Original Research Article
Optimizing Touchless Fingerprint Identification: A Machine Learning Approach to Modelling and Performance Evaluation

ABSTRACT

This paper explored the modelling and performance analysis of a smartphone-based fingerprint identification system using Convolutional Neural Networks (CNN). The research developed a theoretical framework to validate picture-based fingerprint identification as a feasible alternative to traditional touch-based methods. A modified Automated Fingerprint Identification System (AFIS) model served as the study's foundation. To enhance the model's capabilities, data from two databases, IIT India and SOCOFing, were utilized. The evaluation of the CNN architecture focused on mobile device fingerprint recognition.It emphasized key processes such as data pre-processing, model training, and the optimization of the CNN through a Siamese-CNN approach to boost accuracy and efficiency. Python scripts developed for this purpose were converted to Android code using TensorFlow for deployment on Android devices. Performance metrics, including identification accuracy, processing speed, and resource utilization, were analysed to determine the system's feasibility. The results demonstrated that CNN-based fingerprint identification systems hold significant promise for delivering robust and reliable biometric authentication on smartphones, highlighting both their practical applications and limitations. decrease medical as well as financial burden, hence improving the management of cirrhotic patients. These predictors, however, need further work to validate reliability.

Keywords: Convolutional Neural Networks (CNN), identification, fingerprint, Siamese-CNN, picture-based

1. INTRODUCTION

Blau et al. [1] underscored the crucial role of effective human identification in ensuring the proper functioning of society, particularly in areas such as security, criminal prosecution, and the identification of human remains. Among the various identification methods, fingerprints have emerged as a leading technique due to their reliability and extensive use in security devices [2]. The transition from ink-based fingerprinting to modern biometric systems has fuelled the rapid adoption of touch-based fingerprint identification, which is now trusted for its speed, safety, and reliability [3], [4].

Despite its widespread adoption, fingerprint technology still faces challenges, especially regarding the complexity of processing [5]. According to [6], innovations like machine learning (ML) and deep learning (DL) techniques, such as convolutional neural networks (CNNs), are being integrated to improve fingerprint identification systems. The accuracy of these systems is largely dependent on the quality of fingerprint data representation [7].

The integration of fingerprint sensors in smartphones has played a significant role in advancing fingerprint classification techniques, positioning smartphones as a valuable testbed for these improvements [8], [9]. This paper proposes a smartphone-based fingerprint

identification system utilizing DL, specifically a CNN algorithm, to recognize fingerprint features from images captured by smartphone cameras. The goal is to enhance the security and efficiency of image-based fingerprint identification as a viable alternative to traditional minutiae-based methods.

2. REVIEW OF RELATED LITERATURE

The literature review presents several notable studies on photo-based fingerprint recognition systems:

Nur-A-Alam et al. [10] developed an intelligent fingerprint identification system using a combination of Gabor filter features and deep learning (CNN). Their approach achieved high accuracy (99.87%) compared to other methods, but its performance heavily depended on image quality. The research suggested that integrating a semi-automatic minutia-based verification algorithm could reduce this dependency.

Praseetha et al. [11] proposed a secure fingerprint authentication system using CNN as a pre-verification filter and a minutia-based algorithm for user authentication. The system processed optical fingerprint images, achieving high accuracy, but it was limited by its reliance on sensor-generated images rather than camera-captured ones.

Birajadar et al. [12] introduced a smartphone-based touchless fingerprint recognition system using monogenic wavelets. While their system showed promising results, its performance was lower for raw touchless images compared to touch-based systems. The study highlighted the potential improvement from using CNN and integrating additional features like touchless fingerprint quality assessment and liveness detection.

Attrish et al. [13] proposed a contactless fingerprint recognition system using a Siamese CNN to extract features from finger photographs. The system achieved a low equal-error-rate (2.19%) but was limited by the small sample size used in training, impacting the generalizability of the results.

Seow et al. [14] explored image-based fingerprint verification using Inverse Fast Fourier Transform (IFFT) after thinning the fingerprint image. Although the method was efficient, it is considered outdated compared to modern CNN-based techniques, which offer more accurate image processing.

Ref [15] proposed a new fingerprint recognition method designed for low-quality images, like those on Myanmar National Registration Cards (NRCs). Traditional minutiae-based techniques struggled with poor-quality fingerprints, so the authors introduced a ridge feature-based approach. Using contextual filtering, a single-pass thinning algorithm, and Gabor filtering, the system enhanced ridge lines. Ridge features were then extracted and compared using the Euclidean distance metric.

In summary, while these studies provide valuable insights into photo-based fingerprint recognition, gaps remain in image quality dependence, sensor reliance, and the need for more advanced techniques like CNN for improved accuracy and generalization.

3. MATERIAL AND METHODS

The steps taken in realising the proposed fingerprint identification system are as represented in the block diagram shown in Fig. 1. Some blocks in Fig. 1 are examined in further detail in the following subsections.

3.1 Image Capturing

The image capturing in this context involves using a smartphone camera to capture the raw image of fingerprints, as seen in Fig. 2. The customized User Interface (UI) with integrated finger recognition developed by [16] was used to quickly identify the Region of Interest (ROI) for every image captured. This process's effectiveness depended on integrating the High Dynamic Range (HDR) technique in the customized UI for the image capture. HDR technique ensures that the same image quality is achieved in each photograph irrespective of surrounding light intensity.

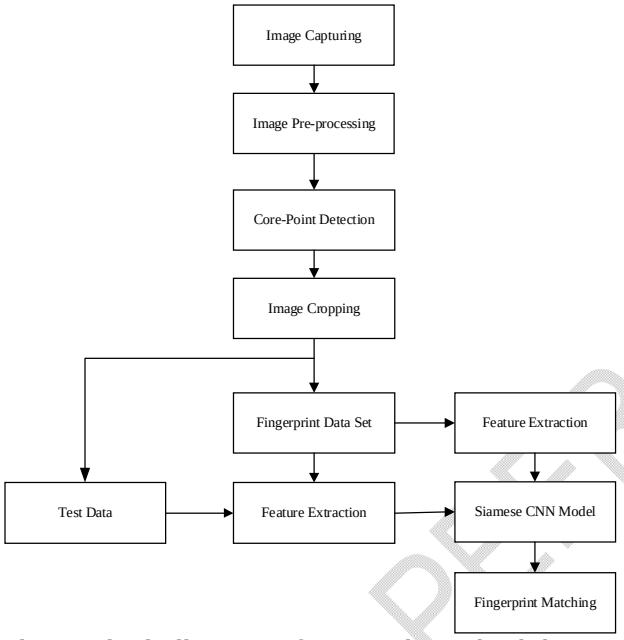

Fig. 1. Block diagram of research methodology

Eight hundred distinct fingerprint images were obtained for the initial CNN model training from the Touchless and Touch-Based Fingerprint Database at IIT Bombay to create a dependable model for the study. The 800 touchless fingerprint images in the collection, each with four samples and an image size of 170×260 are from 200 subjects. It also includes 800 260 × 330 touch-based fingerprint images of the same 200 people.

3.2Image Capturing

Image pre-processing in ML is a general term used for image recognition/formatting; that is, it is a term for operations on images at the lowest level of abstraction. It involves cleaning (filtering), resizing, orienting, and colour-correcting the raw images captured from the smartphone camera to make them recognizable by the ML algorithm before model training. Image background removal and duplicate image sorting are fundamental processes done at the stage of image filtering; this, among several reasons, is to majorly isolate noise from the captured image and sharpen the image quality. A typical architecture in computer vision with fully connected layers in Siamese CNN requires that all photos are the same-sized arrays to reduce the time required for processing. Contrast Limited Adaptive Histogram Equalization (CLAHE) is the suitable noise reduction method. It captures and improves the local edge information, which is critical to the fingerprint extraction process. The pre-processed image

(grey scale image) of Fig. 2 is shown in Fig. 3. The fingerprint image samples from the database are presented in Fig. 4.

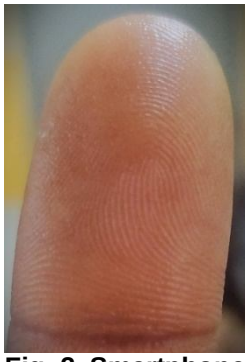

Fig. 2. Smartphone captured fingerprint for customized UI development

Fig. 3. Image after application of pre-processing (greyscale)

Fig. 4. Database sample images. Top row: touchless finger-prints and bottom row: corresponding touch-based images [12]

3.3Core Point Detection

Core point detection isolates a pre-processed fingerprint's upper core (central point) to simplify the identification of each whorl and loop pattern. Image segmentation makes core point detection possible (partitioning images into various regions and boundaries). The pre-processed image serves as the input for this procedure, and the resulting outputs are attributes associated with the provided photos. The RGB image is transformed into grayscale and then divided into non-overlapping tiles of identical sizes. Each tile of the

image undergoes histogram equalization independently in order to enhance contrast. This process involves expanding the intensity range in locations where pixels are densely grouped.

3.4Cropping

This process removes an unwanted subject or irrelevant detail from a photo, changes its aspect ratio, or improves the overall composition. Some background content irrelevant to the required data is removed from the image captured on the customized UI at this stage.

3.5Feature Extraction

A modified version of the Automated Fingerprint Identification System (AFIS) has been adopted to improve the process of extracting and comparing fingerprints. The objective of this exercise was to extract the precise and specific characteristics from fingerprint photos. In order to obtain the ridge-valley map from the fingerprint image, it was necessary to utilize adaptive mean thresholding (AMT) on the grayscale image. This technique separated the foreground pattern of interest and the background image by analyzing the disparity in pixel concentrations in each location. The SciKit Learn Python library facilitated the extraction of features in phases:

i. Terminations: These are the endpoints of the minutiae, characterized by the coordinates of the minutiae point $(LocX, LocY)$ and θ, which represents the angle of the ridge.
ii. Bifurcations: These are spots where a single ridge divides into two separate ridges. The related features include the coordinates of the minutiae point $(LocX, LocY)$, as well as the three angles of the ridges, referred to as θ_1, θ_2, and θ_3

3.6Convolutional Neural Network (CNN) Model

A Convolutional Neural Network (CNN) is a trainable deep neural network. For the IIT database, a cropped image of 227 × 227 pixels were utilized to extract fingerprint features, which were then used to train the CNN algorithm. This algorithm consists of a series of complex layers trailed by a pooling layer, with each layer designated for specific computations. The network design began with an input layer to define the dimensions and characteristics of the input data. The fully connected layer served as the output layer for data classification. The CNN model was created with the succeeding layers:

Convolutional layers: During this stage, the convolutional layer implemented convolutional filters to process the input images. Each image was subjected to a series of mathematical operations to produce a single value in the output feature map. These outputs were then passed through an activation function to bring non-linearities into the model. The rectified linear unit (ReLU) function, mathematically represented in Equation 1, was one of the activation functions adopted in this study.

$$f(x) = max(0, x) \quad \text{Equation 1}$$

where x is the neuron input.

Equation 2 gives a smooth approximation to the rectifier using an analytic function termed the soft plus function;

$$f(x) = ln(1 + e^x) \quad \text{Equation 2}$$

Pooling layers: The pooling layer is an essential component of the CNN architecture. The purpose of this layer is to condense the information contained in a feature map obtained from the convolution layer. This is achieved by applying either the average function or max-pooling, which helps to decrease the sizes of the feature map. As a result, the network had

to absorb a smaller number of parameters, which in turn lowered the amount of time required for computation. The max pooling technique selects subregions from the feature map and retains solitarily the highest worth inside each subregion.

Fully connected layers: The role of this layer is to categorize the characteristics that were retrieved by the convolutional layers and reduce their size through the pooling layers. Each node in this layer is linked to all activations in the previous layer.

Input layer: This layer accepts a pre-processed fingerprint image with dimensions of 227 × 227 $pixels$, where the width and height are both 227 $pixels$.

Two combinations of convolutional and pooling layers: The first convolutional layer used a padding value of 0 and a stride of 4, comprising 96 filters, each with dimensions of 11 × 11. In contrast, the second convolutional layer had 256 filters, each measuring 5 × 5, with a padding of 2 and a stride of 1. A rectifier function, ReLU, was then applied to both layers. Following each convolutional layer, a max-pooling layer with a window size of 3 × 3 and a stride of 2 was added.

Three convolutional layers and a pooling layer: The ReLU functions following the third, fourth, and fifth convolutional layers have 384,384, and 256 filters, respectively. After the three convolutional layers, a max pooling layer with a size of 3 × 3 and stride 2 was added.

Fully connected layer and an output layer: The first and second levels of the three completely connected layers each have 4096 neurons. The output layer, which is the third fully connected layer, was activated using a Softmax regression function. Each layer in a CNN generates a response, or activation, to an input image. Nevertheless, only certain layers within a CNN are effective for extracting image features. The initial layers capture rudimentary image elements, such as edges and blobs, which are then progressively analyzed by subsequent layers, combining these simpler features to form more complex image attributes at higher levels. Utilizing these higher-level features makes recognition tasks easier by merging all the basic components into a more comprehensive visual depiction. In the projected model, characteristics were extracted from the second fully connected layer.

3.7Fingerprint Image Testing and Evaluation

To improve the accuracy and sensitivity of the trained CNN model, additional database from Sokoto Coventry Fingerprint Dataset (SOCOFing) consisting of 55,270 images was used alongside that of IIT India. The SOCOFing fingerprint dataset is divided into Real (6000) and Altered (49,270) datasets. The Altered dataset is further divided into three tiers: Easy (17,931), Medium (17,067) and Hard (14,272) in relation to the level of alteration done to the Real fingerprint images [17]. The goal of the fingerprint matching test and evaluation was to match each 'Altered' fingerprint image to the corresponding 'Real' fingerprint image irrespective of the level of alteration done to the image. The different levels of alteration to the image are illustrated in Fig. 5. Fig. 5 (a) displays the 'Real' 150__M_Right_index_finger_Obl.BMP image while Fig. 5 (b), (c), and (d) are the 'Altered-Easy,' 'Altered-Medium' and 'Altered-Hard' versions of the same image.

Fig. 5: Right index finger (a) real (b) altered-easy (c) altered-medium (d) altered-hard [13]

3.8Android Application for Fingerprint Image Matching and Identification

A simplified approach to code script conversion from Python file (.py) to Android APK (.apk) was chosen for the study. This was done with the main consideration being conversion platform requirements; the platform of choice was Kivy and Buildozer.

3.8.1 Prerequisite

Before initiating the conversion process, the following prerequisites were observed:

i. **Python:** Python was installed on the local computer being used.
ii. **Kivy Framework:**Kivy is a Python framework for creating multi-touch applications, and notably, one of the principal tools used for the Python script translation using the pip command as follows:
```
!pip install kivy
```
iii. **Android Development Environment:** An Android development environment was established on the local computer, together with Android Studio, the Android SDK, and either an Android emulator or a physical device for testing.
iv. **Buildozer:**Buildozer is a build tool used to package Python applications as Android APKs. This was installed and conFig.d on the development machine.

3.8.2Procedure for Android Application Development

The following steps were undertaken to develop the proposed Android application:

i. Develop or modify a Python script to function in the Android environment. Certify the Python code is well-suited with mobile devices and the Kivy framework, which is specifically intended for multi-touch applications.
ii. Set up Buildozer. To set up Buildozer, these stages were followed:
Install Buildozer by means of pip command thus:
```
!pip install buildozer
```
iii. Make new project directory thus:
```
mkdirphonebased_fingerprintid_android_app
cd phonebased_fingerprintid_android_app
```
iv. InitialiseBuildozer inside the new file path (folder):
```
buildozerinit
```
v. The synthax in (iv) produces a build.spec file, which was conFig.d to stipulate the build settings for the Android app thus:
title: Fingerprint Image Matcher
package.name:com.finprintid.github.fl
source.dir:phonebased_fingerprintid_android_app
requirements: python3, kivy==2.0.0, kivymd, pillow
android.permissions: internet access, camera, and file manager.
version: 1.0.
android.sdk: Android 10 (API level 29).
p4a.branch: Python for Android with Android app bundle support.

vi. Build the APK using:

```
buildozer android debug deploy run
```

4. RESULTS AND DISCUSSION

RESULT

This section presents the results obtained in this study. The Python Language scripts were used for fingerprint image pre-processing (sorting in rows and columns based on sex, finger and degree of image distortion), model training and evaluation. Where suitable, image plots and tables of pre-processed and processed fingerprints are presented for easy reference, illustration and inference.

4.1 Core Point Detection, Orientation Estimation and ROI Extraction

As earlier stated, once a finger is detected, the ROI has to be extracted with consequent normalization to a proper width, height, and resolution. The image captured with a smartphone camera presented in Fig. 2 was pre-processed, cropped, processed, and passed through a fully connected layer of the CNN model. All minutia features from the ROI of the fingerprint were extracted and stored using the Python script. A screenshot of the extraction process in Google Colab is shown in Fig. 6. The same process was applied to all the downloaded fingerprint images contained in the IIT and SOCOFing databases. Fig. 7 presents a sample of pre-processed image with an adaptive mean threshold showing enhanced fingerprint features (ROI). According to [18] and [19], approximately 40-100 minutiae are detected in a typical good-quality fingerprint, while a partial or poor-quality fingerprint has approximately 20-30 minutiae. Thus, image enhancement is a precursor to an efficient touchless fingerprint identification system. Basic enhancements such as increased contrast and brightness constitute the majority of the pre-processing procedure. Elaborate pre-processing workflows were required to compensate for the inadequate proficiencies of built-in cameras and the environmental side effects of image capturing especially fingers.

4.2 Core Point Detection, Orientation Estimation and ROI Extraction

The first step in the fingerprint image matching and evaluation adopted in this study was to use an unsupervised ML classifier (kNN) along with a Flann-based algorithm for testing. To proceed with the image matching, all images in the database were resized to the same dimension (96 × 103). The fingerprint-matching procedure was achieved with the aid of a Python programming language script. In the process of testing, identification was initiated, and an accuracy score of 100% was achieved by the classifier, as portrayed in Fig. 8. After the identification test, matching was done with some pixels of the image of Fig. 6 were removed to connote a dirty finger. The output of the kNN-matched fingerprint is shown in Fig. 9. From Fig. 9, some of the minutia features of the original fingerprint image are mapped to the altered fingerprint image, as illustrated by the connections and the various circles, with a matching score of 64%.

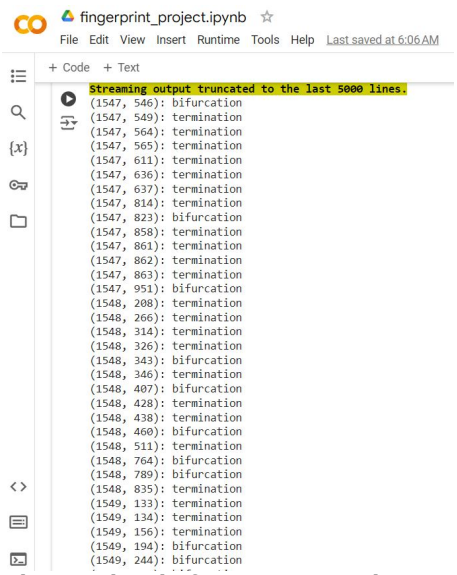

Fig. 6. Minutia feature extraction process

Fig. 7. Fingerprint image pre-processed with adaptive-mean-threshold

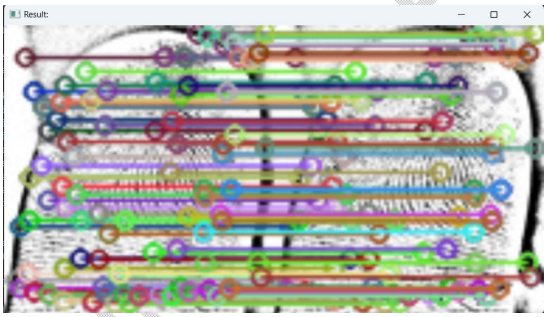

Fig. 8. Matched fingerprint image using kNN (no distortion)

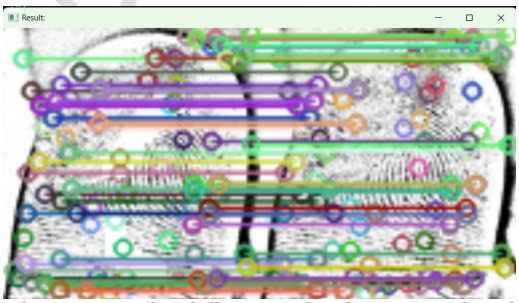

Fig. 9. Matched fingerprint image using kNN (with distortion)

4.3Training and Evaluation of Fingerprint Images using CNN Models

In the case of the DL network, the database was handled in three phases: pre-processing, training and testing. For the pre-processing phase, the label attributes of the 55,270 images in the SOCOFing database were extracted using the 'numpy' python package and stored as a dataset. The dataset created was the input used for the model training. Four CNN models adopted for comparison are the Visual Geometry Group models (VGG16 and VGG19), Residual Network (ResNet50), and Inception-V3 model. The VGG16 CNN model used in the study for fingerprint classification comprised 22 layers, featuring 5 convolutional layers, 3 pooling layers, 3 fully connected layers, and a dropout layer. The final fully connected layer's output was directed to a 5-way softmax classifier that categorizes the input into one of the 5 labels. A fingerprint's global ridge and furrow feature identify its category. This global information needed to be efficiently captured via a feature set that is considered valid for fingerprint categorization. The total number of defined parameters was 8153089, the number of trainable parameters was 8150913, and number of non-trainable parameters was 2176. Fig. 10 shows a set of randomly matched images generated during training from the database with different alterations and image qualities.

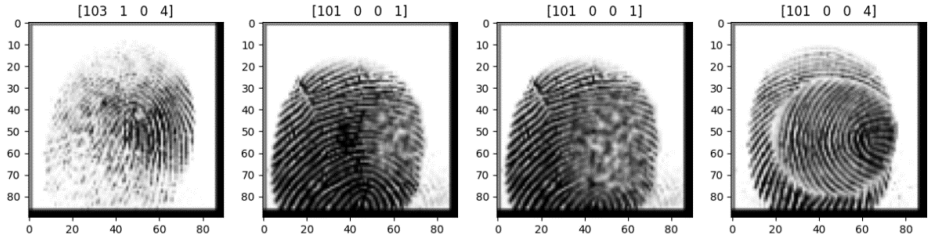

Fig. 10. Randomly matched fingerprints (a) Real (b) Easy (c) Medium (d) Hard

Table 1. Performance evaluation based on NIIT dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16	90	90	91	90
VGG19	92	92	91	92
ResNet50	81	78	84	80
Inception-V3	84	81	85	85

Table 2. Performance evaluation based on SOCOFing dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16	92	92	92	91
VGG19	96	95	96	96
ResNet50	83	84	87	83
Inception-V3	88	86	88	87

Due to their flexibility, the VGG models were further trained with modified parameters in line with the proposal made by [20]. The simulated results for the VGG16 and VGG19 models, both with and without the inversion and multi-augmentation methods, are encapsulated in Table 3.

The VGG16 model achieved a 92% accuracy without parameter augmentation. However, accuracy increased to 93% with the inversion approach and further to 98% with multi-augmentation. In contrast, the VGG19 model also reached 92% accuracy without any modifications. With the inversion approach, accuracy decreased to 89%, and the augmentation approach did not result in any improvement, keeping the accuracy at 93%.

A recognizable feature consistent among the altered images in Fig. 10 is their identical alignment along the y- and x-axis compared to the real image with a precision accuracy of 92%. The details of the data training and evaluation are tabulated in Tables 1 and 2. Table 1 gives a summary of the performance evaluation based on the NIIT dataset, while Table 2 gives a similar comparison based on the SOCOFing dataset.

Table 3. Performance evaluation based on inversion and multi-augmentation

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16 (Normal)	92	92	92	91
VGG16 (Inversion)	93	93	93	93
VGG16 (Multi-augmented)	98	98	98	98
VGG19 (Normal)	92	91	92	92
VGG19 (Inversion)	89	89	89	88
VGG16 (Multi-augmented)	93	93	93	93

Evaluation of the processed smartphone-captured image (see Fig. 2) using CNN models revealed that inception-V3 produced the highest percentage of the metrics (94%). The ResNet50 produced the lowest performance of the bunch (86%), as tabulated in Table 4.

Table 4. Performance evaluation based on SOCOFing dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16	91	91	90	91
VGG19	92	92	92	92
ResNet50	86	86	86	86
Inception-V3	94	94	94	93

All codes and models trained in this research were converted to Android APK for easy deployment in smartphones with the aid of kivy module (see Section 3.4). Testing and evaluation of converted Android APK was done in Android Studio.

Fig. 11 is a pie-chart comparative representation of the percentage accuracy of each model considered in the study.

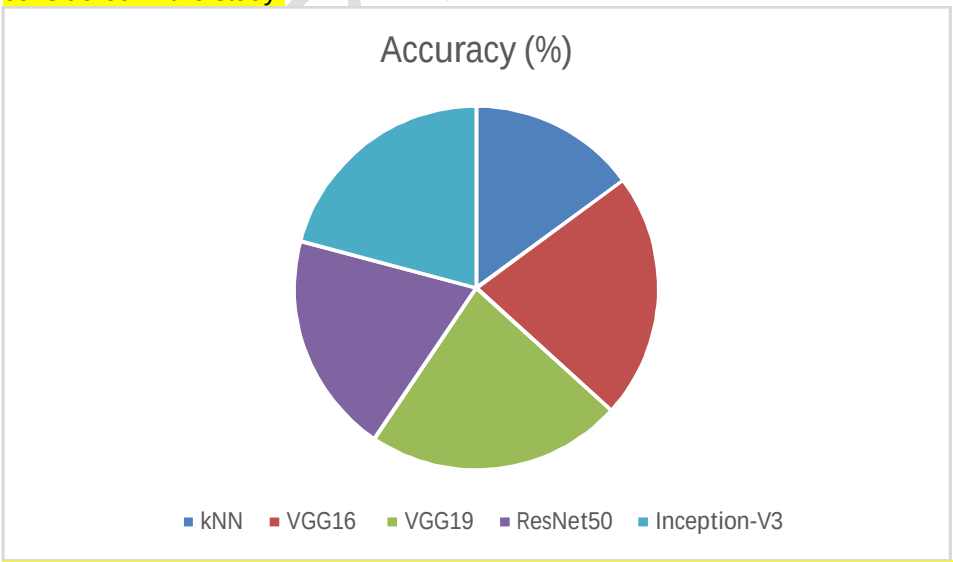

Fig. 11. Percentage accuracy comparison of different models based on SOCOFing dataset

4.4 Discussion

The concept of touchless fingerprint recognition as an alternative to touch-based fingerprint identification was presented in this study.

4.4.1 Data Pre-processing

Data pre-processing was a key step in this study, for which MATLAB was used for image labelling. Python was adopted as the language of choice based on the ease of assessing ML modules. The kNN algorithm was employed to handle missing values, outliers, and noise within the fingerprint datasets. This pre-processing ensured that the images fed into the CNN model were of high quality and standardized, thus improving the model's robustness and accuracy.

4.4.2 Model Architecture and Training

The CNN architecture was designed to effectively capture the intricate patterns and minutiae within the fingerprint images. The model consisted of multiple convolutional layers, each followed by pooling layers, and concluded with fully connected layers. This architecture enabled the extraction of both low-level and high-level features from the fingerprint images.

The model was trained using a combination of NIIT and SOCOFing fingerprint images, which provided a diverse and comprehensive dataset. Data augmentation techniques were applied to enhance the dataset further and prevent overfitting. The training process involved optimizing the model parameters using the Adam optimizer and minimizing the categorical cross-entropy loss.

4.4.3 Performance Evaluation

A fingerprint image captured from a smartphone camera was initially pre-processed using adaptive mean thresholding. An unsupervised machine learning algorithm (kNN) was then employed for fingerprint matching and evaluation, achieving 100% precision for processed unaltered images in a dataset of 50, and a matching precision of 63% for altered images. To enhance the model's accuracy, images from the NIIT and SOCOFing databases were utilized. Four CNN models (VGG16, VGG19, ResNet50, and Inception-V3) were trained and evaluated on various key metrics, as detailed in Tables 1 to 4.
The model proposed by [14] is subpar to the one proposed in this study due to the flexibility and adaptability of the proposed model. Similarly, the models presented by[10], [12], [13] showed 76%, 86%, and 91% accuracy using the distorted smartphone image against 94% showed by the trained Inception-V3 CNN model evaluated in the study.

5. CONCLUSION

The study successfully demonstrated the superior efficacy of Convolutional Neural Networks (CNNs) in fingerprint identification, significantly outperforming traditional k-Nearest Neighbors (kNN) methods. These findings highlight the potential of advanced machine learning techniques to develop more robust and accurate biometric identification systems. A fingerprint image captured from a smartphone camera was pre-processed using adaptive mean thresholding, and an unsupervised kNN algorithm was employed for initial fingerprint matching and evaluation. While kNN achieved 100% precision for unaltered images and 63% for altered images, it fell short in performance compared to CNNs. The use of images from the NIIT and SOCOFing databases further enhanced the model's accuracy. Four CNN

architectures—VGG16, VGG19, ResNet50, and Inception-V3—were trained and evaluated across key performance metrics, demonstrating CNN's superior ability to capture complex image patterns. While kNN proved useful in data pre-processing, CNNs' ability to handle intricate visual details underscores their advantage in fingerprint identification, marking them as the more effective tool for such tasks.

Disclaimer (Artificial intelligence)

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

REFERENCES

[1] S. Blau, J. Graham, L. Smythe, and S. Rowbotham, "Human identification: a review of methods employed within an Australian coronial death investigation system," Int. J. Legal Med., vol. 135, no. 1, pp. 375–385, 2021, doi: 10.1007/s00414-020-02461-3.

[2] Z. Cui, J. Feng, and J. Zhou, "Dense Registration and Mosaicking of Fingerprints by Training na End-to-End Network," IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 627–642, 2020.

[3] A. Holobinko, "Forensic human identification in the United States and Canada: A review of the law, admissible techniques, and the legal implications of their application in forensic cases," Forensic Sci. Int., vol. 222, no. 1–3, pp. 394.e1-394.e13, 2012, doi: 10.1016/j.forsciint.2012.06.001.

[4] T. Thompson, S. Black, and (Eds.), Forensic Human Identification. An Introduction. Boca Raton, FL: CRC Press, 2007.

[5] B. H. Situmorang and D. Andrea, "Identification of Biometrics Using Fingerprint Minutiae Extraction Based on Crossing Number Method," J. Ilm. Ilmu Komput. dan Matermatika, vol. 20, pp. 71–80, 2023.

[6] F. Saeed, M. Hussain, and H. A. Aboalsamh, "Automatic Fingerprint Classification Using Deep Learning Technology (DeepFKTNet)," mathematics, vol. 10, no. 1285, 2022.

[7] J. Bae, H. S. Choi, S. Kim, and M. Kang, "Fingerprint image denoising and inpainting using convolutional neural network," J. Korean Soc. Ind. Appl. Math., vol. 24, pp. 363–374, 2020.

[8] R. Triggs, "How fingerprint scanners work — Optical, capacitive, and other variants," www.androidauthority.com, 2021. https://www.androidauthority.com/how-fingerprint-scanners-work-670934/ (accessed May 12, 2022).

[9] P. Komarinski, P. T. Higgins, K. M. Higgins, and L. K. Fox, Automated Fingerprint Identification Systems (AFIS). Burlington, MA: Elsevier Academic Press, 2005.

[10] Nur-A-Alam, M. Ahsan, M. A. Based, J. Haider, and M. Kowalski, "An intelligent system for automatic fingerprint identification using feature fusion by Gabor filter and deep learning," Comput. Electr. Eng., vol. 95, no. February, p. 107387, 2021, doi: 10.1016/j.compeleceng.2021.107387.

[11] V. M. Praseetha, S. Bayezeed, and S. Vadivel, "Secure fingerprint authentication using deep learning and minutiae verification," J. Intell. Syst., vol. 29, no. 1, pp. 1379–1387, 2020, doi: 10.1515/jisys-2018-0289.

[12] P. Birajadar et al., "Towards smartphone-based touchless fingerprint recognition," Sadhana - Acad. Proc. Eng. Sci., vol. 44, no. 7, 2019, doi: 10.1007/s12046-019-1138-5.

[13] A. Attrish, N. Bharat, V. Anand, and V. Kanhangad, "A Contactless Fingerprint

Recognition System," pp. 1–8, 2021, [Online]. Available: http://arxiv.org/abs/2108.09048
[14] B. C. Seow, S. K. Yeoh, S. L. Lai, and N. A. Abu, "Image based fingerprint verification," 2002 Student Conf. Res. Dev. Glob. Res. Dev. Electr. Electron. Eng. SCOReD 2002 - Proc., no. May, pp. 58–61, 2002, doi: 10.1109/SCORED.2002.1033054.
[15] Z. Mar Win and M. Myint Sein, "An Efficient Fingerprint Matching System for Low Quality Images," Int. J. Comput. Appl., vol. 26, no. 4, pp. 5–12, 2011, doi: 10.5120/3094-4246.
[16] P. Birajadar et al., "Indian Institute of Technology, Bombay, Touchless and Touch Based Fingerprint Database," 2019. https://www.ee.iitb.ac.in/~dsplab/Biometrics/Touchless_Database.html (accessed May 14, 2024).
[17] Y. I. Shehu, A. Ruiz-Garcia, V. Palade, and A. James, "Sokoto Coventry Fingerprint Dataset," 2018, Accessed: Jun. 15, 2024. [Online]. Available: https://www.kaggle.com/datasets/ruizgara/socofing
[18] H. M. Daluz, Fundamentals of Fingerprint Analysis, 2nd ed. Boca Raton, FL: CRC Press, 2019.
[19] B. Zhou, C. Han, Y. Liu, T. Guo, and J. Qin, "Fast minutiae extractor using neural network," Pattern Recognit., vol. 103, p. 107273, 2020.
[20] R. Garg, G. Singh, A. Singh, and M. P. Singh, "Fingerprint recognition using convolution neural network with inversion and augmented techniques," Syst. Soft Comput., vol. 6, no. November 2023, p. 200106, 2024, doi: 10.1016/j.sasc.2024.200106.