volver a aprender. Mantener la media vectorial para el procesamiento. El último paso genera una clave de cifrado basada en el vector de características biométricas únicas de cada persona que se puede utilizar para algoritmos de cifrado simétrico para cifrar documentos personales en la PC personal o en la nube personal. Nuestro método basado en CNN utiliza datos biométricos para reconocer a las personas y crear claves de cifrado seguras y confiables con más del 99 % de precisión en las pruebas. Nuestro clasificador ANN profundo con una precisión del 98% supera la máquina de vectores de soporte y los clasificadores de bosque aleatorios.

Palabras claves: Biometría; Huella dactilar; Modelo CNN; Transferir Aprendizaje; Generación de claves.

1. INTRODUCTION

Biometrics is the analyzing the properties of human (physical and/or behavioral) to identity in a fast and reliable manner by using of unique biological properties. Applications that have traditionally made use of biometrics include military access control, criminal or civil identity, and technological framework. These days, banking, shopping, and buying things online through a mobile device are all part of the same transaction. One of the most popular biometrics is a person's fingerprint. It can be specified by a visual progression of ridges on human fingers as they develop in infancy. Researchers have shown that no two fingerprints are ever exactly the same (Elhoseny et al., 2018). Identification and verification accuracy are quality-sensitive. Noise makes fingerprint pictures blurry. Scars, skin flaws, dampness, filth, and uneven fingerprint reader contact may all contribute. Hence, picture enhancement approaches increase structure and reduce noise. (Chakravarthy et al., 2017; Schuch et al., 2018). The large intra-class differences (differences across images of the same finger) and huge inter-class similarities in fingerprint matching create a pattern-recognition problem (similarity problem). Dry skin, incisions, pressure, rotation, and translation of the scanners can cause class differences. There are just a few fingerprint patterns—whorl, loop, and arch—but groups may share them. Fingerprint matching algorithms compare objects, skeletons, phases, or fine details (Jain et al., 2010). Precise matching algorithms are increasingly used. Local and global minutia-based matching algorithms exist (Mehmandoust & Shahbahrami, 2011; Peralta et al., 2015).

Key generation systems use biometrics to generate cryptographic keys. This research examines key-making processes. Hence, biometric cryptographic key generation alternatives are discussed. (Zaki, 2015), in this study created a fingerprint-based key to secure the system. This method has two parts: The first is 512 numeric values from EPROM fingerprint data enhanced, binarized, and thinned. With an 8x64 EPROM array, the first three linear shift registers supply the row address, while the second through seventh registers provide the column address. (Barman et al., 2015) use a mutually cancellable fingerprint template to generate cryptographic keys in 2015. Key-based steganographic exchanges can safely send cancellable fingerprint templates. They combine the two templates using concatenation-based feature-level fusion. The shuffle key randomly shuffles the combined template components to generate a new session key. Cancellable fingerprint template change provides fingerprint privacy while creating symmetric cryptography revocable keys.

(Partheeba & Radha, 2016) created orientation confidence level to assess fingerprint quality (OCL). If the picture quality is good, Scale Invariant Feature Transform (SIFT) extracts features. Otherwise, the image is ignored. Cover images may hide the cancelable template. Afterwards, Variable Least Significant Bit (VLSB) techniques will convey the hidden picture from sender to receiver and back. (Sanghvi & Mangrulkar, 2023) guarantees key unicity with fingerprints and adds unpredictability with fingerprint combinations. In this study, the key matrix is generated by extracting minute features from the sender and recipient's fingerprints using their combined minute detail template. This system has four stages. Enrolment, Authentication, Key Creation, and Cryptography. (Sarkar & Singh, 2018) offer a method for creating a 128-bit symmetric key from a cancelable fingerprint template shared by sender and recipient. This solution avoids key storage and distribution and confirms fingerprint privacy by one-way changing the

original template into a cancelable one. No one receives or stores the key. The first method applied to the data points yielded nullifiable template values for both sender and receiver.

(Panchal & Samanta, 2018) introduced a novel data safety technique in 2018. This approach generates a codeword using biometric statistical data. Reed-Solomon encodes random codewords. This password will create a key later. Before decryption, they check the user's identification using an SVM Ranking algorithm with no cutoff value. (Abed et al., 2019), a backpropagation neural network is used to encrypt/decrypt data without storing it using a bio-crypto key created via transparent biometrics. Due to background noise, transparent biometric modalities like fingerprints, faces, and keystrokes are difficult to use to create repeatable bio-crypto keys. (Suresh et al., 2020) used points collected at regular intervals, distances computed using Euclidean geometry, sorted by value, and converted to grayscale sequencing. Keys from gray codes. To reduce key discrepancy from duplicate fingerprints, the suggested method uses gray code. This research will focus on selecting minutiae points and adding an error correcting algorithm to produce a reliable cryptographic key.

(Wang et al., 2021) used fingerprints with feature distance as biometric keys. By measuring fingerprint details, they create a unique bio-key. The generation interval method determines and recovers bio-keys. They use two-layer error correction to protect sent data. (Barzut et al., 2021) presented a biometric fingerprint cryptosystem using convolutional neural networks and fuzzy commitment for authentication. By translating them into binary, they created a biometric cryptosystem for key-release systems and fingerprint matching biometric systems. Secure block-level Bose-Chaudhuri-Hocquenghem (BCH) error correction codes, immune to statistical-based attacks, reduce biometric data variability. (Wu et al., 2022) proposed a three-tiered fingerprint bio-key production architecture with a fingerprint biometry preprocessor, an FPBK stabilizer, and a fuzzy extractor. Deep neural network feature selection and layer-by-layer convolution projection eliminate fingerprint sample fluctuation in the FPBK Stabilizer. A multilayer convolutional projection fingerprint bio-key generation model is also created. Generating fingerprint keys from a 100-person fingerprint library tested the proposed framework. The proposed framework generated high-strength, stable, and resilient fingerprint bio-keys with a 98.0% accuracy rate (at 1024 bits) and a misrecognition rate of less than 1.5%.

(Suresh et al., 2022) proposed generating the key pair using fingerprint biometrics and a password. They use gray code to turn fingerprint minutiae gaps into a reliable binary string. Experimentally, gray code encoding reduces discrepancies between bit strings created from two fingerprint occurrences. Thus, the Reed-Solomon error correction algorithm corrects fingerprint duplicate differences to produce more consistent output. XORing the fingerprint and password hashes yields a safe seed. The recommended approach generates two enormous prime numbers from this seed value. These prime numbers produce a public-private key pair using the RSA key creation procedure. This seed value generates the same key pair every time.

As a problem statement, accurate fingerprint image processing relies on the detection of fingerprint image pixels. Low-quality fingerprint images often lack well defined image structures, making it impossible to accurately identify characteristics. Large localization errors in the location and orientation of minutiae pixels may be created, leading to the detection of many false minutiae pixels and the possible disregard of real minutiae pixels. The variant and unstable nature of fingerprint image due to scanner device or other source of noises may effect on the fingerprint features detection. The research introduced a framework to deal with these problems that may effect on best detection and extraction of fingerprint features to generate unique and reliable key from fingerprint image.

2. PROPOSED SYSTEM

Our proposed system deals with the fingerprints of ordinary people in order to generate encryption keys based on biometrics. Initially, the fingerprint has to go through a pre-processing stage, which reduces noise, blur and other problems caused by poor quality. The preprocessing stage also deals with issues such as alignment, rotation resulting from the individual's handling of the scanner. After the completion of the first stage (the pre-processing stage), a version of the enhanced fingerprint is entered into the second stage,

which is the stage of training the proposed model on the fingerprints of several people and according to their classes.

After completing the training process and obtaining the desired results, the model is tested and its efficiency verified. The Tuned VGG-16 model of convolutional neural networks was used for this purpose. Our proposed system exploits the transferability of learning from one convolutional model to another in order to distinguish the fingerprint classes according to their distinctive features. An artificial neural network (ANN) was used to classify each fingerprint according to its class. At this point, the third stage begins where the encryption key is generated by dealing with vectors generated from the convolutional model during the learning process. These vectors belong to the fingerprint classes that were trained in the previous stage. A set of normalization steps performed on vectors to suit the capabilities of generating cryptographic keys in a secure and reliable manner. After generating a biometric encryption key, the required documents and files are encrypted with a secure encryption system.

2.1. Components of the Proposed System

There are four major components of the proposed system:

1. Preprocessing component which performs image enhancement stage and contains sub stages such as dataset images filtering, normalization, resizing and augmentation.
2. Training and testing component which consists of sub stages such as building the proposed deep learning model of CNN, training, testing and saving weights for further processing.
3. Transfer learning Component which transfers learning from one model to another by loading the pre-trained model with the previously saved weights.
4. Feature extraction and mean vector generation component which is responsible for extraction of features of fingerprint from feature map and then generate the mean vector of each class which led to key generation stage.
5. Key generation stage.

Figure 1 depicts the block diagram of the components of the system.

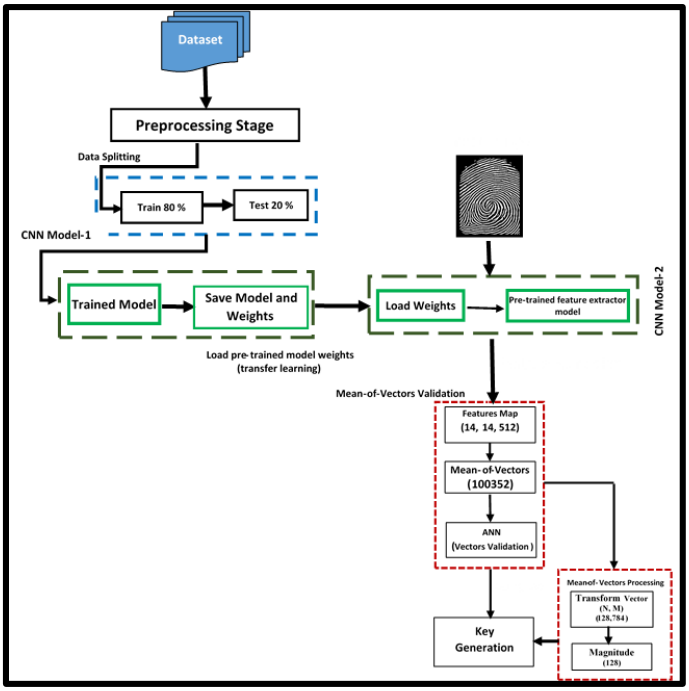


Figure 1. Components of the Proposed System

2.2. Preprocessing Stage

The first stage in the system is preprocessing which consists of two steps as follows:

2.2.1. Fingerprint Image Normalization

In order to create a uniform fingerprint image, the pixel intensity levels are normalized to fall within a predetermined range of grayscale values. The image contrast and brightness are improved by the normalization process, and scanner device noise is eliminated along with changes in grayscale levels caused by variances in finger pressure. This procedure is a pixel-by-pixel operation that does not alter fingerprint structures and simplifies subsequent calculations (Patel et al., 2020). During normalization process, a new value of pixel intensity is assigned to each pixel in the image. It's a pixel-by-pixel adjustment that doesn't affect the sharpness of the hills and valleys. As can be seen in the provided equations, the major goal of this technique is to lessen the range of grayscale values along the ridge and valley (Iwasokun et al., 2012):

$$N(i,j) = \begin{cases} M_0 + \sqrt{\frac{v_0(I(i,j)-M)^2}{V}} \\ M_0 - \sqrt{\frac{v_0(I(i,j)-M)^2}{V}} \end{cases} \quad (1)$$

Where,

$$M = \frac{1}{w\,x\,h} \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I(i,j) \quad (2)$$

$$V = \frac{1}{w\,x\,h} \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} (I(i,j) - M)^2 \quad (3)$$

M and V are the mean and variance of the fingerprint image I (i, j), M_o and V_o are the desired mean and variance values. Figure 2 depicts the normalization operation on the fingerprint image.

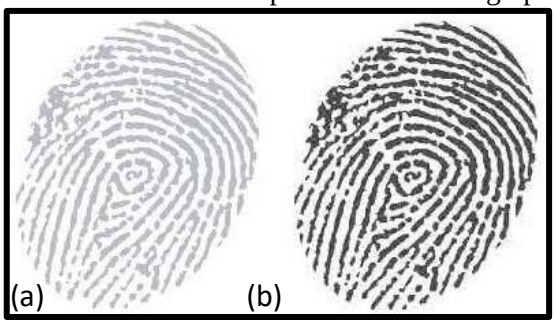


Figure 2. Fingerprint (a) Before Normalization (b) After Normalization

2.2.2. Gabor Image Filtering

The (WxW) blocks will be filtered using a finely calibrated Gabor filter to increase the contrast of the ridges. In the spatial domain, an even-symmetric Gabor filter has the shape described by equation (3). Figure 3 shows the output of applying normalization and Gabor filter to the fingerprint image. Before applying the Gabor filter, the orientation of the (WxW) blocks will be determined based on their position relative to the center of each pixel. Fingerprint image enhancement is a very important step where all other steps depend on the accuracy of the results from this step (Karo et al., 2019)

a. Original Fingerprint Image b. Enhanced Fingerprint Image

Figure 3. Apply Normalization and Gabor Filter on a Fingerprint Image

Gabor filters are formed from two components, sinusoidal and Gaussian. Through the use of a Gabor filter, the spatial domain may be linked to the best possible representation of orientation (frequency). In 1946, Gabor discovered the Gabor function, which he first defined in one dimension (t denoting time) before expanding into two dimensions (space) with the help of the following equations (4-6) (Ahmed et al., 2015; Karo et al., 2019):

$$G(x, y, f, \theta) = Exp\left\{\frac{-1}{2}\left[\frac{x_1^2}{\delta x^2} + \frac{y_1^2}{\delta y^2}\right]\right\}\cos(2\pi f x_1) \quad (4)$$

$$X_1 = x\ sin\ \theta + y\cos\theta \quad (5)$$

$$Y_1 = x\ cos\ \theta + y\sin\theta \quad (6)$$

A fixed distance away from the Gaussian characteristics along the x and y axes is denoted by x and y, where is the orientation direction, f is the cosine wave frequency, and is the cosine of the angle.

2.3. Training the CNN

When training a network, the goal is to discover the kernels in the convolutional layers and the weights in the FC layers that result in the smallest discrepancy between the predictions made by the network and the ground-truth labels on the training dataset. As the loss function and the gradient descent optimization algorithm play crucial roles in the back propagation algorithm, it is the most popular approach for training neural networks. Using a loss function and forward propagation on a training dataset, we may estimate how well a model will perform with a given combination of kernels and weights; then, using back propagation and gradient descent, we can adjust these learnable parameters to improve the model's performance. When training a CNN model, it is important to determine which kernels are most effective for a certain task using the available training data. In the convolutional layer, the only parameter that is automatically learnt during training is the kernel. However, the size of the kernels, the number of kernels, the padding, and the stride are all hyperparameters that must be specified before training can begin as shown in Table 1 (Vojt, 2016).

Table 1. CNN's List of Adjustable Settings

CNN Layers	Parameters	Hyperparameters
Convolution	Kernels	Kernel size, number of kernels, stride, padding, activation function.
Pooling Layer	None	Pooling method, filter size, stride, padding.
FC Layer	Weights	Number of weights, activation functions.

Others	Model architecture, optimizer, learning rate, loss function, mini-batch size, epochs, weight initialization, dataset splitting.

2.4. Transfer Learning Step

At this stage of implementation, the weights are transferred from the pre-trained model to the second model to extract the features that will be included in the later stage of validation through the ANN model. Advantage of the ability is taken to transfer learning from one neural model to another in order to start with the stage of extracting features from the new dataset instead of starting the process from scratch. This led to an increase in the accuracy of the learning results, and thus the classification and extraction of features, which will lead to an increase in the effectiveness of generating a solid key. To create a large-scale model with a limited data set, one useful technique is transfer learning and fine-tuning. In most cases, building a deep learning model will need a large quantity of data. However, it is not always simple to collect a large enough data set for that purpose. As a further caveat, from a training perspective, model creation may sometimes be a lengthy process. I know it bothers you when a single training session encompasses the whole day. It is possible to employ a fine-tune technique to address this issue in some domains, such as image categorization.

2.5. Deep Feature Extraction Stage

After some pre-processing, the fingerprint image is delivered to the deep feature extractor model, which is utilized to extract the important features that will be used to classify the fingerprint images where the extraction of features is a critical stage in the classification of image. The extracted features must be robust against variations in fingerprint illumination, alignment, rotation, noise, and other challenges. So, the features that are important for the biometric based key generation system must be extracted, and these features must contain the information required for discrimination between different persons. The aim of the features extraction process is to obtain the most important information from the original data (image). In this study, this aim was achieved by using two CNN, VGG-16 Tune.

After completing the transfer of the learning weights to the new model type VGG16 and the completion of the validation process on the new fingerprint, the features are extracted from the last convolutional layer in the VGG16 model, which stores the distinctive features of each fingerprint image as a matrix with dimensions (14, 14, 512) of length, width and depth, respectively. If it is assumed that each class contains 50 fingerprint images that have been trained on, then 50 matrices of the same aforementioned dimensions will be generated. Each Matrix within any class is converted into a one-dimensional vector to facilitate future dealing with it. This vector contains 100,352 floating elements whose values are between zero and one. The median is calculated for all vectors of a particular class to obtain the so-called Mean-of-Vectors.

Algorithm1: VGG 16 Tune Based Feature Extraction

Input: Input images

Output: Feature vector for each image within class

Begin

Load input images and create an image dataset.
Make image preprocessing step.
Split the dataset into 80% training and 20% test data.
Load the pre-trained VGG16 Tune model (architecture and weights).
Defining the layer from which features are extracted.
Defining the layer from which features are extracted.
While not end of training images set
Read an image from training set.
Extract training features of the fingerprint image from the layer is last convolutional layer in (block5_conv3).
Add extracted feature vector to "TrainingFeatures" vectors.
End While
Save training feature vectors as "TrainingFeatures".
While not end of test images set
Read an image from test set.
Extract test features of the fingerprint image from the layer is last convolutional layer in model (block5_conv3).
// The extracted features are a vector with (14, 14, 512) = 100,352.
Add extracted feature vector to "TestFeatures" vectors.
End While
Save test feature vectors as "TestFeatures"

End

2.6. Key Generation Stage Algorithm

At this stage, several parameters coming from the previous stages are dealt with, namely: the accuracy threshold as an output from Deep ANN, the distinguished class number c, in addition to the Mean-of-Vectors as a magnitude value which is represented by one dimensional vector of (128). All of these parameters enter into the unique key generation stage. The average vector is dealt with in the form of a matrix to facilitate the task of mathematical dealing with it, as it is converted to 128 784, and this is what generates 128 rows. We will need to normalize these vectors by finding their magnetode value. The values generated by 128 element values represent a summary of the mean vectors, which in turn represent the feature map of the finger image. The key is generated using key derivation functions KDF with SHA256 and salt values.

Algorithm2: Key Generation
Inputs: Fingerprint Image of Specific Class, Load Mean-of-Vectors
Outputs: Generated Key

Begin
Load enhanced fingerprint image.
Load and Run ANN model.
From feature vector of fingerprint image applied to ANN produce predicted class number c.
Load class number c and threshold.
By class number load the identical Encrypted Mean-of-Vectors from Encrypted Mean-of-Vectors Database.
Decrypt Mean-of-Vector.
Transform Seed Vector SV=100,352 elements to the shape (N,M) of (128,784).

For I = 1 to 128 do
Compute the magnitude value for each row as MG(I).
Next I
Store one dimensional vector (128,1) as V.
Load key derivation function (V,SHA256,Salt)
Generate Key String K.
Save Key K for Encryption Stage.
End

3. RESULTS AND DISCUSSION

The proposed system was executed using a personal computer with the following specifications: firstly, the hardware by a processor Intel(R) Core(TM) i5- with Microsoft Windows 10 (64 bit). Secondly, the software tool is by Python implemented on Colab Pro platform produced by Google Cloud.

3.1. Training and Validation

In our proposed system, we used the Fine Tuning VGG16 model to train the first dataset on fingerprints. The modular system was trained on ImageNet, but we modified the model to meet the requirements of our proposed system. The last three layers were trained on the properties of fingerprints only so that the model is able to deal with fingerprints only in a strong and customized way. It is also known that the basic figures in human fingerprints are the core of the fingerprint as a global figure and the ends of the edge and branches as local figures. It is one of the difficult figures in the training process in terms of that it has similar features among all human fingerprints and depends on its discovery on accurate details such as location and rotation in determining discriminant. Table 2 shows the model summary:

Table 2. CNN model (VGG-16 Tuned) Summary:

Model Parameters	No. of Parameters
Total Number of parameters	14,726,996
Trainable parameters No.	4,730,900
Non-trainable parameters No.	9,996,096

Table 3 describes the result (first ten epochs and last ten epochs) of 100 epochs to train this model.

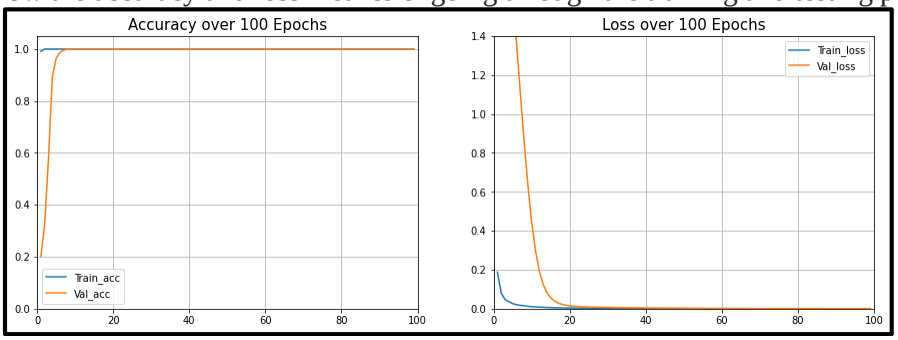
Table 3. Selected twenty epochs from Model Training Report.

Epoch No#	Loss	Accuracy	Val._Loss	Val._Accuracy	Learning Rate
1/100	1.1970	0.7364	2.8133	0.0924	1.0000e-04
2/100	0.1873	0.9929	2.5180	0.1681	1.0000e-04
3/100	0.0793	1.0000	2.2818	0.2773	1.0000e-04
4/100	0.0485	1.0000	2.0474	0.6134	1.0000e-04
5/100	0.0340	1.0000	1.8093	0.8992	1.0000e-04
6/100	0.0268	1.0000	1.5588	0.9748	1.0000e-04
7/100	0.0214	1.0000	1.2995	0.9958	1.0000e-04
8/100	0.0166	1.0000	1.0359	0.9958	1.0000e-04
9/100	0.0142	1.0000	0.7881	1.0000	1.0000e-04
10/100	0.0119	1.0000	0.5674	1.0000	1.0000e-04

.

		.			
		.			
		.			
		.			
		.			
91/100	9.7264e-05	1.0000	0.0012	1.0000	1.0000e-04
92/100	9.1182e-05	1.0000	0.0010	1.0000	1.0000e-04
93/100	8.7003e-05	1.0000	0.0011	1.0000	1.0000e-04
94/100	8.2568e-05	1.0000	0.0011	1.0000	1.0000e-04
95/100	7.6151e-05	1.0000	0.0011	1.0000	1.0000e-04
96/100	8.4102e-05	1.0000	0.0010	1.0000	1.0000e-04
97/100	7.9641e-05	1.0000	0.0011	1.0000	1.0000e-04
98/100	6.6916e-05	1.0000	0.0010	1.0000	1.0000e-04
99/100	6.6890	1.0000	0.0011	1.0000	1.0000e-04
100/100	6.6448e-05	1.0000	0.0010	1.0000	1.0000e-04

Figure 4 will show the accuracy and loss metrics ongoing through the training and testing process.



a. Accuracy over 100 Epochs b. Loss over 100 Epochs

Figure 4. Accuracy, Loss after 100 Epochs

Figure 5 shows accuracy, loss, validation accuracy, validation loss and learning rate over epochs.

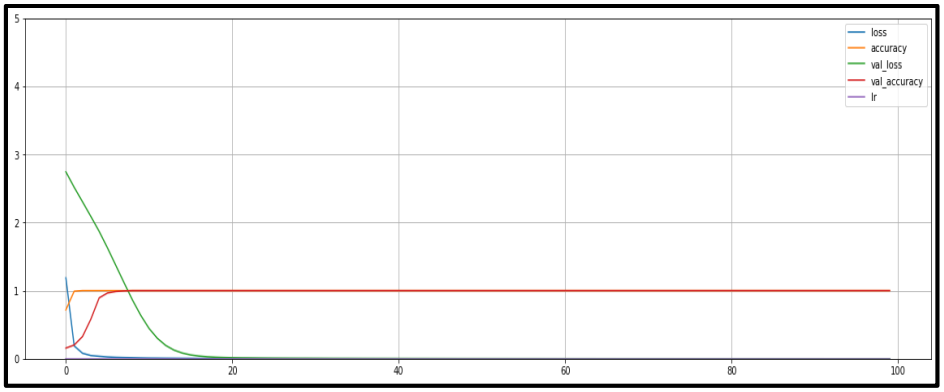


Figure 5. Accuracy, Loss for training and accuracy, loss for Validation and Learning Rate over epochs

Figure 6 shows training and validation accuracy, training and validation loss in split charts.

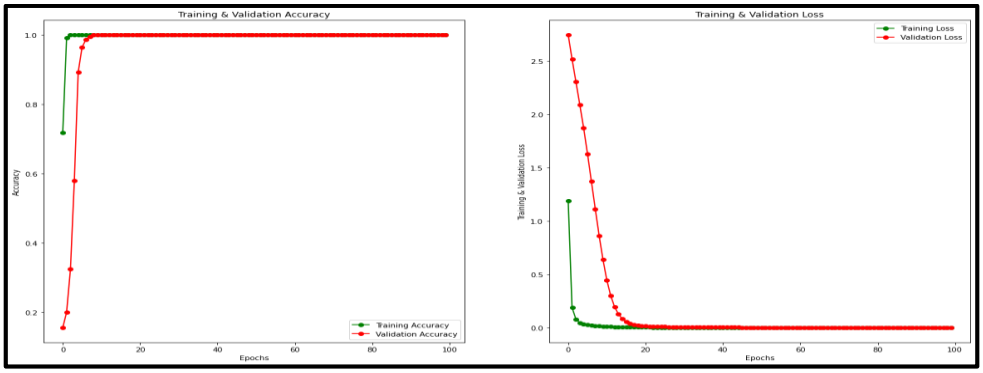


Figure 6. Accuracy and Loss (for Training and Validation stages) over Epochs in Split Charts

Table 4 shows the precision, recall and F1 measures for this model and it exhibits the accuracy %100.

Table 4. Precision Recall, F1-Score and Accuracy Metrics of Training Model:

Class No#	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
Accuracy	1.00		

Figure 7 depicts the confusion matrix as a classification metrics for this model and it shows the complete correspondence between true labels and predicted labels. The accuracy is %100 in testing model.

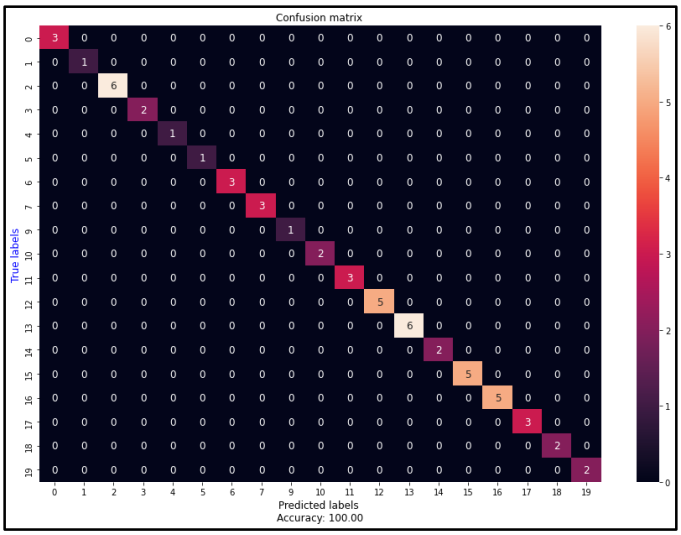


Figure 7. Confusion Matrix (Actual, Predicted) Labels and Accuracy Metrics

3.2. Loading weights to VGG16-fine tuning model

The stored weights are uploaded to a pre-trained Tune model in order to extract the features that play a key role in generating the biometric keys. In our proposed model, the convolutional layers are implemented as usual, and then the figs are extracted from the last convolutional layer of the model structure. Table 5 shows the model summary through training process.

Table 5. Feature Extractor VGG-16 Tune Model Summary:

Model Parameters	No. of Parameters
Total Number of parameters	14,714,688
Trainable parameters No.	14,714,688
Non-trainable parameters No.	0

3.3. Implementation of ANN Classifier

The ANN classifier plays an important role in determining the extent to which a finger image belongs to a specific class according to a predefined threshold. Experiments have proven that the ANN classifier was the best with an accuracy rate of 98%. After classifying the image, its private key is generated from the stored vector. We proposed a deep ANN sequential model for this purpose. Table 6 depicts the structure of this deep ANN classifier summary, while Table 7 depicts the accuracy metrics of the deep ANN classifier.

Table 6. Deep ANN Classifier Model Summary.

Model Parameters	No. of Parameters
Total Number of parameters	412,117,268
Trainable parameters No.	412,108,564
Non-trainable parameters No.	8,704

Table 7. Deep ANN Classifier Recall, Precision and F1-Score Accuracy Metrics

Class No#	Precision	Recall	F1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	0.67	0.80
7	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	0.75	1.00	0.86
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
Accuracy 0.98			

Figure 8 depicts the training and validation accuracy and loss.

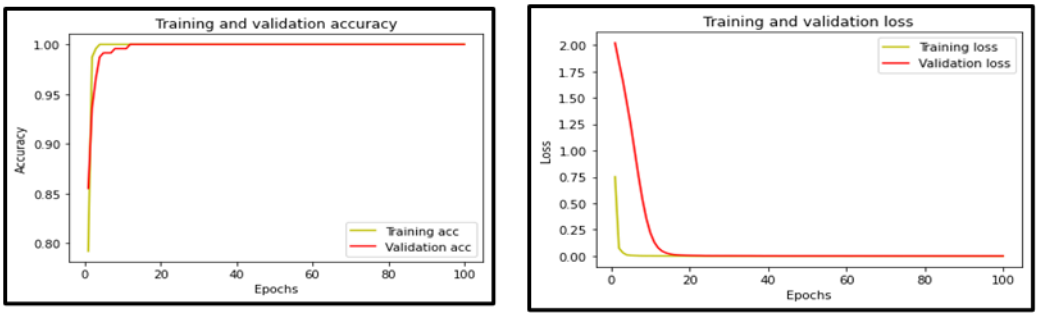


Figure 8. Accuracy and Loss over Epochs

The following chart shown in Figure 9 depicts the accuracy (training/validation), loss (training/validation) and learning rate for this ANN deep sequential model.

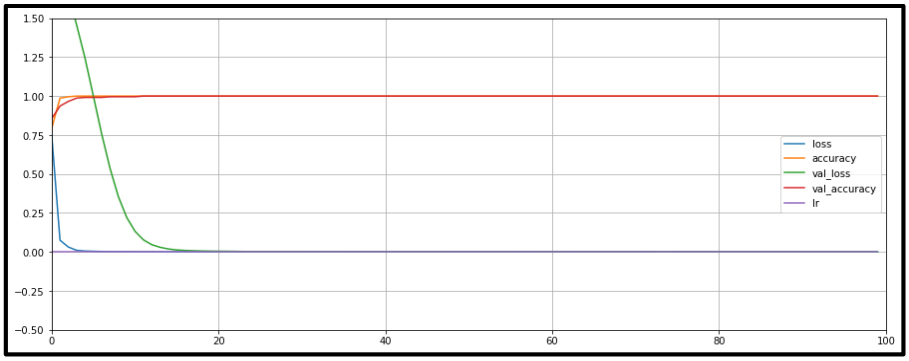


Figure 9. Accuracy, Loss (for Training and Loss) and Learning Rate over Epochs

Figure 10 shows the confusion matrix of this deep classifier.

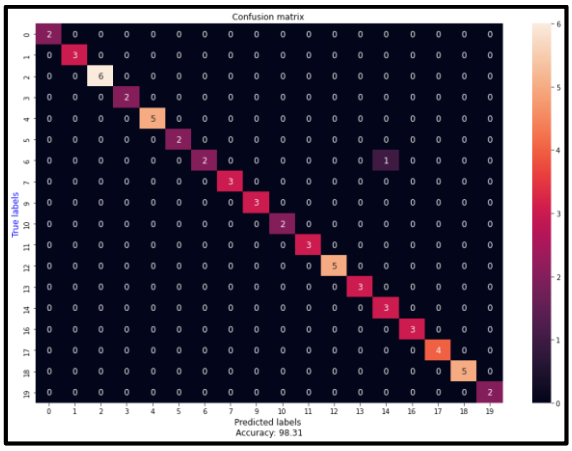


Figure 10. Confusion Matric (Actual, Predicted) Labels and Accuracy Metric

3.4. Performance Evaluation

To discuss the results, accuracy will be taken then precision/recall and F1-score metrics. Then we will compare between our deep Artificial Neural Network (ANN) classifier performances with other classifiers such as Support Vector Machine (SVM) classifier and Random Forest (RF) classifier.

3.4.1. Accuracy

Starting with accuracy as a criterion for our proposed system, relative to the rest of the systems published in the past few years, according to their sources. Practical experiments on our proposed model showed an accuracy of more than % 99. Table 8 shows a comparison of other systems with our proposed system.

Table 8. Accuracy Metrics of Our Proposed system with Several Recent Approaches

Reference	Method	ACC %
(Gupta & Gupta, 2015)	Singular-point	97.80
(Darlow & Rosman, 2017)	Minutiae-and-DL	94.55
(Andono & Supriyanto, 2018)	Bag-of-Visual-Words	90
(Saeed et al., 2018a)	Statistics of D-SIFT descriptor	97.40
(Saeed et al., 2018b)	Modified-HOG descriptor	98.70
(Saeed et al., 2022)	DeepFKTNet model	98.89
(Jeon & Rhee, 2017)	Ensemble-CNN model	97.2
(Zia et al., 2019)	B-DCNNs	95.3
(Nguyen & Nguyen, 2019)	CNN (tested on 3 classes of FVC2004)	96.1
(Nahar et al., 2022)	Modified-LeNet (tested on FVC2004-DB1)	99.1
(Wu et al., 2022)	MCP-FP model	98
(Ang et al., 2018)	CIEC-method	> 92
Proposed System	CNN and DL method	> 99

3.4.2. Recall, Precision and F1-Scores Metrics

Recall, Precision and F1-score metrics can be represented in the following Table 9:

Table 9. Recall, Precision and F1-Score Metrics of our Approach.

	Recall	Precision	F1-score
Proposed System	1.0	1.0	1.0

3.4.3 ANN, SVM and Random Forest Classifiers Comparisons

The Table 10 shows the precision, recall and F1-score metrics between our classifier with other two classifiers. The results shows that our proposed deep learning ANN classifier is higher accuracy.

Table 10. Comparison of Precision, Recall, F1-Score and Accuracy Metrics for Our Proposed Deep ANN Classifier with Support Vecttor Machine (SVM) and Random Forest (RF) Based Classifiers:

Classifier	Precision	Recall	F1-score	Accuracy
Support Vecttor Machine (SVM)	95.0	92.0	92.0	93.0
Random Forest (RF)	95.0	92.0	92.0	93.0
Deep ANN Based Proposed Classifier	99.0	98.0	98.0	98.0

3.5. Case Study of Key Generation

The vectors generated from the previous stage are dealt with the threshold of each class to find the vectors mean. The values are normalized and the vector is converted to a binary matrix. Final vector 128 elements to generate the key. The following list is for a report on the number of images in the classes.
All samples in dataset with shape: (20,)
All samples Label shape: (20,)
Original labels are : ['0', '1', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '2', '3', '4', '5', '6', '7', '8', '9']
New labels read are : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19].
Table 11 is for a report on the number of vectors in each class.

Table 11. Number of Vectors in Classes.

Class #	No. of Vectors
Class: 0	Contains: 70 Vectors.
Class: 1	Contains: 70 Vectors.
Class: 2	Contains: 70 Vectors.
Class: 3	Contains: 70 Vectors.
Class: 4	Contains: 70 Vectors.
Class: 5	Contains: 70 Vectors.
Class: 6	Contains: 70 Vectors.
Class: 7	Contains: 70 Vectors.
Class: 8	Contains: 75 Vectors.
Class: 9	Contains: 77 Vectors.
Class: 10	Contains: 70 Vectors.
Class: 11	Contains: 70 Vectors.
Class: 12	Contains: 84 Vectors.

Class: 13	Contains: 70 Vectors.
Class: 14	Contains: 70 Vectors.
Class: 15	Contains: 84 Vectors.
Class: 16	Contains: 84 Vectors.
Class: 17	Contains: 84 Vectors.
Class: 18	Contains: 70 Vectors.
Class: 19	Contains: 70 Vectors.

The following description is a screenshot of Python results:

```
(20, 100352)
Vector seed: (100352,)
(20, 128, 784)
Matrix M x N: (128, 784)
(20, 128, 1)
Final Vector V: (128, 1)
key generated shape: (1, 14, 14, 512)
shape of transformed embedding key to 1-D vector is (1, 100352)
shape of  normalize 1-D vectors of key is : (1, 100352)
##################################
Predicted finger labels is: 1
The predication ratio is:  0.9994696
Original label is: 1
mean_vectors_1D : (20, 100352)
##################################
(20, 128, 784)
(20, 128)
Vector of predicated class: (128,)
##################################
[14.929769  12.263961  14.816275  11.672906  14.803835  11.599381
 14.9679    11.922448  15.212897  14.27953   10.194674  13.842593
 10.609922  14.018163  10.647506  13.734892  10.399862  13.164744
 12.495237  15.164312  12.84347   16.27667   13.136515  16.273499
 12.785971  15.526098  14.75975   12.830288  14.985918  13.176794
 16.889175  13.975275  16.834654  13.602913  15.984798  11.549228
 15.496771  11.472614  16.423452  14.400638  17.438358  17.454704
 13.755843  16.170088  11.482987  15.201067  11.2428465 16.326849
 14.223474  17.214596  14.106871  17.013025  12.856354  14.863074
 11.620988  14.53986   12.622108  16.75001   17.088324  14.03729
 16.78401   13.205618  14.844906  11.462029  14.524599  12.234268
 16.356714  13.688227  16.685543  13.901848  16.316341  12.16767
 14.125783  15.013947  11.87491   16.310966  13.549695  16.606638
 13.763899  16.19522   12.296909  13.829541  12.135776  15.142895
 13.030238  16.281967  13.743974  16.63217   13.395031  15.158247
 13.963714  12.067517  15.054442  12.678177  15.861125  12.952465
 16.230585  12.604585  14.745344  10.294409  15.083842  11.8574505
 16.351986  13.034914  16.044096  16.306314  12.422688  14.843797
 10.636608  14.671333  10.619393  14.62976   10.953194  14.268913
 10.888568  13.95222   10.163493  13.404393  10.961958  15.500435
 12.51335   15.257783  15.471551  12.135666  14.688079  11.45552
 14.473213  12.435441 ]
```

Key is: '3PO09RAfNA0gG5vyEZqxorBJ4caVYtmff1Q2sD9702c='

3.6. Security analysis

The experimental results shows a high performance of our proposed system in training, testing, validation, classification and then key generation based on these previous issues. The high accuracy of our proposed model, which exceeded 99% in recognizing the fingerprints of the users of the system, produced high accuracy for generating biometric keys, and this was confirmed by the test results of the proposed system.

This section addresses the protection of fingerprint biometrics, password and private key in the context of proposed approach. Understanding of several attacks reveals that the proposed approach prevents the compromise of fingerprint biometrics, password and the private key. Arguments in favor of the above claim are presented in the following subsections.

- Lack of transparency for models based CNN

Many previous studies support the hypothesis of lack of transparency for models based on convolutional neural networks. Therefore, storing the model with weight and other parameters does not leak any useful information to adversaries in order to break the biometric key generation mechanism.

- No storage of generated key

The suggested approach encrypts private data files using a biometrics based generated key. The same cryptographic key is gleaned from the biometrics each time. As the created biometric key is being generated again whenever it is required, there is no need to keep it anywhere.

- Security of fingerprint bio-information

In the memory, which comprises 100352 vector elements that represents a numerical values of machine learning model, only the mean-of-vectors M that are produced during the key vector operation are kept. Since the vector M contains no biometric information, the biometric information is protected from being compromised in the backend system and from being recorded by a multiplicity attack, blended substitution attack, or cross-matching attack.

- Exhaustive search attack

An attacker in this scenario is unaware of the key creation process or any of the supporting data that is kept in the memory. Attackers need to do a lot of extensive search to uncover the real key. The total number of probes is proportional to the size of the produced key. If the produced key length is L, the number of searches an attacker must do is 2^{L-1}.

- Key Randomness

The randomization of deep CNN and ANN parameters and vectors support the randomness entropy of biometric key generation.

4. CONCLUSION

Fingerprint key generation framework based on the more stable feature maps extracted by CNN model has been implemented. Experimental results show that using a pre-trained VGG16 Fine Tune CNN model with transfer learning technique led to higher accuracy results compared to the old traditional minutia-based methods. Experimental results show that deep ANN classifier deployment as a classifier within our proposed model gives an accurate classification result comparing to other classifiers such as SVM and RF classifiers. Experimental results show that the proposed Fine Tuned VGG16 CNN based model gives high accuracy to recognize fingerprint even though the low quality of some fingerprint images within dataset. Experimental results show that distortion in the fingerprint images (i.e. high rates of damaged portions within fingerprint images) can effect on the feature extraction accuracy of the proposed CNN model. Generating the biometric key from hidden parameters of CNN model gives a high security to the key generating mechanism. Security analysis shows that the stored parameters of the CNN model with lack of transparency nature will give more secrecy and protection to the user biometrical attributes.

REFERENCIAS

Abed, L., Clarke, N., Ghita, B., & Alruban, A. (2019). Securing cloud storage by transparent biometric cryptography. *Innovative Security Solutions for Information Technology and Communications: 11th International Conference, SecITC 2018, Bucharest, Romania, November 8–9, 2018, Revised Selected Papers 11*, 97–108.

Ahmed, H. H., Kelash, H. M., Tolba, M., & Badwy, M. (2015). Fingerprint image enhancement based on threshold fast discrete curvelet transform (FDCT) and gabor filters. *International Journal of Computer Applications*, *110*(3), 33–41.

Andono, P., & Supriyanto, C. (2018). Bag-of-visual-words model for fingerprint classification. *Int. Arab J. Inf. Technol.*, *15*(1), 37–43.

Ang, L.-M., Seng, K. P., Ijemaru, G. K., & Zungeru, A. M. (2018). Deployment of IoV for smart cities: Applications, architecture, and challenges. *IEEE Access*, *7*, 6473–6492.

Barman, S., Samanta, D., & Chattopadhyay, S. (2015). Fingerprint-based crypto-biometric system for network security. *EURASIP Journal on Information Security*, *2015*(1), 1–17.

Barzut, S., Milosavljević, M., Adamović, S., Saračević, M., Maček, N., & Gnjatović, M. (2021). A novel fingerprint biometric cryptosystem based on convolutional neural networks. *Mathematics*, *9*(7), 730.

Chakravarthy, I., Balaram, V., & Reddy, B. E. (2017). Overview of Fingerprint Image Enhancement & Minutia Extraction. *International Journal of Computer Science and Information Technologies*, *4*(1).

Darlow, L. N., & Rosman, B. (2017). Fingerprint minutiae extraction using deep learning. *2017 IEEE International Joint Conference on Biometrics (IJCB)*, 22–30.

Elhoseny, M., Elkhateb, A., Sahlol, A., & Hassanien, A. E. (2018). Multimodal biometric personal identification and verification. *Advances in Soft Computing and Machine Learning in Image Processing*, 249–276.

Gupta, P., & Gupta, P. (2015). A robust singular point detection algorithm. *Applied Soft Computing*, *29*, 411–423.

Iwasokun, G. B., Charles, A. O., Kayode, A. B., & Olatubosun, O. (2012). Fingerprint image enhancement: Segmentation to thinning. *International Journal of Advanced Computer Science and Applications*, *3*(1).

Jain, A. K., Feng, J., & Nandakumar, K. (2010). Fingerprint matching. *Computer*, *43*(2), 36–44.

Jeon, W.-S., & Rhee, S.-Y. (2017). Fingerprint pattern classification using convolution neural network. *International Journal of Fuzzy Logic and Intelligent Systems*, *17*(3), 170–176.

Karo, N. N. B., Sari, A. Y., Aziza, N., & Putra, H. K. (2019). The enhancement of fingerprint images using gabor filter. *Journal of Physics: Conference Series*, *1196*(1), 12045.

Mehmandoust, S., & Shahbahrami, A. (2011). A Comparison between Different Fingerprint Matching Techniques. *Digital Information and Communication Technology and Its Applications: International Conference, DICTAP 2011, Dijon, France, June 21-23, 2011. Proceedings, Part I*, 242–253.

Nahar, P., Chaudhari, N. S., & Tanwani, S. K. (2022). Fingerprint classification system using CNN. *Multimedia Tools and Applications*, *81*(17), 24515–24527.

Nguyen, H. T., & Nguyen, L. T. (2019). Fingerprints classification through image analysis and machine learning method. *Algorithms*, *12*(11), 241.

Panchal, G., & Samanta, D. (2018). A novel approach to fingerprint biometric-based cryptographic key generation and its applications to storage security. *Computers & Electrical Engineering*, *69*, 461–478.

Partheeba, S., & Radha, N. (2016). Fingerprint Bio-Crypto key generation using scale invariant feature transform (SIFT). *International Journal of Computer Applications*, *153*(8), 35–41.

Patel, M. B., Parikh, S. M., & Patel, A. R. (2020). Global normalization for fingerprint image enhancement. *Computational Vision and Bio-Inspired Computing: ICCVBIC 2019*, 1059–1066.

Peralta, D., Galar, M., Triguero, I., Paternain, D., García, S., Barrenechea, E., Benítez, J. M., Bustince, H., & Herrera, F. (2015). A survey on fingerprint minutiae-based local matching for verification and identification: Taxonomy and experimental evaluation. *Information Sciences*, *315*, 67–87.

Saeed, F., Hussain, M., & Aboalsamh, H. A. (2018a). Classification of live scanned fingerprints using dense sift based ridge orientation features. *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, 1–4.

Saeed, F., Hussain, M., & Aboalsamh, H. A. (2018b). Classification of live scanned fingerprints using histogram of gradient descriptor. *2018 21st Saudi Computer Society National Computer Conference (NCC)*, 1–5.

Saeed, F., Hussain, M., & Aboalsamh, H. A. (2022). Automatic fingerprint classification using deep learning technology (DeepFKTNet). *Mathematics*, *10*(8), 1285.

Sanghvi, S., & Mangrulkar, R. (2023). BITSAT: an efficient approach to modern image cryptography. *International Journal of Computational Science and Engineering*, *26*(3), 268–282.

Sarkar, A., & Singh, B. K. (2018). Cryptographic key generation from cancelable fingerprint templates. *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, 1–6.

Schuch, P., Schulz, S., & Busch, C. (2018). Survey on the impact of fingerprint image enhancement. *IET Biometrics*, *7*(2), 102–115.

Suresh, K., Pal, R., & Balasundaram, S. R. (2020). Fingerprint based cryptographic key generation. *Intelligent Data Communication Technologies and Internet of Things: ICICI 2019*, 704–713.

Suresh, K., Pal, R., & Balasundaram, S. R. (2022). Two-factor-based RSA key generation from fingerprint biometrics and password for secure communication. *Complex & Intelligent Systems*, *8*(4), 3247–3261.

Vojt, J. (2016). *Deep neural networks and their implementation*.

Wang, P., You, L., Hu, G., Hu, L., Jian, Z., & Xing, C. (2021). Biometric key generation based on

generated intervals and two-layer error correcting technique. *Pattern Recognition*, *111*, 107733.
Wu, Z., Lv, Z., Kang, J., Ding, W., & Zhang, J. (2022). Fingerprint bio-key generation based on a deep neural network. *International Journal of Intelligent Systems*, *37*(7), 4329–4358.
Zaki, H. A. (2015). Cryptographic key Generation Using Fingerprint Biometrics. *J. Thi-Qar Sci*, *5*(2).
Zia, T., Ghafoor, M., Tariq, S. A., & Taj, I. A. (2019). Robust fingerprint classification with Bayesian convolutional networks. *IET Image Processing*, *13*(8), 1280–1288.

PROFILE OF THE AUTHORS

Mithak I. Hashem is a Lecturer and Faculty Member in the Department of Computer Science, College of Education for Pure Sciences, University of Thi Qar, Iraq since 2005. He received the B.Sc. degree with the first honour in Computer Science from University of Thi Qar, Iraq in 2002. He then received M.Sc. degree from Iraqi Commition for Computers and Informatics, Informatics Instutute for Postgraduate Studies. He is now a Ph.D. in the College of Information Technology, University of Babylon, Iraq. His research interests include Artificial Intelligent, Machine Learning, Image Processing and Information Security. He can be contacted at Email: mithaki.sw.hdr@student.uobabylon.edu.iq or mithak@utq.edu.iq.

Kadhim H. Kuban is a Professor and Faculty Member at the Department of Computer Science, University on Thi Qar, Iraq. He received B.Sc. degree in Computer Science from Basra University, Iraq in 1985, and a M.Sc. in Computer Science from Technology University, Iraq in 1992. He then received the Napoli University Postgraduate Scholarship from the Itali government in 2006 and completed his Ph.D. in Computer Science in 2010. His research interests are primarily in the area of Information Security and E-Learning, where he is the author/co-author of multiple research publications. He can be contacted at Email: khalibraheemi@utq.edu.iq.