SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

A

MINI PROJECT REPORT ON "String Matching Visualizer"

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE FULFILLMENT OF THE REQUIREMENT

OF

Laboratory Practice IV

Fourth Year Computer Engineering

Academic Year 2024-25

BY

Name of Students:	**Roll No.:**
Pratik Abhang	4101001
Darshana Borase	4101030
Omkar Darekar	4101032
Chaitanya Gunde	4101056

Under the Guidance of

Mr. S. A. Shinde

DEPARTMENT OF COMPUTER ENGINEERING
STES'S SINHGAD INSTITUTE OF TECHNOLOGY AND SCIENCE
NARHE, PUNE – 411041

Sinhgad Institutes

CERTIFICATE

This is to certify that,

Name of Students:	**Roll No.:**
Pratik Abhang	4101001
Darshana Borase	4101030
Omkar Darekar	4101032
Chaitanya Gunde	4101056

studying in SEM-VII, B.E. Computer Engineering, they have successfully completed their LP-III Lab Mini-Project report titled **"String Matching Visualizer"** under the supervision of **Mr. S. A. Shinde**, in fulfillment of the requirements for Laboratory Practice-III for the academic year 2024-2025, as prescribed by Savitribai Phule Pune University, Pune.

Mr. S. A. Shinde	Dr. G. S. Navale
Guide	**HOD**
Department of Computer Engineering	**Department of Computer Engineering**

Dr. S. D. Markande

Principal

SINHGAD INSTITUTE OF TECHNOLOGY AND SCIENCE, NARHE, PUNE– 411041

Place:

Date:

ACKNOWLEDGEMENT

We take this opportunity to acknowledge each and every one who contributed towards our work. We express our sincere gratitude towards guide **Mr. S. A. Shinde**, Assistant Professor at Sinhgad Institute of Technology and Science, Narhe, Pune for her valuable inputs, guidance and support throughout the course.

We wish to express our thanks to **Dr. G. S. Navale,** Head of Computer Engineering Department, Sinhgad Institute of Technology and Science, Narhe for giving us all the help and important suggestions all over the Work.

We thank all the teaching staff members, for their indispensable support and priceless suggestions. We also thank our friends and family for their help in collecting data, without their help **LP-III** report have not been completed. At the end our special thanks to **Dr. S. D. Markande,** Principal Sinhgad Institute of Technology and Science, Narhe for providing ambience in the college, which motivate us to work.

Name of Students:	**Sign:**
Pratik Abhang	
Darshana Borase	
Omkar Darekar	
Chaitanya Gunde	

CONTENT

INTRODUCTION

Today handwritten signatures are popularly used and acceptable means of biometric authentication. There are two cases of signature verification: Online and offline verification systems. Usually online is more useful than offline, because of the availability of information like stroke order, writing speed, pressure, etc. However, these performances need the cost of special hardware for recording the pen-tip path, improving its system cost, and decreasing the actual application situations.

In the offline system, just a static image of the signature is available, it is more complicated than an online system because dynamic information is not available and it is hard to reach them from the offline images. Offline signature verification depends on pattern recognition, signatures are described using fixed-size feature vectors. There are multiple ways for signature verification, one of them is by using graphs and structural pattern recognition.

Graphs allow extra powerful descriptions that can be useful for signature verification, for instance, by taking local information in nodes and their relationships in the global structure using edges. But the ability of graphs appears at the price of large computational complexity, this is the reason why graphs have only been used rather unusually for signature verification in the past. But in this principle, the model is very different, the model is an offline signature verification by using a convolutional Siamese network.

Siamese networks are double networks by shared weights, which can be trained to learn a feature location where comparable marks are located in proximity. This is performed by displaying the network to a pair of similar and dissimilar measurements reducing the distance between similar sets and simultaneously increasing the distance between dissimilar pairs. Tests conducted on cross-domain datasets indicate the ability of the network to handle forgery in several languages (scripts) and handwriting styles. Signature recognition systems based on machine learning have been shown to be very effective. They can be used in a variety of applications, such as verifying the authenticity of signatures on checks and documents, and authenticating users of electronic devices

PROBLEM STATEMENT

The current manual process of verifying signatures is time-consuming and error-prone. There is a pressing need for an automated signature identification application to streamline authentication processes. The problem at hand is to develop an application that can identify and verify signatures. The system should analyze handwriting styles, considering both inter-personal and intra-personal variations. It must be able to recognize unique patterns in signatures, allowing for reliable authentication.

OBJECTIVES

The primary objectives of this project are as follows:

1. To create a comprehensive dataset of authentic signatures from diverse college students.
2. To implement advanced image processing techniques for signature extraction and enhancement.
3. To employ machine learning algorithms for signature feature extraction and pattern recognition.
4. To ensure high accuracy and reliability in identifying authorized signatures within the dataset.

LIBRARIES USED

1. **Matplotlib**: Matplotlib is a comprehensive data visualization library in Python. It provides a wide array of tools for creating static, animated, and interactive plots. Matplotlib is commonly used for generating plots, histograms, power spectra, bar charts, error charts, scatterplots, and more. It is a fundamental tool for data scientists, engineers, and researchers for visualizing data in a clear and effective manner.

2. **NumPy**: NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. NumPy forms the foundation for many other libraries in the scientific Python ecosystem and is extensively used for tasks like linear algebra, Fourier transforms, random number capabilities, and more.

3. **OpenCV-Python**: OpenCV (Open Source Computer Vision Library) is a versatile computer vision library widely used for tasks related to image and video analysis. The Python wrapper, opencv-python, provides easy access to OpenCV's extensive functionality. It includes a vast set of tools for tasks like object detection, facial recognition, image processing, and more. OpenCV is crucial for applications in computer vision, robotics, and machine learning.

4. **SciPy**: SciPy is a library built on top of NumPy, providing a wide range of scientific and technical computing functions. It includes modules for optimization, integration, interpolation, signal and image processing, statistics, and more. SciPy complements NumPy by adding a variety of high-level functions for common scientific computing tasks, making it an essential tool for scientific research and engineering applications.

5. **scikit-image**: scikit-image is a powerful image processing library that builds on NumPy, SciPy, and Matplotlib. It provides a comprehensive suite of functions for image analysis, including features like image filtering, segmentation, morphological operations, and feature extraction. scikit-image is extensively used in fields like medical imaging, computer vision, and scientific imaging.

6. **tifffile**: tifffile is a Python library for reading and writing TIFF (Tagged Image File Format) files, a popular file format for storing images. It provides a simple and efficient interface for working with multi-dimensional image data. tifffile is particularly useful in scientific applications where handling multi-channel or multi-dimensional images is common.

DESIGN / IMPLEMENTATION

Convolution Neural Networks Algorithm:

Convolutional Neural Networks (CNNs) play a pivotal role in signature recognition due to their exceptional ability to learn and extract intricate features from images. In this context, CNNs analyze signature images through a hierarchical series of convolutional layers, followed by pooling layers for down-sampling and non-linear activation functions. These layers collectively learn meaningful patterns, such as strokes, curves, and unique characteristics of signatures. The network's architecture enables it to capture local relationships within the image, allowing for the detection of complex, spatially-dependent features crucial for accurate recognition. Through training on a diverse dataset, CNNs can autonomously discern distinctive patterns in signatures, ultimately leading to a highly accurate and reliable recognition system.

Algorithm:

```
def captureImage(ent, sign=1):
if(sign == 1)
          filename = os.getcwd()+'\\temp\\test_img1.png'
else:
             filename = os.getcwd()+'\\temp\\test_img2.png'
# messagebox.showinfo(
#     'SUCCESS!!!', 'Press Space Bar to click picture and ESC to exit')
res = None
res = messagebox.askquestion(
  'Click Picture', 'Press Space Bar to click picture and ESC to exit')
if res == 'yes':
  capture_image_from_cam_into_temp(sign=sign)
  ent.delete(0, tk.END)
  ent.insert(tk.END, filename)
return True
```

Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) is a feature extraction technique widely employed in signature recognition. It works by capturing local gradients in an image to represent its texture and shape characteristics. In the context of signature recognition, HOG analyzes the distribution of gradient orientations in small image regions. This allows it to discern significant patterns in signatures, such as strokes, loops, and curves, which are vital for distinguishing genuine from forged signatures. By quantifying these gradients and constructing histograms, HOG effectively transforms complex signature images into a compact and discriminative feature vector. This vector serves as input for machine learning algorithms, enabling accurate signature recognition based on distinctive texture and shape information.

Algorithm:

```
feature_vectors = []
for i in range(0, len(histogram_points_nine) - 1, 1):
  temp = []
  for j in range(0, len(histogram_points_nine[0]) - 1, 1):
    values = [[histogram_points_nine[i][x] for x in range(j, j+2)] for i in range(i, i+2)]
    final_vector = []
    for k in values:
      for l in k:
        for m in l:
          final_vector.append(m)
    k = round(math.sqrt(sum([pow(x, 2) for x in final_vector])), 9)
    final_vector = [round(x/(k + epsilon), 9) for x in final_vector]
    temp.append(final_vector)
  feature_vectors.append(temp)
```

Algorithm for the proposed scheme

Step 1: Handwritten signature is scanned.

Step 2: The signature is preprocessed and converted into binary or grayscale as per requirement, removing noise from signatures, thinning signatures, and finally normalizing the signature.

Step 3: Special domain features such as high-intensity variation points and cross-over points are extracted from genuine as well as test signatures.

Step 4: Compared features with the help of graph matching method as a classifier

Check Similarity:

```
def checkSimilarity(window, path1, path2):
    result = match(path1=path1, path2=path2)
    if(result <= THRESHOLD):
        messagebox.showerror("Failure: Signatures Do Not Match",
                    "Signatures are "+str(result)+f" % similar!!")
        pass
    else:
        messagebox.showinfo("Success: Signatures Match",
                    "Signatures are "+str(result)+f" % similar!!")
    return True
```

The Convolution Neural Network (CNN) algorithm would work as follows:

1. Collect a dataset of signatures. The dataset should be large and representative of the signatures that the system will be used to identify.
2. Preprocess the images. This may involve resizing the images, converting them to grayscale, and normalizing the pixel values.
3. Design the CNN architecture. The CNN architecture should be tailored to the specific task of signature recognition. It should include a number of convolutional layers, followed by fully connected layers.
4. Train the CNN. The CNN is trained on the dataset of signatures. During training, the CNN learns to extract features from the signatures and to classify them into different classes.

Advantages of using CNNs for signature recognition:

1. CNNs are able to learn complex features from the signatures, which can lead to high accuracy.
2. CNNs are robust to variations in the signatures, such as different writing instruments and writing styles.
3. CNNs can be scaled to process large volumes of signatures efficiently.

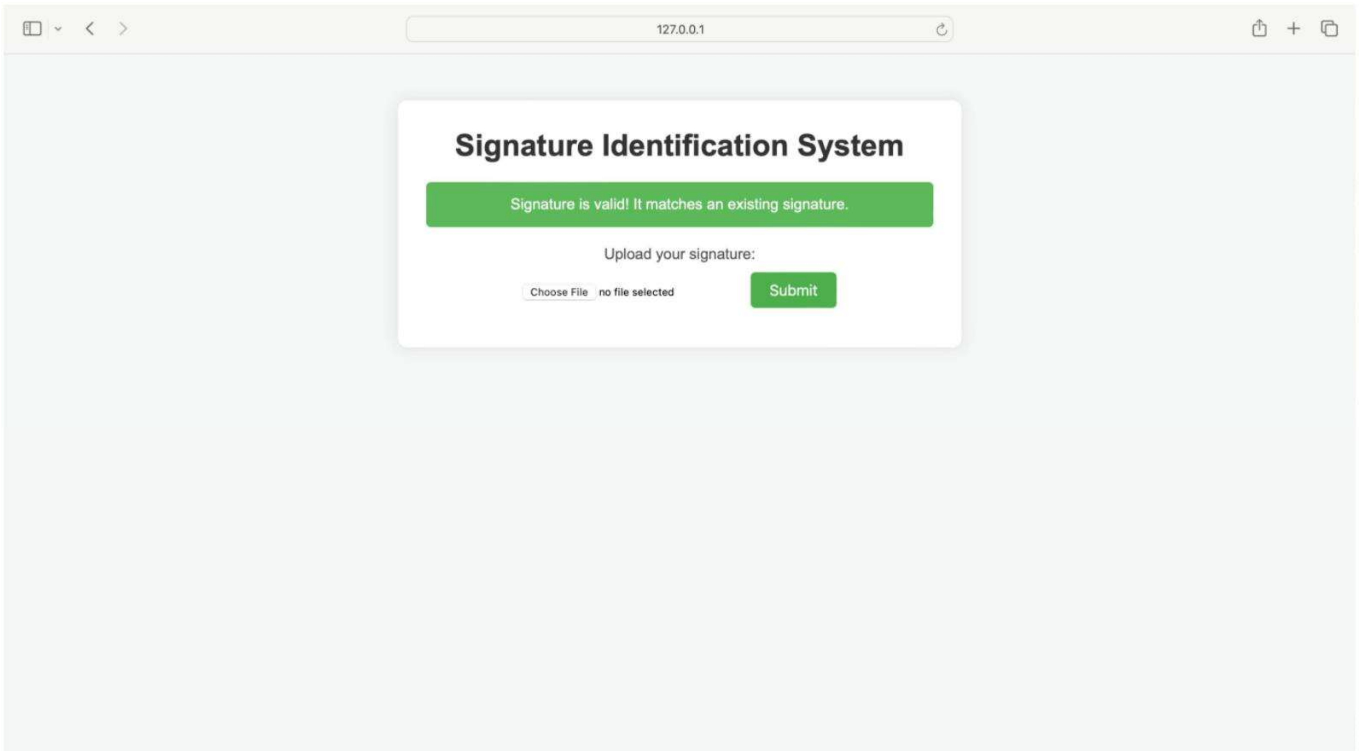
The Histogram of Oriented Gradients (HOG) algorithm would work as follows:

1. Preprocess the image. This may involve resizing the image, converting it to grayscale, and normalizing the pixel values.
2. Compute the HOG feature descriptor for the image. This can be done using a variety of libraries and tools.
3. Train a machine learning algorithm to classify the HOG feature descriptors. Some common machine-learning algorithms for signature recognition include support vector machines (SVMs) and random forests.
4. Evaluate the machine learning algorithm on a held-out dataset. This will help to ensure that the algorithm generalizes well to new data.

Advantages of using Histogram of Oriented Gradients (HOG) recognition:

1. HOG is a simple and efficient feature descriptor to compute.
2. HOG is robust to changes in illumination and translation.
3. HOG has been shown to be effective for signature recognition in a variety of studies.

RESULT/ANALYSIS



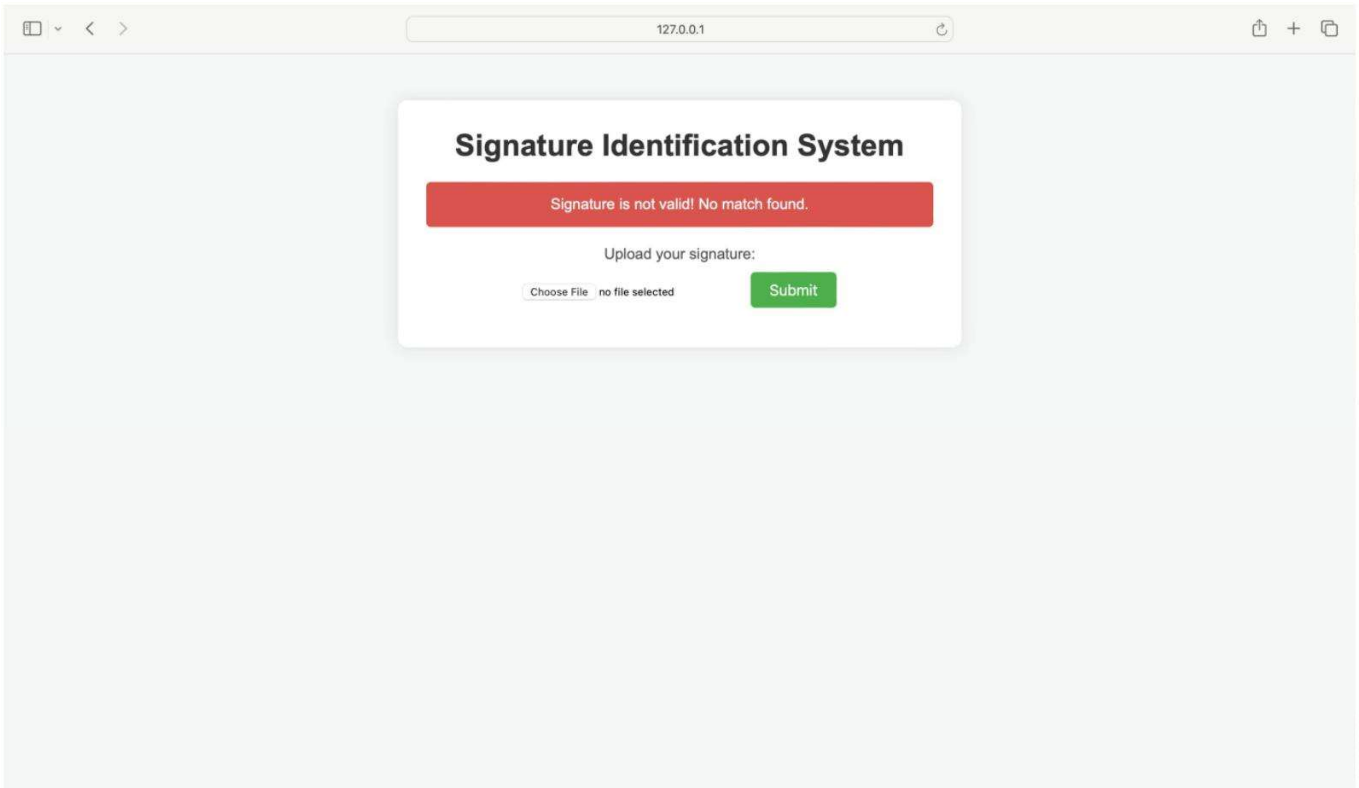
127.0.0.1

Signature Identification System

Signature is valid! It matches an existing signature.

Upload your signature:

Fig.1 Matched Sign



127.0.0.1

Signature Identification System

Signature is not valid! No match found.

Upload your signature:

Fig 2. Mismatched Sign

CONCLUSION

In conclusion, the implementation of a Signature Recognition system using machine learning represents a significant advancement in authentication technology. Through the utilization of sophisticated algorithms and feature extraction techniques, this system offers a robust and reliable means of verifying signatures. By analyzing distinctive patterns and characteristics, the system can effectively differentiate between genuine and forged signatures, mitigating potential security risks associated with manual verification processes. This technology not only streamlines administrative tasks but also bolsters the overall integrity of authentication processes within various domains. Overall, the Signature Recognition system represents a pivotal advancement in authentication solutions, providing a highly secure and efficient means of identity verification in an increasingly security-conscious world.