

HANDWRITTEN DIGIT RECOGNIZER

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of
Technology

in

Computer Science and Engineering/Information Technology

By

Vatsal Agrawal -181359

Under the supervision

of

Dr. Himanshu Jindal

To



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled “**Handwritten Digit Recognizer**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Wakhnaghat is an authentic record of my own work carried out over a period from August 2021 to December 2021 under the supervision of **Dr. Hiamnshu Jindal ,Assistant Professor(SG)**, Department of CSE Jaypee University of Information Technology, Wakhnaghat.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Vatsal Agrawal 181359

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Almighty God for his divine blessing that makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Himanshu Jindal, Assistant Professor(SG)**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of “**Machine learning and Artificial Intelligence**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Himanshu Jindal**, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

TABLE OF CONTENTS

| | |
|-----------------------------------|----|
| 1. Certificate | i |
| 2. Acknowledgement | ii |
| 3. List of Abbreviations | iv |
| 4. List of Figures | v |
| 5. Abstract | vi |
| 6. Chapter-1 Introduction | 1 |
| 6.1. Introduction | |
| 6.2. Problem Statement | |
| 6.3. Objectives | |
| 6.4. Methodology | |
| 7. Chapter-2 Literature Survey | 5 |
| 8. Chapter-3 System Development | 7 |
| 8.1 SVM | 7 |
| 8.2 Neural Network | 13 |
| 8.3 Convolutional Neural Network | 18 |
| 9. Chapter-4 Performance Analysis | 24 |
| 10. Chapter-5 Conclusions | 30 |
| 11. References | 31 |
| 12. Appendices | 32 |

LIST OF ABBREVIATIONS

1. AI -Artificial Intelligence
2. CSV - Comma separated Values
3. HWR - Hand witing recognition
4. ML - Machine Learning
5. RBF -Radial Basis Function
6. SVM - Support Vector Machine
7. SVC - Support Vector Classifier

LIST OF FIGURES

| | | |
|--------|--|----|
| Fig.1 | Methodology of SVM classifier | 2 |
| Fig.2 | Algorithm Flowchart | 3 |
| Fig.3 | Literature survey | 5 |
| Fig.4 | Literature survey | 6 |
| Fig.5 | Svm Algorithm | 7 |
| Fig.6 | Svm Algorithm | 8 |
| Fig.7 | Svm in classifying two class | 9 |
| Fig.8 | Image of data set | 11 |
| Fig.9 | Code snippet for model training | 21 |
| Fig.10 | Output of the code | 22 |
| Fig.11 | Code snippet for plot image of digit 1 | 24 |
| Fig.12 | Code snippet for accuracy 1 | 25 |
| Fig.12 | Code snippet for accuracy 2 | 26 |
| Fig.12 | Code snippet for accuracy 3 | 27 |
| Fig.13 | Output image of digit 9 | 28 |
| Fig.14 | Output image of digit 6 | 28 |
| Fig.15 | Output image of digit 5 | 29 |
| Fig.16 | Output image of digit 0 | 29 |

ABSTRACT

The acknowledgment of Handwritten characters and digits has consistently been a truly challenging errand on account of the numerous varieties of transcribed characters with various composing styles.

Handwriting digit recognition systems are designed to transform handwritten digits into machine-readable representations. Handwritten Numeral Recognition plays an important role in postal automation services mainly in countries like India wherein more than one languages and scripts .The major objectives of this work is to create effective and reliable methods for accurately detecting numerals in order to make banking procedures more convenient and accurate.

This sort of clever framework is applied in different fields: really look at handling, handling of structures, programmed handling of manually written responses to an assessment, and so on. This last application is the subject of this work.

We use various machine learning algorithms to get the best accuracy for our result,we have use three type of algorithms to get best accuracy for our data set these are

SUPPORT VECTOR MACHINE,NEURAL NETWORK,AND CONVOLUTIONAL NEURAL NETWORK we got the best result in CONVOLUTIONAL NEURAL NETWORK which is 98% accuracy.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Handwriting recognition (HWR) is the methodology in which machine can succelly read the handwritten digits and character and can further interpreted as text or Number and convert it into digitised form.Optical Character Recognition (OCR) technology is used to convert images containing written text to machine-readable text data. The digital document becomes a modifiable file.

Text summarization refers to the technique of shortening lengthy text data. The aim is to create a consistent and fluent summary having only the important points sketched in the document. The summary gives insights into the whole document, leaving behind the insignificant and inessential pieces of text. It enhances readability, reduces time and eases the user. Automatic summarization tools are greatly needed to absorb relevant information faster and in an efficient manner.

1.2 Problem Statement

Our problem statement is handwritten digit recognition. Digits from 0-9 can be written in multiple ways by different people because of the variation in handwriting. Sometimes it is difficult to recognise the handwritten digits .Inorder to make this handwritten digit recognition feasible, our aim is to take an image of a handwritten digit and determine what digit it is with the help of our model deployed..

1.3 Objective

The main Objective of this project is to Successfully identify the handwritten digits from 0-9 and in order to achieve this we will be using Support vector Machine to implement a model .The SVM model will be able to identify the handwritten digits on the basis of pixel values as feature.So we can call this as a 10-class classification problem.

1.4 Methodology

For this problem, we are generating our very own dataset, The dataset consist of 600 Images of handwritten digits from 0-9. We are drawing the digit on paint and then capturing the image of the handwritten digit using the pyscreenshot package and then storing them in a folder each named from 0-9.

Each image that we stored in our dataset is 28x 28 pixels so in total there are 784 pixels in the image. So these 784 pixels are in such a way that the area where the digit is not drawn has a value of 0 and the area of the image in which the digit is drawn has a value other than 0. To generate the dataset we will be assigning 1 to the drawn region and 0 to the empty region where no digit is drawn.

In this project, I am trying to experiment with various hyperparameters in SVMs. With a sub-sample of 10-20% of the training data.

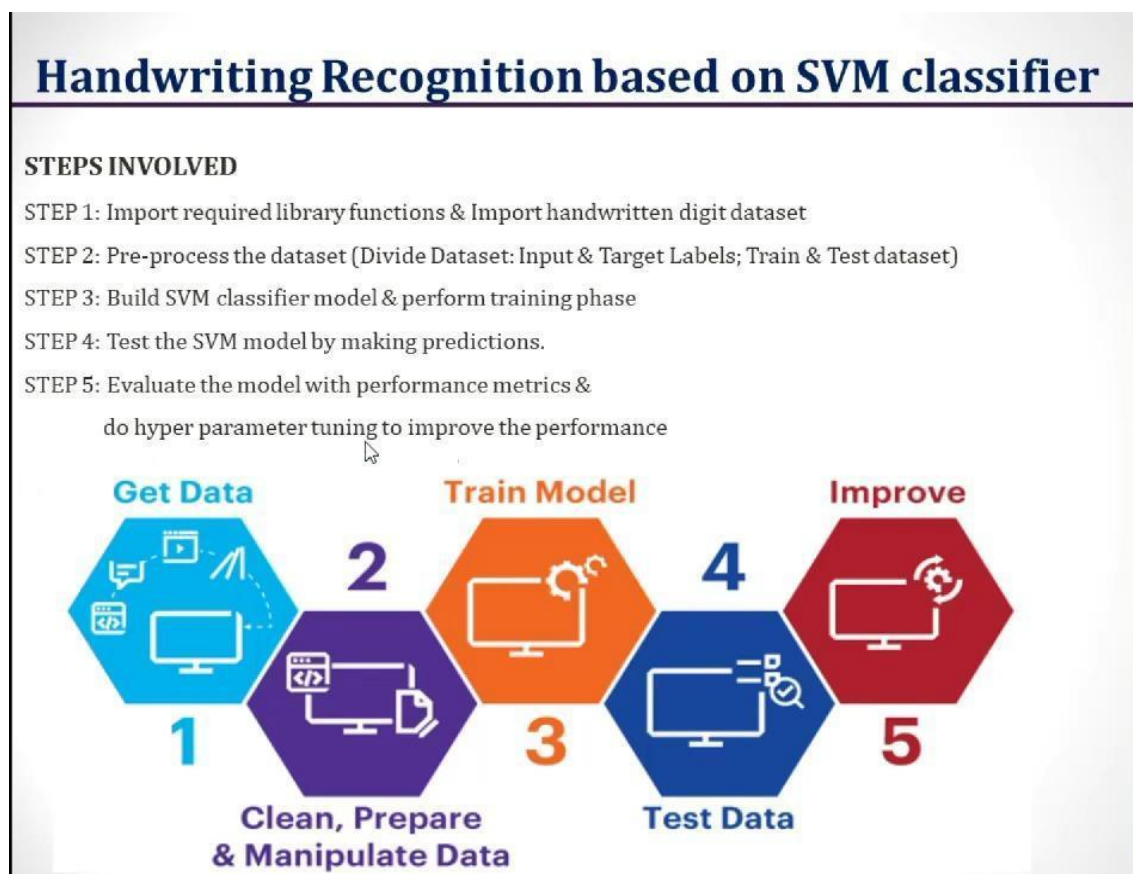


Fig.1

1.4.1 Collecting Dataset :

We have to collect images of digits (from 0 to 9). To collect images, a pyscreenshot package can be used. Through which we are Drawing image of digits and then capturing it and then storing it.

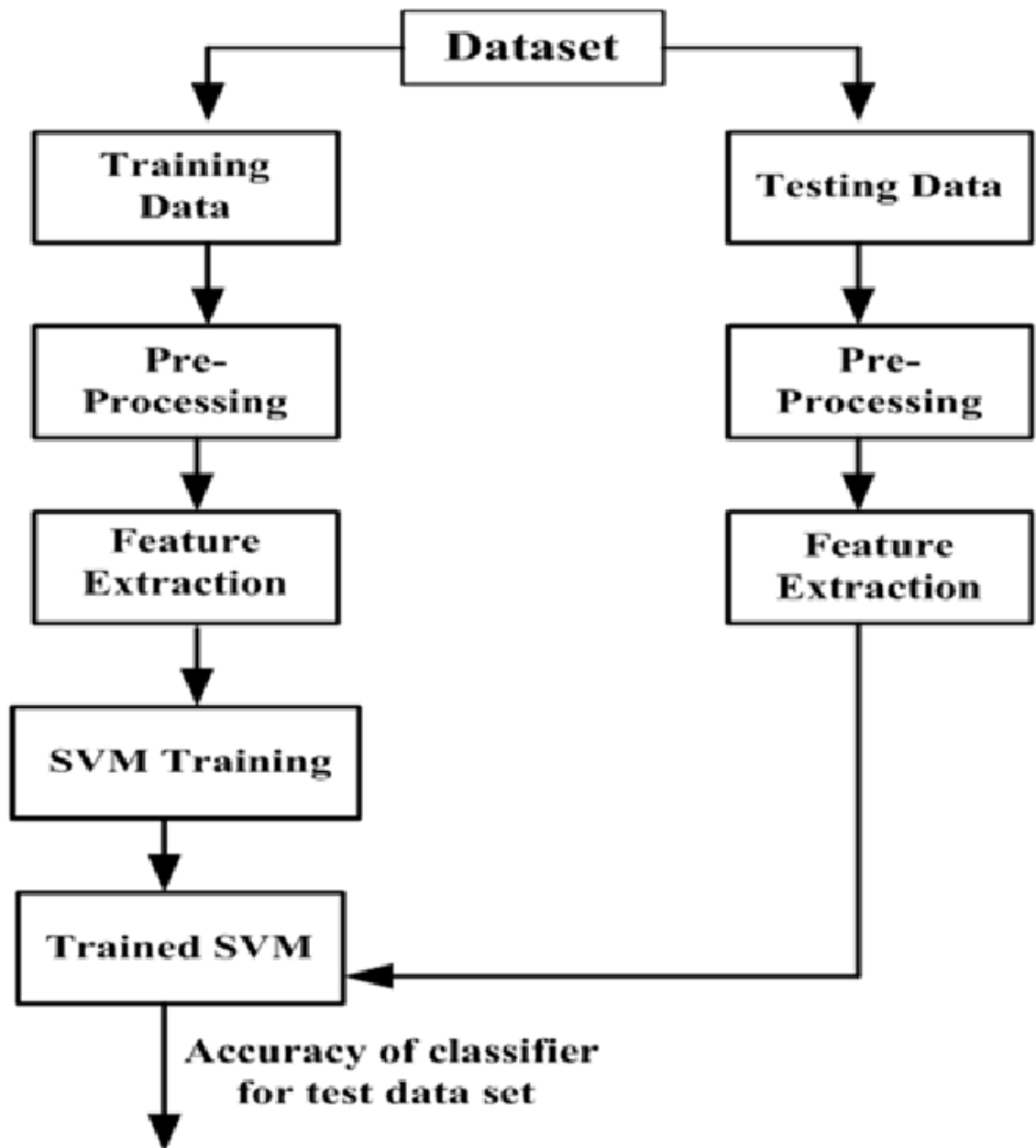


Fig.2

1.4.2 Generating Dataset:

We have to generate our dataset using images that we have collected . To generate a dataset, what we have to do is, we have to assign 1 to the drawn region and 0 to the background. That means, in our dataset, we will be having only two values i.e., 0 and 1. Typically, 0 represents black and 255 represents white. We are assigning 0 to pixel value from 0 to 100 and 1 to pixel value from 100 to 255. Now our pixel value is not from 0 to 255, it is only 0 and 1. In this way, we are generating a dataset (csv file).

Final step is to open the dataset, shuffle it i.e., change the position of each row of data and display it.

1.4.3 Model Training and Testing

We have to separate dependent (Y) and independent variables (X). Our independent variable will be the pixel value (i.e. 0 and 1). And our dependent variable will be our digit (i.e. from 0 to 9). In the training part, we train our model i.e., we give our model pixel value (bunch of 0 and 1) and also label (i.e. which digit is this).

After this step, our model gets learned. Now, in the testing part, we only give a pixel value (bunch of 0 and 1) to our model, our model has to predict that digit. We count how much our model returns a true answer. This way, we calculate accuracy.

1.4.4 Digit Prediction:

We have to predict the digit drawn in paint. We draw a digit in paint and at the same time, pass that digit to the model, our model has to predict that digit.

CHAPTER -2

LITERATURE SURVEY

The first paper we referred to is Handwritten Digit Recognizer Using Machine learning.

Hand Written Digit Recognition using Machine Learning

Publisher: IEEE

Cite This

PDF

Rohan Sethi ; Ila Kaushik [All Authors](#)

Published in: 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)

Date of Conference: 10-12 April 2020

INSPEC Accession Number: 19688997

Date Added to IEEE Xplore: 12 June 2020

DOI: 10.1109/CSNT48778.2020.9115746

► ISBN Information:

Publisher: IEEE

Print on Demand(PoD) ISSN: 2329-7182

Conference Location: Gwalior, India

fig.3

The main goal of this study is to show and illustrate the work that has been done on hand-written digit recognition. Characters and numbers written by hand. One of the most difficult and fascinating aspects of sample recognition and image processing is recognition. Hand-written digit popularity is an extremely difficult task. The numbers in this popularity venture aren't properly written or scripted since they vary in shape and length; as a result, feature extraction and segmentation of hand-written numerical script is time-consuming. For the purpose of segmentation within the suggested artworks, vertical and horizontal projections are employed. SVM is used for reputation and classification.

The next paper we referred is Off-line Handwritten Character recognition System Using SVM

Off-Line Handwritten Character Recognition System Using Support Vector Machine

Gauri Katiyar^{1,*}, Ankita Katiyar², Shabana Mehfuz³

¹Department of Electrical and Electronics Engineering, ITS Engineering College, Gr. Noida (U.P.), India

²School of Computer Science and Engineering, VIT University, Vellore (T.N.), India

³Department of Electrical Engineering, Jamia Millia Islamia, New Delhi, India

Email address:

gaurikatiyar1@its.edu.in (G. Katiyar), ankita.katiyar2013@vitalum.ac.in (A. Katiyar), shabana_mehfuz@yahoo.com (S. Mehfuz)

*Corresponding author

To cite this article:

Gauri Katiyar, Ankita Katiyar, Shabana Mehfuz. Off-Line Handwritten Character Recognition System Using Support Vector Machine. *American Journal of Neural Networks and Applications*. Vol. 3, No. 2, 2017, pp. 22-28. doi: 10.11648/j.ajna.20170302.12

Received: October 24, 2017; **Accepted:** November 21, 2017; **Published:** December 13, 2017

fig.4

A completely off-line handwritten character popularity machine has been developed using the suggested Support Vector Machine. Studies were conducted out using a standard database obtained from CEDAR, as well as four unique function extraction algorithms to generate the final characteristic vector. In order to get great viable category accuracy, classifier selection is critical. The results of the experiments show that SVM outperforms other techniques proposed in the literature in terms of overall performance.

CHAPTER -3

SYSTEM DEVELOPMENT

Support Vector Machine Algorithm:

The goal of the SVM calculation is to make the nice line or desired limit that could isolate n-dimensional space into classes so we will absolutely put the new element inside the proper category afterward. Hyperplane is the first-class preference restriction .

The Support Vector Machine, or SVM, is arguably the most well-known Supervised Learning method, and it's used for both classification and regression problems.SVM selects the outlandish focuses/vectors that aid in the creation of the hyperplane. These outlandish scenarios are referred to as support vectors, and the calculation is referred to as a Support Vector Machine. SVM is mostly used in Machine Learning for Classification problems.

Consider the diagram below, which shows two distinct categories that are defined by a choice limit or hyperplane:

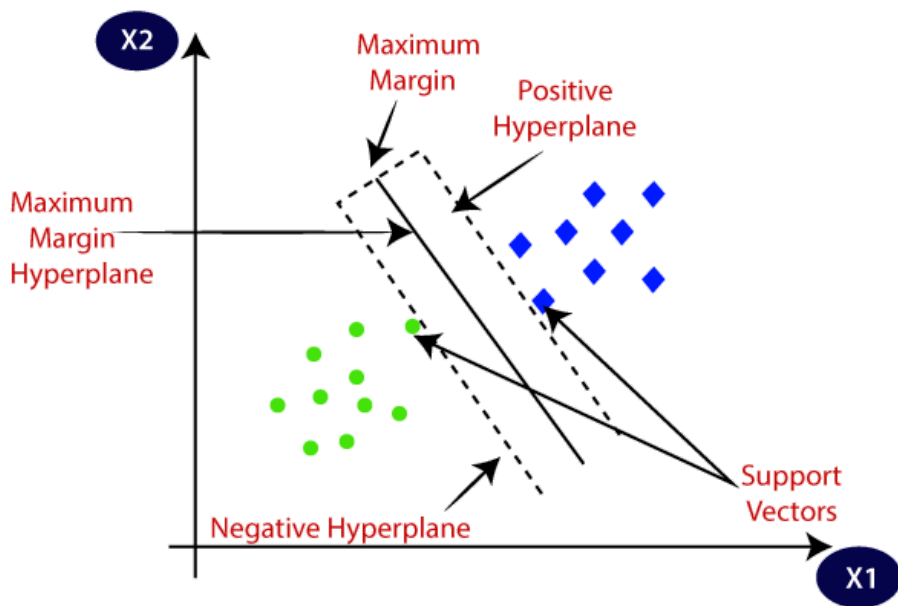


fig.5

ALGORITHM:

1. Distinguish the right hyperplane which isolates the two classes better.
2. Estimate the distance as an edge by finding the maximum extreme distance between the nearest information point and the hard hyperplane. Similarly, seek for a hyperplane on both sides with the most extreme edge. The hyperplane with a higher edge is more strong, even though the low edge has been changed for misclassification.
3. SVM selects the classifier for the enhanced edge with precision.
4. SVM is a powerful classifier with a feature that allows it to ignore anomalies and seek for the hyperplane with the largest edge.

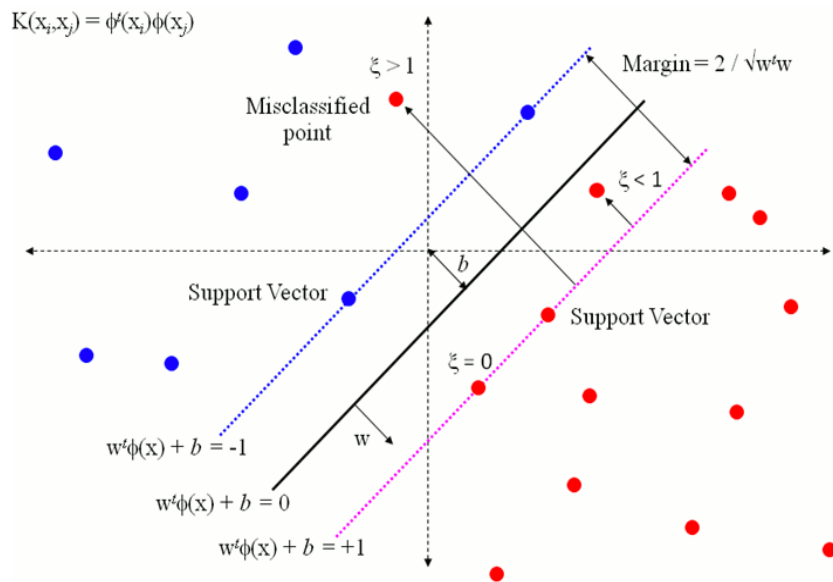


Fig.6

MATHEMATICAL

The linear algebra that can be used to generate hyper - plane learning in linear SVM is known as Kernel. To overcome these challenges, SVM contains many of the most frequent kernels. SVM features a kernel technique that allows it to achieve higher accuracy.

The radial basis function (RBF) kernel is the most preferred. The Linear kernel is the simplest of them all. The Polynomial kernel and the Sigmoid kernel are kernels for normalising data issues.

- Kernel : $F(x) = B(0) + \sum (a_i * (X, X_i))$

- The linear kernel: $K(x, y) = x \times y$
- RBF kernel: $K(x, y) = \exp(-\gamma \|x - y\|^2)$
- The Sigmoid kernel: $K(x, y) = \tanh(\beta_0 x y + \beta_1)$
- The polynomial kernel: $K(x, y) = [(x \times y) + 1]^d$

Steps Of Implementation :-

1. First we will Import the necessary Python libraries and then we will load the image dataset of our handwritten digit
2. Next step is Preprocessing and splitting of data into train & test dataset
3. Build the Support vector machine model
4. Perform the training of model
5. Evaluating the efficiency of model using performance metrics
6. Digit prediction with the help of GUI.

TECHNIQUES:

3.1 SVM

The Handwritten digit Recognizer and Summarizer have been programmed using Python language on Jupyter Notebook.

Following are the steps followed to build the model:

1. Obtain the dataset of images of handwritten characters. My data set contains digits from 0-9.

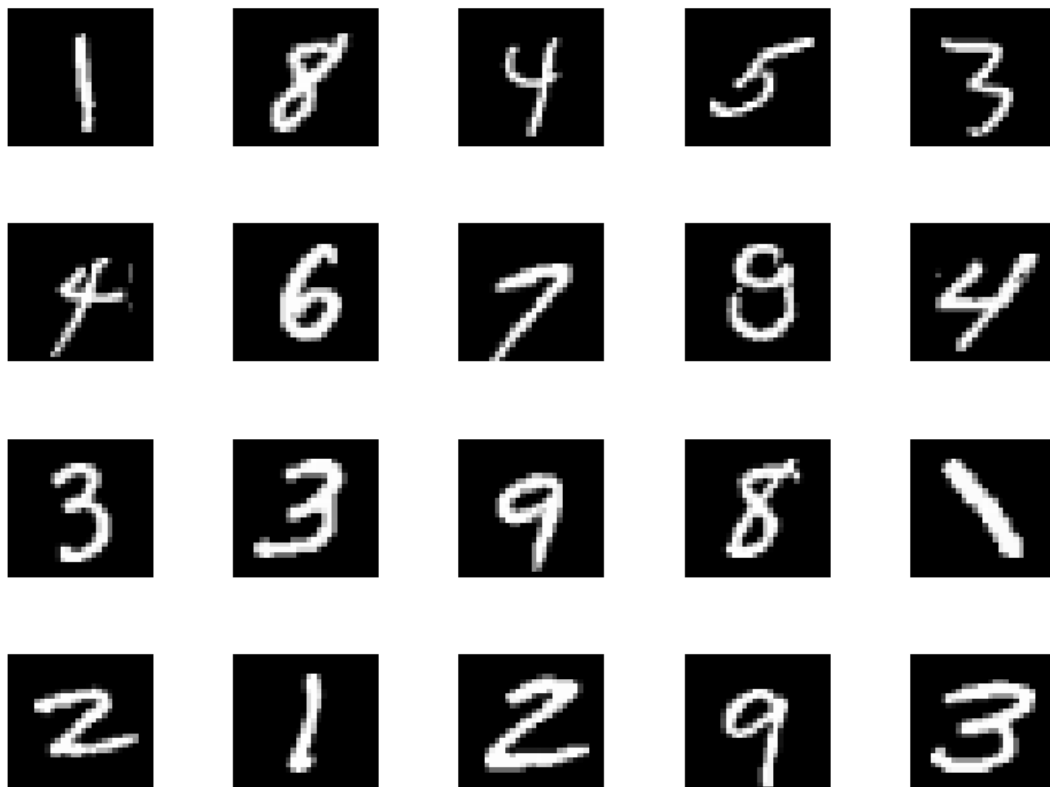
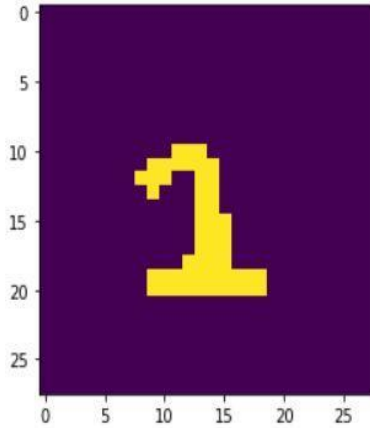


Fig 7

2. Train the model using a training set of images

Out[6]: <matplotlib.image.AxesImage at 0x1bcb1c7bcd0>



```
In [7]: from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size = 0.2)
```

```
In [8]: import joblib
from sklearn.svm import SVC
classifier=SVC(kernel="linear", random_state=6)
classifier.fit(train_x, train_y)
joblib.dump(classifier, "model/digit_recognizer")
```

Out[8]: ['model/digit_recognizer']

2. Validation and testing of the model trained using a training set.
3. Obtain digits dataset of images of handwritten digits.
4. Apply contour analysis technique so as to segment the Digit images into individual images.
5. Once digits have been segregated, they are passed to the model for digit recognition.
6. And then for any input digit drawn through paint we are predicting the result through an interface.

3.2 NEURAL NETWORK

A Neural organisation is a progression of calculations that undertakes to perceive hidden connections in a bunch of information through an interaction that emulates the manner in which the human mind works. In this sense, neural organisations allude to frameworks of neurons, either natural or fake in nature.

Neural organisations can adjust to evolving input; so the organisation produces the most ideal outcome without expecting to upgrade the result measures. The idea of neural organisations, which has its foundations in man-made brainpower, is quickly acquiring fame in the advancement of exchanging frameworks.

Manually written digit acknowledgment utilising MNIST dataset is a significant venture made with the assistance of Neural Network. It essentially distinguishes the filtered pictures of manually written digits.

We have made this a stride further where our manually written digit acknowledgment framework identifies filtered pictures of transcribed digits as well as permits composing digits on the screen with the assistance of a coordinated GUI for acknowledgment.

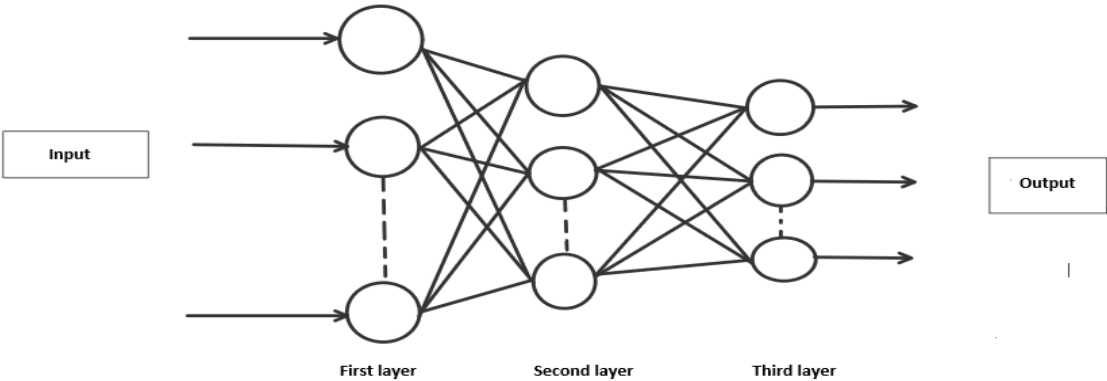


Fig 8

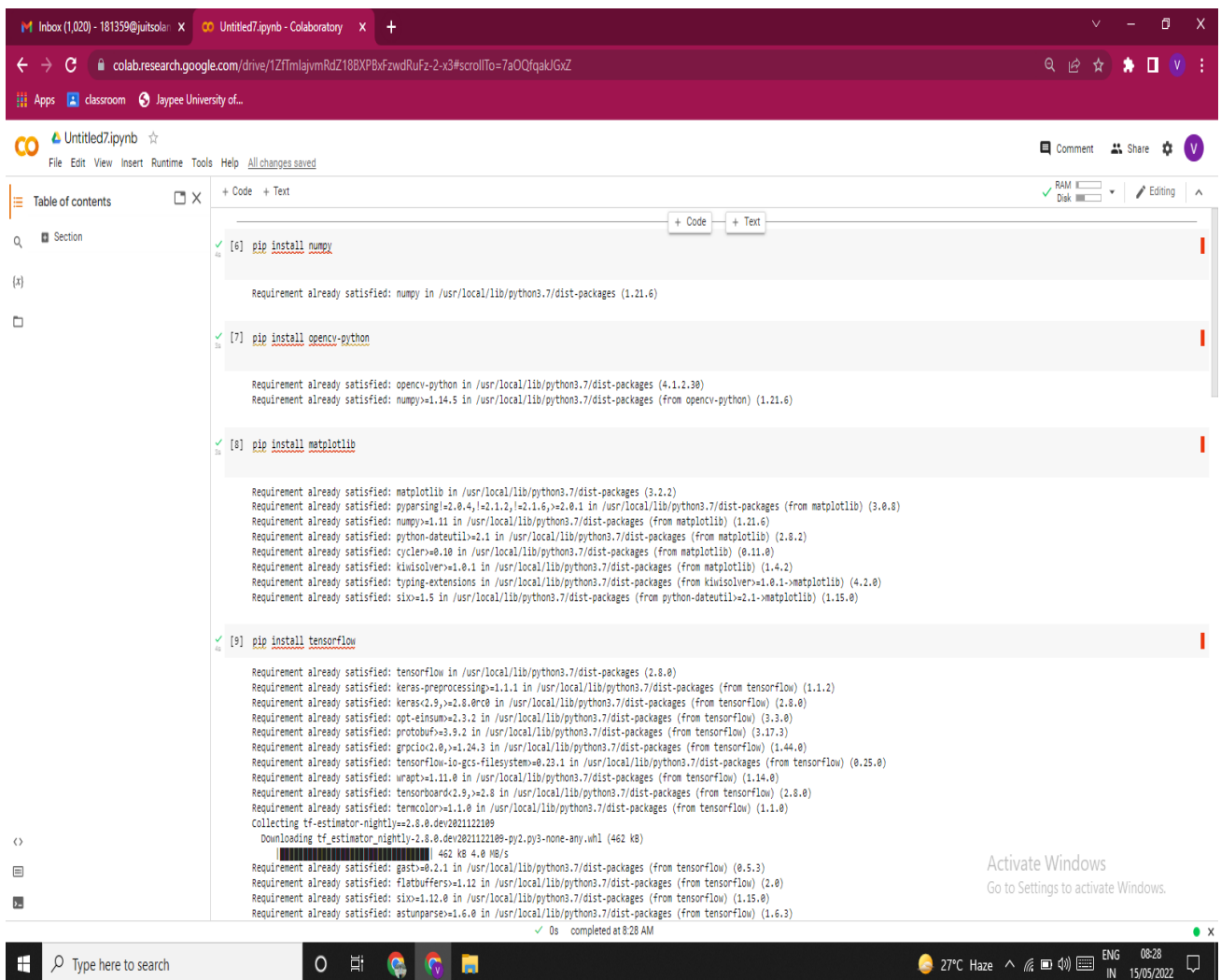
ALGORITHM:

1. First we will Import the necessary Python libraries and then we will load the image dataset of our handwritten digit
2. Next step is Preprocessing and splitting of data into train & test dataset
3. Build the Support vector machine model
4. Perform the training of model
5. Evaluating the efficiency of model using performance metrics

Steps Of Implementation :-

The Handwritten digit Recognizer and Summarizer have been programmed using Python language on Jupyter Notebook.

Following are the steps followed to build the model:
Obtain the dataset of images of handwritten characters. My data set contains digits from 0-9.



The screenshot shows a Jupyter Notebook interface with the following code cells and their outputs:

```
[6]: pip install numpy
```

Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.21.6)

```
[7]: pip install opencv-python
```

Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python) (1.21.6)

```
[8]: pip install matplotlib
```

Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (3.2.2)
Requirement already satisfied: pyparsing=2.0.4,!=2.1.2,!=2.1.6,!=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.8)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.21.6)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib) (1.15.0)

```
[9]: pip install tensorflow
```

Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.8.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.17.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.44.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.25.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.0)
Requirement already satisfied: tensorboard<2.9,>=2.8 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.0)
Collecting tf-estimator-nightly==2.8.0.dev2021112109
 Downloading tf_estimator_nightly-2.8.0.dev2021112109-py3-none-any.whl (462 kB)
 462 kB 4.0 MB/s
Requirement already satisfied: gast>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: flatbuffers>=1.12 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3)

0s completed at 8:28 AM

```
[ ] import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test,y_test)=mnist.load_data()
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
11493376/11490434 [=====] - 0s 0us/step

2. Train the model using a training set of images

```
✓ [11] import tensorflow as tf
43s x_train = tf.keras.utils.normalize(x_train,axis=1)
x_test = tf.keras.utils.normalize(x_test,axis=1)
model=tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(128,activation='relu'))
model.add(tf.keras.layers.Dense(128,activation='relu'))
model.add(tf.keras.layers.Dense(10,activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train,epochs=3)
model.save('handwritten.model')
```

```
Epoch 1/3
1875/1875 [=====] - 6s 3ms/step - loss: 0.2624 - accuracy: 0.9237
Epoch 2/3
1875/1875 [=====] - 9s 5ms/step - loss: 0.1075 - accuracy: 0.9671
Epoch 3/3
1875/1875 [=====] - 6s 3ms/step - loss: 0.0722 - accuracy: 0.9769
INFO:tensorflow:Assets written to: handwritten.model/assets
```

Validation and testing of the model trained using a training set.

3. Obtain digits dataset of images of handwritten digits.

4. Apply contour analysis technique so as to segment the Digit images into individual images.

5. Once digits have been segregated, they are passed to the model for digit recognition.

6. And then for any input digit drawn through paint we are predicting the result through an interface.

```
✓ [12] model = tf.keras.models.load_model('handwritten.model')
```

```
✓ [13] loss, accuracy = model.evaluate(x_test, y_test)  
      print(loss)  
      print(accuracy)
```

```
313/313 [=====] - 1s 2ms/step - loss: 0.0914 - accuracy: 0.9736
```

```
0.09140726178884506
```

```
0.9735999703407288
```


3.3 CONVOLUTIONAL NEURAL NETWORK:

A Convolutional Neural Network or CNN is a Deep Learning Algorithm which is extremely successful in taking care of picture characterization assignments. Catching the Temporal and Spatial conditions in a picture with the assistance of channels or kernels is capable.

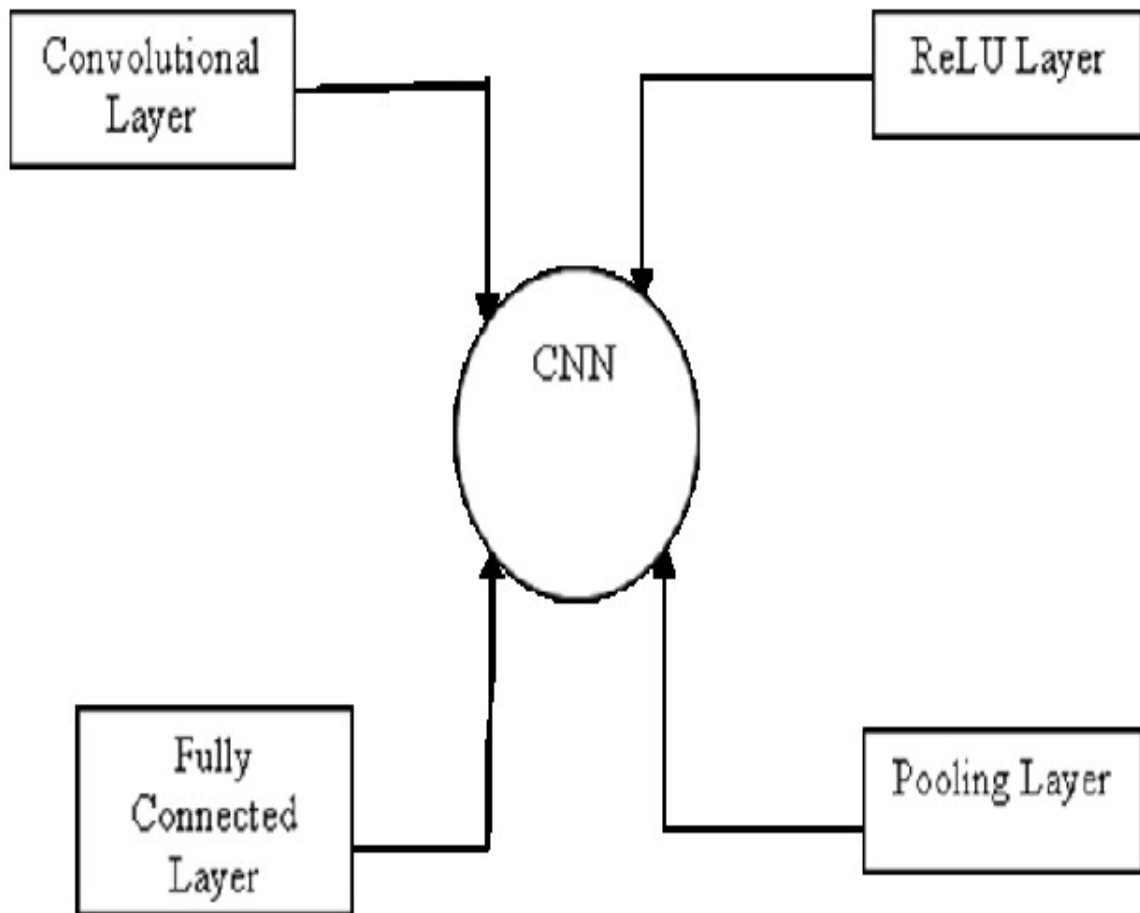


Fig. 2. Layers in CNN

ALGORITHM:

1. First we will Import the necessary Python libraries and then we will load the image dataset of our handwritten digit
2. Next step is Preprocessing and splitting of data into train & test dataset
3. Build the Support vector machine model
4. Perform the training of model
5. Evaluating the efficiency of model using performance metrics

Steps Of Implementation :-

1. Obtain the dataset of images of handwritten characters. My data set contains digits from 0-9.

```
▶ from keras.layers import *  
from keras.models import Sequential  
import keras  
  
[ ] #building Model  
  
model=Sequential()  
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))  
model.add(MaxPool2D(2,2))  
  
model.add(Conv2D(64,(3,3),activation='relu'))  
model.add(MaxPool2D(2,2))  
  
model.add(Conv2D(64,(3,3),activation='relu'))  
model.add(MaxPool2D(2,2))  
  
model.add(Flatten())  
model.add(Dense(64,activation='relu'))  
model.add(Dense(10,activation='softmax'))
```

2. Model building

```
▶ model.summary()
```

```
⊙ Model: "sequential_1"
```

| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d_3 (Conv2D) | (None, 26, 26, 32) | 320 |
| max_pooling2d_3 (MaxPooling2D) | (None, 13, 13, 32) | 0 |
| conv2d_4 (Conv2D) | (None, 11, 11, 64) | 18496 |
| max_pooling2d_4 (MaxPooling2D) | (None, 5, 5, 64) | 0 |
| conv2d_5 (Conv2D) | (None, 3, 3, 64) | 36928 |
| max_pooling2d_5 (MaxPooling2D) | (None, 1, 1, 64) | 0 |
| flatten (Flatten) | (None, 64) | 0 |
| dense (Dense) | (None, 64) | 4160 |
| dense_1 (Dense) | (None, 10) | 650 |
| ===== | | |
| Total params: 60,554 | | |
| Trainable params: 60,554 | | |
| Non-trainable params: 0 | | |

3.Next step is Preprocessing and splitting of data into train & test dataset

```
[ ] from keras.utils.np_utils import to_categorical
    from keras.datasets import mnist
    (xtrain,ytrain),(xtest,ytest)=mnist.load_data()
```

```
[ ] def process_data(x,y):
    x=x.reshape((-1,28,28,1))
    x=x/255.0
    y=to_categorical(y)
    return x,y
```

```
[ ] xtrain,ytrain=process_data(xtrain,ytrain)
    xtest,ytest=process_data(xtest,ytest)

    print(xtrain.shape,ytrain.shape)
    print(xtest.shape,ytest.shape)
```

```
(60000, 28, 28, 1) (60000, 10)
(10000, 28, 28, 1) (10000, 10)
```

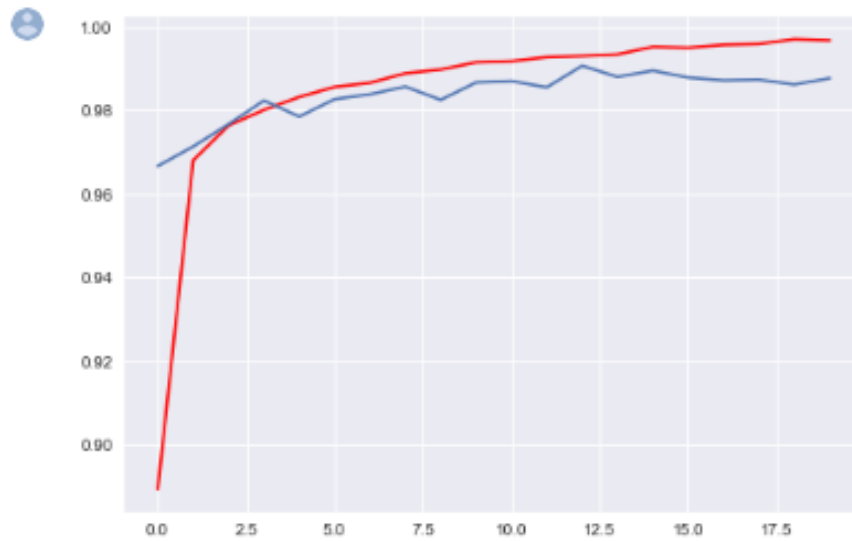
```
▶ model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
  hist=model.fit(xtrain,ytrain,epochs=20,batch_size=128,validation_split=0.1)
```

```
[ ] model.evaluate(xtest,ytest)
```

```
313/313 [=====] - 2s 5ms/step - loss: 0.0659 - accuracy: 0.9868
[0.065882109105587, 0.9868000745773315]
```

4. Plotting the results in histogram.

```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
accuracy=hist.history['accuracy']
plt.plot(accuracy,color='red')
plt.plot(hist.history['val_accuracy'])
plt.show()
```



CHAPTER -4

PERFORMANCE ANALYSIS

4.1 SVM:

```
In [5]: X = data.drop(["label"],axis=1)
        Y= data["label"]
```

```
In [6]: %matplotlib inline
import matplotlib.pyplot as plt
import cv2
idx = 118
img = X.loc[idx].values.reshape(28,28)
print(Y[idx])
plt.imshow(img)
```

1

```
Out[6]: <matplotlib.image.AxesImage at 0x1bcb1c7bcd0>
```

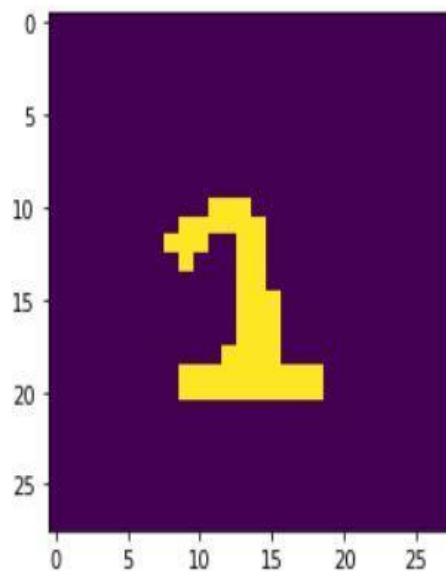


Fig 10

The Accuracy Of our Model is 83.33%

```
In [7]: from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size = 0.2)
```

```
In [8]: import joblib
from sklearn.svm import SVC
classifier=SVC(kernel="linear", random_state=6)
classifier.fit(train_x, train_y)
joblib.dump(classifier, "model/digit_recognizer")
```

```
Out[8]: ['model/digit_recognizer']
```

```
In [9]: from sklearn import metrics
prediction=classifier.predict(test_x)
print("Accuracy= ", metrics.accuracy_score(prediction, test_y))
```

```
Accuracy= 0.8333333333333334
```

Fig 11

4.2 NEURAL NETWORK:

The Accuracy Of our Model is 97.35%

```
✓ [12] model = tf.keras.models.load_model('handwritten.model')
```

```
✓ [13] loss, accuracy = model.evaluate(x_test, y_test)  
      print(loss)  
      print(accuracy)
```

```
313/313 [=====] - 1s 2ms/step - loss: 0.0914 - accuracy: 0.9736  
0.09140726178884506  
0.9735999703407288
```

4.3 CONVOLUTIONAL NEURAL NETWORK:

The Accuracy Of our Model is 98.68%

```
[ ] model.evaluate(xtest,ytest)
```

```
313/313 [=====] - 2s 5ms/step - loss: 0.0659 - accuracy: 0.9868  
[0.065882109105587, 0.9868000745773315]
```

The following are the snippets through which we can analyse our result ,through the digit we have provided in input.

Input digit is 9 and the predicted result is 9.

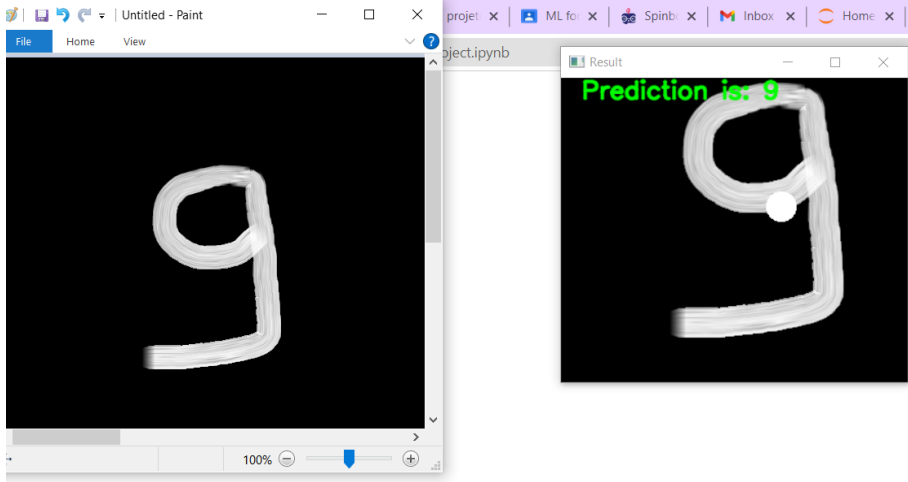


Fig 12

The input digit is 6 and the predicted result is 6.

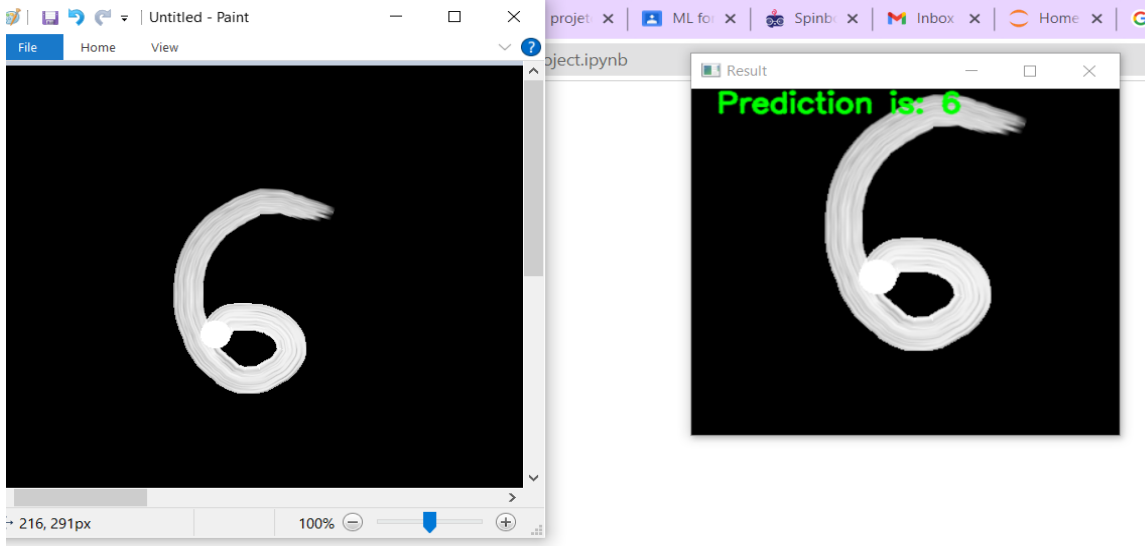


Fig 13

The input digit is 5 and the predicted result is 5.

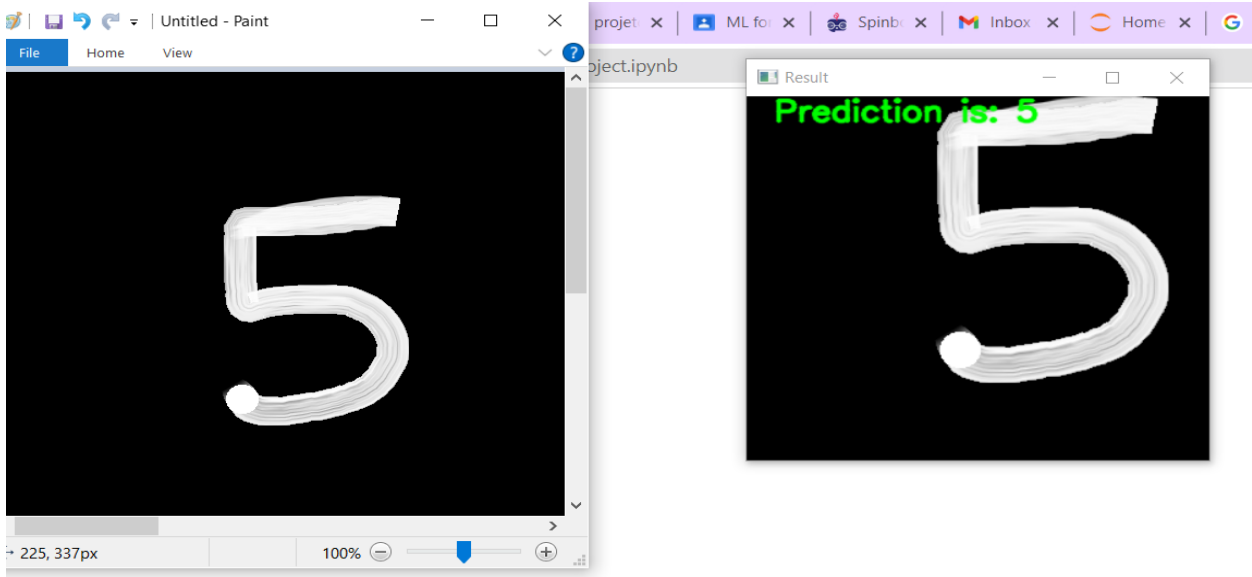


Fig 14

The input digit is 0 and the predicted result is 0.

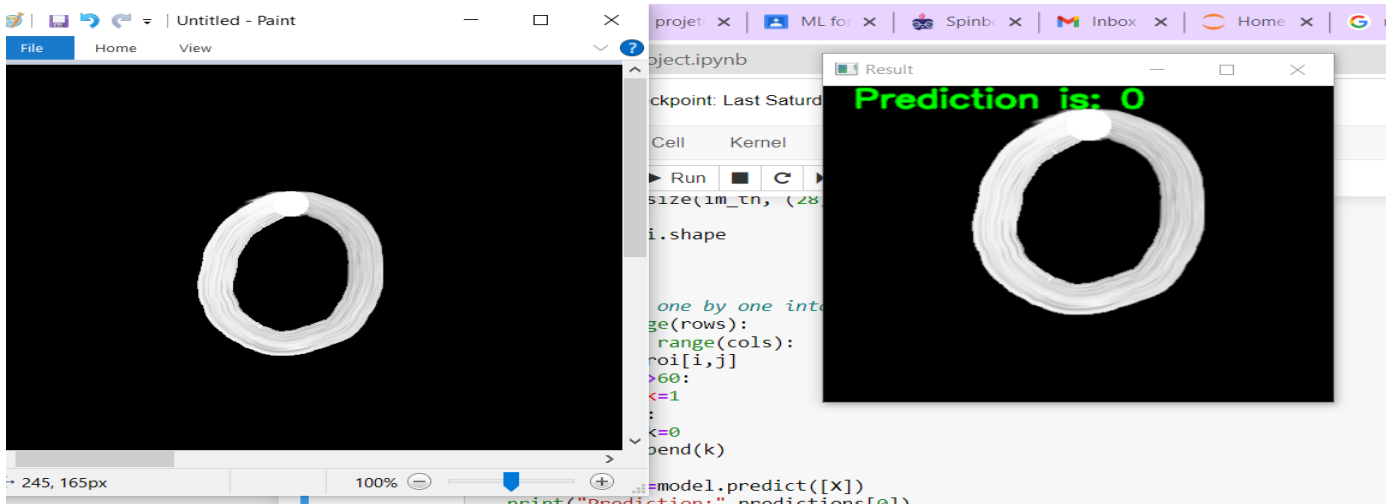


Fig 15

CHAPTER -5

CONCLUSION

5.1 Conclusion and Future work

We developed a model using SVM and designed a GUI through which we can predict handwritten digits. And in the project we were able to train model for 600 image dataset and predict the values of digit we have provided as input using SVM. As the amount of information increments grows, there is always the possibility of improving accuracy. Accuracy can fluctuate depending on how training data and testing data information is distributed, and it can also rise when more training and testing data is supplied.

Further to improve the performance of the model we can use a hybrid classifier of SVM and NN(Nearest Neighbour) technique to solve the problem.

REFERENCES

1. https://www.researchgate.net/publication/221710787_Handwritten_digit_Recognition_using_Support_Vector_Machine
2. Support vector machines in handwritten digits classification
 - a. Publisher: IEEE U. Markowska-Kaczmar; P. Kubacki
3. SVM based off-line handwritten digit recognition
4. Publisher: IEEE Gauri Katiyar; Shabana Mehfuz
5. <https://ieeexplore.ieee.org/document/8237400> Gu, J., Wang, G., Cai, J.,&Chen, T. (2017). An empirical study of language cnn for image captioning. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1222-1231).
6. <https://techvidvan.com/tutorials/handwritten-digit-recognition-with-python-cnn>

APPENDIX

| | pgno |
|--------------------------------|------|
| 1. Linear kernel | 9 |
| 2. RBF kernel | 9 |
| 3. Sigmoid Kernel | 9 |
| 4. Kernel | 9 |
| 5. Svm classifying two classes | 8 |

