model.py:

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define constants

train\_data\_dir = r'C:\Users\manas\Downloads\CEDAR\signatures'  # Replace with your training data directory

validation\_data\_dir = r'C:\Users\manas\Downloads\archive (2)\Dataset\_Signature\_Final\Dataset\dataset3'  # Replace with your validation data directory

img\_width, img\_height = 128, 128

batch\_size = 32

epochs = 20

# Create a CNN model

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_width, img\_height, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(1, activation='sigmoid'))  # Output layer for binary classification

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Data augmentation and preprocessing

train\_datagen = ImageDataGenerator(

    rescale=1.0/255.0,  # Normalize pixel values to [0, 1]

    rotation\_range=20,  # Randomly rotate images by up to 20 degrees

    width\_shift\_range=0.2,  # Randomly shift the width of images

    height\_shift\_range=0.2,  # Randomly shift the height of images

    shear\_range=0.2,  # Randomly apply shear transformations

    zoom\_range=0.2,  # Randomly zoom into images

    horizontal\_flip=True,  # Randomly flip images horizontally

    fill\_mode='nearest'

)

validation\_datagen = ImageDataGenerator(rescale=1.0/255.0)  # Only rescale for validation data

# Load and prepare the data

train\_generator = train\_datagen.flow\_from\_directory(

    train\_data\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='binary'

)

validation\_generator = validation\_datagen.flow\_from\_directory(

    validation\_data\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='binary'

)

# Train the model

history = model.fit(

    train\_generator,

    steps\_per\_epoch=train\_generator.samples // batch\_size,

    epochs=epochs,

    validation\_data=validation\_generator,

    validation\_steps=validation\_generator.samples // batch\_size

)

# Save the trained model

model.save('my\_model.h5')

import tensorflow as tf

from tensorflow.keras.preprocessing import image

import numpy as np

# Load the trained model

model = tf.keras.models.load\_model('my\_model.h5')  # Replace with your model file path

# Load a new signature image for testing

signature\_image\_path = r'C:\Users\manas\Downloads\CEDAR\signatures\full\_org\original\_17\_22.png'  # Replace with the path to the image you want to testC:\Users\manas\Downloads\CEDAR\signatures\full\_org\original\_4\_16.png

# Load and preprocess the test image

img = image.load\_img(signature\_image\_path, target\_size=(128, 128))

img = image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img /= 255.0  # Normalize the input image

# Make a predictionC:\Users\manas\Downloads\CEDAR\signatures\full\_org\original\_30\_2.png

prediction = model.predict(img)

# Interpret the prediction

if prediction[0][0] >= 0.5:

    print("True signature")

elif prediction[0][0] <= 0.5:

    print("False signature")

else:

    print("invalid signature")

app.py:

import os

from flask import Flask, render\_template, request, jsonify

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

app = Flask(\_\_name\_\_)

# Define the absolute path to your model file

model\_path = os.path.join(os.getcwd(), 'my\_model.keras')  # Adjust the path as needed

# Load the trained model

model = load\_model(model\_path)

# Set the maximum file upload size (adjust as needed)

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024  # 16MB

# Define a function to perform signature verification

def verify\_signature(signature\_image):

    try:

        # Load and preprocess the test image

        img = image.load\_img(signature\_image, target\_size=(128, 128))

        img = image.img\_to\_array(img)

        img = np.expand\_dims(img, axis=0)

        img /= 255.0  # Normalize the input image

        # Make a prediction

        prediction = model.predict(img)

        # Interpret the prediction

        if prediction[0][0] >= 0.5:

            result = "True signature"

        else:

            result = "False signature"

        return result

    except Exception as e:

        return str(e)

@app.route('/')

def homepage():

    return render\_template('index.html')

@app.route('/verify', methods=['POST'])

def verify():

    signature =request.form.get('signature')

    try:

        # if signature not in request.files:

        #     return jsonify({'error': 'No file part'})

        # signature\_image = request.files['signature']

        # if signature\_image.filename == '':

        #     return jsonify({'error': 'No selected file'})

        result = verify\_signature(signature)

        if result:

            return result

        else:

            return result

        return jsonify({'result': result})

    except Exception as e:

        return jsonify({'error': str(e)})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

index.html:

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}">

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Signature Verification</title>

</head>

<body>

    <div class="left-image">

    <div class="container">

        <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTLQ70ihHgTkVgyYSXTMyk8tdkQlVGcBMkA5ChyJ71hbv7CaUfxPL44No-kVR\_pEcQDnTI&usqp=CAU" alt="image" width="35%">

        <h1>Signature Verification</h1>

        <form id="signatureForm" action="/verify" method="post">

            <label for="signatureImage">Upload Signature</label>

            <input type="file" accept="image" id="signatureImage" name="signature" required>

            <button type="submit">Verify Signature</button>

        </form>

        <div id="result"></div>

    </div>

</body>

</html>

style.css:

       body {

            font-family: arial;

            margin: 0;

            padding: 0;

            display: flex;

            flex-direction: row; /\* Change to row to place image at the left \*/

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        /\* New style for the left image container \*/

        .left-image {

            flex: 1; /\* Takes up 1/2 of the available space \*/

            padding: 20px; /\* Optional padding \*/

        }

        .container {

            /\* margin-left: 570px; \*/

            background-color: rgba(255, 255, 255, 0.8);

            padding: 20px;

            border-radius: 5px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

            text-align: center;

            flex: 1; /\* Takes up 1/2 of the available space \*/

        }

        h1{

            color: #333;

        }

        form{

            display: flex;

            flex-direction: column;

            align-items: center;

            margin-top: 20px;

        }

        input[type="file"]{

            display: none;

        }

        label {

            background-color: #007bff;

            color: #fff;

            padding: 10px 20px;

            border-radius: 5px;

            cursor: pointer;

        }

        label:hover {

            background-color: #0056b3;

        }

        button {

            background-color: #007bff;

            color: #fff;

            padding: 10px 20px;

            border: none;

            border-radius: 5px;

            margin-top: 10px;

            cursor: pointer;

        }

        button:hover {

            background-color: #0056b3;

        }

        #result {

            margin-top: 20px;

            font-weight: bold;

        }