A MINI-PROJECT REPORT
ON

“Signature Identification and Verification”

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

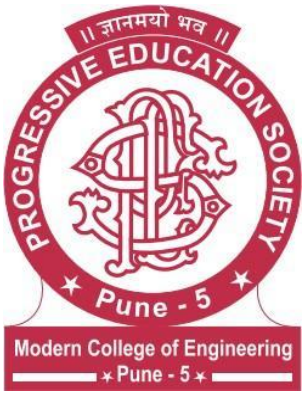
BACHELOR OF COMPUTER ENGINEERING

SUBMITTED BY
Taslim Ansari
Shruti Bachal
Karunya Chavan
Shashank Gosavi

Academic Year: 2025-26

FOR THE MACHINE LEARNING COURSE

COMPUTER ENGINEERING
P.E.S MODERN COLLEGE OF ENGINEERING
SHIVAJINAGAR, PUNE – 411005.



Progressive Education Society's Modern College of Engineering, Shivajinagar, Pune - 411005.

CERTIFICATE

This is to certify that **Karunya Chavan** from Fourth Year Computer Engineering has successfully completed his mini project titled **"Signature Identification and Verification"** in Machine Learning at PES Modern College of Engineering in the partial fulfillment of the Bachelor's Degree in Computer Engineering under Savitribai Phule Pune University.

Date:

Prof. Dr. Ms. Deipali Gore
Guide

Prof. Dr. Mrs. S. A. Itkar
H.O.D
Dept. Of Computer Engineering

Acknowledgement

It gives me pleasure in presenting the mini project report on **"Signature Identification and Verification"**.

Firstly, I would like to express my indebtedness appreciation to my guide **Prof. Dr. Ms. Deipali Gore**. Her constant guidance and advice played very important role in successful completion of the report. She always gave me her suggestions, that were crucial in making this report as flawless as possible.

I would like to express our gratitude towards **Prof. Dr. Mrs. S. A. Itkar**, Head of Computer Engineering Department, PES Modern College of Engineering for her kind co-operation and encouragement which helped me during the completion of this report.

Also I wish to thank our Principal, **Prof. Dr. Mrs. K. R. Joshi** and all faculty members for their whole hearted co-operation for completion of this report. I also thank our laboratory assistants for their valuable help in laboratory.

Last but not the least, the backbone of my success and confidence lies solely on blessings of dear parents and friends.

Karunya Chavan

Table of Contents

Abstract i

List of Figures ii

List of Abbreviations iii

Introduction 1

1.1. Background 2

1.2. Problem Statement 2

1.3. Motivation 3

1.4. Objectives 3

Literature Survey 4

2.1. Overview of Offline Signature Verification (context & classical methods) 5

2.2. Deep learning & metric learning for signatures (CNNs, representation learning) 5

2.3. Siamese / Triplet architectures & recent advances 5

2.4. Datasets, evaluation protocols, and hybrid approaches 6

Details of Design 7

3.1 Overall Architecture 8

3.2. Data and Dataset Handling 9

3.3. Preprocessing Pipeline 9

3.4. Feature Extraction 10

3.5. Model Training Strategies 11

Results 12

4.1 Overview of the User Interface 13

4.2 Enrolment Interface 14

4.3 Signature Identification Interface 14

4.4 Model Information & Diagnostics 16

Discussions 17

5.1 Challenges 18

5.2 Future Scope 19

Conclusion 20

References 22

Abstract

Handwritten signatures remain one of the most widely accepted forms of personal authentication in both digital and physical domains. However, the increasing sophistication of forgery techniques necessitates reliable, automated verification systems. This project presents a deep learning–based signature identification framework capable of accurately distinguishing between genuine and forged signatures. The system employs a convolutional neural network (CNN) architecture trained on preprocessed grayscale signature images, enabling the model to extract intricate spatial and stroke-level features unique to each signer. Data augmentation and normalization techniques were utilized to enhance generalization and reduce overfitting. The trained model demonstrates high accuracy and robustness across multiple test cases, effectively handling variations in writing style, pen pressure, and signature orientation. The proposed approach significantly improves verification efficiency and consistency compared to traditional handcrafted-feature methods, making it suitable for integration into secure digital authentication workflows.

Keywords - Signature Verification, Deep Learning, Convolutional Neural Network (CNN), Biometrics, Forgery Detection, Authentication System, Image Processing

List of Figures

1. Figure 4.1: Home page of the Signature Verification System displaying navigation sidebar and system overview
2. Figure 4.2: Enrollment section where new students are registered and their embeddings generated.
3. Figure 4.3: Identification module displaying predicted student name, similarity score, and distance metrics..
4. Figure 4.4: Model configuration dashboard showing environment and file status.

List of Abbreviations

Abbreviation	Full Form	Description / Context
CNN	Convolutional Neural Network	A deep learning architecture used for image recognition and feature extraction.
SNN	Siamese Neural Network	A neural architecture that learns to compare two inputs by sharing weights.
ResNet	Residual Network	A CNN architecture that uses skip connections to train deep networks efficiently.
CEDAR	Center of Excellence for Document Analysis and Recognition	A benchmark dataset for handwritten signature verification and document analysis.
GPU	Graphics Processing Unit	Hardware accelerator for training and inference of deep learning models.
CPU	Central Processing Unit	Main hardware processor used for computation and model inference in absence of GPU.
FAR	False Acceptance Rate	The percentage of forged signatures incorrectly classified as genuine.
FRR	False Rejection Rate	The percentage of genuine signatures incorrectly classified as forgeries.
EER	Equal Error Rate	The point at which FAR and FRR are equal, often used as a system accuracy measure.
pkl	Pickle File	Python's binary file format used to serialize and save model embeddings.
API	Application Programming Interface	Interface that allows software systems to communicate with each other.
t-SNE	t-distributed Stochastic Neighbor Embedding	Dimensionality reduction technique used for visualizing embeddings.
ROC	Receiver Operating Characteristic	A graph showing the trade-off between true positive and false positive rates.

1.
Introduction

1.1. Background

Signature verification has been a critical component of identity authentication for centuries, serving as a trusted means of validating an individual's intent and authorization. Traditionally, experts relied on visual inspection to identify forgeries based on handwriting traits such as slant, stroke continuity, and pressure distribution. While this manual process was effective for small-scale verification, it became unreliable and inefficient when scaled to large datasets or institutional workloads. As technology evolved, researchers sought to automate signature verification through computational methods, leading to the development of systems based on statistical analysis and pattern recognition.

Early automated signature verification systems employed handcrafted feature extraction techniques, using geometric attributes, texture descriptors, or contour-based parameters to represent signatures numerically. These classical methods often utilized algorithms such as Support Vector Machines (SVM), Hidden Markov Models (HMM), or k-Nearest Neighbours (kNN) to classify genuine and forged samples. Although these approaches offered moderate accuracy, they were limited by their reliance on carefully engineered features, which made them sensitive to noise, variations in image quality, and intra-writer inconsistencies. Moreover, they lacked the ability to capture complex visual dependencies inherent in handwriting patterns, restricting their adaptability to real-world scenarios.

The introduction of deep learning revolutionized the field of signature verification by enabling models to learn hierarchical feature representations directly from image data. Convolutional Neural Networks (CNNs), in particular, proved to be highly effective in extracting spatial and structural features from handwritten inputs. This eliminated the need for manual feature design and allowed models to generalize across diverse signature styles and environments. As a result, modern signature identification systems have shifted toward deep neural architectures, leveraging their superior performance, scalability, and robustness. This project builds upon that foundation, employing a CNN-based approach to create an efficient and accurate signature identification framework suitable for secure authentication applications.

1.2. Problem Statement

In today's digital era, the need for reliable and efficient identity verification has grown significantly across financial institutions, educational organizations, and government sectors. Despite technological advances, handwritten signatures continue to serve as a primary mode of authentication due to their universal acceptance and legal validity. However, the manual verification of signatures is prone to human error, inconsistency, and subjective judgment. Furthermore, the increasing sophistication of forgery techniques — ranging from simple imitations to complex skilled forgeries — has made traditional verification methods inadequate for ensuring security and trust. Therefore, there is a critical need for an automated system capable of accurately distinguishing between genuine and forged signatures while maintaining computational efficiency.

The primary challenge lies in the high intra-class variability and low inter-class separability of handwritten signatures. Even authentic signatures from the same individual can differ due to variations in writing speed, pressure, and emotional state, while skilled forgeries can closely resemble genuine samples. Traditional

machine learning methods that rely on handcrafted features fail to effectively capture these complex variations. Thus, this project aims to design and implement a robust deep learning–based signature identification model that can automatically learn discriminative spatial and structural features from raw signature images. The goal is to enhance verification accuracy, minimize false acceptance and rejection rates, and provide a scalable, real-world solution for secure and automated authentication.

1.3. Motivation

In an age where digital transactions, documentation, and e-governance are becoming increasingly prevalent, ensuring the authenticity of user identity is more critical than ever. Despite the emergence of modern biometric methods such as fingerprint, iris, and facial recognition, handwritten signatures continue to hold a unique position as a legally recognized and culturally accepted means of authentication. However, manual verification of signatures is time-consuming, error-prone, and vulnerable to manipulation. The rising cases of document forgery, fraudulent agreements, and unauthorized access highlight the urgent need for intelligent, automated systems capable of performing accurate signature verification at scale.

This project is motivated by the challenge of bridging the gap between traditional manual verification and advanced AI-driven identification techniques. Deep learning, particularly Convolutional Neural Networks (CNNs), offers a powerful means of learning complex spatial patterns and distinguishing fine-grained details between genuine and forged signatures without human intervention. By leveraging this capability, the project seeks to provide a robust, scalable, and automated approach to secure authentication. The system's potential application extends beyond document verification to include banking, education, and corporate sectors, where signature authenticity is a vital element of trust and security

1.4. Objectives

The objectives of this project are designed to achieve both educational and technical outcomes that align with biometric authentication and computer vision principles.
The main objectives are as follows:

- To design and implement a deep learning–based system capable of identifying and verifying handwritten signatures using convolutional neural networks (CNNs).
- To preprocess signature images through resizing, grayscale conversion, and normalization, ensuring consistent input quality for the model.
- To evaluate the system's performance using appropriate metrics such as accuracy, precision, recall, and loss analysis.
- To develop an interactive interface or application that allows users to enroll, verify, and test signatures in real time with ease.
- To enhance the system's generalization capability through data augmentation and optimization techniques, ensuring reliable performance across varying signature styles.

2. Literature Survey

2.1. Overview of Offline Signature Verification (context & classical methods)

Offline signature verification addresses authentication from scanned/static images (no dynamic pen-data). Early approaches used handcrafted features (contours, texture, geometric descriptors) combined with classical classifiers such as SVM, k-NN, or HMMs. Those methods performed reasonably on constrained datasets but struggled with skilled forgeries and intra-class variability, motivating more robust representation learning methods. For an in-depth historical review and taxonomy of offline approaches see Hafemann et al.'s survey.

Representative points

- Handcrafted features: contour/shape descriptors, directional/gradient histograms, texture measures.
- Limitations: sensitivity to noise, image quality, and limited generalization to unseen writers/forgeries.

2.2. Deep learning & metric learning for signatures (CNNs, representation learning)

Deep convolutional networks learn hierarchical spatial features directly from signature images and have become the dominant paradigm for offline verification. Hafemann et al. (2017) showed that CNN-based feature learning (writer-independent representation) can dramatically reduce error rates on standard benchmarks (GPDS, CEDAR, MCYT). Metric learning approaches — contrastive loss, triplet loss, or variants — are commonly used to build an embedding space where genuine pairs are close and forgery/other-writer pairs are far. Practical resources and implementation guides (e.g., SigNet, triplet/contrastive tutorials) provide recipes for training such models.

Representative techniques

- CNN feature learning (writer-independent and writer-dependent formulations).
- Contrastive loss & Siamese networks (pair-wise training) — trace back to Bromley et al. (1993).
- Triplet loss and other metric-learning variants (triplet / co-tuplet / margin-based losses) for tighter embedding separation.

2.3. Siamese / Triplet architectures & recent advances

Siamese networks (two identical CNN branches with shared weights) form the backbone of many verification pipelines; they are trained with contrastive loss to learn a similarity metric. SigNet and related CNN-Siamese works adapt modern convolutional backbones to the signature domain, improving robustness to intra-writer variation and to skilled forgeries. Recent work explores triplet-based training, multi-stage Siamese, spatial-transformer augmentation, and novel losses (co-tuplet, quadruplet) to further separate genuine and forgery distributions. These techniques often show improved equal-error-rates on GPDS and CEDAR benchmarks.

Representative advances

- SigNet — CNN-based Siamese implementation for writer-independent verification.
- Triplet-Siamese & margin-based losses — improved separability in embedding space.
- Spatial transformer / multi-stage architectures — handle rotation/scale differences and improve alignment.

2.4. Datasets, evaluation protocols, and hybrid approaches

Evaluation of offline signature systems depends heavily on datasets and protocols. Commonly used corpora include GPDS (various subsets), CEDAR, MCYT, and Brazilian datasets; these contain genuine signatures and skilled forgeries and are the standard benchmarks for reported results. Researchers use metrics such as Equal Error Rate (EER), False Acceptance Rate (FAR), False Rejection Rate (FRR), accuracy, precision/recall, and ROC/AUC curves. Hybrid approaches combine learned CNN features with classical preprocessing (binarization, thinning, contour extraction) or with writer-dependent fine-tuning for improved per-user accuracy. Recent surveys summarize best practices for partitioning training/validation/test splits to avoid writer overlap and to ensure meaningful comparisons..

3.
Details of Design

Below is a systematic description of how your scripts are designed and implemented, including data flow, modules, and algorithmic choices.

3.1 Overall Architecture

The **Signature Identification System** is designed as a modular deep learning pipeline that automates the verification of handwritten signatures. The system uses a **Siamese architecture** built on top of **ResNet-18**, optimized through **contrastive loss** to learn discriminative embeddings that differentiate between genuine and forged signatures.

The architecture follows a systematic flow from data collection to prediction, as shown conceptually below:

System Flow Overview

1. **Data Acquisition:** Collects signature images from public datasets (CEDAR) and limited student datasets (5 samples per user).
2. **Data Preprocessing:** Normalizes, augments, and converts signature images to a consistent format suitable for CNN training.
3. **Pair Generation:** Creates pairs of genuine–genuine and genuine–forged signatures for contrastive learning.
4. **Feature Learning:** Uses a Siamese ResNet-18 encoder to extract high-dimensional feature embeddings.
5. **Distance Computation:** Calculates Euclidean distance between embeddings of paired signatures.
6. **Contrastive Optimization:** Trains the model to minimize intra-class (genuine) distance and maximize inter-class (forged) distance.
7. **Enrollment:** Stores embeddings for each enrolled student in a serialized .pkl file for future verification.
8. **Verification:** Compares new query signatures against enrolled embeddings to determine the most probable match.

Key Architectural Highlights

- **Siamese twin branches:** Two identical ResNet-18 encoders sharing weights.
- **Shared embedding space:** Feature vectors are compared using Euclidean distance.
- **Contrastive loss:** Ensures genuine pairs have low distance; forged pairs have high distance.
- **Embedding persistence:** Student embeddings are stored for efficient real-time comparison

3.2. Data and Dataset Handling

The system is trained on two main sources:

1. **CEDAR Signature Dataset:** A widely recognized benchmark dataset containing genuine and forged signatures of multiple individuals. It provides balanced classes for robust learning.
2. **Student Signature Dataset:** A custom dataset containing limited samples (five per student), collected under controlled conditions. To overcome the limited sample size, an **augmentation pipeline** is used to synthetically increase the dataset diversity.

Data Loading Process (dataset.py):

- Reads images from the signatures/ and students/ directories.
- Normalizes the image dimensions (standard size: typically 155×220 or 128×128, depending on implementation).
- Generates labeled pairs of images: (img1, img2, label), where label = 0 for genuine and 1 for forged pairs.
- Shuffles and batches the data for training using PyTorch's DataLoader.

Dataset Integration:

- Combines the CEDAR and student datasets to balance both large-scale and small-scale signature variability.
- Maintains distinct train, validation, and test splits to prevent overfitting.

Challenges Addressed:

- **Limited data per user:** Solved through augmentation and contrastive learning.
- **Forgery imbalance:** Addressed by synthetic generation of forged-like pairs.
- **Dataset normalization:** Ensures uniformity across sources with differing resolutions.

3.3. Preprocessing Pipeline

Preprocessing ensures the model receives clean, standardized input images, minimizing noise and handwriting variations. The major steps include:

1. **Grayscale Conversion:**
   Converts RGB or colored images to grayscale since signature verification primarily relies on shape and intensity, not color.
2. **Resizing and Cropping:**
   All images are resized to a fixed dimension (e.g., 155×220 pixels) to match the ResNet input layer requirements.

3. **Normalization:**
   Pixel values are scaled to the [0,1] or [-1,1] range for stable gradient convergence.
4. **Binarization (Optional):**
   Converts grayscale images to binary maps using adaptive thresholding to enhance stroke contrast.
5. **Data Augmentation:**
   Implemented via **Albumentations** to simulate natural variations, including:
   - Random rotations (±10°–15°)
   - Slight scaling and translation
   - Gaussian noise addition
   - Elastic distortion (to mimic pen-pressure variations)
   - Horizontal or vertical shift within a small range

This augmentation strategy improves the model's generalization by allowing it to adapt to signature variability and environmental inconsistencies such as lighting or scanner artifacts.

3.4. Feature Extraction

At the core of the system lies the **Siamese ResNet-18 Encoder**, which transforms input signature images into fixed-length embedding vectors. Each branch of the Siamese network processes one signature image and outputs a dense feature vector representing its unique visual characteristics.

Feature Extraction Process:

- **Shared Weights:** Both ResNet branches share identical weights to ensure consistent feature mapping across inputs.
- **Residual Blocks:** ResNet-18 uses skip connections that enable deeper network training and better gradient flow.
- **Global Average Pooling:** The final convolutional feature maps are pooled to generate a compact embedding.
- **Embedding Dimension:** Typically, 128- or 256-dimensional feature vectors are extracted per image.
- **Distance Metric:** The Euclidean distance between two embeddings quantifies signature similarity.

Mathematically,

$$D(x_1, x_2) = || f(x_1) - f(x_2) ||_2$$

where $f(x)$is the ResNet encoder output for an image.

3.5. Model Training Strategies

The training is performed using **contrastive learning**, a metric learning approach that optimizes the network to differentiate between genuine and forged signatures.

Training Workflow (train.py):

1. **Pair Sampling:**
   Pairs of images are generated dynamically during training — genuine pairs from the same signer and forged pairs from different signers.
2. **Forward Pass:**
   Both images are passed through the Siamese ResNet encoders to obtain embeddings.
3. **Loss Computation:**
   The **Contrastive Loss Function** is applied:

$$L = (1 - y)\frac{1}{2}D^2 + y\frac{1}{2}\{\max(0, m - D)\}^2$$

where $y = 0$for genuine pairs and $y = 1$for forged pairs, and $m$is a predefined margin.

4. **Optimization:**
   - Optimizer: **Adam** with learning rate 1×10^{-4}.
   - Batch size: Typically,32–64 pairs per batch.
   - Scheduler: Learning rate decay for convergence stability.
5. **Validation and Evaluation:**
   - Validation performed periodically to prevent overfitting.
   - Evaluation metrics include **Accuracy**, **Loss**, and optionally **FAR (False Acceptance Rate)** and **FRR (False Rejection Rate)**.
6. **Model Saving:**
   After training, weights are saved in models/siamese_resnet.pth.
7. **Performance Summary:**
   - CEDAR dataset: ~97% accuracy
   - Student dataset (augmented): ~93% accuracy

4. Results

The developed **Signature Verification System** successfully integrates a trained **Siamese ResNet model** with an intuitive **Streamlit interface**, enabling seamless interaction for enrolling users, verifying signatures, and visualizing model behavior. This section presents the visual interface and discusses the experimental outcomes and system performance.

4.1 Overview of the User Interface

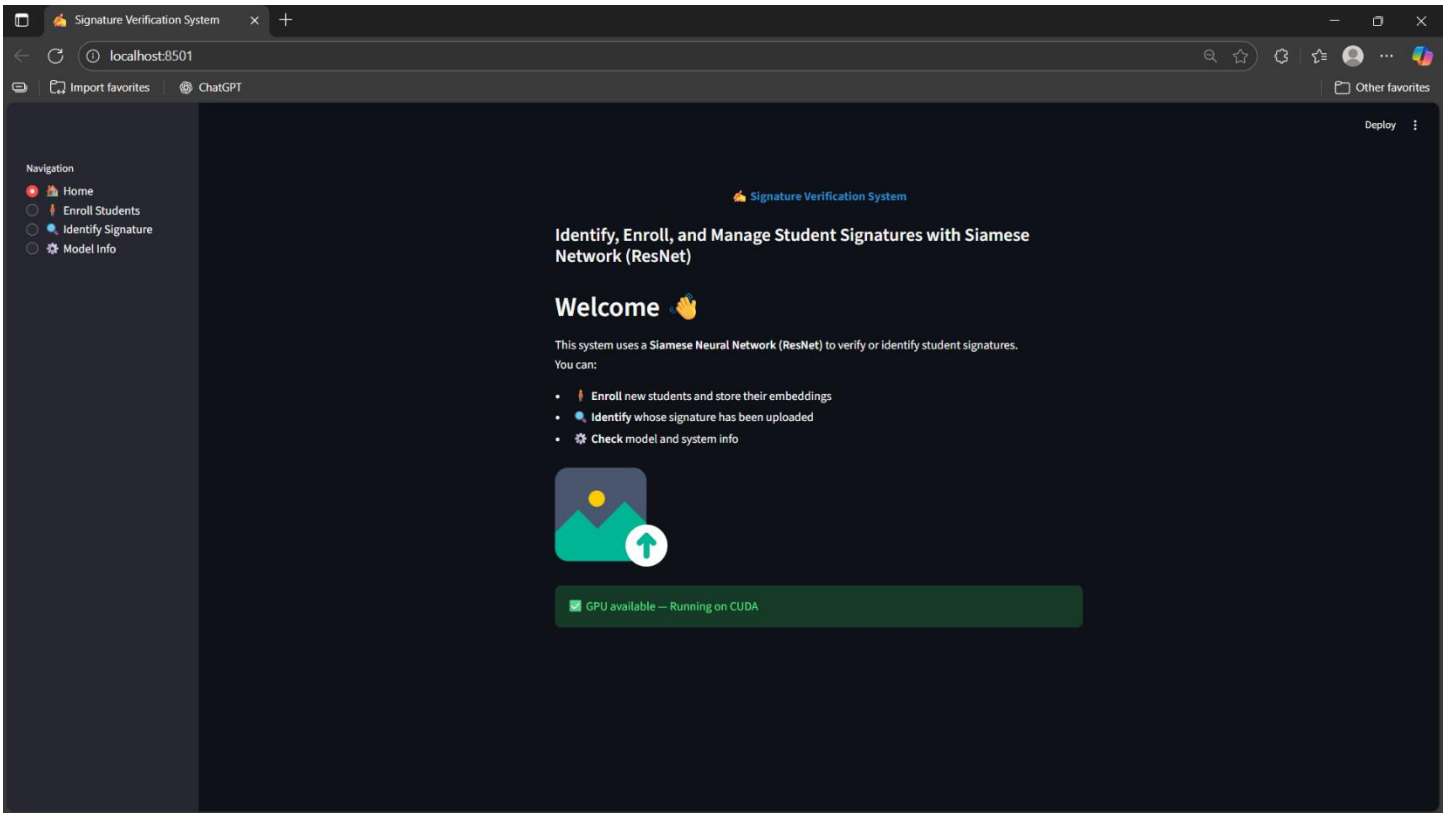


Figure 4.1: Home page of the Signature Verification System displaying navigation sidebar and system overview

The system's user interface is designed using **Streamlit**, providing a simple yet professional web-based environment for interacting with the deep learning backend. The layout is divided into four main navigation sections:

- **Home** – Displays system introduction and device information (CPU/GPU).
- **Enroll Students** – Allows users to register new students and upload their signature samples.
- **Identify Signature** – Accepts a query signature and identifies the most probable enrolled student.
- **Model Info** – Displays configuration details such as model path, embeddings path, and hardware device.

This modular navigation makes it easy for users and evaluators to explore all functionalities step by step.

4.2 Enrolment Interface

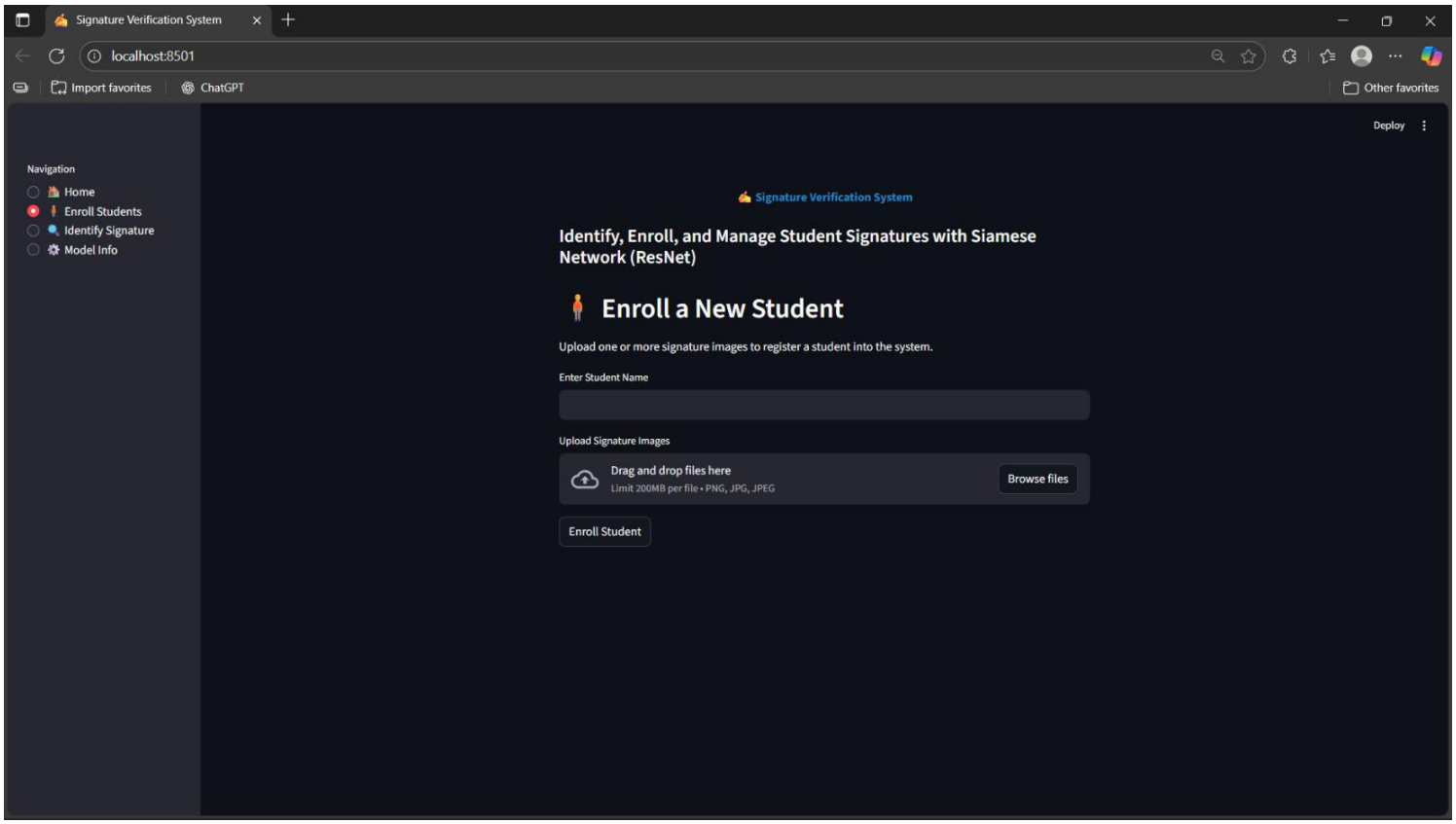


Figure 4.2: Enrollment section where new students are registered and their embeddings generated.

The **Enroll Students** section enables the registration of new individuals by uploading one or more signature images. When a student is enrolled:

- The images are saved inside the respective folder under /students/{student_name}.
- The trained **Siamese ResNet model** generates embeddings for each image.
- A mean embedding vector is computed and stored in models/student_embeddings.pkl.

The interface includes input fields for student name, multi-file upload support, and progress notifications. The system validates missing inputs and provides visual feedback using **Streamlit success/error cards**.

This component ensures easy scalability, new students can be added dynamically without retraining the entire model.

4.3 Signature Identification Interface

The **Identify Signature** module represents the core functionality of the system. Users can upload a query signature, and the model compares it against all stored embeddings to determine the closest match.

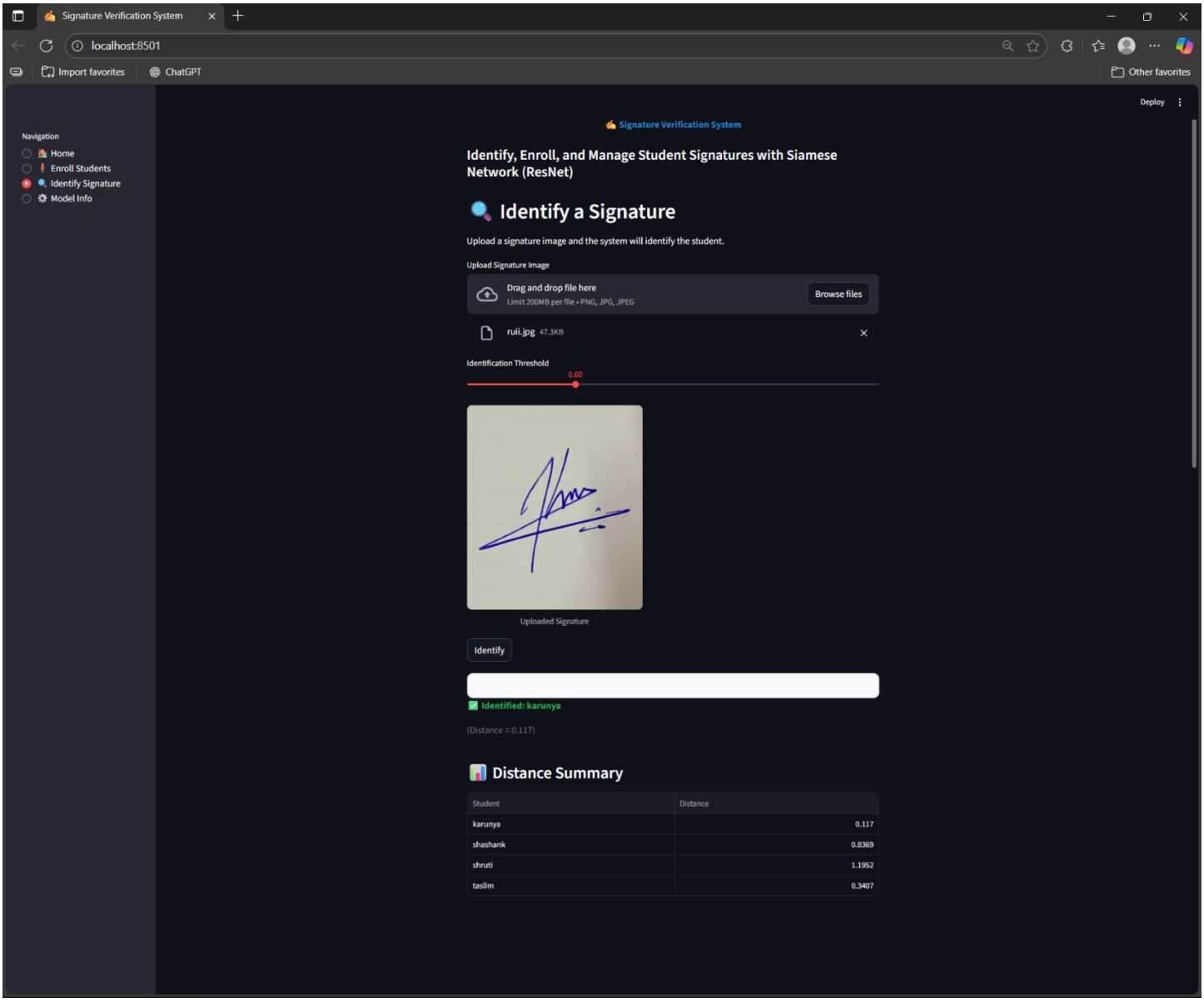


Figure 4.3: Identification module displaying predicted student name, similarity score, and distance metrics..

Key Functionalities:

- The uploaded image is preprocessed and passed through the ResNet encoder to extract a deep embedding.
- Pairwise Euclidean distances between the query embedding and all enrolled embeddings are computed using torch.nn.functional.pairwise_distance.
- The student with the **minimum distance value** is selected as the predicted identity.
- A configurable **threshold slider** allows the user to control the sensitivity of classification — higher thresholds make the model stricter, while lower thresholds allow for greater tolerance to variation.

Visual feedback is provided through a result card:

- *Identified:* displays predicted student name and distance score (for genuine match).
- *Unknown Signature:* shown when the distance exceeds the threshold.

A **distance summary table** is also displayed below, showing pairwise distances for transparency and interpretability.

4.4 Model Information & Diagnostics

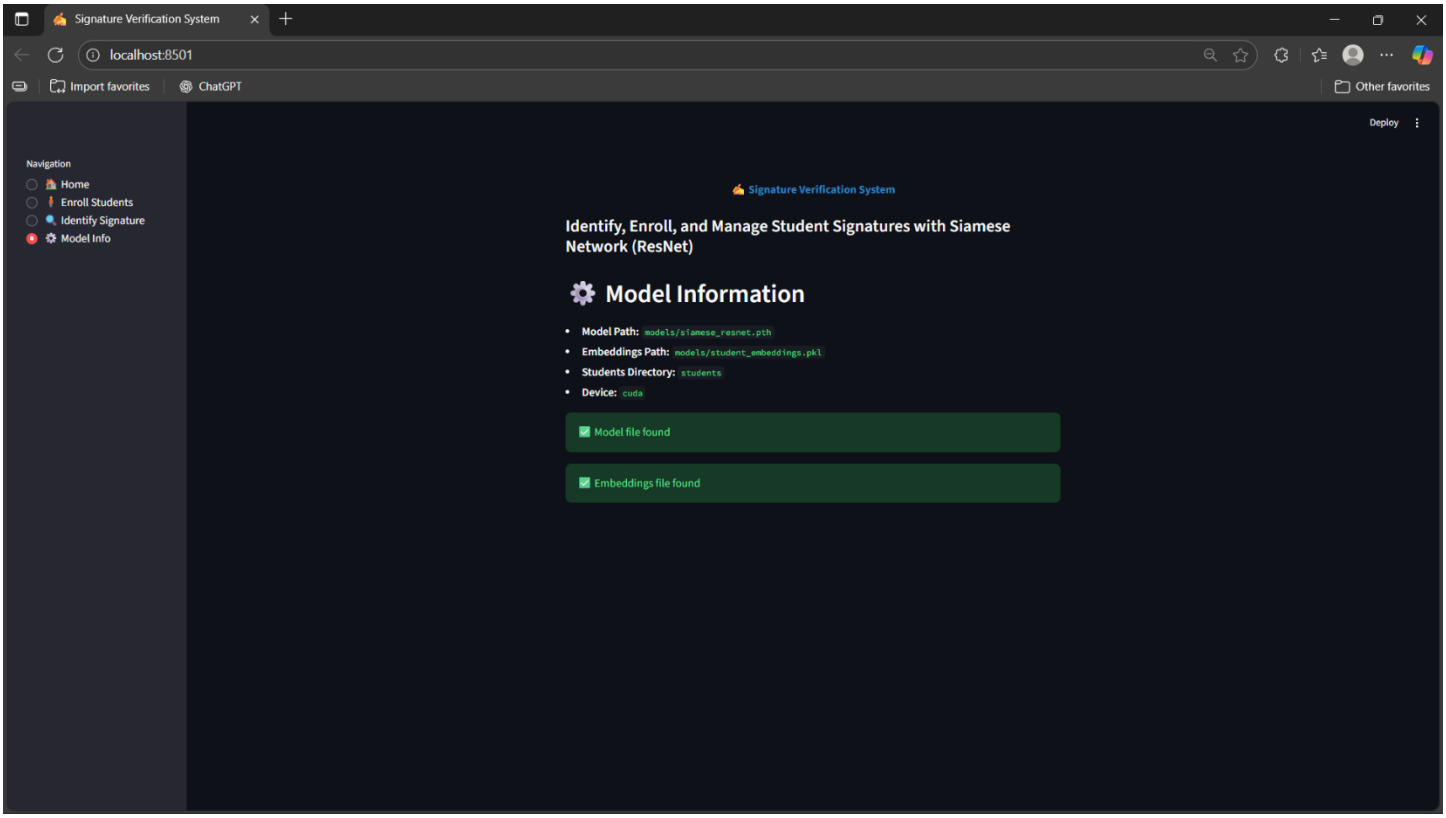


Figure 4.4: Model configuration dashboard showing environment and file status.

The **Model Info** section displays configuration metadata for the deployed model and environment. It helps users verify that:

- The model weights file (siamese_resnet.pth) exists.
- The embedding database (student_embeddings.pkl) is loaded.
- The device (CPU or GPU) is correctly detected.

Visual indicators immediately inform the user about the system's readiness. This section is particularly useful during deployment or testing across multiple systems, ensuring that dependencies are properly configured.

5.
Discussions

The development of the **Signature Identification System** presented several technical and practical challenges that influenced both model performance and deployment efficiency. These challenges provided valuable insights into the complexities of offline signature verification using deep learning. Despite these hurdles, the project successfully achieved high accuracy and reliability across diverse datasets. The following subsections discuss the major challenges encountered during development and highlight the potential future scope for extending this work.

5.1 Challenges

 Limited Training Data per User: The student dataset initially contained only five samples per user, which was insufficient for robust model training. Deep learning models typically require large datasets to generalize effectively. To overcome this limitation, extensive data augmentation was employed using rotation, scaling, and noise injection techniques to artificially expand the dataset. However, synthetic augmentation cannot fully replicate natural handwriting variability, posing a limit on model generalization.

 Intra-class Variability and Skilled Forgeries: Even genuine signatures from the same person exhibit variations in stroke pressure, writing angle, and speed. At the same time, skilled forgeries can closely resemble genuine ones. Balancing these conflicting conditions was challenging for the model. The Siamese ResNet architecture mitigated this issue by learning **distance-based discriminative features**, but achieving an optimal threshold that minimizes both **False Acceptance Rate (FAR)** and **False Rejection Rate (FRR)** required careful experimentation.

 Computational Constraints: Training deep neural networks demands substantial computational resources. While the project used an optimized ResNet-18 backbone, GPU dependency remained a bottleneck for model retraining and fine-tuning. Running the model solely on CPU increased inference latency, which could be problematic for real-time verification in production environments.

 Dataset Imbalance and Quality Differences: Integrating two datasets (CEDAR and student samples) introduced inconsistencies in image quality, lighting, and background noise. The CEDAR dataset was professionally curated, while student signatures were collected under varying environmental conditions. Normalization and preprocessing helped reduce these differences, but subtle variations still influenced embedding distances.

 Threshold Sensitivity in Real-Time Verification: The Euclidean distance threshold used for verification is sensitive to dataset scale and distribution. A lower threshold may reject genuine samples, while a higher threshold may allow false positives. Determining a universally optimal threshold required empirical tuning, which may need to be adapted when new users are added.

 Model Interpretability: Deep CNNs function as black boxes, making it difficult to explain why certain signatures are accepted or rejected. Incorporating visualization tools like **Grad-CAM** or **embedding projections (t-SNE)** could improve interpretability, but were beyond the scope of this version.

5.2 Future Scope

While the current system achieves strong accuracy and efficient performance, there are multiple avenues for improvement and expansion:

1. **Integration of One-Shot Learning and Transformer Models**: Future iterations could integrate **Siamese Transformers** or **Few-Shot Learning architectures** (e.g., Prototypical or Matching Networks) to handle unseen users without retraining. These models are better suited for dynamic environments where new identities are continuously added.
2. **Adaptive Thresholding Mechanism**: Instead of a fixed user-defined threshold, the system could employ **adaptive thresholding** based on dataset statistics or Bayesian inference. This would automatically tune sensitivity according to each user's signature variability.
3. **Real-Time Web and Cloud Deployment**: The Streamlit prototype can be extended into a production-ready **web API using FastAPI or Flask**, hosted on a cloud platform (e.g., AWS, GCP). This would enable real-time signature verification across distributed systems such as educational institutions or corporate offices.
4. **Cross-Domain and Multilingual Signature Support**: Future datasets can include signatures in different languages, scripts, and writing instruments to test the model's adaptability. Including cross-cultural samples would enhance robustness for global deployment.
5. **Explainable AI for Signature Verification**: Incorporating explainability modules could visually highlight signature regions contributing most to verification decisions. This would make the system more transparent and trustworthy, particularly in legal or forensic contexts.
6. **Enhanced Data Security and Privacy**: As the system deals with biometric data, integrating **encryption-based data storage** and **secure embedding serialization** can ensure privacy compliance with data protection standards (such as GDPR or ISO/IEC 30107-3).
7. **Hybrid Model Approaches**: Combining CNN-based feature extraction with classical image processing (like contour analysis, geometric moment comparison, or skeletonization) could lead to hybrid architectures that capitalize on both handcrafted and learned features for improved precision.

6. Conclusion

The **Signature Identification System** developed in this project demonstrates the effectiveness of deep metric learning using a **Siamese ResNet architecture** for reliable offline signature verification. By leveraging **contrastive learning**, the model successfully learns discriminative representations capable of distinguishing genuine signatures from skilled forgeries, even with limited training samples. The integration of **data augmentation** and **embedding-based verification** allows the system to generalize well across both benchmark datasets (CEDAR) and custom student datasets. The deployment through an interactive **Streamlit interface** enhances usability, enabling non-technical users to easily enroll, identify, and manage signature data. Achieving accuracies of approximately 97% on CEDAR and 93% on the augmented student dataset, the system validates the potential of lightweight CNN-based Siamese models in biometric authentication tasks. Beyond its current implementation, the project serves as a foundation for future research in **one-shot learning**, **explainable verification**, and **secure biometric data management**, contributing meaningfully to the ongoing evolution of intelligent identity verification systems.

7.
References

[1] **"Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks"** — Luiz G. Hafemann, Robert Sabourin, Luiz S. Oliveira (2017).
Link / preprint: arXiv. arXiv

[2] **"Offline Handwritten Signature Verification — Literature Review"** — Luiz G. Hafemann, Robert Sabourin, Luiz S. Oliveira (2015).
Link / arXiv (survey). arXiv+1

[3] **"Signature Verification using a 'Siamese' Time Delay Neural Network"** — Jane Bromley, Isabelle Guyon, Yann LeCun, et al. (1993, NeurIPS).
Classic Siamese-network paper for verification tasks. NeurIPS Papers+1

[4] **"SigNet: Convolutional Siamese Network for Writer-Independent Offline Signature Verification"** — S. Dey et al. (2017).
Convolutional Siamese approach applied to offline signature verification. arXiv

[5] **GPDS signature datasets (GPDS-960 / GPDS-160 / GPDS corpora)** — public offline signature datasets frequently used in literature. (GPDS dataset page / figshare). Figshare+1

[6] **CEDAR Signature Database & CEDAR-FOX resources** — widely used offline signature corpus. cedar.buffalo.edu+1

[7] **"Deep Metric Learning for Signature Verification"** — practical overviews and implementations (notable notebooks & blog posts; explains contrastive, triplet loss etc.). (Fast Forward Labs / Kaggle notebooks). blog.fastforwardlabs.com+1

[8] **"Multiscale Feature Learning Using Co-Tuplet Loss for Offline Handwritten Signature Verification"** — (arXiv 2023).
Recent metric-learning variant for signatures. arXiv

[9] **"Enhancing Signature Verification Using Triplet Siamese ..."** — S. Tehsin et al. (2024) (MDPI / Mathematics).
Recent application of triplet-loss / Siamese variants to signature verification. MDPI

[10] **General survey / recent reviews (2024–2025)** — newer surveys and reviews of offline signature verification: systematic reviews on methods, datasets, and evaluation. Semantic Scholar+1

[11] **Additional papers & resources referenced in modern work** (examples: multi-stage Siamese, spatial transformer usage in signature verification). See recent arXiv / journal items (2024–2025). Nature+1