Signature VERIFICATION SYSTEM

AuthentiSign

SIGNATURES YOU CAN TRUST

PRESNTED BY:

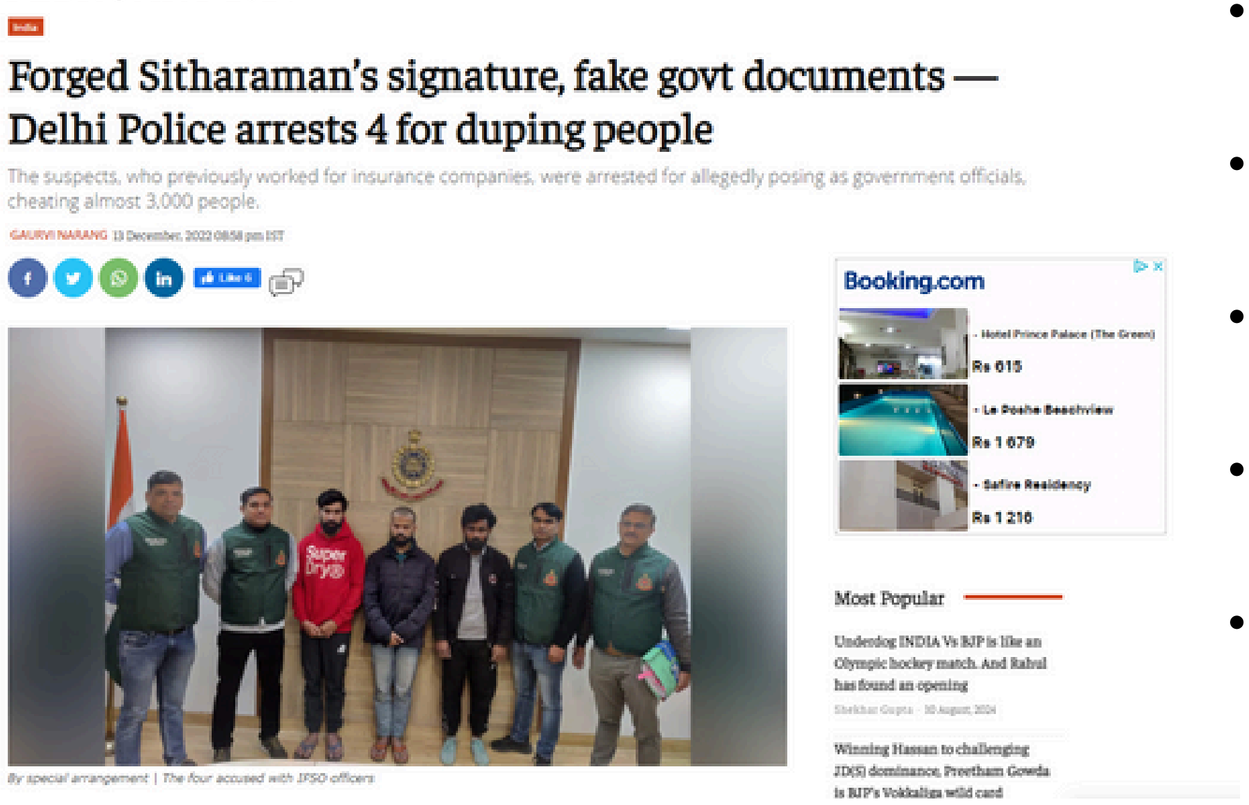
B AMALA FRANCIS

GUIDE:RAMYA KRISHNAN S

DATE:15/10/24

CHALLENGES

FEBRUARY 2024



India

Forged Sitharaman's signature, fake govt documents — Delhi Police arrests 4 for duping people

The suspects, who previously worked for insurance companies, were arrested for allegedly posing as government officials, cheating almost 3,000 people.

GAURVI NARANG 13 December, 2022 08:58 pm IST

By special arrangement | The four accused with IFSO officers

Most Popular

Underdog INDIA Vs BJP is like an Olympic hockey match. And Rahul has found an opening

Sheikhar Gupta - 10 August, 2024

Winning Hassan to challenging JD(S) dominance, Preetham Gowda is BJP's Vokkaliga wild card

Issues in Signature verification ,

- **Forgery:** Easy to fake or forge.
- **Time-Consuming**: Manual verification is time-consuming.
- **Error-Prone:** Manual verification is prone to errors.
- **Disputes:** Can lead to disputes and disagreements.
- **Financial Loss:** Can result in financial loss or damage.
- **Reputation Damage:** Can damage reputation and credibility.

OVERVIEW

Signature Verification System (Offline): A machine learning-based approach to detect and prevent forgery by comparing original and comparison signatures and tells whether the signatures match or not (real or fake), reducing financial and reputational risks

Key Features includes,

- Machine Learning AlgorithmS (CNN,RNN,SVM)
- Signature Analysis.
- Improved security and reliability

01 Upload original signature image.

02 Upload the signature to be verified.

03 Click compare.

04 Get the result.(Real or Fake)

FRONT END

01

Streamlit

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

02

CSS

Cascading Style Sheet (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

TECHNOLOGIES USED

Data Preprocessing and Feature Extraction

01

Python

For data preprocessing, including cleaning, normalization, and augmentation of signature images.

02

OpenCV

To perform image processing tasks such as resizing, binarization, and feature extraction.

Machine Learning Models

03

TensorFlow/Keras

For deep learning models, especially Convolutional Neural Networks (CNNs) to learn features from signature images.

04

Scikit-learn

For traditional machine learning models like SVM, Random Forest, or KNN.

05

Pre-trained Models

Using models like ResNet, VGG, or Inception as feature extractors.

Algorithms

06

Convolutional Neural Networks (CNNs)

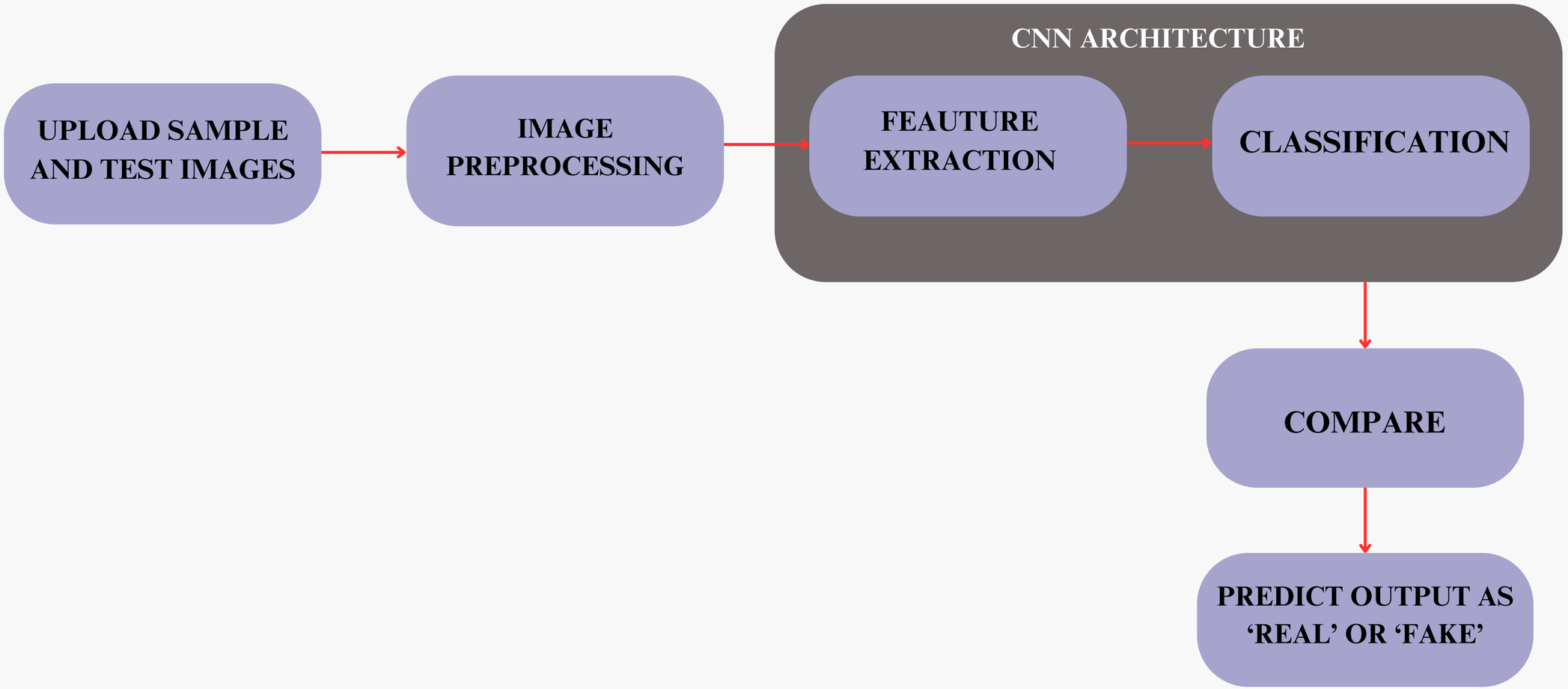
Used for feature extraction and classification of signature images.

07

Support Vector Machines (SVM)

Often used as a classifier after feature extraction. SVM is used to classify signatures based on extracted features, such as edges, curves, and texture patterns.

WORKFLOW


UPLOAD SAMPLE AND TEST IMAGES
IMAGE PREPROCESSING
CNN ARCHITECTURE
FEAUTURE EXTRACTION
CLASSIFICATION
COMPARE
PREDICT OUTPUT AS ‘REAL’ OR ‘FAKE’

DESIGN MODULES

PROGRAM DESIGN

- User Interface Module
- Image Preprocessing Module
- Feature Extraction Module

INPUT DESIGN

- Upload Original and Test Signatures for Comapring
- Back_End Input Processing include Preprocessing, Feature Extraction, Classification and Comparison

OUTPUT DESIGN

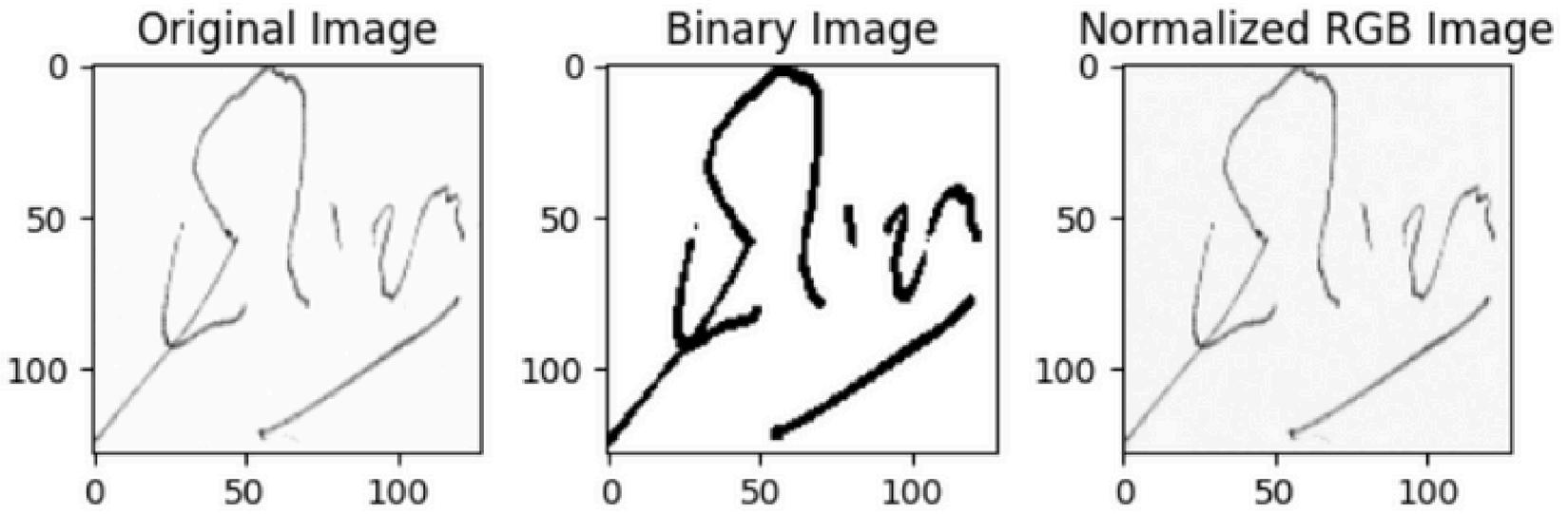
- Verification Result
- User Feedback and Guidance
- Error Messages

DATA PREPARATION

BINARY AND NORMALIZED IMAGES

```
# Display the normalized RGB image
plt.subplot(1, 3, 3)
plt.imshow(processed_images["rgb"])
plt.title('Normalized RGB Image')

plt.tight_layout()
plt.show()
```



MODEL TRAINING

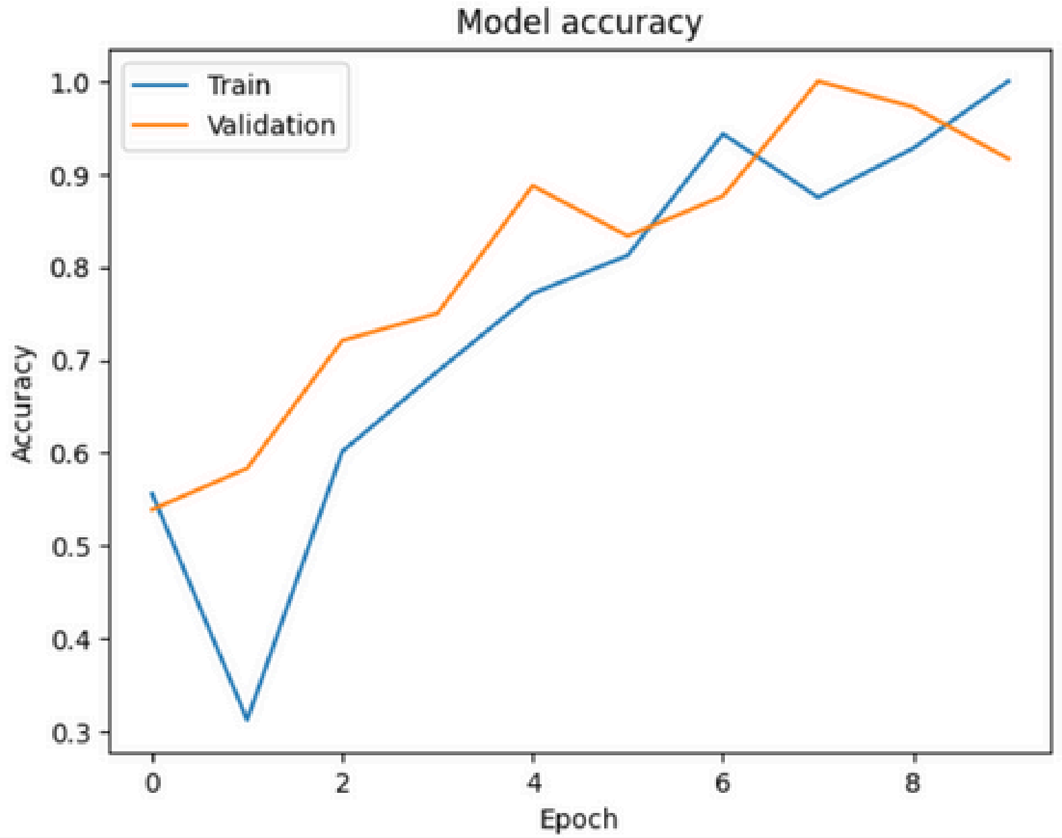
```
57/57 ━━━━━━━━━━━━━━━━━━━━ 130s 2s/step - accuracy: 0.5874 - loss: 0.6718 - val_accuracy: 0.7208 - val_loss: 0.6038
Epoch 4/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.6875 - loss: 0.5792 - val_accuracy: 0.7500 - val_loss: 0.5990
Epoch 5/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 127s 2s/step - accuracy: 0.7221 - loss: 0.5446 - val_accuracy: 0.8875 - val_loss: 0.2656
Epoch 6/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.8125 - loss: 0.2206 - val_accuracy: 0.8333 - val_loss: 0.2538
Epoch 7/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 166s 3s/step - accuracy: 0.9317 - loss: 0.2057 - val_accuracy: 0.8764 - val_loss: 0.4150
Epoch 8/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8750 - loss: 0.4753 - val_accuracy: 1.0000 - val_loss: 0.0113
Epoch 9/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 128s 2s/step - accuracy: 0.9102 - loss: 0.2635 - val_accuracy: 0.9722 - val_loss: 0.0877
Epoch 10/10
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 1.0000 - loss: 0.0620 - val_accuracy: 0.9167 - val_loss: 0.1318
```

[87]:
```python
# Evaluate the model
loss, accuracy = model.evaluate(test_generator)
print(f"Accuracy: {accuracy:.2f}")
```

```
46/46 ━━━━━━━━━━━━━━━━━━━━ 21s 451ms/step - accuracy: 0.9699 - loss: 0.1029
Accuracy: 0.97
```

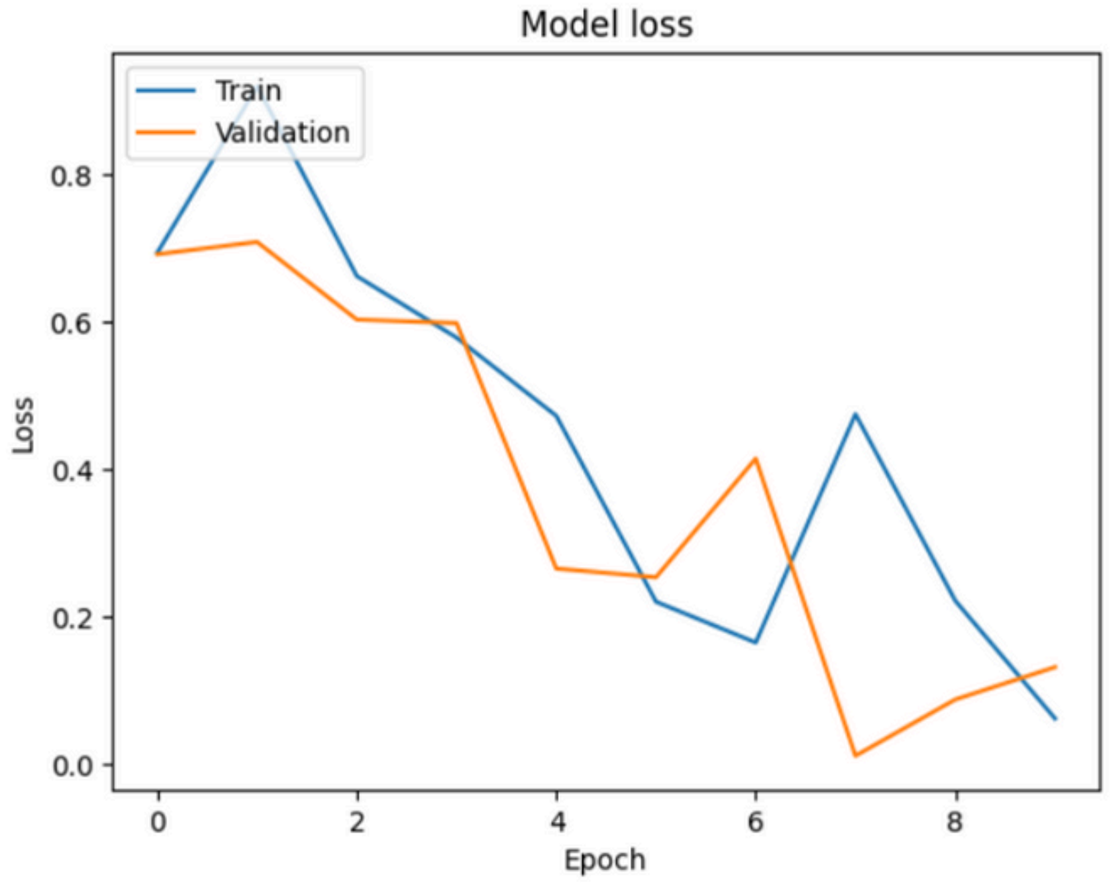
MODEL EVALUATION

[88]:
```
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

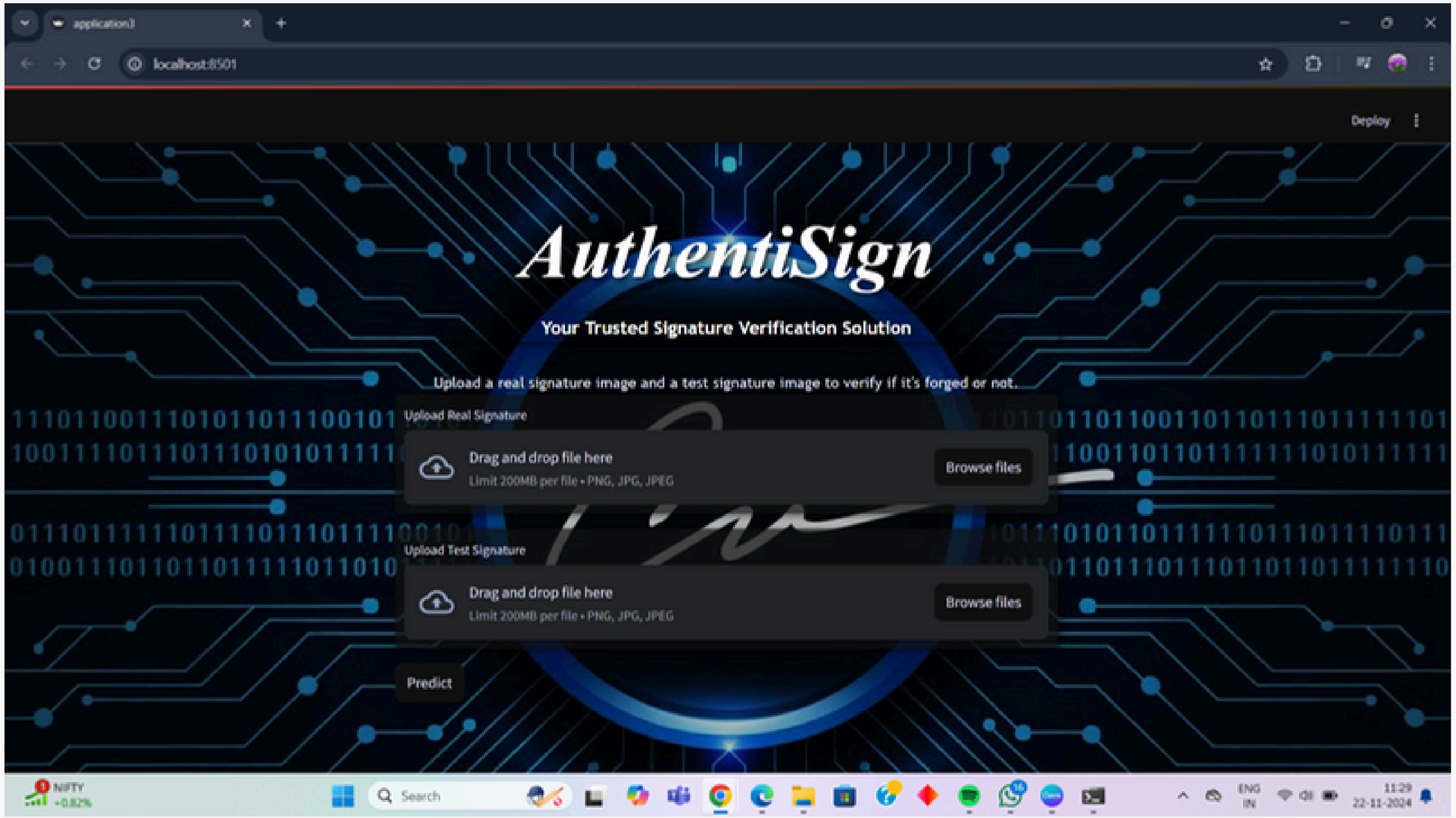


MODEL EVALUATION

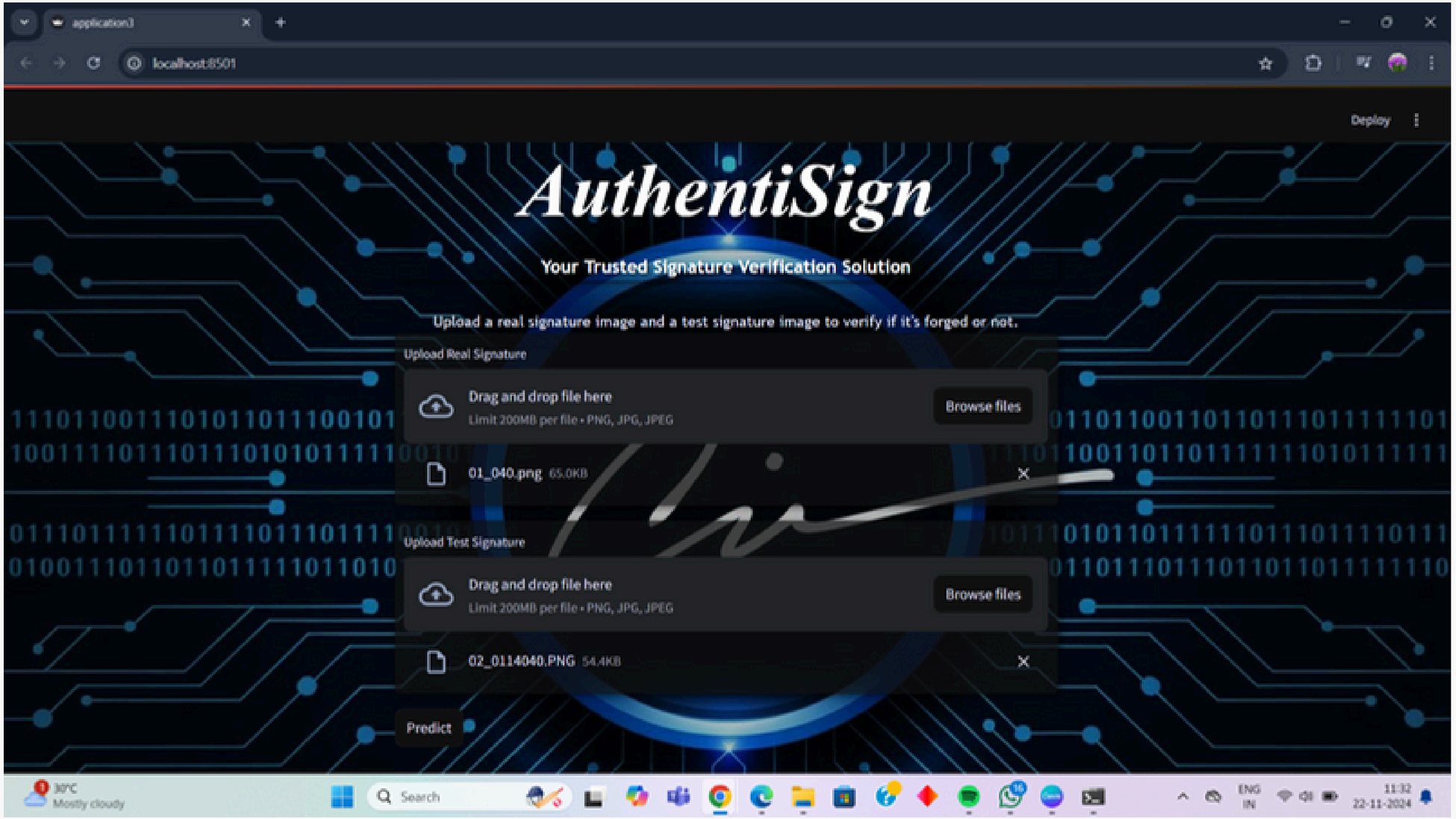
[89]:
```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



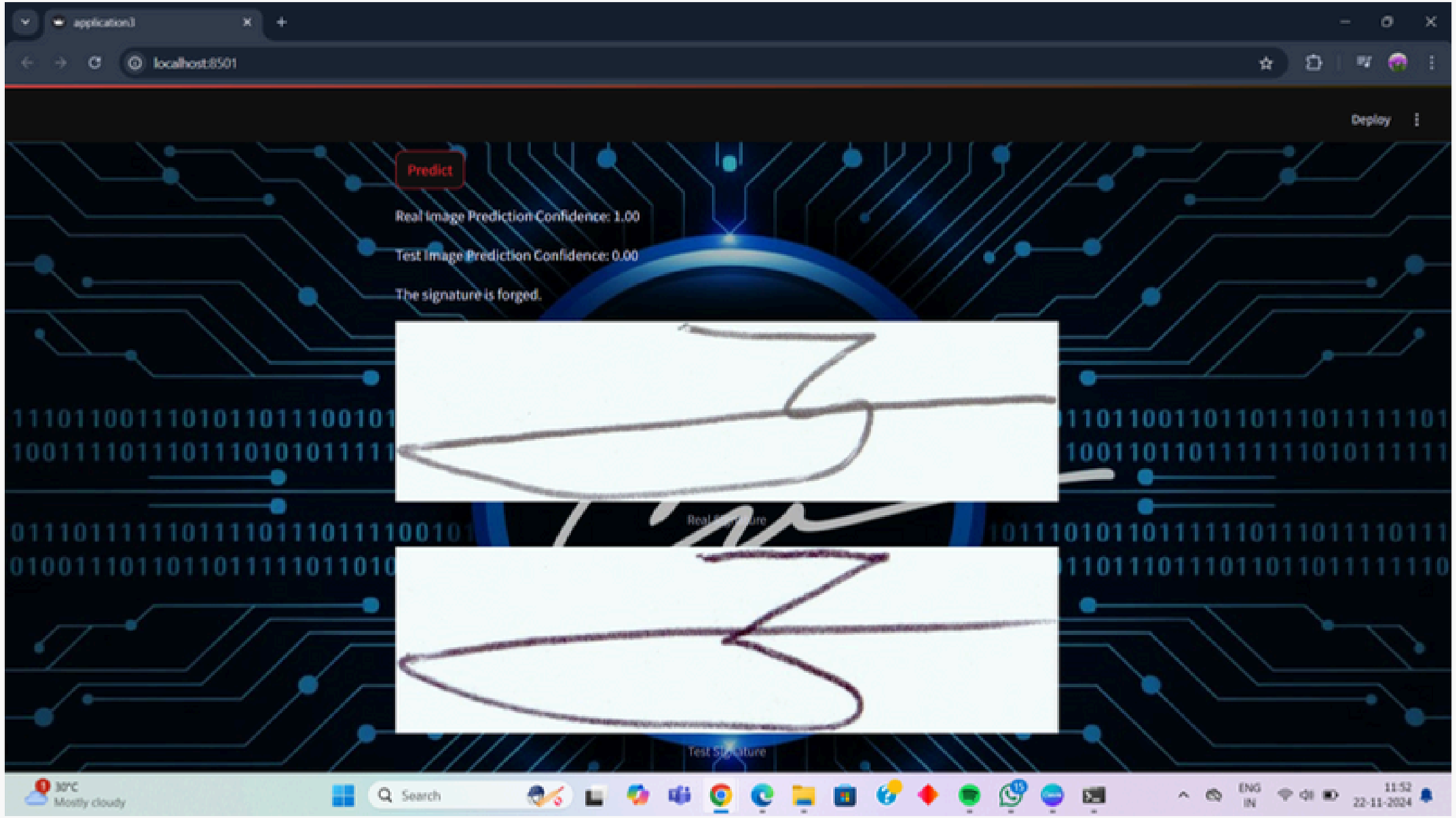
SCREENSHOTS



SCREENSHOTS



SCREENSHOTS




Authenti Sign
Thank You
MacBook Pro