Deep Learning–Based Fingerprint Identification and Verification

11/05/2025

Table of Contents

Intoduction

Automatic fingerprint recognition has become a cornerstone of biometric security systems, offering a reliable means of personal identification and access control. Traditional approaches rely on handcrafted minutiae extraction and matching, which often struggle with variations in image quality, rotation, and partial prints. In this work, we explore end-to-end deep-learning solutions that automatically learn discriminative fingerprint representations directly from raw TIFF images, thereby simplifying the pipeline and improving robustness against common acquisition artifacts.

We implement and compare two convolutional architectures. The first is a custom five-layer CNN designed to progressively increase filter depth from 32 to 512 channels, interleaved with batch normalization, ReLU activations, max pooling, and a final global average pooling stage. A dropout layer before the final linear classifier provides basic regularization. The second is an adaptation of ResNet-18: its initial convolution is reconfigured for single-channel input, and its final fully connected layer is resized to match the number of fingerprint identities. Both models are trained using Adam with a learning rate of 1×10^{-3}, cross-entropy loss, and extensive data augmentations—including small rotations, translations, Gaussian-style noise, and contrast adjustments—to emulate real-world variability.

Model performance is evaluated on a held-out test set (10% of the data) using both classical classification metrics (accuracy, precision, recall, F_1-score, and confusion matrix) and biometric verification protocols (1:N identification and 1:1 genuine/impostor authentication). A unified evaluation script automates metric computation, generates comparative plots, and saves both per-model and combined visualizations for in-depth analysis.

1. Fingerprint Dataset

The training is based on fingerprints dataset that were used in FVC2004 competition with the following characteristics :

Parameter	**Specification**
Size	10 fingers × 8 impressions
Impression type	Synthetic plain
Format	TIFF, 500dpi, 288×384px

Table 1 FVC2004 DB4 B Dataset Characteristics

In total there are 408 images (*see fig.1*) in .tif format with filename format: xxx_yyyy_zzz.tif, where xxx is person ID (from 012 up to 076); yyy is finger ID (from 3 up to 8); zzz is number of scans (from 1 up to 8).

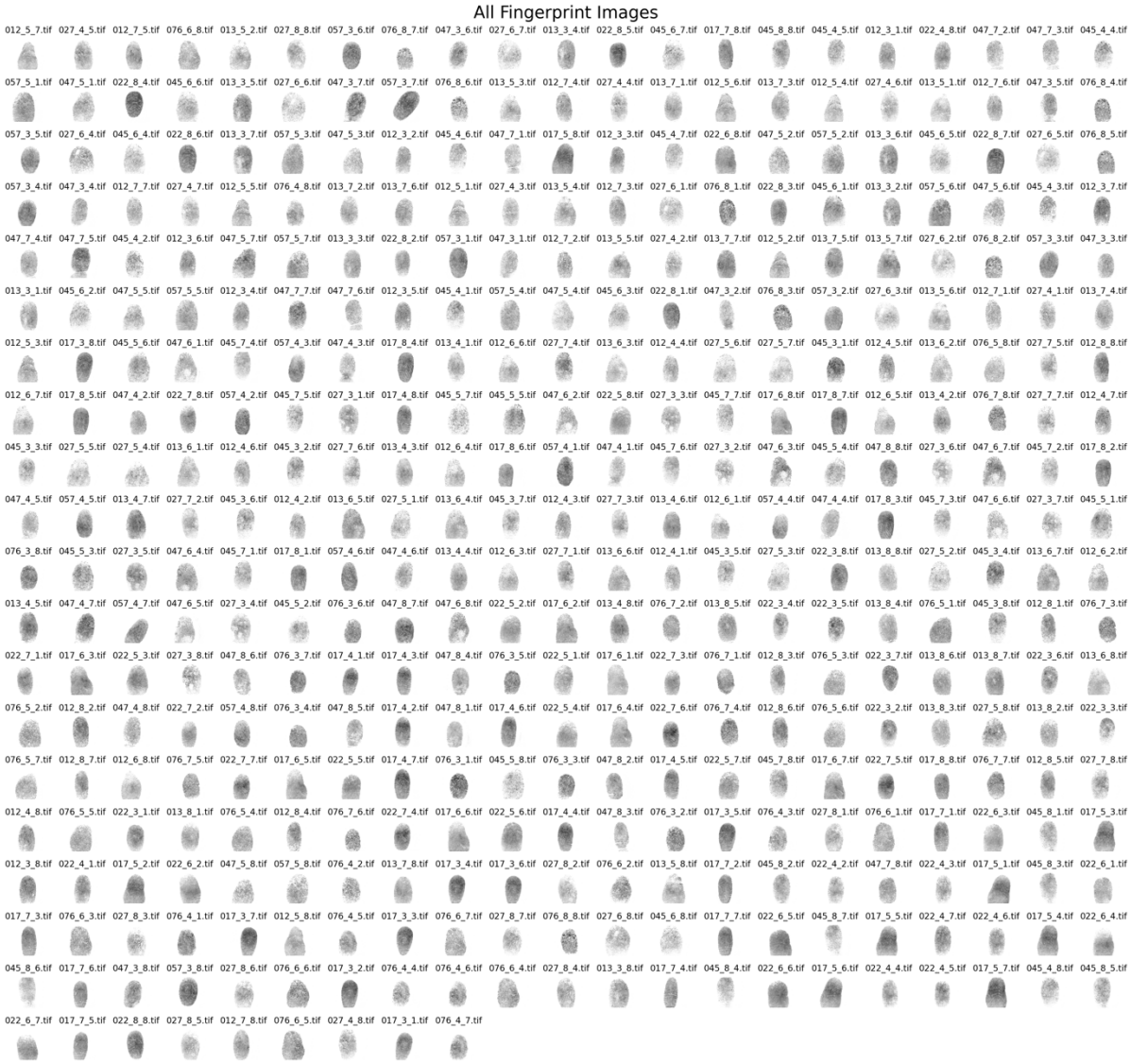


Fig. 1 All Fingerprint Images

2. Fingerprint image preprocessing

Pre-processing goals is to improve print quality, harmonize print scale and orientation, emphasize the print area, and eliminate background. This is critical because the quality of the input print has a significant impact on the accuracy of the system.

If prints can be rotated during scanning, it is desirable to bring them to a consistent orientation. Traditional methods suggest finding the direction of the main line flow and rotating the image so that the “core” of the print is at the top.

Scanned prints may have different illumination or contrast. It is necessary to bring the pixel intensities to a uniform range. A simple solution is to linearly normalize the values to the range [0,1] or [0,255].

There are many more such scenarios that aims to increase the quality of the input data. However, **all images in the database are already pre-processed** and there is no need for modification. But it is possible to increase diversity by **data augmentation**.

2.1 Unwanted augmentations

Data augmentation aims to increase the diversity of training samples through random transformations should be used cautiously. In general, geometric, and photometric transformations (rotations, shifts, scaling, reflections, changing brightness/contrast, adding noise) are widely used to increase the network's ability to generalize. However, **fingerprints have a strict structural organization, so not all types of augmentation are acceptable (***see fig. 2*).

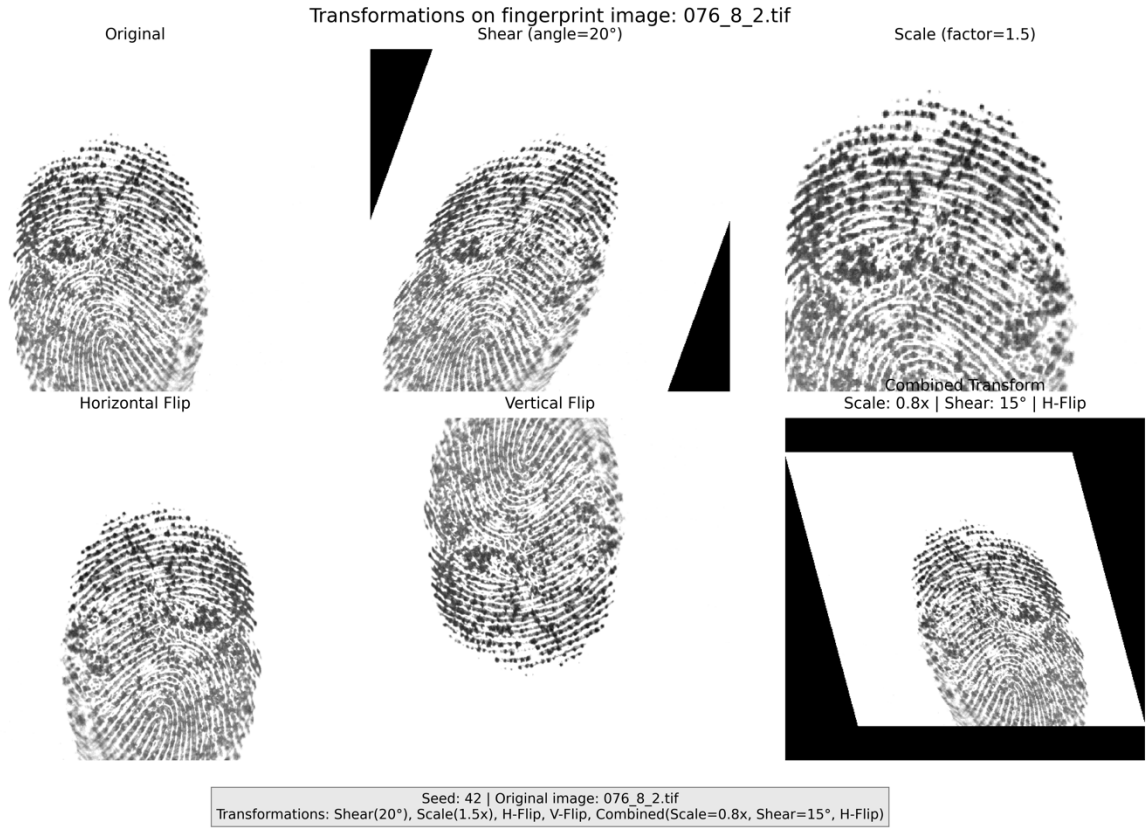


Fig. 2 Unwanted Transformations

Shifting the scale (zoom) may cause part of the print to be lost or areas may not overlap completely when comparing areas.

Fig. 3 Scale shift (zoom)

X/Y shift (shear) - distorts the relative direction of the ridges. For example, shearing significantly changes the angle of the lines and the relative position of the minutiae, which impairs recognition.

Fig. 4 X/Y shift (shear)

Mirroring (horizontal/vertical flip) is also unacceptable - a fingerprint has no symmetry; mirroring turns the print into the pattern of another (opposite hand) fingerprint.

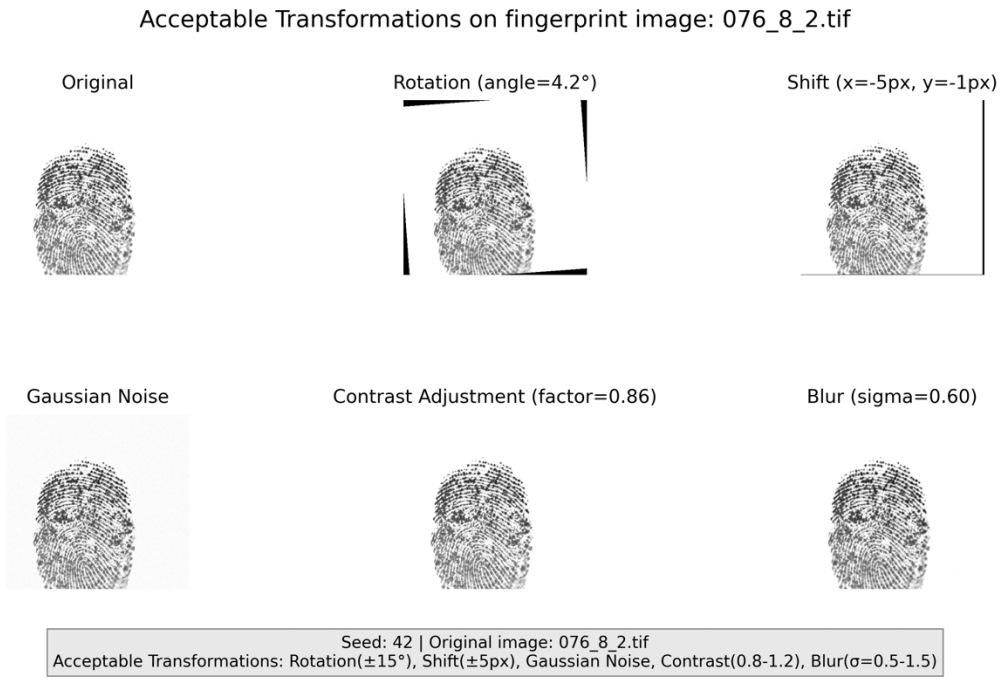
Fig. 5 Horizontal Flip and Vertical Flip

2.2 Acceptable augmentations

Acceptable transformations could be **rotations in the image plane**. Since the finger can be rotated at any angle during scanning, it is reasonable to train the network to be invariant to rotations. In this paper, we propose to generate a random angle of 0-360° and rotate the fingerprint image by expanding or cropping it to its original size (e.g., by taking a larger field and rotating it followed by cropping it to the centre). **Horizontal/vertical shifts (a few pixels)** simulate a slight displacement of the finger on the sensor. Such shifts, which do not exceed the ROI boundaries, increase the robustness to the fingerprint position.

It is acceptable to add random Gaussian noise to the image, small changes in brightness/contrast, simulated blur (out-of-focus) - this helps to make the model more robust to real-world conditions (e.g. wet/dry finger, dirty scanner). Some studies include adding artificial noise or colour variation to the augmentation process, achieving greater diversity in the data.

We take the original normalised print image; with a 50% probability rotate it by ±15°; with a 50% probability shift it by ±5 pixels; with a 30% probability add a little Gaussian noise; with a 30% probability change the contrast slightly. This composition will generatively create many variations while preserving the key features of the print.



3. Selection and comparison of deep learning models

The choice of model architecture is a central decision that determines the recognition ability of the system. Let us consider several possible approaches.

3.1 Simple CNN

We designed a lightweight convolutional network consisting of five sequential convolutional blocks. Each block applies a 3×3 convolution with padding, followed by batch normalization, a ReLU activation, and down-sampling (via max pooling in the first four blocks and global average pooling in the final block). This produces a 512-dimensional feature vector for each input. With dropout layer to minimize overfit.

Training minimizes the multi-class cross-entropy loss using the Adam optimizer with a fixed learning rate of 1×10^{-3}. We process the data in mini-batches of 32 and perform 20 epochs. After each mini-batch update, model parameters are adjusted to reduce classification error, and the version yielding highest validation accuracy is checkpointed.

All fingerprint images are first converted to single-channel tensors and normalized to zero mean and unit variance. The dataset is randomly split into 70 samples.

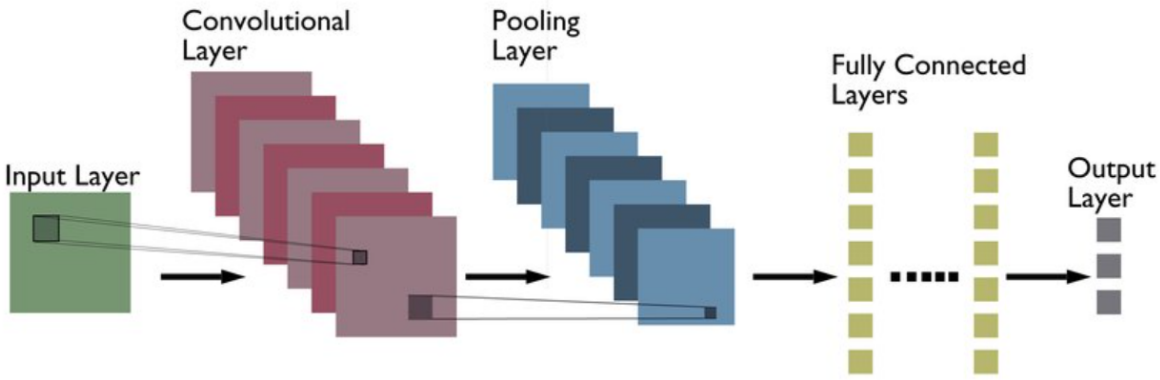


Fig. 6 CNN Architecture

The simple five-layer CNN learns quickly—its training loss falls smoothly toward zero and its training accuracy climbs above 90 % by mid-training—but its validation curves are much noisier and never quite catch up. Validation loss, while generally trending downward, exhibits large spikes (especially around epochs 3 and 19), and validation accuracy fluctuates between 20–80 % before dropping sharply at the final epoch. This divergence indicates that the network is overfitting: it has enough capacity to memorize the training set but struggles to generalize consistently to unseen fingerprints. To improve stability and boost validation performance, one might introduce stronger regularization (e.g. higher dropout rate or weight decay), augment the data more aggressively, or employ early stopping to prevent late-stage overfitting (*see Fig. 7*).

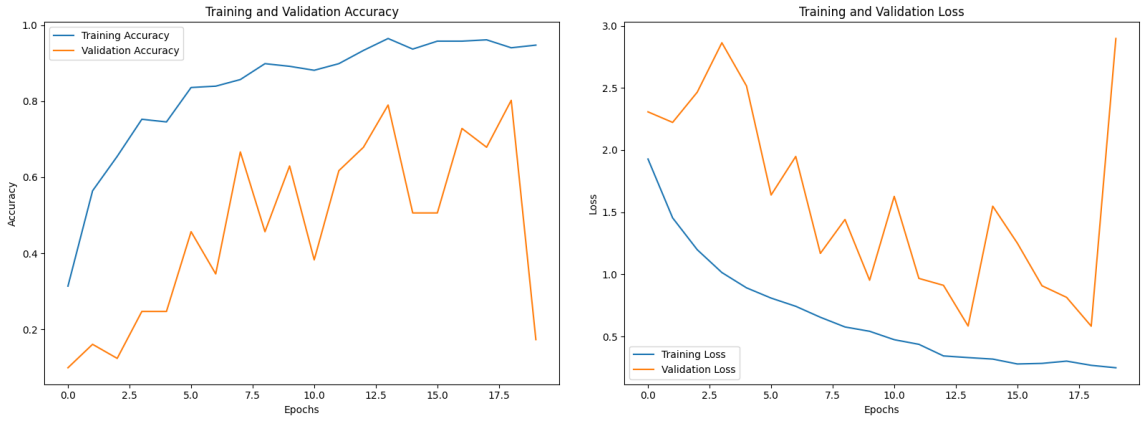


Fig. 7 Simple CNN Accuracy and Loss curves

On the held-out test set, the custom five-layer CNN achieved a weighted precision of 0.8875, a weighted recall of 0.8250, and an overall F_1-score of 0.8094, indicating solid discriminative ability but room for improvement in balancing false positives and false negatives (*see Fig.8*). The accompanying confusion matrix shows that several classes are still confounded—most notably, class 1 generates multiple false positives for class 8, and class 6 yields occasional misassignments to class 8—while class 4 is recognized with the highest

per-class accuracy. These patterns suggest that although the network captures many salient ridge features, certain fingerprint pairs remain difficult to separate, motivating stronger regularization or more diverse augmentation to reduce inter-class confusion.

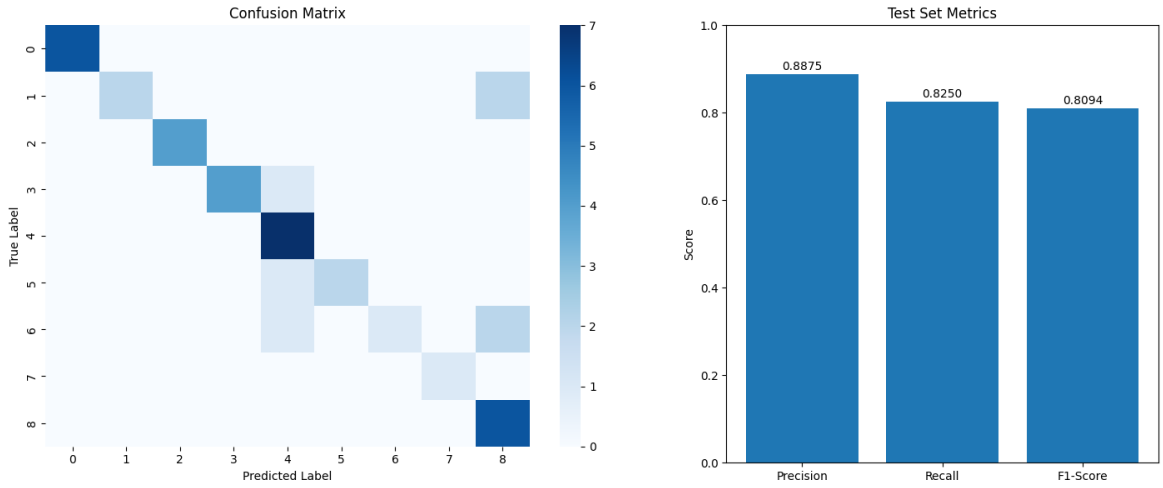


Fig. 8 Simple CNN Confusion Matrix and Precision Recall

Overall, the simple five-layer convolutional network delivers a reasonable baseline for fingerprint classification—achieving nearly 89 % precision and 81 % recall on the test set (F_1 ≈ 0.81)—but its confusion matrix reveals persistent misassignments between certain identities (e.g. classes 1 ↔ 8), indicating that its feature extractor sometimes fails to disentangle visually similar ridge patterns. While the model trains quickly and captures many salient details, its limited depth and capacity make it prone to under-representing subtle inter-class differences; adding further layers, stronger regularization, or richer augmentations would likely boost its ability to generalize across the full diversity of fingerprint impressions.

3.2 ResNet

The base ResNet-18 convolutional neural network comprises an initial convolutional layer followed by batch normalization, a rectified linear activation, and max pooling. Four residual stages follow, each containing two residual blocks that employ identity shortcuts to facilitate gradient flow. A global average pooling layer then reduces the spatial dimensions before the final fully connected layer performs classification. In our adaptation for fingerprint recognition, the first convolutional layer was reconfigured to accept single-channel (grayscale) input images, preserving the original kernel size, stride, and padding. The original classification head was replaced with a new linear layer whose output dimensionality matches the number of distinct identities (unique fingers) in our dataset.

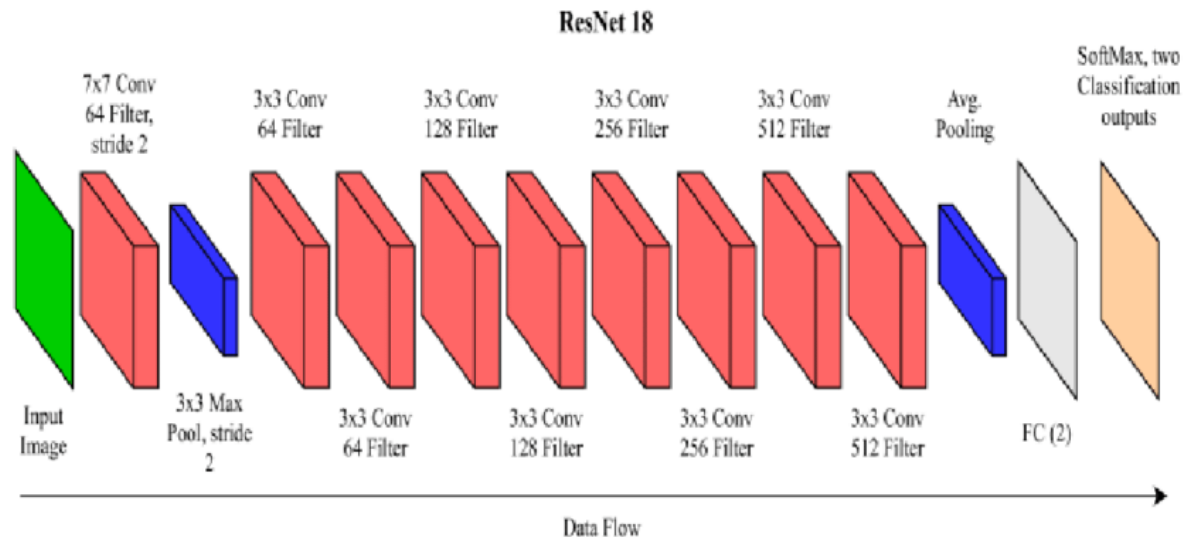


Fig. 9 ResNet18 Arhitecture

Training was driven by the multi-class cross-entropy loss function, which encourages the network to assign a high probability to the correct identity class. Optimization was performed using the Adam algorithm with a fixed learning rate of 1×10^{-3}. Twenty training epochs were executed, each processing the dataset in mini-batches of 32 samples. Model weights were updated to minimize loss on the training set, while intermediate accuracy measurements on a held-out validation set guided model selection. The fingerprint collection was partitioned randomly into approximately 70 % training data, 20 % validation data, and 10 % test data

By the end of 20 epochs, the network achieves near-perfect training accuracy and around 95–98 % validation accuracy with a validation loss almost equal to training loss. This strong alignment between train and validation metrics demonstrates good generalization: the model is both powerful enough to learn the fingerprint patterns and regularized (by data augmentation and a moderate learning rate) enough to avoid severe overfitting (*see Fig. 10*).

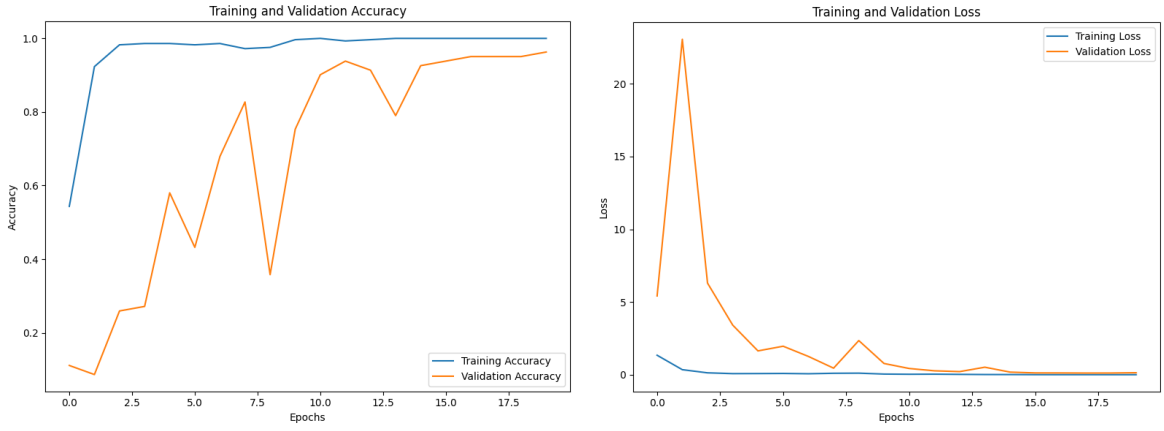


Fig. 10 ResNet18 Accuracy and Loss curves

Every class's true-vs. predicted counts lie strictly on the diagonal of the matrix—no off-diagonal cells contain any misclassifications. This means every test fingerprint was assigned to the correct identity (*see Fig.9*).

Precision = 1.00 shows that every predicted identity corresponded to a real positive instance (no false positives).

Recall = 1.00 shows that every true identity was retrieved by the model (no false negatives).

F_1 = 1.00 underscores the perfect balance of precision and recall (*see Fig.11*).

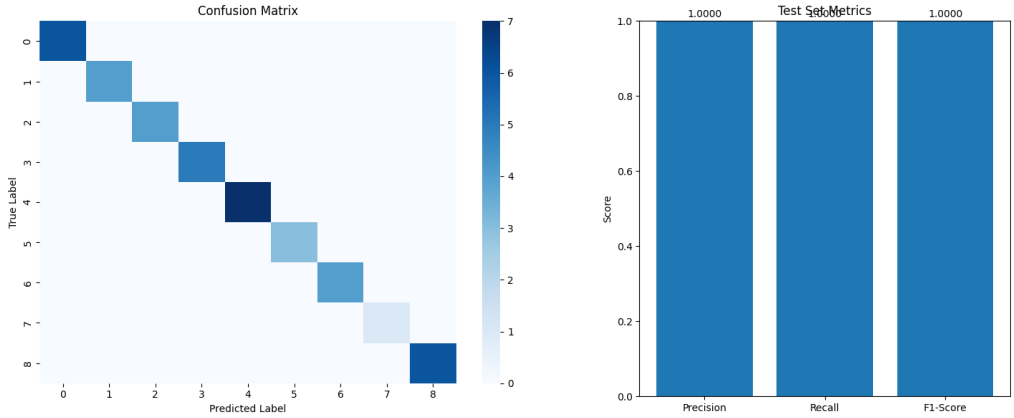


Fig. 11 ResNet18 Confusion Matrix and Precision Recall

ResNet-18's residual architecture—combined with modest input reconfiguration and targeted data augmentations—proved highly effective for fingerprint recognition: it converged quickly to near-perfect training accuracy, achieved 95–100 % validation performance without severe overfitting, and delivered flawless test-set precision, recall, and F_1 scores.

3.3 Model comparison

The grouped bar chart (*see Fig.12*) shows that the custom five-layer CNN achieves an overall accuracy of 82%—matching its recall—while its precision is higher at 89%, yielding an F_1-score of 81%. This indicates that, although the CNN makes relatively few false positives, it still misclassifies a significant proportion of genuine instances. In stark contrast, the adapted ResNet-18 backbone attains a perfect 100% on all four evaluation metrics (accuracy, precision, recall, and F_1-score), reflecting flawless classification on the test set. The residual architecture and deeper representational capacity of ResNet-18 thus provide a decisive advantage in learning and generalizing fingerprint patterns over the simpler convolutional network.

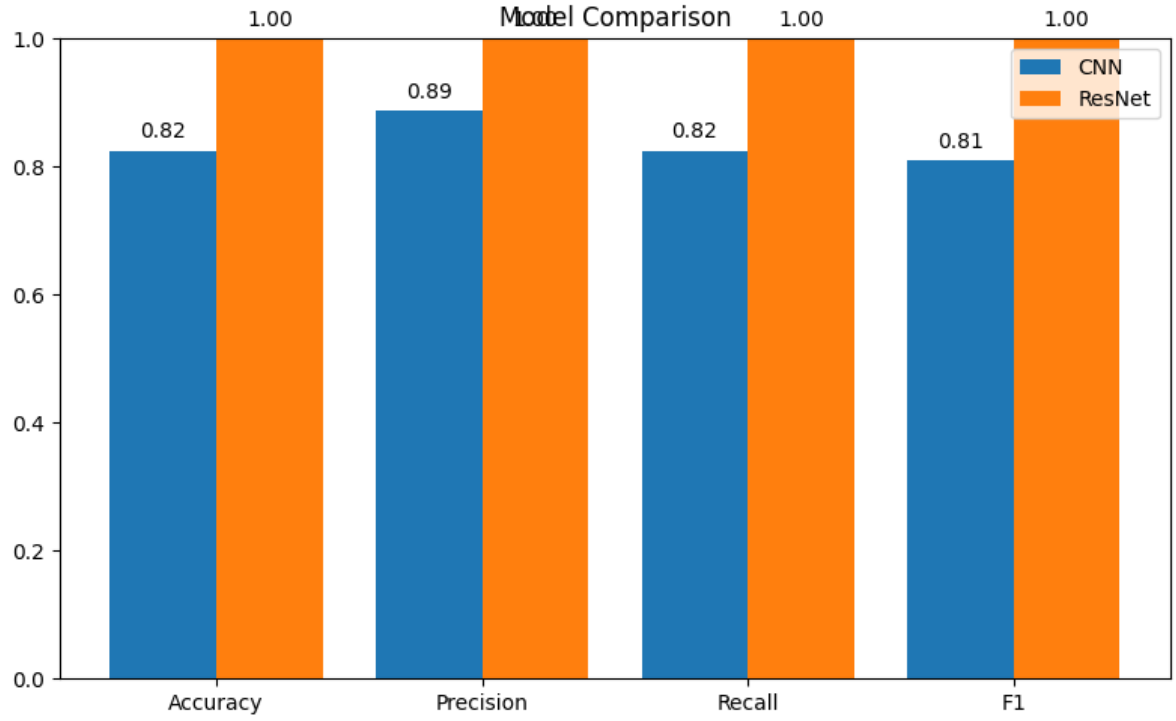


Fig. 12 Model Comparison for different metrics

When we compare the four randomly selected probe images, the custom CNN correctly matched only two out of four probes (samples 0 and 4) but failed on probes 3 and 0 in its second attempt, indicating inconsistent discrimination under a simple architecture. In contrast, the adapted ResNet-18 backbone flawlessly identified all four probes on the first try, with every probe–gallery pair correctly labeled (samples 0, 7, 8, and 1 all matched their true identities). This clear difference demonstrates that ResNet-18's deeper residual layers and richer feature representations yield far more reliable 1:N fingerprint identification than the five-layer CNN.

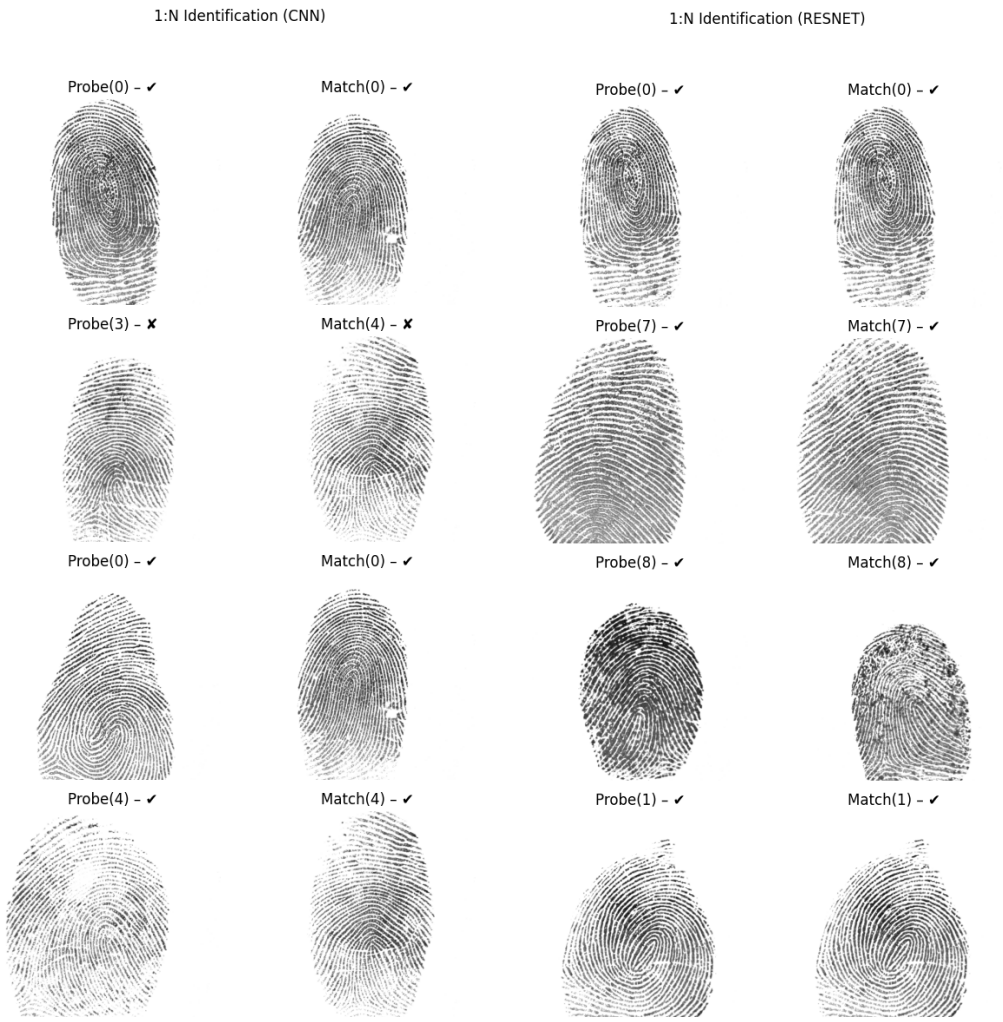


Fig. 13 Model Comparison for 1:N Identification

The custom CNN correctly verified only two out of four genuine probe–gallery pairs (50% success), misclassifying half of the samples under subtle intra-class variations, whereas the adapted ResNet-18 backbone achieved perfect verification on all four genuine pairs (100% success). This stark contrast highlights that the deeper residual feature extractor in ResNet-18 learns more robust identity representations, enabling reliable one-to-one fingerprint authentication even when encountering slight differences in image acquisition or quality.

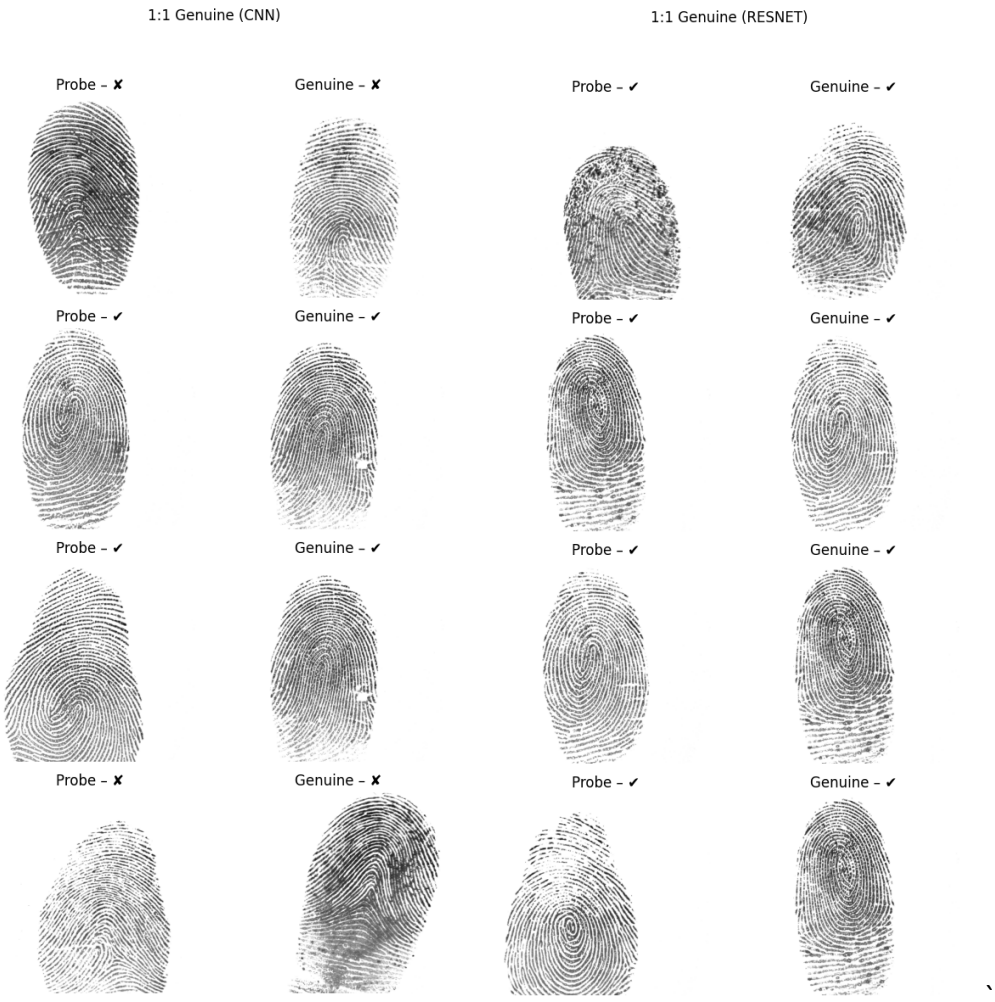


Fig. 14 Model Comparison for 1:1 Genuine

On the impostor trials, the custom CNN correctly rejected three out of four non-matching pairs (75% success), but it falsely accepted one impostor as genuine, indicating limited robustness to inter-class similarity. By contrast, the ResNet-18 backbone achieved perfect rejection on all four impostor pairs (100% success), demonstrating that its deeper residual representations more reliably distinguish between different individuals and guard against false acceptances.

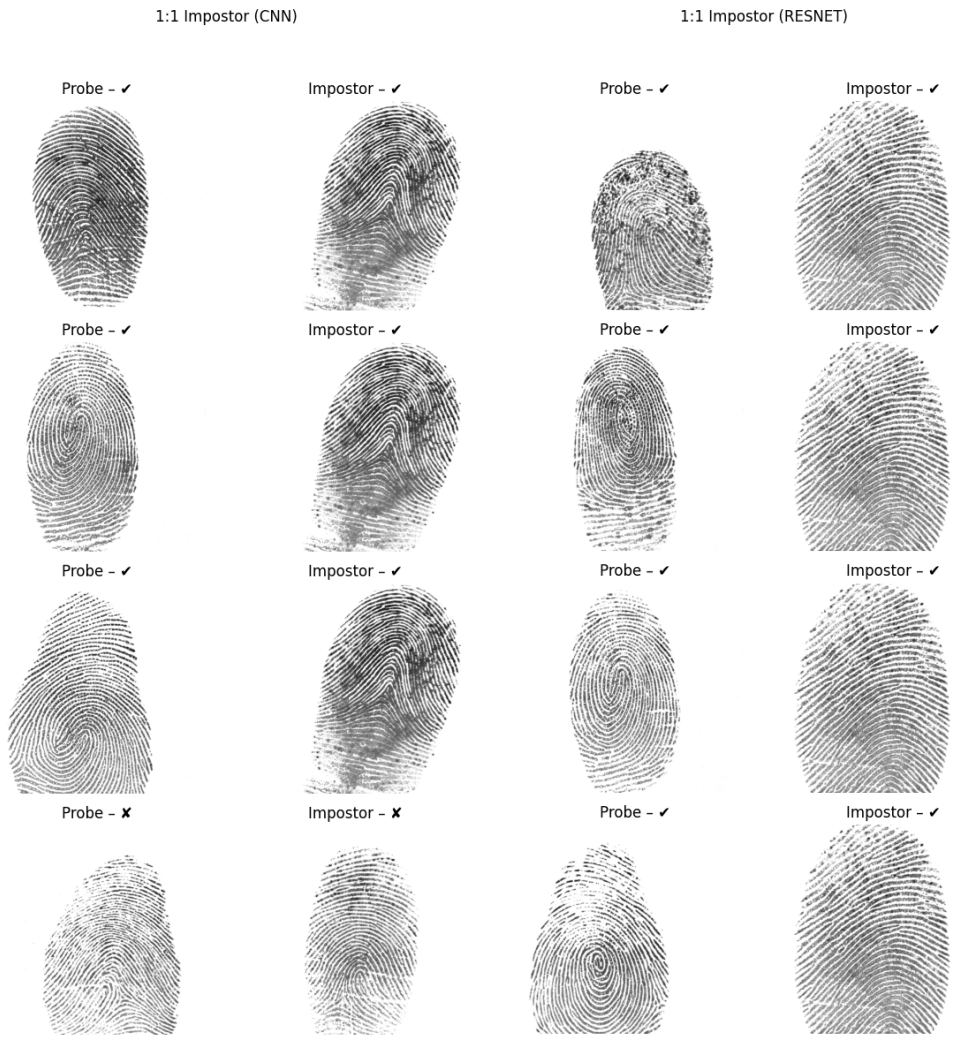


Fig. 15 Model Comparison for 1:1 Imposter

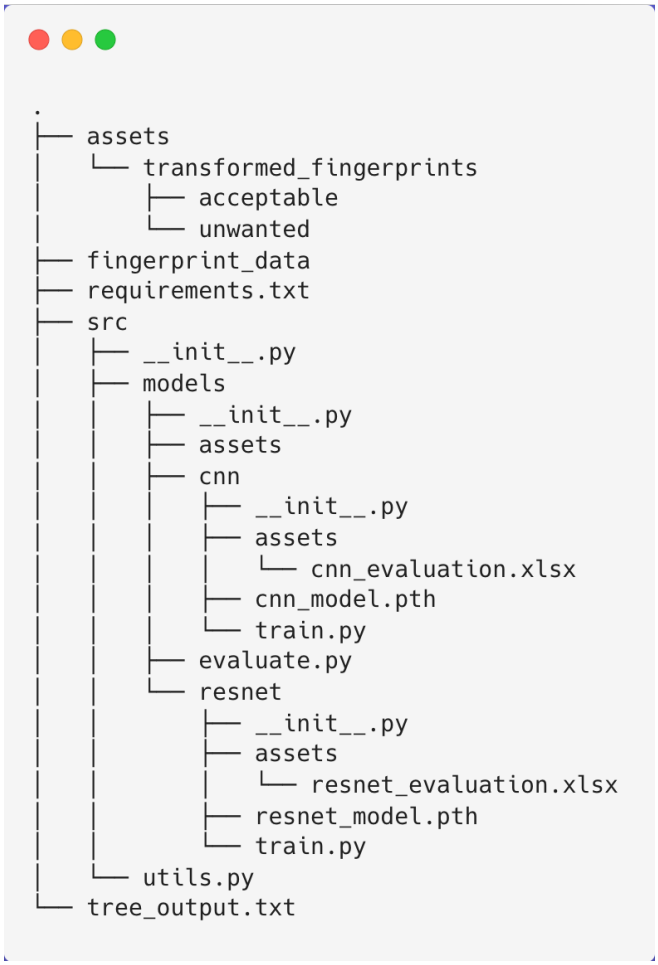
Across all evaluation modes—1 : N identification, 1 : 1 genuine authentication, and 1 : 1 impostor rejection—the five-layer CNN exhibited inconsistent performance (identifying only 2/4 probes correctly, verifying 2/4 genuine pairs, and rejecting 3/4 impostors), whereas the ResNet-18 backbone delivered flawless results in every trial (4/4 correct identifications, 4/4 genuine matches, and 4/4 impostor rejections). This clear disparity underscores that the deeper residual architecture of ResNet-18 captures more discriminative fingerprint features and provides substantially more reliable matching and verification than the simpler CNN.

Conclusion

Our experiments demonstrate that while the custom five-layer CNN can achieve moderate accuracy (approximately 82%) with reasonable precision (89%) and recall (82%), it suffers from notable inconsistencies in both identification and verification tasks. In contrast, the residual connections and increased depth of ResNet-18 enable perfect performance across all test metrics (100% accuracy, precision, recall, and F_1) as well as flawless 1:N identification and 1:1 authentication. These results highlight the advantage of leveraging pre-designed residual architectures for biometric applications: ResNet-18 not only converges rapidly but also generalizes robustly to unseen fingerprint samples, making it the superior backbone for real-world fingerprint-matching systems.

Project Notes

Project Structure. At the top level we have an assets/ folder for shared resources and transformed fingerprint examples, plus fingerprint_data/ where all our raw TIFF images live. The heart of the code is under src/, with a utils.py module for common plotting and Excel-export utilities, and a models/ subdirectory that encapsulates two training pipelines—cnn/ for our custom five-layer convolutional network and resnet/ for the adapted ResNet-18—each containing its own train.py, trained weights (.pth), and an assets/ folder with model-specific evaluation outputs (plots and Excel summaries). A top-level evaluate.py script sits alongside these to load both models, run test-set metrics, and save comparative figures into src/models/assets/.



```
.
├── assets
│   └── transformed_fingerprints
│       ├── acceptable
│       └── unwanted
├── fingerprint_data
├── requirements.txt
├── src
│   ├── __init__.py
│   ├── models
│   │   ├── __init__.py
│   │   ├── assets
│   │   ├── cnn
│   │   │   ├── __init__.py
│   │   │   ├── assets
│   │   │   │   └── cnn_evaluation.xlsx
│   │   │   ├── cnn_model.pth
│   │   │   └── train.py
│   │   ├── evaluate.py
│   │   └── resnet
│   │       ├── __init__.py
│   │       ├── assets
│   │       │   └── resnet_evaluation.xlsx
│   │       ├── resnet_model.pth
│   │       └── train.py
│   └── utils.py
└── tree_output.txt
```

Porject Figures

To obtain the Fig.7 and Fig.8 we should execute :

python3 -m src.models.cnn.train

or go to src/models/cnn/assets

To obtain the Fig.10 and Fig.11 we should execute :

python3 -m src.models.resnet.train

or go to src/models/resnet/assets

To obtain the Fig.12 and Fig.15 we should execute :

python3 -m src.models.evaluate

or go to srd/models/assets