Signature Forgery Detection

Balusu Revanth
2021BCS-017

Pathlavath Sudeendra
2021BCS-052

Rajashekar Reddy
2021BCS-058

Supervisor :

Prof. Shashikala Tapaswi

OVERVIEW

1. Introduction
2. Motivation
3. Objectives
4. Dataset
5. Methodology
6. Results
7. Limitations
8. References

Introduction

- Handwritten signature verification is an essential and ubiquitous task in modern society, finding extensive applications in financial transactions, document authentication, and identity verification.
- Online signature verification employs digital devices to capture signatures in real-time. This approach incorporates dynamic elements, such as the speed and pressure of the signature, to enhance security and enable biometric authentication.
- In contrast, offline signatures are handwritten on physical mediums, like paper, and lack dynamic data.
- Despite the absence of real-time verification and biometric features, offline signatures continue to be widely accepted for legal and traditional purposes.

Motivation

- Traditional methods of signature verification are often time consuming, subjective, and prone to errors.
- Leveraging CNNs for signature forgery detection presents an opportunity to automate and enhance the accuracy of this process. By training the CNN on a large dataset of genuine and forged signatures, the model can learn to automatically extract intricate features and patterns that distinguish authentic signatures from forgeries.
- The potential impact of such a project spans across various industries, including banking, legal, and government sectors, where the ability to reliably detect forged signatures is crucial for preventing financial losses, legal disputes, and maintaining trust in digital transactions.

Objectives

- To design a robust CNN architecture
- To curate and preprocess the dataset
- To train and validate the designed model
- To evaluate and benchmark the model
- To build a website and integrate the model

Dataset

Data Acquisition

- The dataset gathering process for the project involves obtaining the CEDAR-Dataset from Kaggle.
- This dataset contains original and forged signatures, providing a diverse range of samples for training a reliable forgery detection model.
- The dataset consists of 2640 signature images, categorized as genuine or forged. The images are in the .png format and consist of three channels: red, green, and blue (RGB).

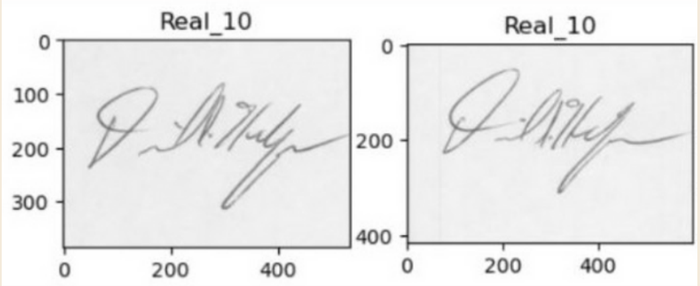


Figure 1: Original Signature

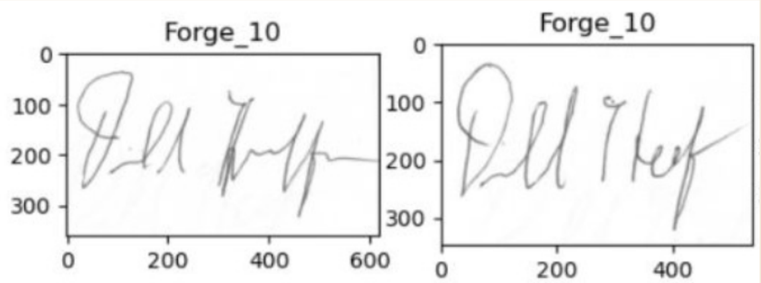


Figure 2: Forged Signature

Data Preprocessing

Data Resizing

- The images are resized to dimensions of 224x224. This resizing ensures that all images have a consistent size, which is essential for machine learning models that require fixed size inputs. By standardizing the dimensions, we eliminate potential issues that may arise from varying image sizes.

RGB to Grayscale

- The RGB images are converted to grayscale. This conversion reduces the colour channels from three (red, green, and blue) to one (gray). Since signature forgery detection relies more on shape and texture rather than colour information, converting to grayscale simplifies the data while retaining the important features needed for accurate detection.

Data Augmentation

- Data augmentation techniques are applied specifically to the training data. It involves applying various transformations to artificially increase the size and diversity of the training dataset. For the signature forgery detection project, the training data is augmented with techniques such as rotation, shear, zoom, and horizontal flip.

Methodology

Model Architecture

- The CNN architecture plays a crucial role in the model's performance. The CNN model for signature forgery detection consists of multiple convolutional layers to extract relevant features from the input images.
- These convolutional layers are often followed by pooling layers to reduce spatial dimensions and introduce translation invariance.
- Additional layers like batch normalization and dropout are included to improve model generalization and prevent overfitting.
- Fully connected layers are responsible for the final classification.

Convolutional Layer

- A convolutional layer is a fundamental building block of Convolutional Neural Networks (CNNs), which are primarily used for image recognition task.
- The key idea behind this layer is to detect local patterns or features in the input data by sliding a small filter over the input and computing dot products between the filter and local regions of the input.
- The output of this operation is called a feature map, and it represents the presence of specific features in the input data.

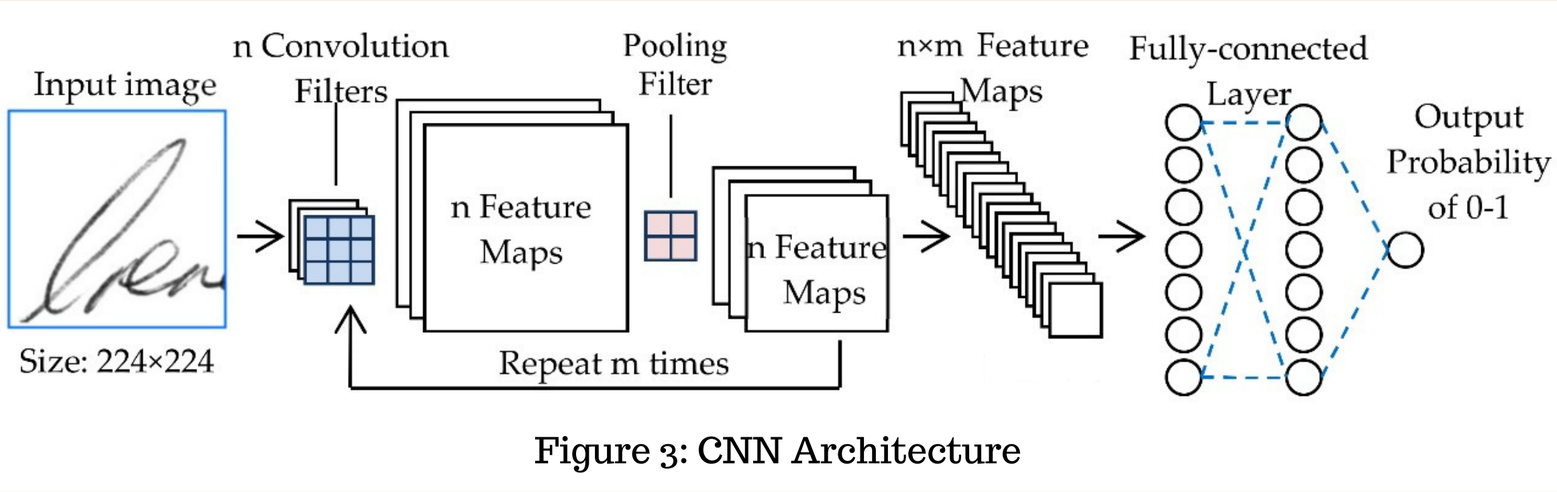


Figure 3: CNN Architecture

Batch Normalization

- Batch Normalization is a technique used to improve the training and performance of deep neural networks.
- It addresses the problem of internal covariate shift.
- Batch normalization normalizes the input of each layer by standardizing it to have zero mean and unit variance.

Maxpooling

- Maxpooling is a down sampling operation typically used in CNNs to reduce the spatial dimensions of the feature maps.
- This helps to reduce the computational complexity of the network, make it less sensitive to small variations in input, and introduce a degree of translation invariance since the maximum value represents the most prominent feature in each region.

Dropout

- Dropout is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly sets a fraction of the neurons' outputs to zero with a certain probability (dropout rate).
- This effectively removes those neurons from the network for that particular forward and backward pass.
- By doing this, dropout prevents the network from relying too much on any particular set of neurons, forcing it to learn more robust and general features.

Flatten

- Flatten is an operation that converts multidimensional data (e.g., a 2D image or a 3D volume) into a 1D vector.
- In CNNs, the convolutional and max-pooling layers produce 3D feature maps, and before passing the data to fully connected (dense) layers, it needs to be flattened into a 1D vector.

Dense (Fully Connected) Layer

- A dense layer is a standard layer in a neural network where each neuron is connected to every neuron in the previous layer.
- These layers are typically used at the end of a CNN or other types of networks to map the extracted features to the final output.

Model Architecture

Layer(type)	Output Shape	Param#
conv2d (Conv2D)	(None, 222, 222, 32)	320
batch_*ormalization*(*BatchNormalization*)	(None, 222, 222, 32)	128
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
dropout (Dropout)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_1 (MaxPooling 2D)	(None, 54, 54, 64)	0
dropout_1 (Dropout)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 52, 52, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
dropout_2 (Dropout)	(None, 26, 26, 128)	0
conv2d_3 (Conv2D)	(None, 24, 24, 256)	295168
batch_normalization_3 (BatchNormalization)	(None, 24, 24, 256)	1024
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 256)	0
dropout_3 (Dropout)	(None, 12, 12, 256)	0
conv2d_4 (Conv2D)	(None, 10, 10, 256)	590080
batch_normalization_4 (Batc hNormalization)	(None, 10, 10, 256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 256)	0
dropout_4 (Dropout)	(None, 5, 5, 256)	0
conv2d_5 (Conv2D)	(None, 3, 3, 512)	1180160
batch_normalization_5 (BatchNormalization)	(None, 3, 3, 512)	2048
max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 512)	0
dropout_5 (Dropout)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 256)	131328
batch_normalization_6 (BatchNormalization)	(None, 256)	1024
dropout_6 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

Figure 4: Model Architecture

Early stopping

- Early stopping is employed to prevent overfitting and enhance generalization. It involves monitoring the model's performance on a validation set during training.
- If the performance does not improve for a certain number of epochs (controlled by a patience value), the training process is stopped. Early stopping prevents the model from memorizing the training data and ensures it learns useful patterns.

Learning Rate Reduction

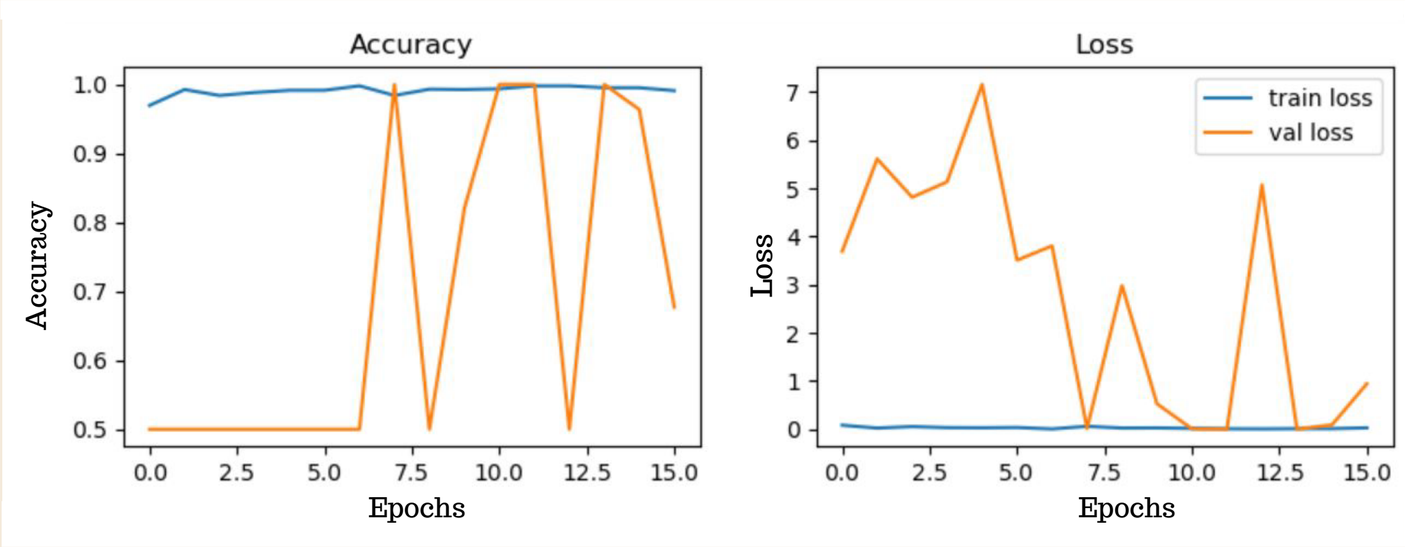
- Learning rate reduction is used to optimize the training process and model convergence.
- The learning rate determines the step size taken during parameter updates. By dynamically adjusting the learning rate, typically when validation performance plateaus or the loss function stops decreasing, the model can fine-tune its parameters more effectively.
- This adjustment aids in reaching better local or global minima during optimization.

Results

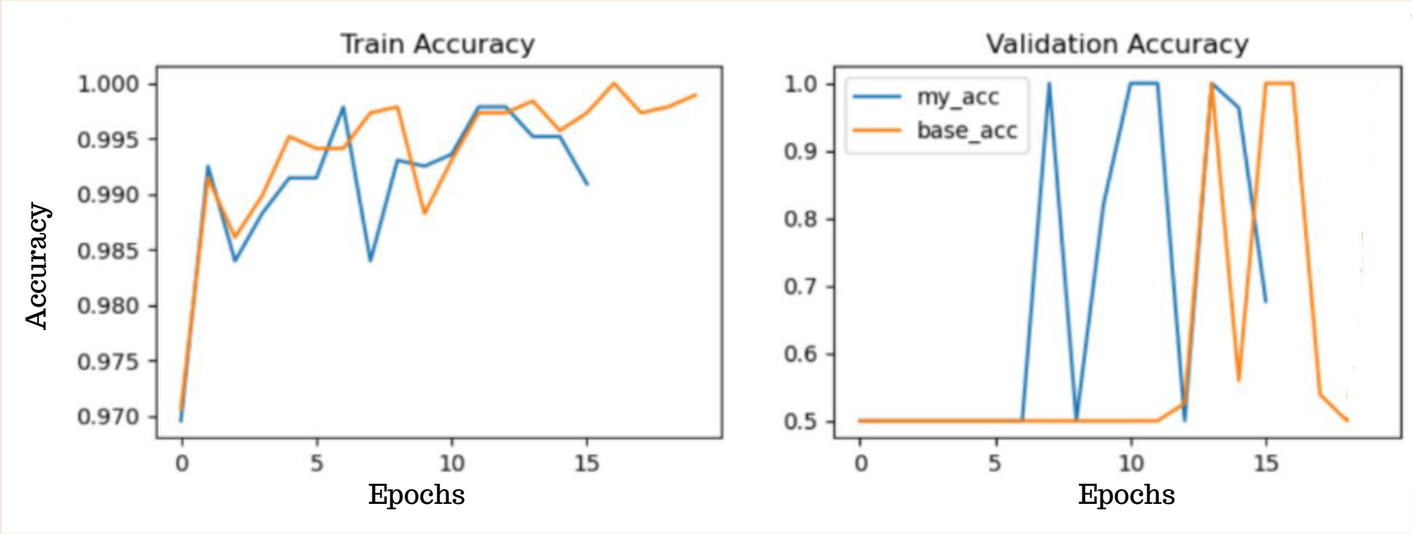
Performance Comparision

- To assess the performance of our CNN model, it is compared to a baseline CNN model.
- The CNN model with data augmentation and early stopping with learning rate reduction showcased the most promising results. Its accuracy on both the training and validation datasets improved significantly compared to the baseline CNN model.
- We achieved a testing accuracy of 95.4% with the train, validation, test split of ratio 70%, 15%, 15%.

Training Graph



Comparison of our model with base-line model



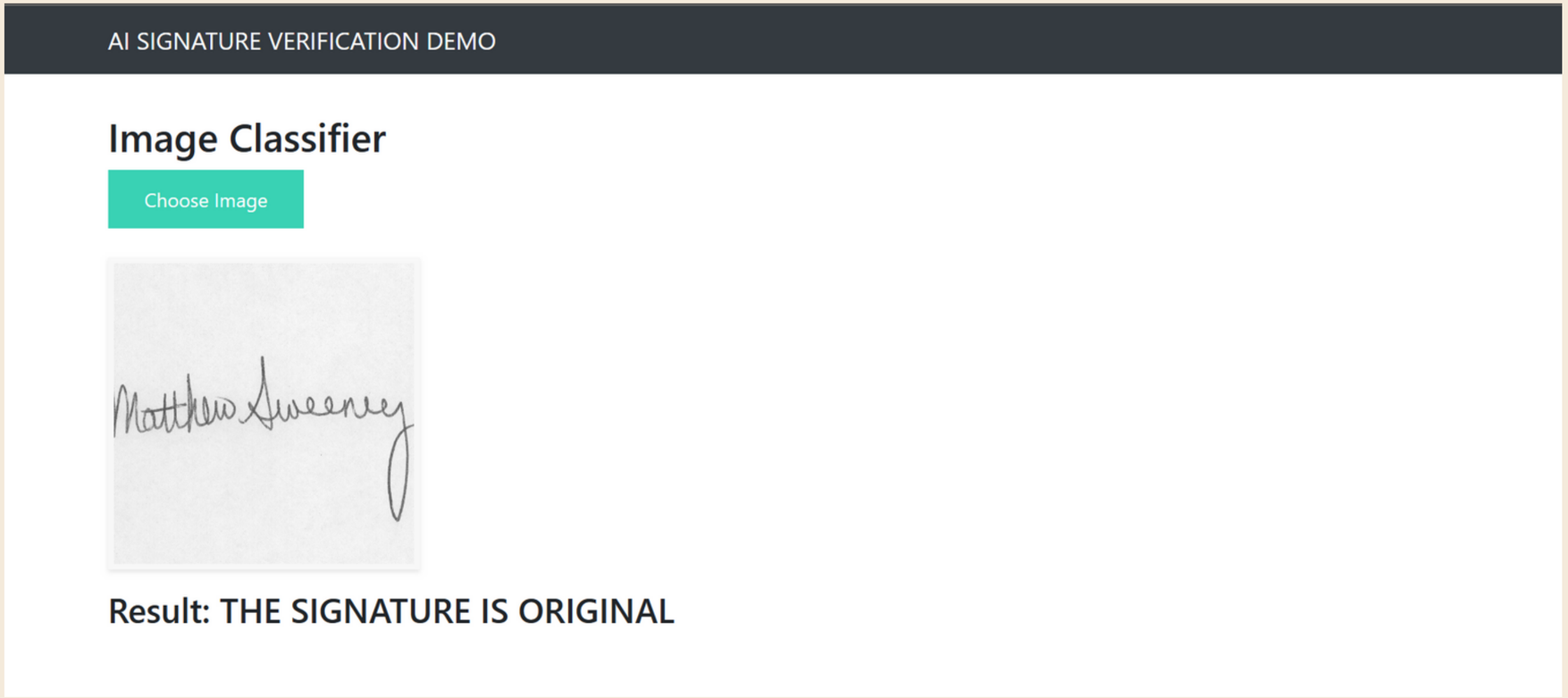
Comparision Table

Model	Training Set Accuracy	Validation Set Accuracy	Testing Set Accuracy
Basic CNN model	100 %	88.14 %	87.80 %
CNN model with Regularization	96.64 %	88.62 %	88.53 %
CNN model with Data Augmentation	97.82 %	91.07 %	90.78 %
Our Model	98.56%	95.78%	95.41%
Siamese Network model	99.88%	99.17%	98.94 %

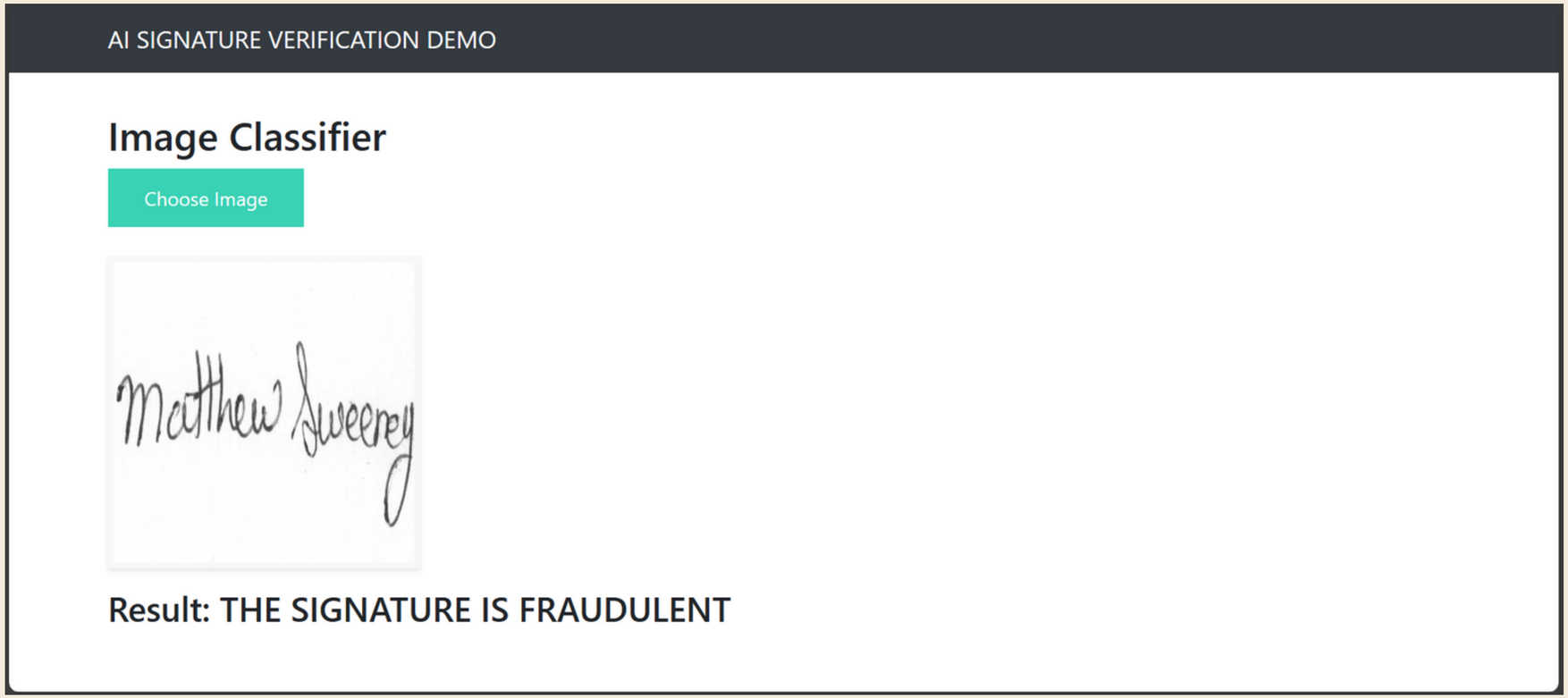
Website

- The signature forgery detection website is developed using HTML, CSS, JavaScript, and Python Flask.
- Its primary objective is to detect the authenticity of signatures by allowing users to upload a photo containing the signature in question.
- Through a seamless and user-friendly interface, users can easily submit the signature images for analysis.
- The combination of front-end web technologies and backend Python Flask framework ensures smooth interaction and communication between the user and the machine learning system.

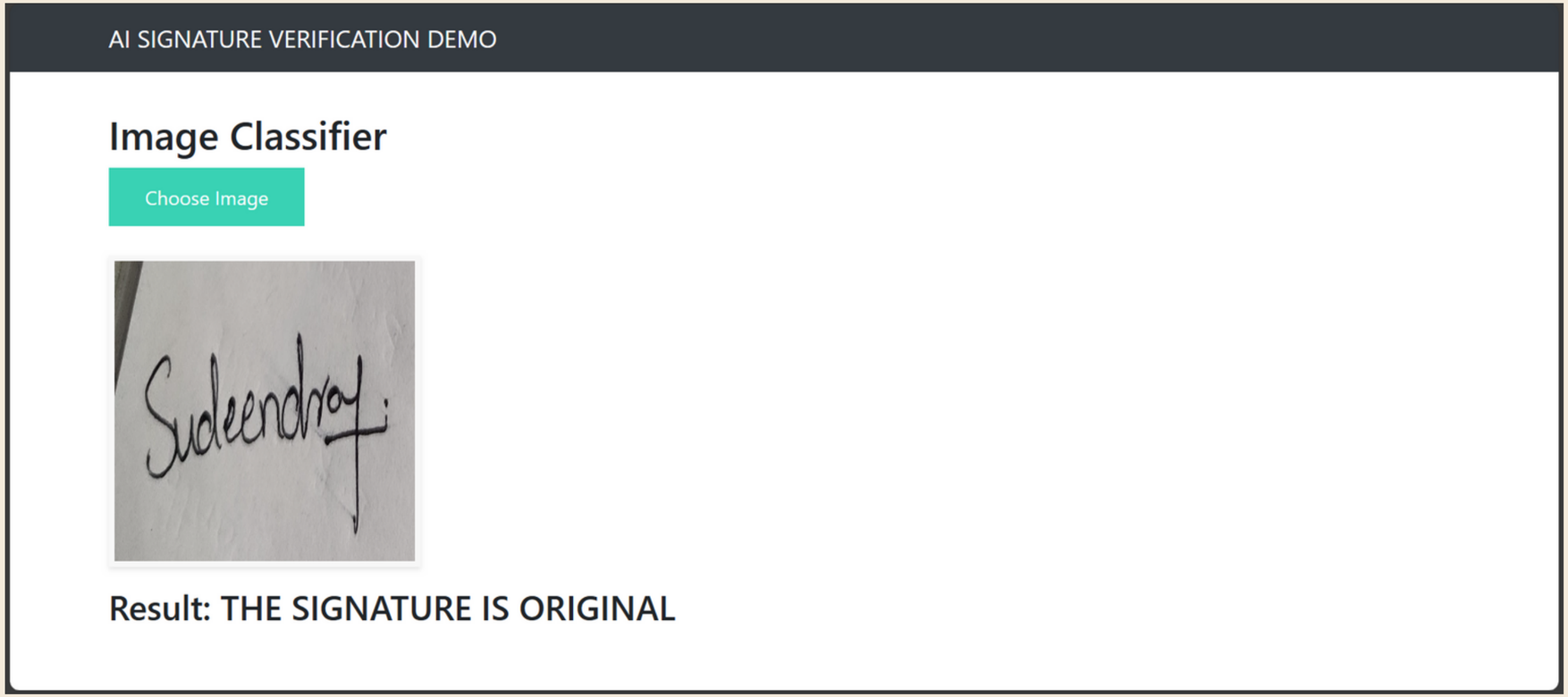
Website


AI SIGNATURE VERIFICATION DEMO
Image Classifier
Choose Image
Result: THE SIGNATURE IS ORIGINAL

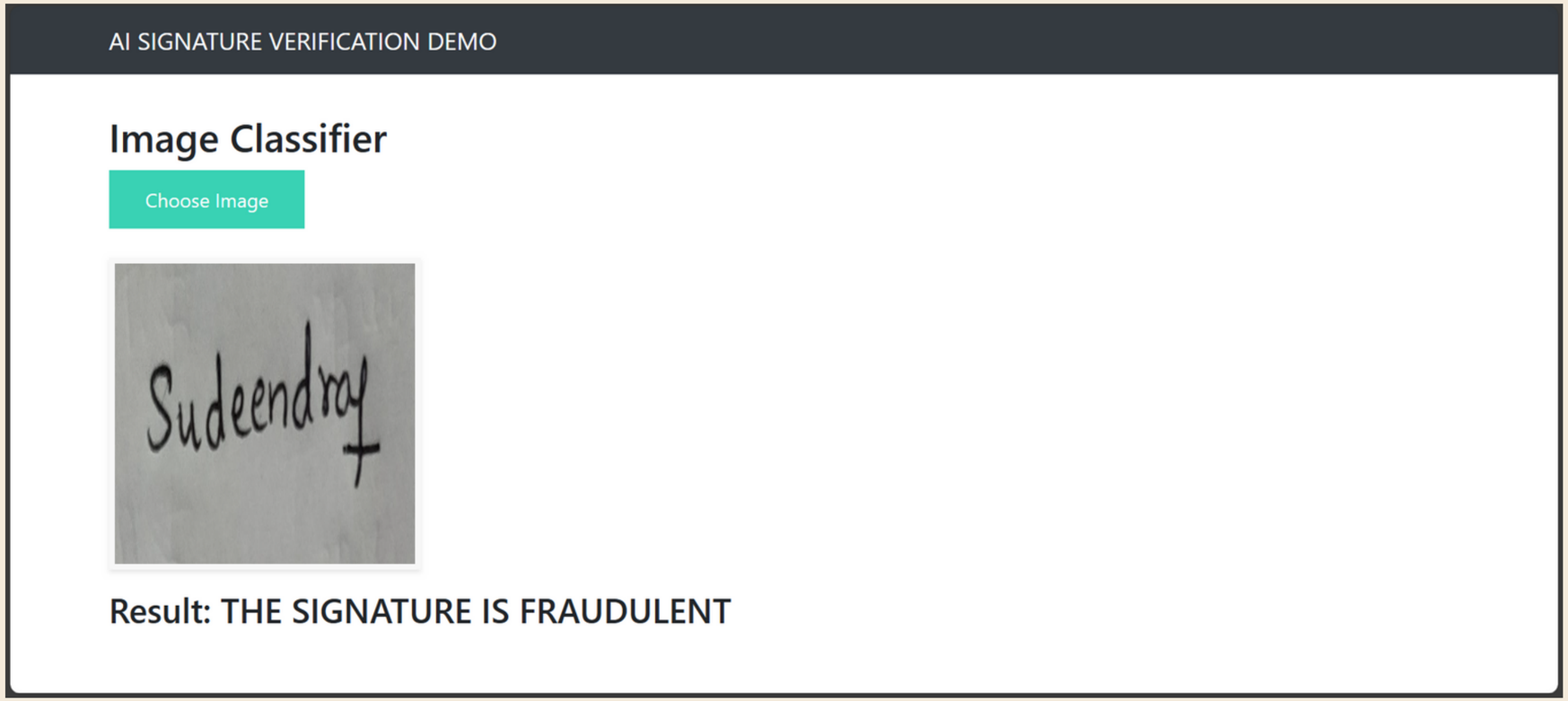
Website


AI SIGNATURE VERIFICATION DEMO
Image Classifier
Choose Image
Result: THE SIGNATURE IS FRAUDULENT

Website


AI SIGNATURE VERIFICATION DEMO
Image Classifier
Choose Image
Result: THE SIGNATURE IS ORIGINAL

Website


AI SIGNATURE VERIFICATION DEMO
Image Classifier
Choose Image
Sudeendra
Result: THE SIGNATURE IS FRAUDULENT

Limitations

- Limited Datasets: One of the primary challenges in training CNNs for signature forgery detection is the availability of large-scale, diverse, and annotated datasets.
- Variations in Signature Styles: Signatures can exhibit significant variations even for the same individual, depending on factors such as writing instruments, writing surface, mood, and age.
- Skilled Forgeries: Detecting skilled forgeries is particularly challenging because forgers attempt to mimic the genuine signer's style closely
- Online vs Offline Signatures: Online signatures carry temporal information about the signing process, whereas offline signatures lack this temporal context. Combining both types of signatures to create a robust CNN model is a non-trivial task.

References

- [1] E. Alajrami, B. S. Abu-Nasser, and M. M. Musleh, "Handwritten Signature Verification using Deep Learning," *International Journal of Academic Multidisciplinary Research (IJAMR)*, pp. 39-44, 2019.
- [2] J. Poddara, V. Parikha, and S. K. Bhartia, "Offline Signature Recognition and Forgery Detection using Deep Learning," in *International Conference on Emerging Data and Industry 4.0 (EDI40)*, 2020, pp. 610-617.
- [3] J. G. S. Jerome, A. Kandulna, A. A. Kujur, and D. A., "Handwritten Signature Forgery Detection using Convolutional Neural Networks," in *International Conference on Advances in Computing and Communication (ICACC-2018)*, 2018, pp. 978-987.

THANK YOU