## A Project Report

**on**

**OFFLINE SIGNATURE FORGERY DETECTION**

**submitted in partial fulfillment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**by**

**17WH1A05B2 Ms. A. HARI PRASANNA**

**18WH5A0522 Ms. G. SUDHASREE**

**17WH1A0577 Ms. P. ANILA**

**under the esteemed guidance of**

**MS. B. NAGAVENI**

**Assistant Professor**



**Department of Computer Science and Engineering**

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**

**Bachupally, Hyderabad – 500090**

**May, 2021**

**DECLARATION**

We hereby declare that the work presented in this project entitled **“OFFLINE SIGNATURE FORGERY DETECTION**” submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women**’**, Hyderabad is an authentic record of our original work carried out under the guidance of Ms. B. Nagaveni, Assistant Professor, Department of CSE.

Sign. with date:

**Ms. A. HARI PRASANNA**

**(17WH1A05B2)**

Sign. with date:

**Ms. G. SUDHASREE**

**(18WH5A0522)**

Sign. with date:

**Ms. P. ANILA**

**(17WH1A0577)**

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**

**Bachupally, Hyderabad – 500090**

**Department of Computer Science and Engineering**



###### CERTIFICATE

This is to certify that the Project Work report on “**OFFLINE SIGNATURE FORGERY DETECTION**” is a bonafide work carried out by Ms. A. HARI PRASANNA(17WH1A05B2) ; Ms. G. SUDHASREE(18WH5A0522) ; Ms. P. ANILA (17WH1A0577) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

**Head of the Department Guide**

**Dr. K. Srinivasa Reddy MS. B. Nagaveni**

**Professor and HoD, Assistant Professor**

**Department of CSE**

**External Examiner**

**ACKNOWLEDMENT**

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K. Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. B. Nagaveni, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE**Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

**Ms. A. HARI PRASANNA**

**(17WH1A05B2)**

**Ms. G. SUDHASREE**

**(18WH5A0522)**

**Ms. P. ANILA**

**(17WH1A0577)**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topic** | **Page No.** |
|  | Abstract | 7 |
|  | List of Figures | 8 |
| 1. | Introduction | 9 |
|  | 1.1 Objectives | 9 |
|  | 1.2 Methodology | 9 |
|  | 1.2.1 Dataset | 10 |
|  | 1.2.2 The proposed models | 12 |
| 2. | Theoretical Analysis of the proposed project | 15 |
|  | 2.1 Requirements Gathering | 15 |
|  | 2.1.1 Software Requirements | 15 |
|  | 2.1.2 Hardware Requirements | 15 |
|  | 2.2 Technologies Description | 15 |
| 3. | Design | 19 |
|  | 3.1 Introduction | 19 |
|  | 3.2 Architecture Diagram | 20 |
|  | 3.3 UML Diagrams | 21 |
|  | 3.3.1 Use Case Diagram | 21 |
|  | 3.3.2 Sequence Diagram | 22 |
|  | 3.3.3 Activity Diagram | 23 |
|  | 3.3.4 Collaboration Diagram | 24 |
|  | 3.3.5 Class Diagram | 25 |
| 4. | Implementation | 25 |
|  | 4.1 Coding | 25 |
|  | 4.6 Output Screenshots | 34 |
| 5. | Conclusion and Future Scope | 35 |
| 6. | References | 35 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ABSTRACT**

Signature forgery is one of the most frequently committed frauds. This drives cooperates and business organizations to very huge consequences which might include financial or reputational losses. In this respect, signature verification is quite crucial.

Therefore, this project aims to model, train and calculate the accuracies by comparing the predictions of genuine or forged signatures using a few models which acts as an application for automating the prediction. This is implemented by modeling our image dataset with different classification models. The dataset contains 1021 images. The dataset taken is the “Handwritten Signatures” dataset containing both forged and genuine signature examples in a folder structure. The accuracies of various classifiers upon the dataset are found.

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | Fig No. | Fig Name | Page No. |
| 1. | 1.2.2 | Models used | 11 |
| 2. | 1.2.2.1 | Sequential Model | 11 |
| 3. | 1.2.2.1.1 | Max Pooling in Sequential | 11 |
| 4. | 1.2.2.1.2 | Flattening in CNN | 12 |
| 5. | 1.2.2.1.3 | Full Connection in CNN | 12 |
| 6. | 1.2.2.2 | Support Vector Machine(SVM) | 13 |
| 7. | 1.2.2.3 | Principle Component Analysis(PCA) + SVM | 14 |
| 8. | 1.2.2.4 | Ensemble Learning | 14 |
| 9. | 3.2 | Architecture Diagram | 19 |
| 10. | 3.3 | UML diagrams | 20 |
| 11. | 3.3.1 | Use case Diagram | 20 |
| 12. | 3.3.2 | Sequence Diagram | 21 |
| 13. | 3.3.3 | Activity Diagram | 22 |
| 14. | 3.3.4 | Collaboration Diagram | 23 |
| 15. | 3.3.5 | Class Diagram | 24 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. INTRODUCTION**

Offline signatures are still in use for many banking purposes, several account verification and confirmation, legal documentation, and doctor prescriptions. It is often considered as the security identity for a person, misusing which can lead to economic losses, reputational losses, and personal losses as well. In this respect, signature verification is very crucial in the field of biometrics. Such recognition systems are useful for authorization and achieving a high degree of security.

Although signatures forgery are often manually detected by experts, still high accuracy is not always achieved. There are many difficulties in manual forgery detection due to variations in handwriting style and professionalism of forgers. Therefore, a high degree of verification is necessary to avoid any errors and machine learning is one such alternative that might help us in reducing human errors.

**1.1 Objectives**

This project aims to calculate the accuracy for various classification approaches by training the models with a public handwritten dataset. We then try to compare the accuracies of various models and predict them using the model that best fits. The classifiers used in this project are Vanilla SVM(Support Vector Machine), PCA(Principal Component Analysis) + SVM, Sequential model with layers, and Ensemble learning technique.Through this project, we would like to explore various classifiers that are not too frequently used and obtain the accuracies using the dataset passing the data through the models.

**1.2 Methodology**

To classify the signatures, a dataset is required to train. That dataset happened to be the handwritten dataset from the kaggle site. The session includes detail step by step methodology

**1.2.1 Dataset**

The dataset folder contains four folders that included the following structure. The dataset contains genuine and forged signatures of 30 people. Each person has 5 genuine signatures which they made themselves and 5 forged signatures someone else made.

The naming of images is explained in detail. To exemplify, NFI-00602023(say) is an image of the signature of person number 023 done by person 006. This is a forged signature. NFI-02103021 is an image of the signature of person number 021 done by person 021. This is a genuine signature. This is how the images are included in the folder structure.

## Introduction

This project helps us in exploring various classifiers and the difference of those classifiers with dimensionality reduction techniques like Principal Component Analysis that are not straightforwardly trained with the conventional classifiers like Convolutional Neural Networks and Deep neural network. This project does include a sequential network but does not completely rely on the most frequently used or standard machine learning models.

As the current public dataset is small compared to various research datasets we tried to focus more on the classifiers other than using Convolutional neural network or Siamese model(Deep neural network) as they are a lot simpler and faster to implement when compared to the multi-perceptron model. We often miss the concept that other classifiers perform well and outperform when compared to the neural networks when our dataset is small.

What are the advantages that these techniques provide:

* These classifiers outperform when using a simple dataset like ours.
* They are usually simpler and easier to implement than complex classifiers and understand the various implementation of a problem.
* We get to understand the details more when we use a only a neural network and the code explicability increases too.

## Loading the dataset

To load the images from the dataset folder, we use the glob library which helps in loading the data into the list with regex expression provided along with the file path in the runtime. The dataset contains all signatures in .png files. To recollect, we have two classes for this project - genuine and forged. Hence, this is a binary classification problem.

Loading the dataset is a common step for all the models. However, the steps succeeding this steps vary basing on the model type.

**Pre-processing step and feature extraction**

The model requires converting the images into vectors basing on the content. Therefore, to let that happen, we need to convert the image into RGB composition or Grayscale, since it helps in extracting vectors and is usually used in many of the preprocessing steps while working with images.

This RGB composition here is used for models which do not have dimensionality reduction step. The dimensionality reduction technique used for SVM is principal component analysis. For this technique, We convert our images to black & white and take their negative. cv2 library contains functions which help us save time from writing function from scratch. Hence, this allows us to convert it easily.

To convert the images into vectors we need to transform each image into a matrix representing the pixel activation(a zero for a black pixel, and a 255 for a white one). We then rescale each element of the matrix by dividing it by the maximum possible value (255), before flattening the matrix.Though the conversion of images are different the purpose behind that is the same.

**Dealing with data**

While converting the dataset into vectors (a.k.a feature vector), we also try to store the data into the train and test lists which are used while training and accuracy calculation steps. These lists are not necessarily single-dimensional. Aligning these lists according to the fit transform parameters is quite essential. Therefore, this project has stored the train and test datasets differently.

**1.2.2 The proposed models**

**1. Sequential Model**

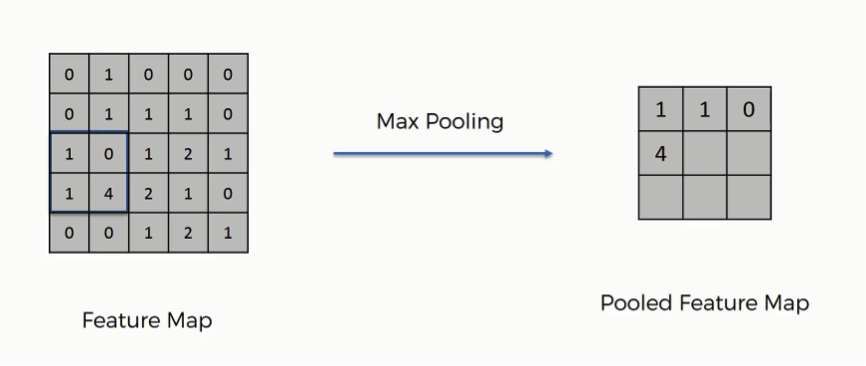
Sequential model’s architecture varies with the type of problem at hand. The proposed sequential model has 9 layers in total. ReLu activation function is applied to the output of every convolutional layer and fully connected layer. After max pooling is applied, the output is flattened and again sent dense layer. The output of this layer is given to the soft-max function which produces a probability distribution of the fouroutput classes. The model is trained using adaptive moment estimation.

A diagram of a microchip

Description automatically generated with medium confidence

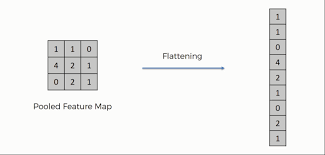
**Fig 1.2.2.1: Convolution Neural Network**

**Max pooling:** Max pooling is a sample-based discretization process. The objective is to down-sample an input representation(image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to bemade about features contained in the sub-regions binned**.**

****

**Fig 1.2.2.1.2: Max Pooling in CNN**

**Flattening:** Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector.



**Fig 1.2.2.1.3: Flattening in CNN**

**Full Connection:** At the end of a CNN, the output of the last Pooling Layer acts as a input to theso called Fully Connected Layer. There can be one or more of these layers(“fully connected” means that every node in the first layer is connected to every node in the second layer).

As you see from the image below, we have three layers in the full connection step:

* Input layer
* Fully-connected layer
* Output layer



**Fig 1.2.2.1.4: Full Connection in CNN**

**2. Support Vector Machine Model**

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However,  it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).

**Diagram

Description automatically generated**

**Fig 1.2.2.2.4: Support vector machine model**

**2. Support Vector Machine with Dimensionality reduction Model**

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

And then the SVM is applied normally. This is just to check if that builds accuracy. But, as the dataset is small dimensionality reduction is not of much use as there are high chances of informational loss. This technique is usually performed on large datasets when there are very huge number of dimensions to handle. Therefore, the output of the PCA is fit into the SVM using the grid search cv function in sklearn library.

**Chart, scatter chart

Description automatically generated**

**Fig 1.2.2.2.5 : PCA**

**Ensemble learning**

The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions. In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.

We have a quite of a lot of ensemble learning techniques in usage, out of which “boosting” is one such efficient one. We also have many subdivisions in boosting as well. So, out of this, classic gradient boosting is a good alternative. The reason out of using all these techniques is to get minimum loss by re-estimating the parameters at the leaves of the tree.

Diagram

Description automatically generated

**Fig 1.2.2.2.6: Boosting**

**2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT**

**2.1 Requirements Gathering**

**2.1.1 Software Requirements**

Programming Language : Python 3.6

Dataset : Handwritten Signature Dataset

Libraries Used : Numpy, Pandas, CV2, Matplotlib, Scikit-learn

Tool : Colab notebook

**2.1.2 Hardware Requirements**

Operating System: Any

Processor : Intel Core i5

Required Unit : GPU

Memory : 8 GB (RAM)

**2.2 Technologies Description**

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Handwritten Signature Dataset**

The dataset for the experiment is downloaded from the handwritten signatures database which contains various signatures of people both forged and genuine. It contains a collection of images taken from 30 people.

The dataset is made of a bunch of diverse people who have shared both genuine and forged signatures.

Contains Genuine and Forged signatures of 30 people. Each person has 5 Genuine signatures which they made themselves and 5 Forged signatures someone else made.  
The naming of images is explained here.NFI-00602023 is an image of signature of person number 023 done by person 006. This is a forged signature. NFI-02103021 is an image of signature of person number 021 done by person 021. This is a genuine signature.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

* **NumPy**: Base n-dimensional array package
* **SciPy**: Fundamental library for scientific computing
* **Matplotlib**: Comprehensive 2D/3D plotting
* **IPython**: Enhanced interactive console
* **Sympy**: Symbolic mathematics
* **Pandas**: Data structures and analysis
* Extensions or modules for SciPy care conventionally named [SciKits](http://scikits.appspot.com/scikits). As such, the module provides learning algorithms and is named scikit-learn.

**Google Colaboratory Notebook**

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

* Zero configuration required
* Free access to GPUs
* Easy sharing

Whether you're a **student**, a **datascientist** or an **AIresearcher**, Colab can make your work easier. Colab notebooks allow you to combine **executablecode** and **richtext** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.

ColabNotebook is maintained by the people at Google

**3. DESIGN**

**3.1 Introduction**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

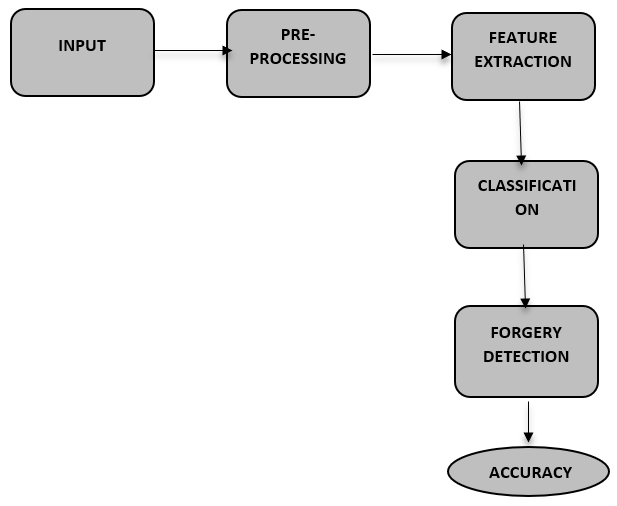
The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**3.2 Architecture Diagram**

Applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.



**Fig 3.2: Architecture Diagram**

**3.3 UML Diagrams**

**3.3.1 Use Case Diagram**

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.Hence to model the entire system, a number of use case diagrams are used.



**Fig 3.2.1: Use Case Diagram**

**3.3.2 Sequence Diagram**

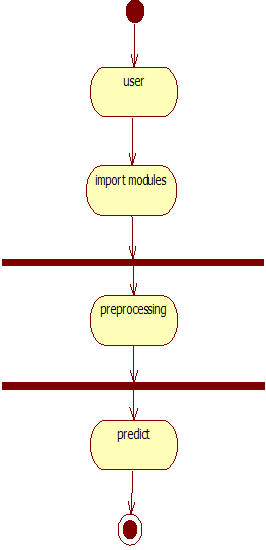
Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.



**Fig 3.3.2: Sequence Diagram**

**3.3.3 Activity Diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Fig 3.2.3: Activity Diagram**

**3.3.4 Collaboration Diagram**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

****

**Fig 3.2.4:Collabration Diagram**

**3.3.5 Class Diagram**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



**Fig 3.2.5: Class Diagram**

**4. IMPLEMENTATION**

**4.1 Coding**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

img = plt.imread('/content/drive/MyDrive/hadwritten-signatures/sample\_Signature/sample\_Signature/forged/NFI-00301001.png')

plt.imshow(img)

<matplotlib.image.AxesImage at 0x7fc3f4032910>

import glob

gen = [glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset1/real/\*.\*"),

       glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset2/real/\*.\*"),

       glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset3/real/\*.\*"),

       glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset4/real/\*.\*")]

forg = [glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset1/forge/\*.png"),

        glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset2/forge/\*.png"),

        glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset3/forge/\*.png"),

        glob.glob("/content/drive/MyDrive/hadwritten-signatures/Dataset\_Signature\_Final/Dataset/dataset4/forge/\*.png")]

import glob

import keras

import cv2

train\_data = []

train\_labels = []

train\_labels\_linear = []

test\_data = []

test\_labels = []

for data in range(len(gen)):

for i in gen[data]:

if data == 3:

image = cv2.imread(i)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image = cv2.resize(image, (224, 224))

test\_data.append(image)

test\_labels.append(0)

else:

image = cv2.imread(i)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image = cv2.resize(image, (224, 224))

train\_data.append(image)

train\_labels.append([0,1]) #genuine = 0

train\_labels\_linear.append(0)

for data in range(len(forg)):

for j in forg[data]:

if data == 3:

image = cv2.imread(j)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image = cv2.resize(image, (224, 224))

test\_data.append(image)

test\_labels.append(1)

else:

image = cv2.imread(j)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image = cv2.resize(image, (224, 224))

train\_data.append(image)

train\_labels.append([1,0]) #forged = 1

train\_labels\_linear.append(1)

train\_data = np.array(train\_data)/255.0

train\_labels = np.array(train\_labels)

test\_data = np.array(test\_data)/255.0

test\_labels = np.array(test\_labels)

train\_data.shape

(540, 224, 224, 3)

from sklearn.utils import shuffle

train\_data,train\_labels = shuffle(train\_data,train\_labels)

test\_data,test\_labels = shuffle(test\_data,test\_labels)

from keras.models import Sequential

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

from keras\_preprocessing.image import ImageDataGenerator

from sklearn.metrics import confusion\_matrix as CM

from keras.optimizers import Adam

network = Sequential()

network.add(Conv2D(64,(3,3),input\_shape=(224,224,3),activation='relu'))

network.add(MaxPooling2D(3,3))

network.add(Conv2D(32,(3,3),activation='relu'))

network.add(MaxPooling2D(2,2))

network.add(Flatten())

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

network.add(Dropout(rate=0.3))

network.add(Dense(2,activation = 'softmax'))

network.compile(optimizer=Adam(lr=0.001),loss="binary\_crossentropy",metrics=["accuracy"])

network.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 222, 222, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 74, 74, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 72, 72, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 36, 36, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 41472) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 128) 5308544

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 2) 258

=================================================================

Total params: 5,329,058

Trainable params: 5,329,058

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer\_v2/optimizer\_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

"The `lr` argument is deprecated, use `learning\_rate` instead.")

from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBoard

earlyStopping = EarlyStopping(monitor='val\_loss',

min\_delta=0,

patience=3,

verbose=1)

EPOCHS = 5

BS = 1

progess = network.fit(train\_data,train\_labels, batch\_size=BS,epochs=EPOCHS, validation\_split=.05)

Epoch 1/5

513/513 [==============================] - 46s 5ms/step - loss: 0.8158 - accuracy: 0.4540 - val\_loss: 0.6952 - val\_accuracy: 0.3333

Epoch 2/5

513/513 [==============================] - 2s 4ms/step - loss: 0.7021 - accuracy: 0.5339 - val\_loss: 0.6928 - val\_accuracy: 0.6667

Epoch 3/5

513/513 [==============================] - 2s 4ms/step - loss: 0.7025 - accuracy: 0.4806 - val\_loss: 0.6976 - val\_accuracy: 0.3333

Epoch 4/5

513/513 [==============================] - 2s 4ms/step - loss: 0.7064 - accuracy: 0.5072 - val\_loss: 0.6992 - val\_accuracy: 0.3333

Epoch 5/5

513/513 [==============================] - 2s 4ms/step - loss: 0.6933 - accuracy: 0.5410 - val\_loss: 0.6985 - val\_accuracy: 0.3333

acc = progess.history['accuracy']

val\_acc = progess.history['val\_accuracy']

loss = progess.history['loss']

val\_loss = progess.history['val\_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training acc')

plt.plot(epochs, val\_acc, 'r', label='Validation acc')

plt.title('Training and validation accuracy')

plt.legend()

plt.plot(epochs, loss, 'b', label='Training loss')

plt.plot(epochs, val\_loss, 'r', label='Validation loss')

plt.title('Training and validation loss')

plt.legend()

plt.show()

plt.figure()

pred = network.predict(test\_data)

from sklearn.metrics import accuracy\_score

accuracy\_score(pred.argmax(axis=1), test\_labels)

**SVM**

from sklearn.svm import SVC

SupportVectorClassModel = SVC()

train\_data.shape

(540, 2)

X\_train = train\_data.reshape(540,3\*224\*224)

X\_test = test\_data.reshape(90,150528)

(90, 224, 224, 3)

train\_labels\_linear = []

for i in train\_labels:

if(i[0] == 0):

train\_labels\_linear += [0]

else: train\_labels\_linear += [1]

SupportVectorClassModel.fit(X\_train, train\_labels\_linear)

SVC(C=1.0, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

y\_pred = SupportVectorClassModel.predict(X\_test)

y\_test = test\_labels

svm\_accuracy = accuracy\_score(y\_test, y\_pred)\*100

print(svm\_accuracy)

**PCA + SVM**

import glob

import keras

import cv2

train\_data1 = []

train\_labels1 = []

train\_labels\_linear1 = []

test\_data1 = []

test\_labels1 = []

for data in range(len(gen)):

for i in gen[data]:

if data == 3:

image = cv2.imread(i)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (224, 224))

test\_data1.append(image)

test\_labels1.append(0)

else:

image = cv2.imread(i)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (224, 224))

train\_data1.append(image)

train\_labels1.append([0,1])

train\_labels\_linear1.append(0)

for data in range(len(forg)):

for j in forg[data]:

if data == 3:

image = cv2.imread(j)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (224, 224))

test\_data1.append(image)

test\_labels1.append(1)

else:

image = cv2.imread(j)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (224, 224))

train\_data1.append(image)

train\_labels1.append([1,0])

train\_labels\_linear1.append(1)

train\_data1 = np.array(train\_data1)/255.0

train\_labels1 = np.array(train\_labels1)

test\_data1 = np.array(test\_data1)/255.0

test\_labels1 = np.array(test\_labels1)

test\_data1.shape

(90, 224, 224)

X\_train = train\_data1.reshape(540,224\*224)

X\_test = test\_data1.reshape(90, 50176)

from sklearn.model\_selection import GridSearchCV

def best\_SVC(X,y):

svc\_model = SVC()

param\_dic = {'C':[1,10,100],

'gamma':[0.001,0.005,0.01]}

clf = GridSearchCV(svc\_model, param\_dic, n\_jobs=-1)

clf.fit(X, y)

print("Best parameters: ", clf.best\_params\_)

return clf.best\_estimator\_

from sklearn.model\_selection import train\_test\_split

from sklearn.decomposition import PCA

from sklearn.metrics import accuracy\_score

def benchmark(X,y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y)

pca = PCA(n\_components = 24)

pca.fit(X\_train)

reduced\_X\_train, reduced\_X\_test = pca.transform(X\_train), pca.transform(X\_test)

best\_model = best\_SVC(reduced\_X\_train,y\_train)

predictions = best\_model.predict(reduced\_X\_test)

return accuracy\_score(y\_test, predictions)

## score\_on\_images = benchmark(X\_train, train\_labels\_linear1)

## print("Best accuracy on images: {}".format(score\_on\_images))

**Ensemble learning - Boosting**

model = GradientBoostingRegressor()

X\_train\_bagg = train\_data.reshape(540, 150528)

model.fit(X\_train\_bagg, train\_labels\_linear)

GradientBoostingRegressor(alpha=0.9, ccp\_alpha=0.0, criterion='friedman\_mse',

init=None, learning\_rate=0.1, loss='ls', max\_depth=3,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=100,

n\_iter\_no\_change=None, presort='deprecated',

random\_state=None, subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0, warm\_start=False)

X\_test\_bagg = test\_data.reshape(90, 150528)

test\_pred = model.predict(X\_test\_bagg)

test\_pred = test\_pred.reshape(90)

test\_labels = test\_labels.reshape(90)

from sklearn.metrics import mean\_squared\_error

print(mean\_squared\_error(test\_labels, test\_pred))

**4.2 OUTPUT SCREENSHOTS**

**Graphical user interface, text, application, Word

Description automatically generated**

**Fig 4.2.1 : Sequential Model Accuracy**

**Graphical user interface, application, website

Description automatically generated**

**Fig 4.2.2: Vanilla SVM Accuracy**

**Graphical user interface, Word

Description automatically generated**

**Fig 4.2.3 : SVM after Principal Component Analysis accuracy**

Graphical user interface, application, Word

Description automatically generated

**Fig 4.2.4: Gradient boosting Ensemble learning mean\_square\_error**

**5. CONCLUSION AND FUTURE SCOPE**

The application helps predicts whether the given signature is forged or genuine. However, we cannot rely on the current accuracy. Using a public dataset which is this small does not fetch great accuracies. Therefore, this model can be fed with large datasets to model to fit appropriately. Apparently that will not lead to a dilemma of overfitting or underfitting.

The future enhancement of this application is

* To run it with a large dataset.
* To display it with a user input in a more understandable way.

**6. REFERENCES**

1. TaraggyMghanim and Ayman MNabil, “Offline Signature Verification and Forgery Detection Approach”, IEEE 2018 – *base paper.*
2. Sjerome Gideon, Anurag Kandulna, Aron Abhishek Kujur, Kumudha Raimond, “Handwritten signature forgery detection using convolutional neural networks”, published at the 8th international conference on advances of computing and communication 2018.
3. Sung-Hyuk Cha and Charles C. Tappert, “Automatic Detection of Handwriting Forgery”, IEEE 2002.
4. Naouel Arab, Hassiba Nemmour, Youcef Chibani, “New Local Difference Feature for Off-Line Handwritten Signature Verification ”, IEEE 2019.
5. J. Hu, Y. Chen, “Offline signature verification using real adaboost classifier combination of pseudo-dynamic features, in: Document Analysis and Recognition (ICDAR)”, 2013 12th International Conference on, IEEE, 2013