Fingerprint Recognition System

Synopsis Report submitted in partial fulfillment

of the requirement for the degree of

B. E. (Computer Engineering)

Submitted By

SARVESH GAONKADKAR

OM GHAG

TEJAS KUMBHAR

ANURAG KUMBHARE

Under the Guidance of

Prof. SNEHA ANNAPPANAVAR

Department of Computer Engineering

VIT | Vidyalankar Institute of Technology

ACCREDITED A+ BY NAAC

(An Autonomous Institute Affiliated to University of Mumbai)

Vidyalankar Institute of Technology

Wadala(E), Mumbai 400 037

University of Mumbai

2024-25

CERTIFICATE OF APPROVAL

For

Project Synopsis

This is to Certify that

SARVESH GAONKADKAR

OM GHAG

TEJAS KUMBHAR

ANURAG KUMBHARE

Have successfully carried out Project Synopsis work entitled

Fingerprint Recognition System

in partial fulfillment of degree course in

Computer Engineering

As laid down by University of Mumbai during the academic year

2024-25

Under the Guidance of

Prof. SNEHA ANNAPPANAVAR

Signature of Guide	Head of Department

Examiner 1	Examiner 2	Principal

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Name of student	Roll No.	Signature
1.	Sarvesh Gaonkadkar	21102B0001	
2.	Om Ghag	21102B0002	
3.	Tejas Kumbhar	21102B0004	
4.	Anurag Kumbhare	21102B0005	

Date:

Acknowledgements

This Project wouldn’t have been possible without the support, assistance, and guidance of a number of people whom we would like to express our gratitude to. First, we would like to convey our gratitude and regards to our mentor Prof. Sneha Annappanavar for guiding us with her constructive and valuable feedback and for her time and efforts. It was a great privilege to work and study under her guidance.

We would like to extend our heartfelt thanks to our Head of Department, Dr. Sachin Bojewar, for overseeing this initiative which will in turn provide every Vidyalankar student a distinctive competitive edge over others.

We appreciate everyone who spared time from their busy schedules and participated in the survey. Lastly, we are extremely grateful to all those who have contributed and shared their useful insights throughout the entire process and helped us acquire the right direction during this research project.

Table of Contents

1. Abstract

Since the advancements in the deep learning and computer vision, Convolutional Neural Networks (CNNs), have become a key framework for image related tasks including fingerprint recognition system. On one hand, fingerprints have been matched with traditional techniques such as Scale Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB) along with CNNs. However, previous studies have demonstrated that CNNs are not well suited to working with large and complex datasets. SIFT, although accurate, is computationally expensive whereas ORB is fast, but lacks precision.

For fingerprint recognition tasks of fingerprint matching, Siamese Neural Networks (SNNs) have been employed to overcome these challenges. SNNs are unlike traditional methods, which learn the similarity among input fingerprints, and hence are very good for verification and matching tasks. Fingerprint pattern recognition is shown to be improved with SNNs, as they are capable of distinguishing genuine from false fingerprints even when there is noise or partial prints. For mobile security systems with limited computational resources, high efficiency is of utmost importance.

Fingerprint recognition systems based on SNNs can provide the advantages of the pairwise comparison rather than global classification, and more accurate and reliable fingerprint matching. While SNNs have been designed for local feature extraction and matching, their lightweight architecture enables them to be deployed on mobile security applications, which demand resource efficiency.

In this paper, the application of SNNs in mobile fingerprint recognition systems is evaluated, from a comprehensive point of view, in order to understand the mobile security context in which SNNs can be applied. Results indicate that for real time biometric authentication on mobile devices, SNNs offer an excellent balance between accuracy and computational efficiency and are thus a good solution. Through this study, the potential of SNNs to improve the performance and security of mobile based fingerprint recognition systems is validated, at a time when there is growing demand for secure and efficient mobile biometric solutions.

2. Introduction

Fingerprint recognition systems have largely relied on Convolutional Neural Networks (CNNs) for feature extraction as well as pattern recognition tasks. Strong performance across multiple biometric applications has been made possible due to their ability to capture local spatial hierarchies within images. CNNs, however, suffer in computational efficiency and the handling of global contextual information — both of which are important in mobile security environments with limited resources.

In order to tackle these problems, fingerprint recognition employs the techniques of Siamese Neural Networks (SNNs) in mobile security applications. Using a twin network architecture, which learns similarity metrics between two inputs, SNNs are particularly well suited to tasks like fingerprint matching and verification. SNNs can compare pairs of fingerprint images to check if the produced images are from the same individual, with the additional advantage of having greater accuracy and robustness in a biometric environment that poses a challenge.

As SNNs are computationally efficient and naturally focused on pairwise comparisons, they are especially advantageous for mobile security, where fingerprint verification requires such comparisons. Unlike traditional CNNs, SNNs are specifically designed to compare individual image pairs, and results indicate that they are robust to fingerprints that are noisy or partially captured. SNNs also have the appropriate resource constrained nature of mobile platforms, where computational demands can be balanced with accuracy.

This research is the demonstration of how the architecture of SNNs addresses the unique challenges of mobile security and apply them to mobile fingerprint recognition systems. The combination of high accuracy and speed makes SNNs ideal for mobile device biometric authentication systems, allowing for fingerprint recognition systems to take advantage of SNNs to achieve both.

3. Aim and Objectives

Aim:

This project aims to design and implement an efficient scalable and secure Siamese Neural Network (SNN) based fingerprint recognition system designed for mobile devices. The application involves real time fingerprint matching for user authentication in order to minimize computational overhead and enhance data security. It is essential as personal and financial transactions are increasingly moving to mobile devices, and fingerprint recognition is becoming a robust biometric authentication system.

Key system aspects include:

- Efficiency: For low latency and resource constrained mobile environments.
- Accuracy: Fingerprinting that is reliable to distinguish between different fingerprints.
- Security: Fingerprint protection during profile registration.

An SNN is used in the project that can compute similarity score between fingerprints images to do correct identification. Stored biometric data will be protected by AES encryption, and TFLite will optimize the model for mobile performance.

Objectives:

- Develop an SNN model: For fingerprint recognition, to create and train a Siamese Neural Network addressing challenges like lighting, rotation, and partial fingerprints.
- Ensure data security through AES encryption: To store fingerprint data based on AES encryption to control the unauthorized access.
- Optimize the model for mobile platforms using TensorFlow Lite: To convert and optimize the SNN model for efficient mobile deployment using model quantization and GPU delegation.
- Integrate augmentation techniques: It is the focus to augment training with the aim of improving the model’s robustness to variations in the images of fingerprints.
- Enable real-time fingerprint matching: To create the system for quick comparisons on the platform using mobile hardware resources in order to authenticate users seamlessly.
- Deploy the system on mobile devices: Install the fingerprint recognition system as a mobile application where the users can enrol and identify fingerprints securely.

4. Literature Survey

Fingerprint recognition systems have evolved from conventional Convolutional Neural Networks (CNNs) to more complex Siamese Neural Networks (SNNs). The purpose of these advances is to drive biometric systems to higher accuracy, greater efficiency, and increased robustness. In this review, an overview of key SNNs and CNNs developments is provided, and their application to fingerprint recognition in mobile applications.

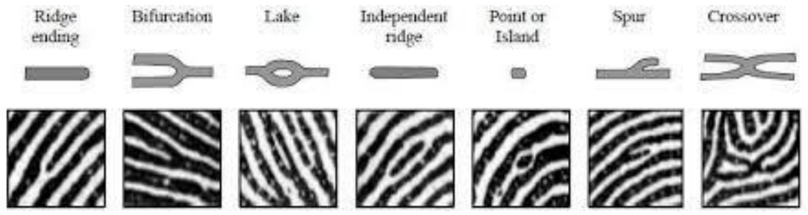
Fingerprints: Fingerprint patterns are probably the most relied upon and widely used biometric identifiers for security purposes due to their uniquely unalterable patterns and ease of acquisition. The fingerprints of a person are unique compared to any other identical twins, as these patterns develop in the fetal stages of human development and remain invariant across a lifetime. Due to the inherent distinctiveness and permanence, fingerprints prove to be a good metric for identity verification and security applications.

Why Fingerprint Matching as a Security Measure: Fingerprint security is a reliable biometric identification method, owing to the uniqueness and permanence of fingerprints, as well as their ease of capture. Some of the main reasons why they suit secure systems are:

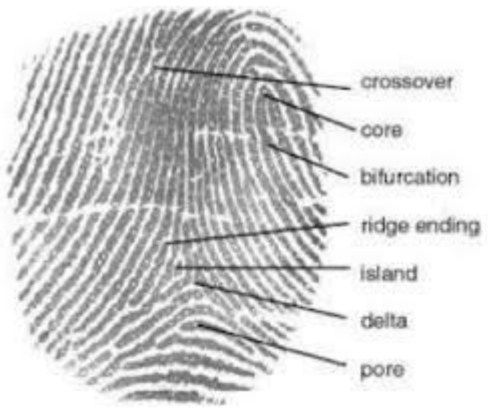
- Uniqueness: Different ridge patterns along with the different bifurcations of human beings create a unique fingerprint for no two human beings; therefore, it automatically makes fingerprints very reliable to identification systems.
- Immutable: Fingerprints are immutable during one's lifetime unlike facial features or voice patterns. Once printed, they do not need to be updated like other biometrics. This makes them extremely reliable over time in security systems; enrolled fingerprints cannot be deleted and remain valid for identity verification over time.
- Universality: Fingerprint identification is quite extensively used and very easy to implement for any population. It is also widely available for use in all fields, be it financial or law enforcement agencies as well as on the handsets of mobile devices.
- User-Friendly: Fingerprint collection is quick and not intrusive, using commonly available devices like optical sensors or scanners. This authentication method is practical and secure to users and organizations since risks of unauthorized access do not exist, as fingerprints cannot be guessed, hacked, or shared like passwords.

Fingerprint Matching Key Features: Appropriately chosen features from fingerprints are extracted and form a biometric template to enable comparison and matching for authentication or identification purposes in fingerprint security systems.

- Ridge Patterns: Fingerprints contain ridge patterns in loops, whorls, and arches, mainly due to their usage for classification and basic differentiation.
- Minutiae Points: The minutiae points, including ridge endings and bifurcations, primarily help systems identify and match partially captured fingerprints.



- Ridge count and orientation: In this feature, the ridges are counted at the minutiae points for accurate matching in fingerprints. The orientation of ridges is compared for better results. The core is the center of a fingerprint, where the ridges form loops or spirals, while the delta is formed by ridges, which split in a triangular fashion. Such reference points will guide the alignment of the fingerprint images to be compared, especially if the images have been rotated or distorted.



- Pores and Sweat Glands: Other fingerprint systems account for microscopic sweat gland pores, which strengthens security and increases accuracy in some applications.

Advantages of Fingerprint Authentication Over the Traditional Security Systems: Fingerprint security has many key advantages over old methods like passwords, PINs, and other biometrics. It improves security by:

- Comparing with Other Biometrics for Resistance to Attacks: Facial recognition is easily compromised using photos or videos. Voice recognition can easily be compromised using a recording, whereas fingerprint systems are quite stronger since there is a mechanism for liveness detection.

– Multi-dimensional Security Features: Fingerprint authentication utilizes more details like minutiae points and ridge patterns, compared to passwords, which rely entirely on a one-dimensional input. Therefore, it provides higher reliability on multiple dimensions as compared to a one-dimensional password or PIN.
– Higher Entropy Compared to Passwords: Entropy calculates the complexity of security elements. Low entropy corresponds to weak passwords and is easy prey to brute-forcing attacks. In contrast, fingerprints correspond to much higher entropy because of the complex and unique patterns they represent. Thus, predicting or guessing a fingerprint is almost impossible.
– Low Effort-High Compliance: People opt for easy passwords. The reason is convenience. Moreover, people forget to use advanced security, such as two-factor authentication. Fingerprint Authentication is safe and easy to handle. Only a quick finger scan can do the trick to increase compliance and ensure minimum weak security issues.
– Local matching for better privacy: New devices such as smartphones process fingerprint data internally that prevents exposing associated biometric information to any other device. Encryption, illustrated in systems such as Apple's Touch ID and Google's mechanism, makes up for this deficiency.

Convolutional Neural Networks (CNNs): CNNs have become centre of advancements in image recognition, from fingerprint recognition to image recognition, because they are capable of recognizing spatial patterns and hierarchies of images. Major contributions in this area are:

- AlexNet (2012): The AlexNet was introduced in the world of image classification by Krizhevsky et al, that brought a lot of improvements using deep layers, ReLU and drop out. In large scale image datasets, it proved successful, and it has established itself as an important milestone in CNN development, beyond biometric tasks such as fingerprint recognition.

Despite the success of CNNs in feature extraction, they remain constrained in the pairwise comparison tasks that are key to fingerprint matching and are consequently employed for fingerprint matching using Siamese Neural Networks (SNNs).

Siamese Neural Networks (SNNs): The advantage of SNNs is that they are specially made for tasks where there is a need to measure the similarity between two inputs, as in biometric matching systems such as fingerprint recognition.

- Fingerprint Recognition with SNNs (2017): SNNs have been used on fingerprint recognition tasks where the model learns to compare two fingerprint images and learn a similarity function. A robust matching of fingerprints is provided by this method, in the case where the prints are noisy or incomplete.

The selection of SNNs' architecture through which pairwise comparison is performed is particularly well suited for mobile fingerprint recognition systems due to the fact that new fingerprints can be compared against stored biometric templates in an efficient manner.

Application of SNNs in Mobile Fingerprint Matching: As compared to conventional SNNs, implementation of SNNs to fingerprint recognition in mobile devices presents several advantages, such as low computational overhead and high accuracy.

- Pairwise Matching: The twin architecture of SNNs allows the system to achieve high accuracy in matching tasks by comparing pairs of fingerprints, by attenuating the differences between fingerprint images.
- Efficiency: A variety of SNNs are optimized for mobile platforms, which provides efficient performance without sacrificing accuracy, an important property for mobile biometric systems.
- Robustness: The advantage of SNNs is that they are able to handle noisy or partial fingerprint data, increasing system robustness in real world mobile applications where fingerprint scans may not be the best, and making it easier to design for such situations.

5. Problem Statement

Problem Statement:

As reliance on mobile devices increases for personal and work, authentication systems are becoming more important. Password based authentication has been traditionally the standard form to control who can access a resource, however with the ever-growing risk of password breach, phishing attacks and poor password management, traditional password-based authentication isn't good enough anymore. Fingerprint recognition as an alternative appears to offer improved security and convenience. Yet, fingerprint recognition becomes a viable solution on mobile devices but faces with several difficult problems.

By design, mobile devices have limited computational power and memory, and therefore it is hard to efficiently implement resource heavy fingerprint recognition algorithms. The overall speed and practicality of fingerprint recognition systems are limited by this. There is also a large problem of accuracy, because fingerprints are affected by various conditions like partial prints or environmental factors. This can cause system reliability problems in which variations can allow unauthorized access or block legitimate users.

The problem of the handing of biometric data is an additional concern. It is not merely, unlike passwords, that one can be easily changed out after a security breach, but consequences can remain a problem with long-lasting consequences. As well, fingerprint recognition systems are difficult to integrate consistently and securely across all the hardware configurations of mobile devices. Together, these factors indicate the need to address the core problems of efficiency, accuracy and data security in mobile fingerprint recognition systems.

6. Scope

The focus of this work is building and deploying a Siamese Neural Network (SNN) based fingerprint recognition system that is secure, efficient, and accurate for mobile devices. It covers several important areas like:

- Development of the SNN Model for Fingerprint Recognition: It will be the focus of the project to build a strong Siamese Neural Network that can accurately compare fingerprint images. Different augmentation techniques are used to train the model to handle variations in fingerprint presentation like difference in orientation, pressure as well as partial captures. It is to make sure the system is still able to separate normal users from those allowed.
- Security of Fingerprint Data through AES Encryption: The system will ensure that encrypted biometric data is used and that it will be implemented during the profile registration process using AES. This is this critical measure that protects user's fingerprint data from unauthorized access and if data storage is compromised then also biometric information does not come out from the safe unless proper decryption key is not present.
- Optimization for Mobile Devices using TensorFlow Lite: Since the resource constraint of mobile platforms, the project will make use of TensorFlow Lite to convert and optimize the SNN model to reduce the time consumption in mobile devices. The focus will be on reducing the computational overhead and making the system work smoothly within the limited resources such as processing power and memory provided by such devices as smartphones and tablets.
- Real-Time Fingerprint Matching for User Authentication: A real time fingerprint matching system will be designed for real time fingerprint matching with a requirement of low latency to provide a quick and smooth user authentication. The model provides a means to use mobile hardware resources effectively and perform fast matching between the stored fingerprint and the one during login attempts for usability.
- Integration with Real-World Mobile Applications: The last objective is to deploy the fingerprint recognition system on Android devices. Then the system will be integrated with mobile app, where users can register their fingerprints securely, store them safely and use them for authentication during actions like unlocking their device or authorising transactions.
- Augmentation and Robustness: During the training phase, multiple image augmentation techniques will be used to increase robustness to the system. The model generalizes better to real world conditions, and it can effectively handle variations in the fingerprint quality and presentation.

7. Proposed System

For fingerprint recognition in mobile security, the proposed system implements Siamese Neural Networks (SNNs). In contrast to standard biometric recognition techniques, the SNN architecture is specially developed to perform pairwise comparisons, thus being a good candidate for fingerprint matching. The goal of this system is to produce a highly efficient and accurate fingerprint recognition model that can be deployed on mobile devices and run in real-time with secure data processing.

System Overview:

An SNN is used as the core of the system to compare two fingerprint images to identify their similarity. With architecture optimized for small/mobile footprint, great computational overhead is minimized, and maximum battery is attained. The design consists of a chain of convolutional layers which extract salient features from input fingerprint images, and then dense layers that map each image to a compact feature embedding. These embeddings are used for comparing fingerprint during authentication. The user takes their fingerprint at the time of initial profile registration, then securely stores It on their mobile device and uses it to generate the user's biometric profile. In this process, the background adds fingerprint augmentation to the fingerprint image (rotations, translations, and flips etc.) to train the model and make it generalize. In this phase, the fingerprint data are stored secured by AES encryption so that it is secure from sensitive information while being stored. During login attempts, the user's input fingerprint is then compared to the stored fingerprint data. After this initial phase the system processes fingerprint inputs in real time without storing further fingerprint data. This is done to increase security because encryption is handled by the mobile device's operating system (OS) and sensitive data is stored.

Key Components:

- Siamese Neural Network Architecture: The twin network architecture is an architecture to compare two inputs and decide their similarity. The network is split into two identical branches, and each of them processes one of the input fingerprints. These branches are shared so that each fingerprint goes through the same transformation, simplifying learning a similarity metric.
- Fingerprint Matching: The network is fed with two fingerprint images as inputs, outputs feature embeddings for each, and then the system uses these features to generate features for each fingerprint image. By taking these embeddings and comparing them using a similarity metric

(such as cosine similarity), to decide if the fingerprints match or not, and then set a threshold that verify if fingerprints belong to the same individual.

- Fingerprint Storage and AES Encryption: On initial registration, fingerprints are securely stored on the device, protected with AES encryption, to protect biometric data. The fingerprint images are augmented in the background using data augmentation techniques (e.g. flipping, rotation, and translation) to improve training of model, and to make the model robust to variations in fingerprint input.
- Real-time Processing: After initial registration, the system operates for real time processing without storing new fingerprint data. The SNN model is placed into TensorFlow Lite (TFLite) format, fastening the model for mobile devices. By the use of frequency and power scaling, fingerprint matching can be achieved in milliseconds, and the system can be made responsive and practicable for everyday use.

Advantages of the Proposed System:

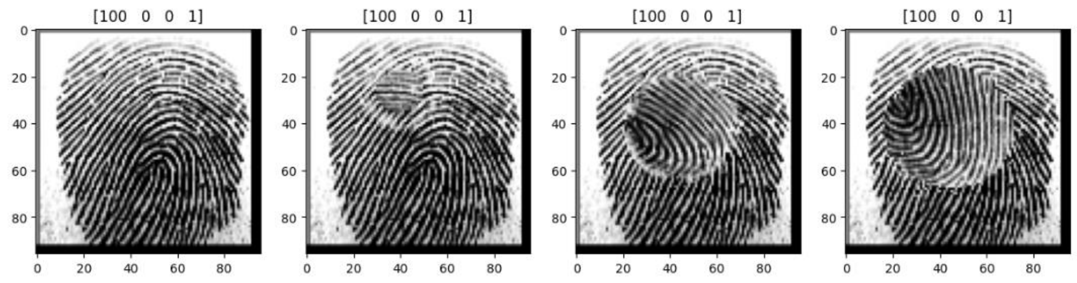
- Lightweight Architecture: Traditional CNN based models require much more computational power, and especially the SNN architecture, when optimized for mobile, demands far less. Due to the efficiency required for use on smartphones, it is highly suitable for use in such a form.
- Secure Fingerprint Storage: During the initial registration, the user's fingerprints are stored in the system using AES encrypted, so that sensitive biometric data is protected. This feature enables system to customize the fingerprint recognition process for each user. During this phase augmentation is used to add an extra accuracy of the model.
- Scalability: The SNN model is lightweight and once registered does not require the permanent storage of new fingerprint templates. It is easy to deploy across different mobile platforms, irrespective of the hardware specifications. It is designed to scale, with modifications to the underlying architecture needed only for a small amount.
- Battery Efficiency: During inference, power consumption is minimized for mobile application, which is critical, once converted to TFLite, the SNN model. The system prevents fingerprint recognition from using up the device's battery with excessive energy.

8. Methodology

A structured approach to developing the Siamese Neural Network (SNN) for mobile fingerprint recognition is presented, beginning with real time data collection, preprocessing, model training, encryption, mobile deployment and evaluation of performance.

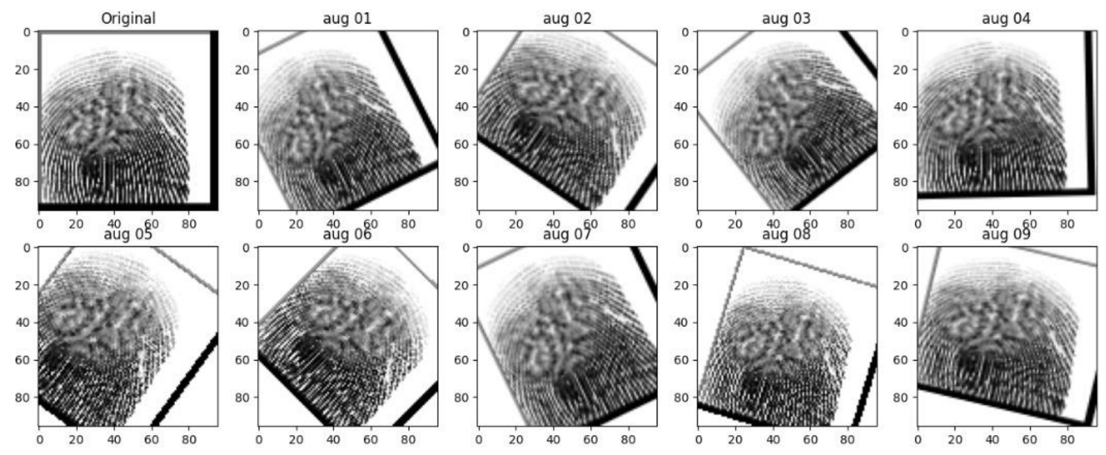
Data Collection and Preprocessing:

Fingerprint data is collected in real time when the user first registers their identity and credentials on the mobile device in this system. When this registration phase takes place, the fingerprint image is captured and safely stored by AES encryption. The foundation of the user's biometric profile is based on this fingerprint data.



The preprocessing steps are essential to maintain consistency and quality in the input data before training the model:

- Image Resizing: All fingerprint images are resized into a fixed size of 96x96 pixels. The standardization that comes from this size also reduces computational resources and speeds up processing time because this size is small enough. Smaller images, as the system is more efficient on mobile devices, most of them have limited processing power.
- Normalization: Once resized, pixel values are scaled between 0 and 1. This last step makes the model converge faster in training and adds the model's ability to generalize under a variety of lighting conditions, or fingerprint quality.
- Data Augmentation: During training, the use data augmentation techniques increase the robustness of the model. Rotation, flipping, and translation are techniques such as these which aid a SNN to be able to recognize fingerprints in different orientations or in different conditions. It is important in dealing with the variability inherent in real world applications.



Model Architecture and Training:

A convolutional layer, max pooling layer and dense layer are all used in building the SNN model. The architecture is designed to extract distinctive features from fingerprint images and generate embeddings for comparison.

- Training Objective: A contrastive loss function is used to train the model, minimizing the distance between embeddings of matching fingerprints and maximizing the distance between nonmatching pairs. For the pairwise comparison tasks, this is an ideal approach, and the SNN also accurately distinguishes genuine and impostor fingerprints during the authentication process.
- Hyperparameter Tuning: These key hyperparameters, learning rate, batch size and number of epochs are fine-tuned in order to find the optimal model performance. In order to avoid overfitting, early stopping is employed, allowing the model to generalize well among new unseen data.
- Encryption and Secure Storage: Biometric data is stored securely with the fingerprint data stored using AES encryption at the time of registration. During a login attempt, the encrypted fingerprint data is used for future comparison. Once the fingerprint has been matched against the user's input fingerprint, the system executes according to the mobile device's security protocols of data handling validating both data integrity and privacy.

Model Architecture and Training:

The SNN model is built with convolutional layers, max pooling layers and dense layers. The architecture is designed to extract distinctive features from fingerprint images and generate embeddings for comparison.

- Training Objective: A contrastive loss function is used to train the model, minimizing the distance between embeddings of matching fingerprints and maximizing the distance between nonmatching pairs. For the pairwise comparison tasks, this is an ideal approach, and the SNN also accurately distinguishes genuine and impostor fingerprints during the authentication process.
- Hyperparameter Tuning: The model performance is fine-tuned on key hyper parameters such as learning rate, batch size and the number of epochs. To prevent overfitting, early stopping is performed, so that once the model sees new, unseen data, it generalizes well.
- Encryption and Secure Storage: In order to guarantee sensitive biometric information's security, the user's fingerprint data is stored at the time of registration, using AES encryption. During a login attempt, the encrypted fingerprint data is used for future comparison. A comparison between the pre-stored fingerprint and the user input fingerprint is done, and after the comparison, the data is processed in line with the mobile device's security protocols to match data integrity and privacy.

9. Analysis

Figure A: The relationship between time per matching and similarity score:

The relationship between two important variables in the matching process is illustrated in this scatter plot.

Key Observations:

- X-axis (Time per Matching in seconds): Usually indicates the time needed to do a matching operation, e.g. nearest neighbours or similarity matching algorithm.
- Y-axis (Similarity Score per Matching): It is a measure of similarity between two entities in terms of matching process, from score of 0 to 1, where 1 is the maximal similarity.

Analysis:

- The plot most of the data points cluster in the top left area, around a similarity score of 1 and time per matching around 0.06 seconds. This means that a large number of matching operations are both very accurate (similarity close to 1) and very fast (small time per matching).
- Lower similarity scores (between 0.5 and 0.9) with different matching time (between 0.065 and 0.085 seconds) are shown in a few outlier points. These cases correspond to situations where matching took longer or produced less similarity.

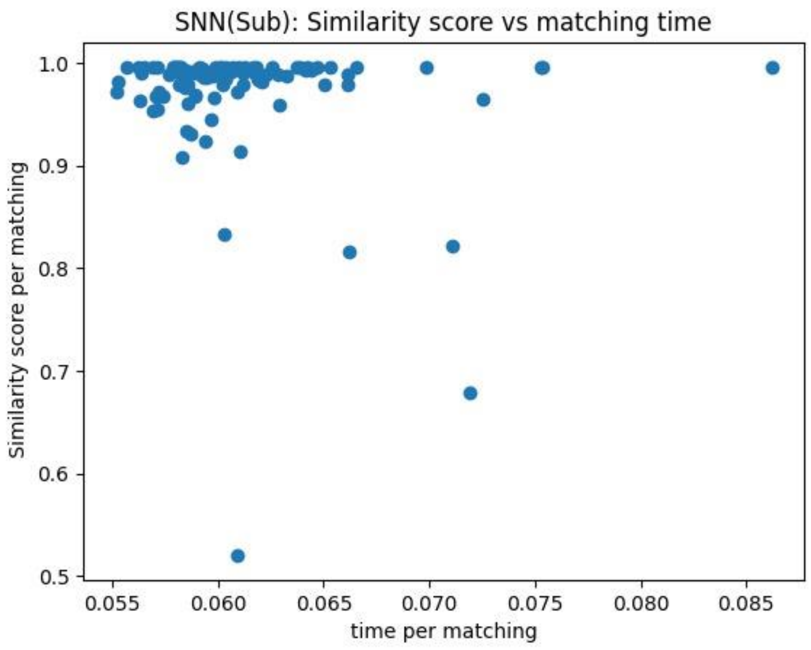


Figure B: SNN (Subtract) Model Train and Test Accuracy Across 100 Epochs:

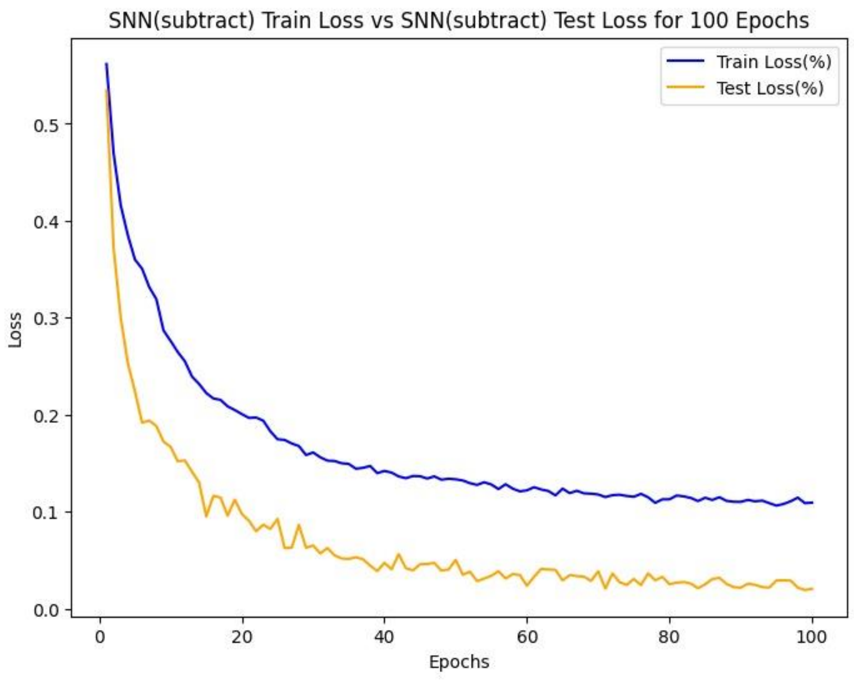
Here the comparison of train and test accuracy for a Spiking Neural Network (SNN) model with subtract operation over 100 epochs is plotted in this graph.

Key Observations:

- X-axis (Epochs): Number of training iterations, from 0 to 100.
- Y-axis (Accuracy): Accuracy is measured from 0.70 to 1.0 (70% to 100%).

Analysis:

- The blue line represents the train accuracy of the SNN (subtract) model for which the accuracy begins at approximately 70% and increases up to a little above 96% at the end of training indicating effective learning.
- The orange line represents the test accuracy, starting at a comparable value, and increasing more quickly to almost 98% after the 20th epoch and then staying at a stable level. Generalization to unseen data shows strong evidence of minimal overfitting and this trend.



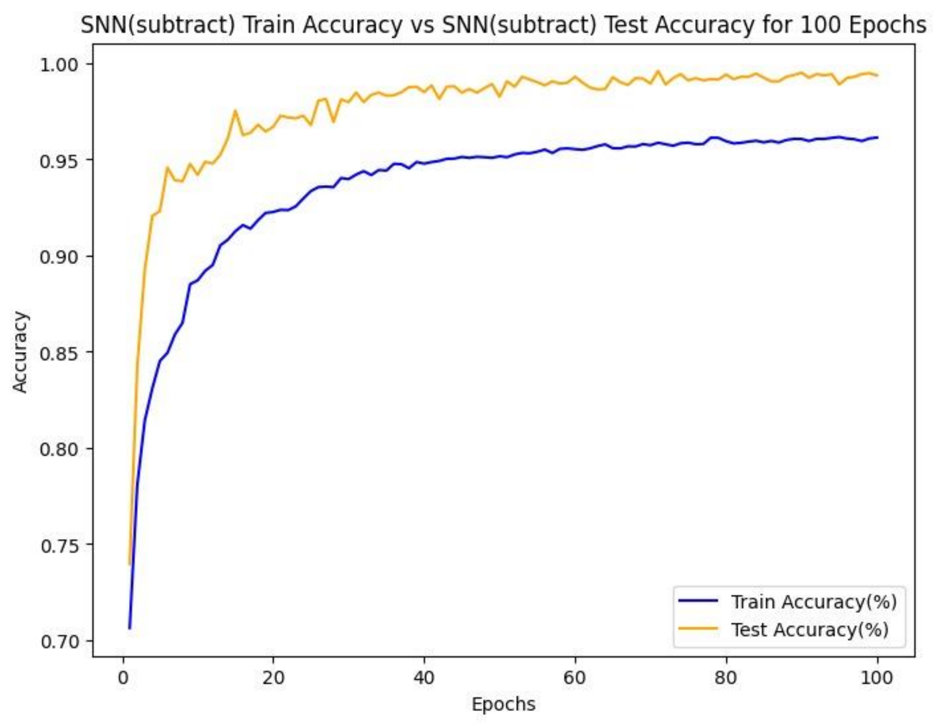


Figure C: Using SNN(L1): Similarity Score vs Matching Time

Here similarity scores are compared against matching time of a fingerprint recognition system based on the Siamese Neural Network (SNN) with L1 distance.

Key Observations:

- Y-axis (Similarity score per matching): It is a measure of how similar two fingerprint templates are (0.55 to 0.90).
- X-axis (Time per matching): It shows the time taken for each matching from 0.055 to 0.090 seconds.

Analysis:

- Most matches fall into the top left (higher similarity scores, faster matching times), with some points close to a maximum similarity score (0.90) at low times (0.055 to 0.065 seconds).
- At slightly higher matching times (above 0.065 seconds) there are some outliers with lower similarity scores (around 0.55-0.75).
- A clear trade-off between similarity scores at different time intervals, with the highest performance in both accuracy and time near 0.90 score mark.

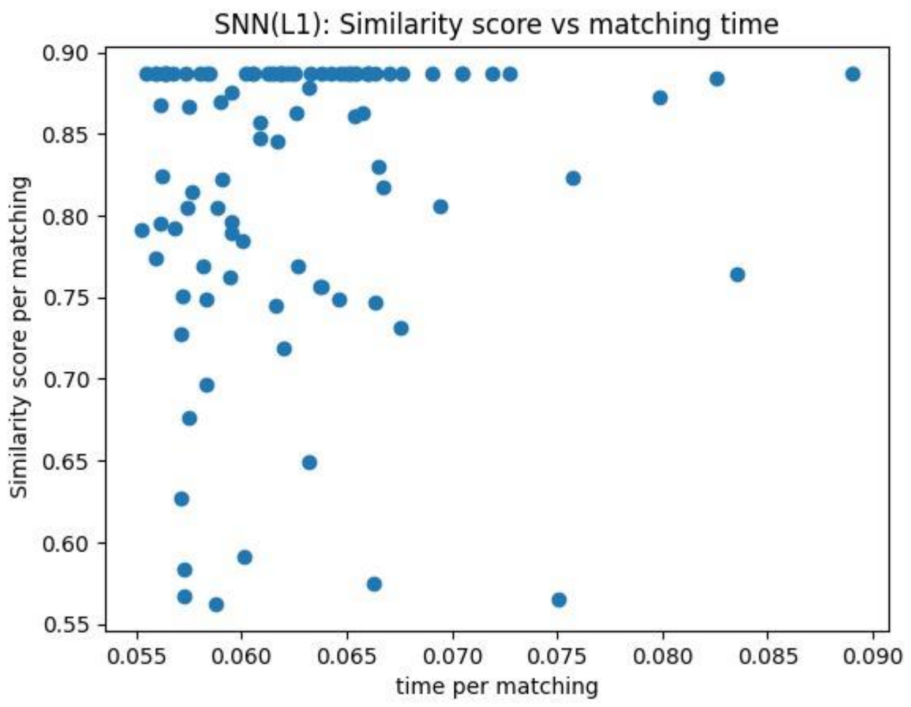


Figure D: Training Accuracy Comparison of SNN Models

This line graph compares the training accuracy of two SNN (Siamese Neural Network) models over 15 epochs.

Key Observations:

- Y-axis (Accuracy): Represents the accuracy of each model during training.
- X-axis (Epochs): Number of epochs during training, ranging from 1 to 15.

Analysis:

- L1 SNN (Blue Line): Shows a gradual increase in accuracy, starting at around 60% and steadily climbing to approximately 78% over 15 epochs. This model improves consistently but at a slower rate.
- Subtract SNN (Yellow Line): Begins with a higher initial accuracy, around 83%, and reaches a peak of about 92% accuracy in the first few epochs. The improvement slows after 5 epochs but maintains a higher accuracy compared to the L1 SNN.

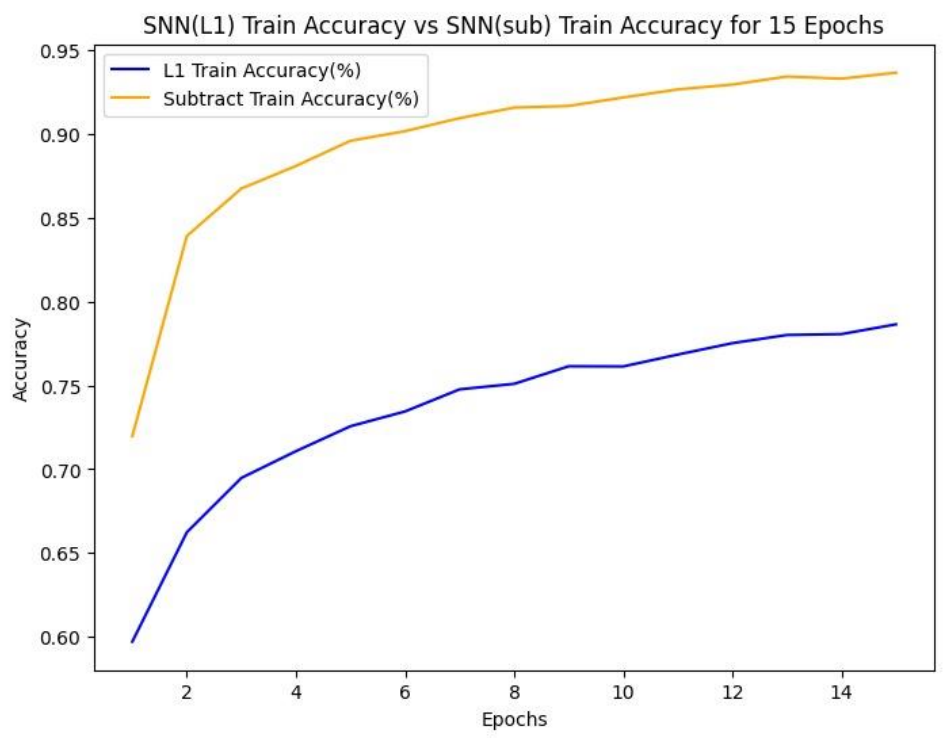


Figure E: Test Accuracy Comparison of SNN Models over 100 Epochs

This line graph compares the test accuracy of two SNN (Siamese Neural Network) models over 100 epochs.

Key Observations:

- Y-axis (Accuracy): Represents the test accuracy of each model.
- X-axis (Epochs): Number of epochs during the testing phase, ranging from 1 to 100.

Analysis:

- L1 SNN (Blue Line): The test accuracy of the L1 model remains fairly consistent, fluctuating between 55% and 60% throughout the epochs. There is no significant improvement over time.
- Subtract SNN (Yellow Line): The Subtract SNN model starts with a high accuracy of approximately 90% and shows slight improvements, peaking close to 98% before stabilizing.

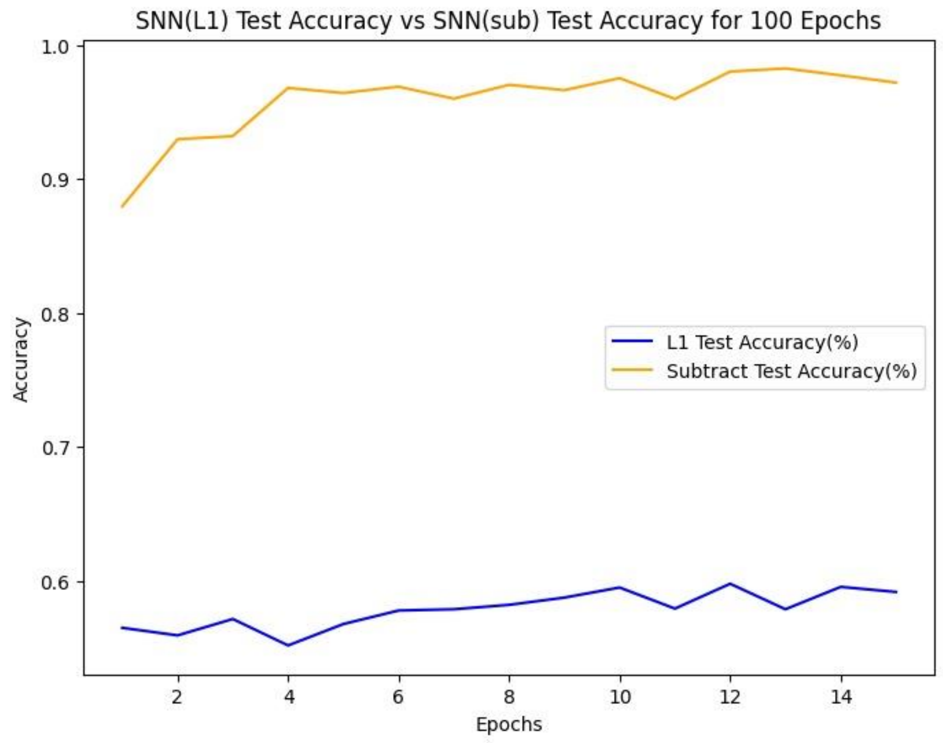


Figure F: Comparison of Similarity Score of Models:

In the comparison, comparison of the similarity scores of two SNN (Siamese Neural Network) models with different matching methods for the same matching time.

Key Observations:

- Y-axis (Similarity Score per Matching): This is the amount of how well each model can match the data. The closer the score is to 1, the stronger the match.
- X-axis (Time per Matching): It is the time taken by each model to match.

Analysis:

- SNN (Sub) (Blue Dots): This method shows high similarity scores consistently and most of the clusters are near 1.0, indicating strong performance. And the time per matching stays low and is concentrated around 0.06 seconds.
- SNN (L1) (Orange Dots): The similarity scores of this method are more spread out, with values ranging from 0.4 to 0.9 which exhibit more variance in performance. The matching times are more widespread, beyond 0.1 seconds.

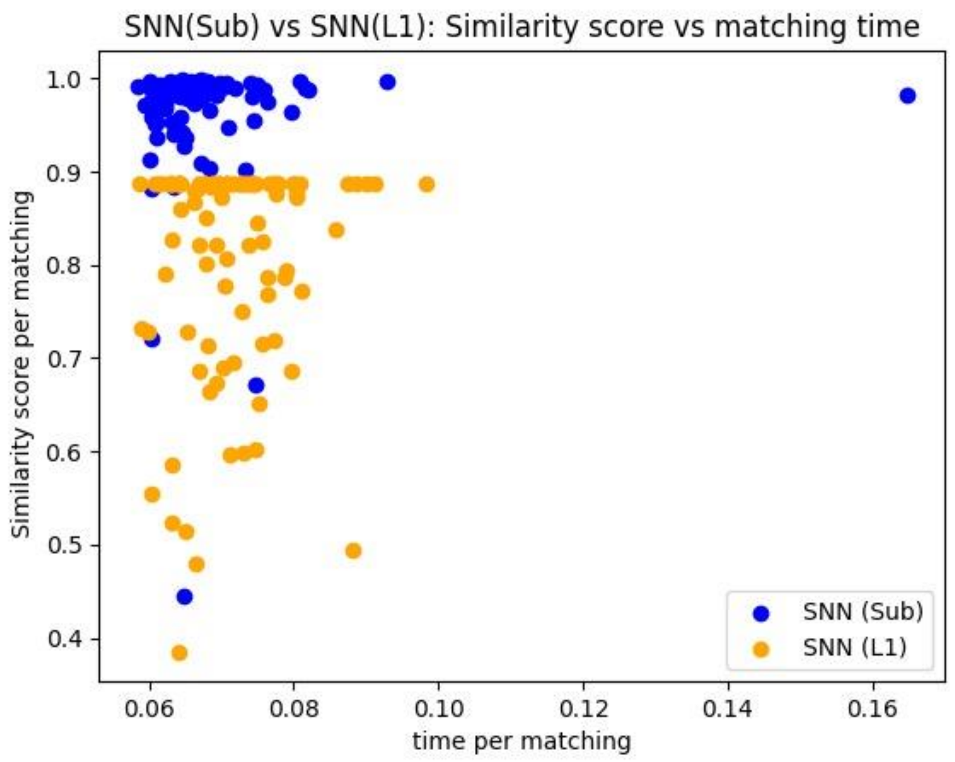


Figure G: The Matching Rate Comparison of Models:

The matching rates of three models are compared in a verification or classification task and this bar graph compares them.

Key Observations:

- Y-axis (Matching Rate %): It represents the accuracy of each model to identify or match data.
- X-axis (Models): It compares three different approaches.

Analysis:

- Traditional Minutiae-based (Gray Bar): It achieves a matching rate of about 60% which is relatively low.
- CNN-based (Orange Bar): It demonstrates a matching rate of around 85% which is a better accuracy compared with the traditional method.
- Siamese Neural Networks (SNNs) (Blue Bar): It achieves matching rates close to 100% and demonstrates the best performance of the models.

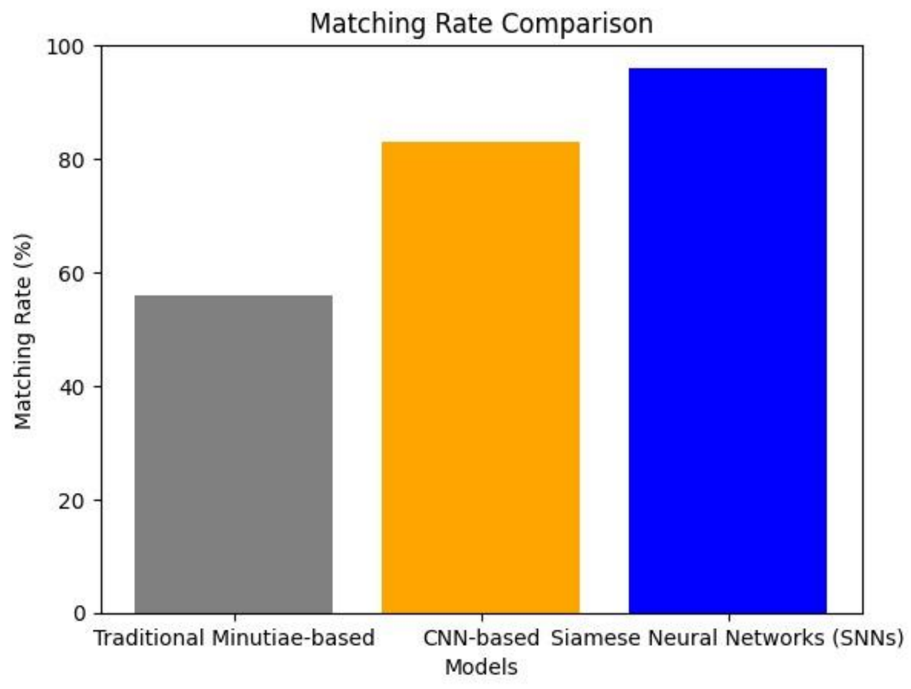


Figure H: Comparison of Storage Requirements of Models:

The storage requirements for each of the three models are shown in this bar graph.

Key Observations:

- Y-axis (Storage Requirement in MB): It is the memory required per model.
- X-axis (Models): The same three models are then compared.

Analysis:

- Traditional Minutiae-based (Gray Bar): It takes about 1 MB of storage, making it the least memory demanding option.
- CNN-based (Orange Bar): This means that the storage requirements are significantly higher than that.
- Siamese Neural Networks (SNNs) (Blue Bar): Strikes the balance between minutiae-based model and CNN based model, using around 3 MB.

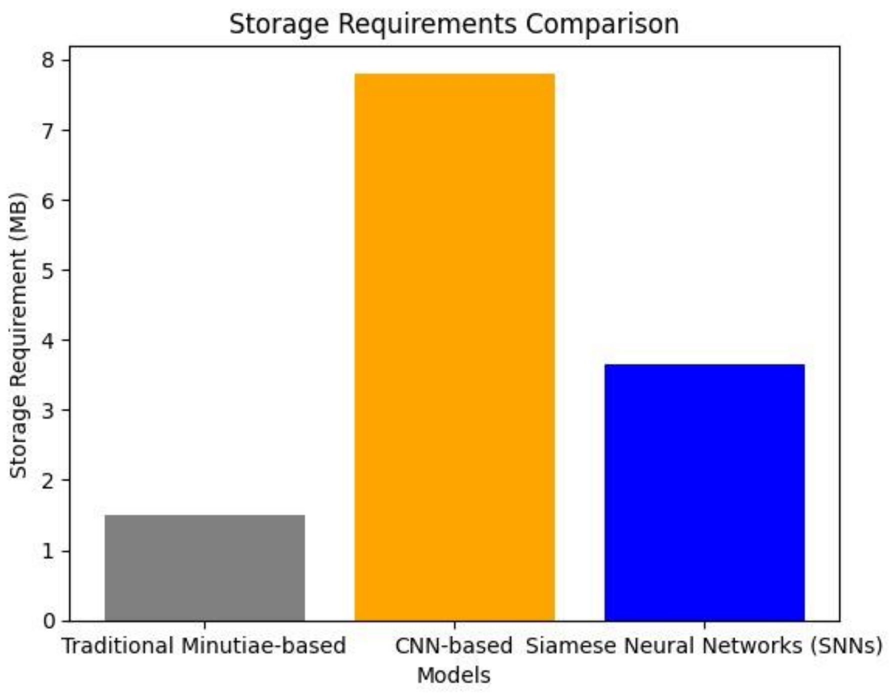


Figure I: Speed Comparison of Models:

Below is a graph depicting the time (in milliseconds) taken per match by each model.

Key Observations:

- Y-axis (Time per Match in ms): Indicates how long each model takes to perform a match.
- X-axis (Models): Compared to the following models:

Analysis:

- Traditional Minutiae-based Model: The time of shows the highest at around 40 ms which means it is one of the slowest to do among the options.
- CNN-based Model: One that is faster than the minutiae based (approximately 30 ms per match).
- Siamese Neural Network (SNN): With a time per match below 10 ms, the fastest model.

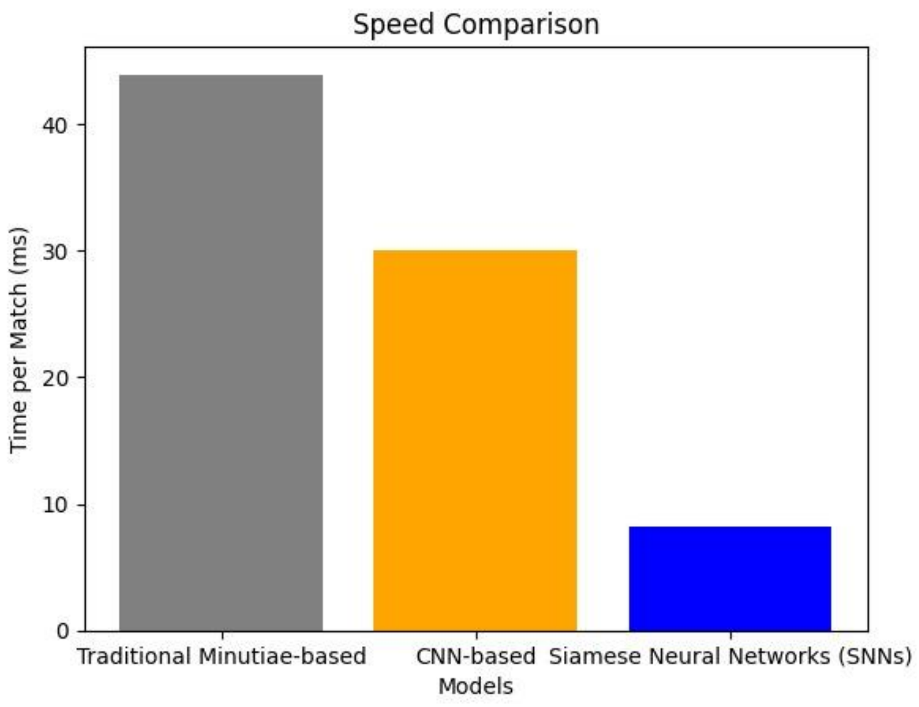


Figure J: Accuracy Comparison of Models:

This graph shows the accuracy of each model, as a percentage.

Key Observations:

- Y-axis (Accuracy %): They represent the accuracy of each model.
- X-axis (Models): Compares them in the following models.

Analysis:

- Traditional Minutiae-based Model: It has an accuracy around 70%.
- CNN-based Model: Approximately 90% accuracy shows improved.
- Siamese Neural Network (SNN): It achieves the highest accuracy, approximately 95%.

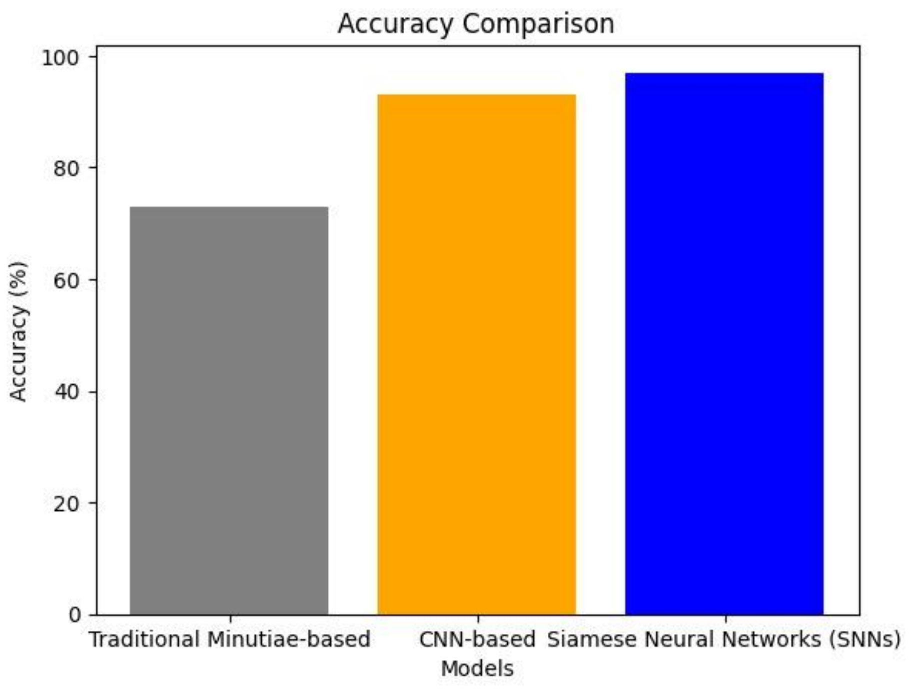


Figure K: Comparison of Fingerprint-Matching Models in Computational Complexity:

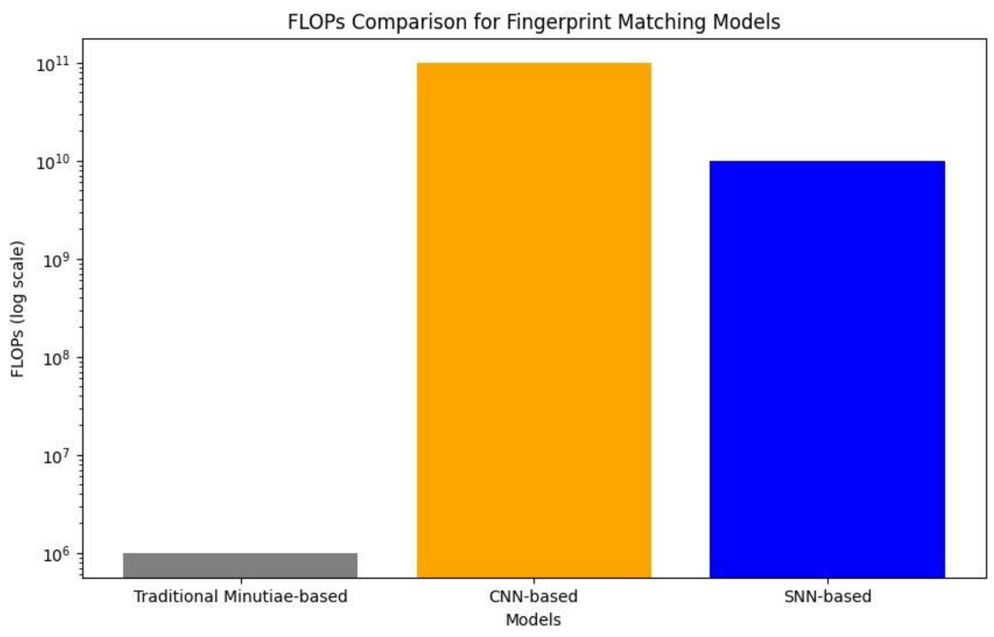
On a logarithmic scale, this graph compares the computational complexity of Traditional Minutiae based, CNN based, and SNN based fingerprint matching models using the number of Floating-Point Operations per Second (FLOPs).

Key Observations:

- Y-axis (FLOPs): Is the computational complexity represented on a logarithmic scale.
- X-axis (Models): It compares the following models.

Analysis:

- Traditional Minutiae-based Model (Gray Bar): Has the lowest FLOPs, around 10^6. It is found that this method demands much less computational resources than the other models.
- CNN-based Model (Orange Bar): The FLOPs are displayed at approximately 10^{11} This suggests that CNN based models are very computationally expensive and require much higher number of operations to do fingerprint matching.
- Siamese Neural Network (SNN)-based Model (Blue Bar): Shows a FLOPs value around 10^{10} Still orders of magnitude above traditional minutiae based approaches, , which is slightly lower than CNN based models.

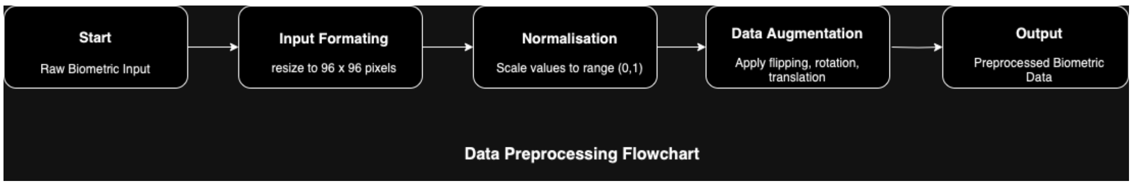

FLOPs Comparison for Fingerprint Matching Models
FLOPs (log scale)
10^{11}
10^{10}
10^{9}
10^{8}
10^{7}
10^{6}
Traditional Minutiae-based
CNN-based
SNN-based
Models

10. Design

1. **Data Preprocessing Flowchart:**

This flowchart describes the critical steps for the preparation of a biometric input data for processing within the model.

- Start (Raw Biometric Input): Biometric data, specifically, fingerprint images are captured at the raw level, and after that the process starts.
- Input Formatting (Resize to 96 x 96 Pixels): The raw input images are then resized to a common dimension of 96x96 pixels to standardize the image size to help ease efficient processing. Along with this, the computational load is reduced and the resizing guarantees consistency in the data set.
- Normalization (Scale Values to Range (0,1)): All images are normalized to the range from 0 to 1 in terms of pixel values. The scaling improves the training efficiency and achieves higher consistency of the model input to learn from the data.
- Data Augmentation (Apply Flipping, Rotation, Translation): Different augmentation techniques are used to change the images, such as flipping, rotating and translating in order to enlarge the training dataset. In this approach, the model generalizes better and is more robust to the variation of fingerprint presentation.
- Output (Pre-processed Biometric Data): To a certain extent, the biometric data is rendered suitable for model input by following these preprocessing steps; therefore, this biometric data can be further processed and analysed.

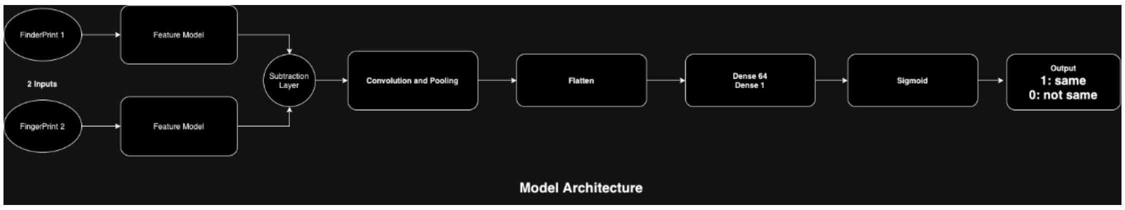

Start
Raw Biometric Input
Input Formating
resize to 96 x 96 pixels
Normalisation
Scale values to range (0,1)
Data Augmentation
Apply flipping, rotation, translation
Output
Preprocessed Biometric Data
Data Preprocessing Flowchart

2. **Model Architecture:**

The architecture of the Siamese Neural Network used for fingerprint recognition is detailed in this flowchart which showcases its multistage design.

- Two Inputs (Fingerprint 1 and Fingerprint 2): An architecture is presented, which starts with two different inputs, each corresponding to a fingerprint. First, the user's profile (previously registered fingerprint) provides one input, and the second input is captured when the user attempts to login. The setup allows to compare the biometric data.

- Feature Model: Each fingerprint is processed by a specialized feature extraction model that extracts salient features necessary for good comparison.
- Subtraction Layer: A subtraction layer turns into feature representations of both inputs, and they are compared using it. That allows the model to work with the differences in the fingerprints.
- Convolution and Pooling: The resultant differences are processed through a series of convolutional layers followed by pooling operations. At this stage, there is a need to extract hierarchical features whilst simultaneously keeping dimensionality low, in order to enable the model to find relevant patterns.
- Flatten: The flattened processed features are then fed to feed to the following dense layers.
- Dense 64, Dense 1: It is flattened vector through fully connected (dense) layers. The first dense layer with 64 neurons is intended to capture complex feature interactions, and the second, dense layer contains a single neuron which outputs a single prediction score.
- Sigmoid: Finally, a final dense layer feeds to an application of a sigmoid activation function to transform raw output into a probability score indicating the probability of match.
- Output (1: final output binary (Same, 0: Not Same): The final output is binary, a value of 1 means match (inputs belong to the same person), a value 0 means do not match (inputs do NOT belong to the same person).

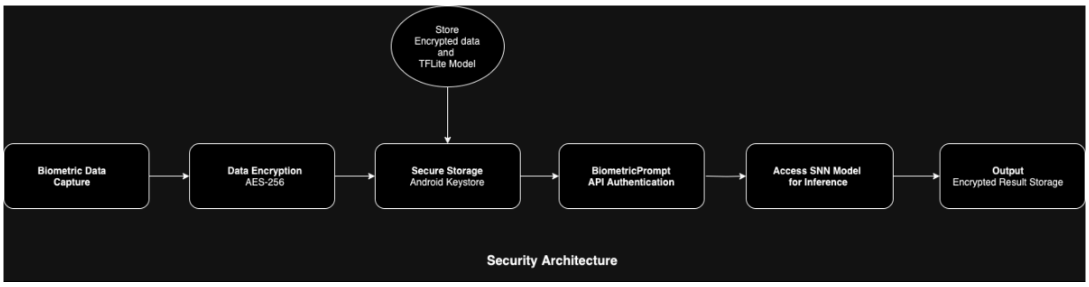


3. **Security Architecture:**

The secure fingerprint data processing flow using a Siamese Neural Network (SNN) for fingerprint matching in an Android device context is shown in this diagram. It highlights the key security items that protect biometric data from the time of capture to the time of processing, and ultimately to the time of storage.

- Biometric Data Capture: The process of security begins with capturing biometric data which is fingerprints via the device sensors. This initial stage is very important to make sure that the input data is handled securely.

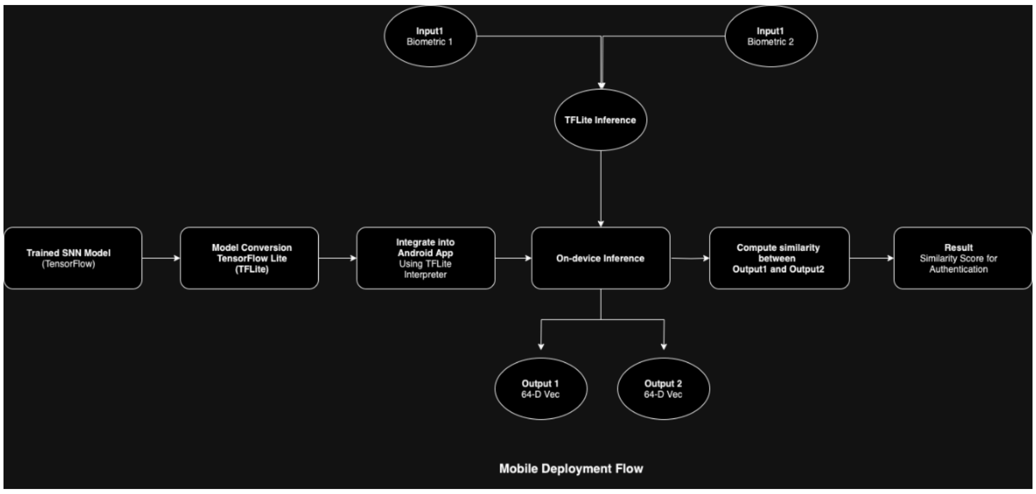
- Data Encryption (AES-256): The biometric info is captured and after that the biometric information goes through AES-256 encryption, which is considered one of the most robust encryptions. This step in fact protects from exposure of sensitive biometric data raw form transmission, storage, or via unauthorized access.
- Secure Storage (Android Keystore): And finally, in the Android Keystore is the encrypted biometric data and the TFLite model used for inference. This component ensures that cryptographic keys stay contained within the device, so that defense against unauthorized tampering and access to encrypted fingerprint data and the machine learning model used to learn who is who is strengthened.
- BiometricPrompt API Authentication: Before further processing it relies on the BiometricPrompt API to check for user authentication. This API makes it easy for a secure authentication flow using the biometric capabilities of the device, e.g., fingerprint or facial recognition, to be used only with those who are allowed to activate the SNN model for fingerprint matching.
- Access SNN Model for Inference: The system retrieves the Siamese Neural Network (SNN) model for inference upon successful authentication. The fingerprint comparison process with this model is highlighted with the secure TensorFlow Lite model and the input fingerprint is run through it to compare the stored biometric data to make a similarity determination and produce a matching score.
- Output (Encrypted Result Storage): The SNN inference produces a result, i.e. does it match or not, which is securely stored. The matching decision is encrypted and stored in a storage location that limits access by only authorized personnel, leaving the confidentiality of the matching decision undisturbed. It protects the results from possible data breach or unauthorised exploitation.


Store Encrypted data and TFLite Model
Biometric Data Capture
Data Encryption AES-256
Secure Storage Android Keystore
BiometricPrompt API Authentication
Access SNN Model for Inference
Output Encrypted Result Storage
Security Architecture

4. Mobile Deployment Flow:

The deployment of a trained Siamese Network model (SNN) for fingerprint authentication on a mobile device is illustrated by this flowchart.

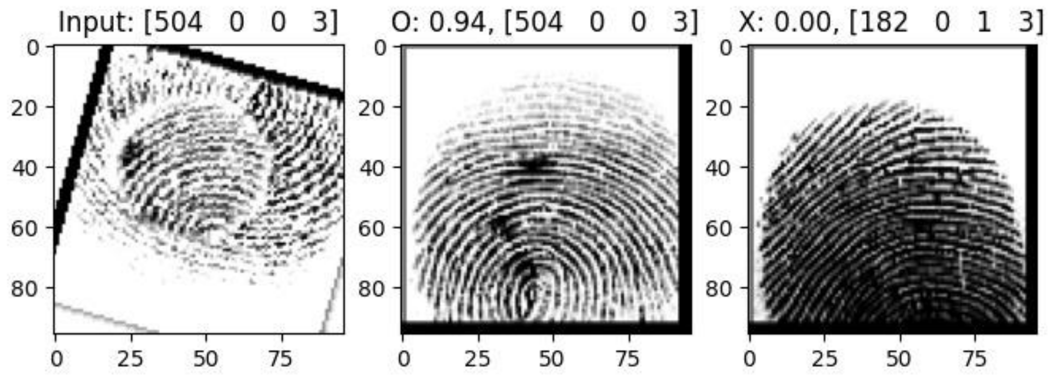
- Trained SNN Model (TensorFlow): Using TensorFlow framework provides robust tools to build and optimal deep learning models, the process starts with SNN model training.
- Model Conversion (TensorFlow Lite): The model is then trained after which the model is converted into TensorFlow Lite (TFLite) format. It is crucial for the model to be able to operate efficiently on mobile devices to achieve performance optimality and accuracy at the same time.
- Integrate into Android App Using TFLite Interpreter: Finally, the converted model is used in an Android application using the TFLite interpreter to do on device inference. It reduces the reliance on external servers, and it improves user privacy.
- Input1 and Input2 (Biometric 1 and Biometric 2): For comparison, the model takes two biometric inputs (fingerprints). The first input is taken from the user's profile (previously registered fingerprint), while the second input is captured when the user is trying to log in. With this setup, the user's authentication data can be directly analysed.
- TFLite Inference: The on-device inference engine is used to process the inputs and generate embeddings, generating 64 dimensional vectors which represent the feature sets of the input fingerprints.
- Output1 and Output2 (64-D Vector): Outputs are two 64 dimensional vectors, one for each biometric input, which contain the essential features for comparison.
- Compute Similarity: Between the two output vectors, a similarity measure is computed, which measures degree of similarity between the fingerprints and aid in authentication decisions.
- Result (Similarity Score for Authentication): The system calculates the similarity score and based on the similarity score decides if the two fingerprints come from the same person. If the score is high, there is a match, and authentication occurs; if low, there is a mismatch.


Input1 Biometric 1
Input1 Biometric 2
TFLite Inference
Trained SNN Model (TensorFlow)
Model Conversion TensorFlow Lite (TFLite)
Integrate into Android App Using TFLite Interpreter
On-device Inference
Compute similarity between Output1 and Output2
Result Similarity Score for Authentication
Output 1 64-D Vec
Output 2 64-D Vec
Mobile Deployment Flow

11. Conclusion

Conclusion from fingerprint recognition of image: The fingerprint recognition image is made up of three panels that present the fingerprint recognition process in the context of biometric security. The different labels and scores demonstrate how the system evaluates the similarity between fingerprints:

1. Left Panel (Input Image): - Label: Input: [504 0 0 3]
   It is an input fingerprint image on this panel. The fingerprint looks a bit rotated perhaps an initial raw (or unprocessed image) that needs correction or normalization. The print is clear, ridge lines are visible, but there are some borders or artifacts around the edges of the image.
2. Middle Panel (Matched Fingerprint): - Label: O: 0.94, [504 0 0 3]
   This panel has a high match score (0.94) making this is a successful matching attempt or recognition. The fingerprint pattern looks corrected or aligned with a well-centred clear fingerprint pattern. This score of 0.94 means that input fingerprint and a stored template are very similar (i.e. a success match).
3. Right Panel (Unmatched Fingerprint): - Label: X: 0.00, [182 0 1 3]
   As this score is 0.00, this is a failed matching attempt. Here the fingerprint seems darker and more distorted, and a few ridge lines seem less defined than in the middle panel. This fingerprint does not match because the lower quality or distortion of it.



Conclusion derived from the figures: The figures are used for an in-depth comparison of different fingerprint matching models including Siamese Neural Networks (SNNs), CNN based models and traditional minutiae-based methods.

1. Performance and Accuracy:

- The Siamese Neural Networks (SNNs) are nearly perfect with accuracy rates nearly 100% and also display high similarity scores. For the fingerprint recognition tasks, the SNN (Subtract) model outperforms the SNN (L1) model, and it is the most effective model.
- As with traditional methods, CNN based models have accuracy rates of 85–90% but are dramatically slower and less efficient than SNNs.
- The matching rates of traditional minutiae-based models are lowest (around 60%-70%) and thus the least suitable for high demanding fingerprint matching tasks.

2. Speed and Efficiency:
   - The fastest matching times (below 10 milliseconds) make SNNs particularly well suited to real time fingerprint recognition, for example unlocking mobile devices or secure access points.
   - SNNs are moderately slow, but much more efficient than CNNs.
   - Slowest to perform are traditional minutiae-based methods, which take about 40 milliseconds to match, which makes them impractical for use in real time applications like mobile security.
3. Computational and Memory Efficiency:
   - The SNNs come up with its good trade-off that uses less computational power than any CNN model and yet produces good performance. They also have moderate storage requirements which are suitable for devices with little resources, such as mobile devices.
   - CNN based models are computationally expensive in terms of memory and processor requirement and may not be convenient for mobile devices constrained with hardware limitations.
   - On the other hand, models based on traditional minutiae can make use of the least resources, However, their accuracy and speed are not high enough to be desirable in secure fingerprint recognition.
4. Training and Generalization:
   - The SNN models learn extremely well and there is no overfitting on both training and unseen data. For instance, the SNN (Subtract) model obtains 98% test accuracy, which is very accurate for mobile applications.
   - SNN models generalize as well but are outperformed by the CNN models that are more efficient.
   - Training and generalization are not evaluated on traditional methods, yet they are considered less adaptable to more complex tasks like fingerprint matching.

Implications for Mobile Fingerprint Security:

- Compared to other security systems, such as those based on keystroke dynamics, this analysis is particularly relevant for mobile fingerprint security systems that need to achieve high accuracy, fast response times, and efficient use of computational resources.
- Mobile security applications require a quick and accurate fingerprint matching and SNNs are the best choice for mobile security. They are able to achieve near perfect accuracy within milliseconds so that it doesn't impact the user experience and at the same time keep the security strong.
- Mobile use for CNN based models may be plausible but they would face a ceiling on energy consumption possibly because of greater computational requirements or memory, causing slower unlock times or battery drain.
- Modern mobile security requires models that are faster and more accurate, but traditional minutiae-based models are too slow and too inaccurate, resulting in frequent misidentifications or slower unlocking times, decreasing the user experience.

Therefore, the conclusion is that Siamese Neural Networks (SNNs) are the most appropriate models for mobile fingerprint security due to the excellent balance between speed, accuracy and resource efficiency, compared to other proposed models. Thanks to their quick response time and high accuracy, mobile devices can be unlocked quickly and securely in a modern mobile environments.

12. Hardware and Software Requirements

Hardware Requirements:

- Mobile Device: Android smartphones or tablets with built in fingerprint sensor, minimum 2GB RAM, running Android 8.0 or later.
- PC for Development: Processor: At least 2.5 GHz Intel i5 or equivalent. RAM: Training and preprocessing tasks take up 8GB or higher.
- Storage: Datasets and models require at least 100GB available. GPU: Accelerated model training recommended NVIDIA GPU.

Software Requirements:

- Operating Systems:
  - Development: Windows 10, macOS, or Linux.
  - Deployment: If you have Android, version 8.0 or later.
- Development Tools: Python: For SNN implementation and the model training.
- TensorFlow & Keras: For building, training and optimizing of the SNN model.
- Android Studio: It can be used to develop the mobile application.
- AES Encryption Tools: AES encryption is implemented using PyCryptodome.
- Libraries and Dependencies: OpenCV, NumPy, Pandas, TensorFlow/Keras, TFLite Converter.
- Version Control and Collaboration Tools:
  - Git: For version control.
  - GitHub/GitLab: It is for cloud-based code repositories.

13. References

1. "Fingerprint Matching Algorithm for Android." International Journal of Engineering Research and Technology (IJERT), vol. 2, no. 10, 2013
2. Chen, Y., & Wang, H. "Efficient Fingerprinting-Based Android Device Identification With Zero-Permission Identifiers." ResearchGate, 2016
3. Gune, S., & Khedkar, S. "The Recognition of Fingerprints on Mobile Applications - An Android Case Study." ResearchGate, 2016.
4. "Fingerprint Hardware Abstraction Layer (HAL)." Android Developer Documentation.
5. Vangala, P. "How Fingerprint Scanners Work." Android Authority, 2017.
6. B. Bakhshi and H. Veisi, "End to End Fingerprint Verification Based on Convolutional Neural Network," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 1994-1998, doi: 10.1109/IranianCEE.2019.8786720.