Forgery Detection using Signature Verification with MobileNetV2 and OTP Authentication

Sreeya Rudrangi
University of Florida
sreeyarudrangi@ufl.edu

&
Hemanth Krishna Maraboina
University of Florida
maraboinhemanthk@ufl.edu

Abstract

Signature forgery detection is a difficult problem in security-relevant applications. In this work, we suggest and deploy a machine learning-based system that verifies handwritten signatures using feature extraction based on MobileNetV2 and L1 distance computation for similarity comparison. One more OTP authentication process is integrated for additional security. We demonstrate the system with sample signature images and examine its performance in detecting forgery. This paper presents the motivation, problem statement, algorithmic solution, experimental setup, and future work.

1 Introduction

The ability to verify handwritten signatures reliably is essential in banking, legal, and governmental scenarios. Authentication by hand is prone to human error and bias. With advances in machine learning algorithms, verification of signatures automatically is becoming more of a possibility. In this project, offline static signature verification through the use of deep learning to extract prominent features and compare them with enrolled signature samples is taken into consideration. Use of MobileNetV2, a lightweight CNN model pre-trained on ImageNet, enables efficient feature extraction. In addition, two-factor authentication (2FA) based on OTP verification gives confirmation of identity.

2 Problem Statement

Given a query signature image and a set of five registered genuine signature images for a user, the goal is to classify the query signature as genuine or forged. This can be formulated as a supervised binary classification task, where:

- **Input**: Image of the signature (after preprocessing)
- **Output**: Decision (genuine or forgery)

Mathematically, the problem can be modeled as minimizing the error rate:

$$\text{Minimize} \quad E[\text{Loss}(y, \hat{y})]$$

where y is the true label (genuine/forgery) and $\hat{y}$ is the predicted label based on similarity computation.

3 Algorithm

The system architecture consists of the following main components:

1. **Feature Extraction**: We use MobileNetV2 (without its top classification layers) to extract a dense 128-dimensional feature vector for each signature image. Global average pooling is applied to reduce the feature maps.

2. **Similarity Computation**: After obtaining embeddings for both the uploaded and registered signatures, we compute the L1 distance between the feature vectors:

$$\text{Distance}(A, B) = \frac{1}{n} \sum_{i=1}^{n} |A_i - B_i|$$

   A similarity score is then derived as $1 - \text{Distance}$. Higher scores indicate higher similarity.

3. **Verification Decision**: The average similarity across the five registered samples is computed. If the similarity score exceeds a threshold (0.75), the signature is accepted; otherwise, it is flagged as potentially forged.

4. **OTP Authentication**: Upon successful signature verification, an OTP is sent via Twilio's Verify API. The user must input the received OTP to complete the verification.

4 Experiments

4.1 Dataset

For testing, user datasets were simulated with five genuine signature images per user and various uploaded signature samples. Images were resized to 160x160 pixels and preprocessed according to MobileNetV2 requirements (RGB, normalized).

4.2 Implementation Details

- Framework: TensorFlow 2.13, Keras
- Front-end: Streamlit for user interaction
- OTP Service: Twilio Verify API
- Hardware: Apple MacBook Air (8GB RAM)

4.3 Sample Screenshots

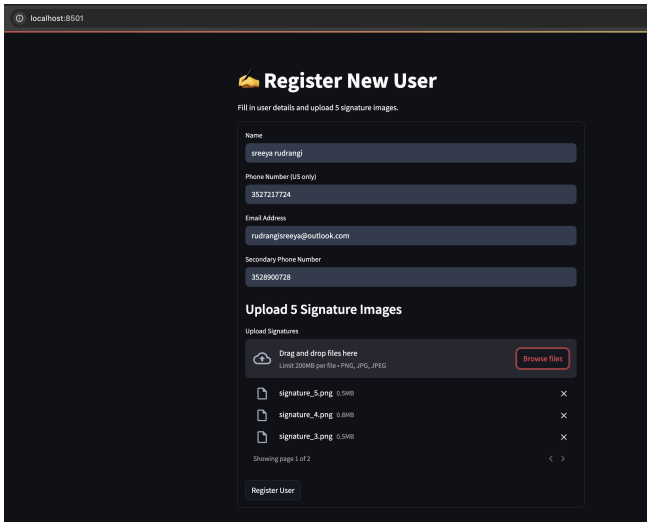

Figure 1: User Registration Page: Entering User Details and Uploading 5 Signature Images

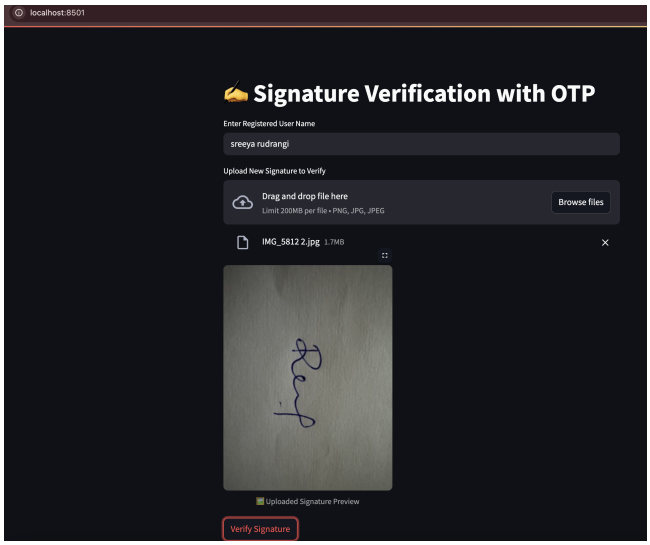

Figure 2: Streamlit App: Signature Upload Preview

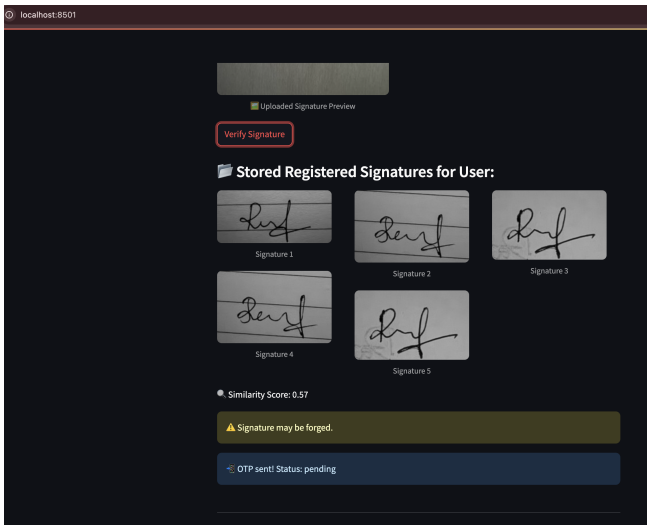

Figure 3: Verification Step: Similarity Score and Stored Samples

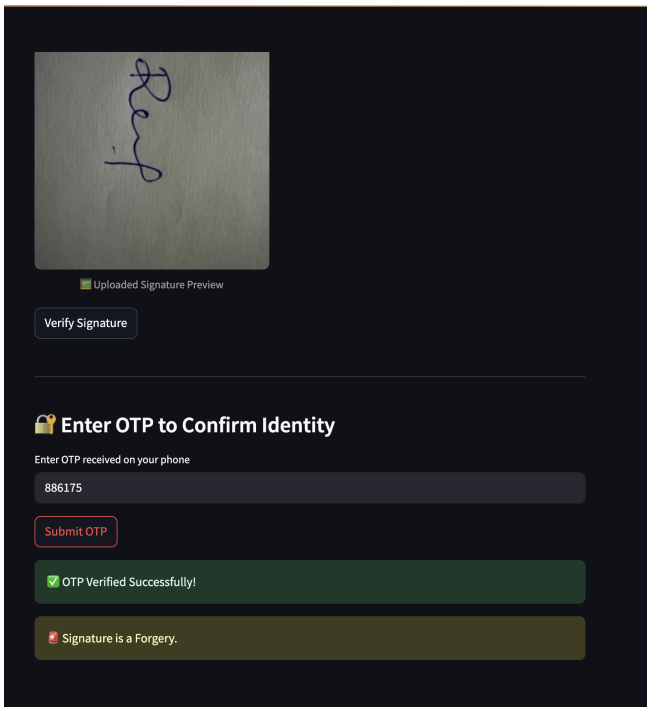

Figure 4: OTP Verification Step

5 Conclusion

Through this project, we were able to successfully integrate an offline signature verification system with another OTP-based two-factor authentication system for the sake of enhancing security. MobileNetV2 was employed as a feature extractor to facilitate the generation of short and discriminative embeddings for handwritten signatures. L1 distance was employed since it provided a simple and effective way of measuring feature similarity between signature samples. Streamlit enabled seamless front-end deployment with an interactive user interface, and Twilio Verify API ensured secure and effective OTP verification. Future development can potentially include optimization of MobileNetV2 on domain-specialized signature datasets for further improvement of performance in a specific domain, or exploring end-to-end training of Siamese Network for optimizing similarity learning specifically for handwritten signature verification problems.

References

[1] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1994). *Signature verification using a "Siamese" time delay neural network.* International Journal of Pattern Recognition and Artificial Intelligence, 7(04), 669–688.

[2] Hafemann, L. G., Oliveira, L. S., and Sabourin, R. (2017). *Learning features for offline handwritten signature verification using deep convolutional neural networks.* Pattern Recognition, 70, 163–176.

[3] Chopra, S., Hadsell, R., and LeCun, Y. (2005). *Learning a similarity metric discriminatively, with application to face verification.* In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Bengio, Y., Courville, A., and Vincent, P. (2013). *Representation learning: A review and new perspectives.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798–1828.

[5] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2009). *Self-taught learning: Transfer learning from unlabeled data.* In Proceedings of the 24th International Conference on Machine Learning (ICML).

[6] Verizon (2023). *2023 Data Breach Investigations Report (DBIR).* Retrieved from: magenta`https://www.verizon.com/business/resources/reports/dbir/`

[7] Twilio Docs. (2024). *Twilio Verify API: Adding Two-Factor Authentication (2FA) to Applications.* Retrieved from: magenta`https://www.twilio.com/docs/verify`