VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi -590018, Karnataka

A Mini Project Report On

"DYNAMIC HANDWRITTEN DIGIT RECOGNITION SYSTEM"

Submitted in the partial fulfillment for award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

ABHISHEK K N	**4MN21AD002**
DARSHAN N	**4MN21AD011**
NITHISH GOWDA B R	**4MN21AD027**
VINAY B K	**4MN21AD050**

Under The Guidance Of

Prof . MADHAN KUMAR G S

Assistant professor , Dept. Of AI & DS

MIT Thandavapura

Maharaja Institute Of Technology Thandavapura

Department of Artificial Intelligence And Data Science

NH 766, Nanjangud taluk, Mysore district-571302

2023-2024

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA
NH 766, Nanjangud Taluk, Mysuru- 571 302
(Approved by AICTE & Affiliated to Visveswaraya Technological University, Belagavi)
(Certified by ISO 91001:2015 & ISO 21001:2018)
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

Certified that the mini project work entitled **"DYNAMIC HANDWRITTEN DIGIT RECOGNITION SYSTEM"** is a bonafede work carried out by **ABHISHEK K N (4MN21AD002) , DARSHAN N (4MN21AD011) , NITHISH GOWDA B R (4MN21AD027)** AND **VINAY B K (4MN21AD050)** for the **MINI PROJECT** with course code **21ADMP67** of Sixth Semester in **Artificial Intelligence And Data Science** under **Visvesvaraya Technological University, Belagavi** during academic year 2023-24. It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

Signature Of The Guide	Signature Of The HOD
Prof. Madhan Kumar G S	**Dr. Swarnalatha K**
Assistant Professor	Associate Professor & HOD
Dept. Of AI & Data Science	Dept. Of AI & Data Science

ACKNOWLEDGEMENT

It Is The Time To Acknowledgement All Those Who Have Extended Their Guidance, Inspiration And Their Whole Hearted Co-Operation All Along Our Project Work.

We Would Like To Extend Our Thanks To **Dr. Y T Krishne Gowda,** Principal, Maharaja Institute Of Technology Thandavapura, For His Co-Operation Throughout The Academics. That Has Helped Us In Satisfactory Completion Of Project.

We Express Our Sincere Thanks To **Dr. H K Chethan,** Joint Secretory Of MET Mysuru (**.** For His Continuous Support And Appreciation During This Program.

We Are Grateful To **Dr. Swarnalatha K,** Associate Professor And HOD, Department Of Artificial Intelligence And Data Science, For Providing Us A Valuable Support Throughout The Period Of Project.

We Express Our Deepest Sense Of Gratitude To Our Guide **Prof. Madhan Kumar G S ,** Assistant Professor, Department Of Artificial Intelligence And Data Science, For His Valuable Guidance, Suggestions And Cheerful Encouragement During The Entire Period Of Project.

We Also Take An Opportunity To Thank All The Teaching And Non-Teaching Staff Members Of AI&DS Department.

Finally, We Thank Almighty God And We Are Grateful To Our Parents For Their Faith, Love And Kindness In Believing Us. We Are Indebted To MITT **,** Who Has Directly Or Indirectly Supported Us During The Period Of Mini Project.

ABHISHEK K N	**4MN21AD002**
DARSHAN N	**4MN21AD011**
NITHISH GOWDA B R	**4MN21AD027**
VINAY B K	**4MN21AD050**

ABSTRACT

In the realm of interactive machine learning applications, this project introduces a dynamic handwritten digit recognition system that combines a robust Convolutional Neural Network (CNN) with a user-friendly painting interface. The system leverages the MNIST dataset to train a CNN model capable of accurately identifying handwritten digits. The trained model is then utilized within an interactive application, where users can draw digits directly onto a canvas using their mouse. The system integrates real-time digit classification with OpenCV's graphical capabilities, allowing users to see immediate predictions of their drawn digits. This is achieved by continuously capturing and processing user inputs, resizing and normalizing the drawn digits, and making predictions using the trained CNN model. The application displays these predictions on the canvas, providing instant feedback to users. Additionally, the interface includes functionality to clear the canvas, enhancing user interaction. This project demonstrates the potential of combining machine learning with interactive tools to create intuitive and engaging applications. By bridging the gap between user input and real-time predictions, it showcases the practical application of CNNs in dynamic environments and offers a foundation for further development in digital handwriting recognition.

CONTENT

Chapter 1

INTRODUCTION

This chapter introduces handwriting recognition technology and its importance, and outlines the problem of accurately recognizing handwritten digits due to variability in handwriting styles.

1.1 Overview

Handwriting recognition (HWR) is a technology that enables machines to interpret and understand handwritten characters and digits. This process is critical for converting physical handwritten documents into digital formats that can be edited, searched, and analysed. Optical Character Recognition (OCR) is a well-established technology in this domain, converting images of text into machine-readable text data. OCR enables digital transformation by allowing scanned documents to be edited and searched like any other text file.

Text summarization is another key technology that simplifies the extraction of meaningful information from large volumes of text. By summarizing text, users can quickly grasp the core ideas and important points without having to read the entire document. Automatic summarization tools are increasingly vital in an information-rich world, helping users to efficiently absorb relevant content and focus on essential information.

1.2 Problem Statement

The core problem addressed by this project is the accurate recognition of handwritten digits. Handwritten digits, ranging from 0 to 9, exhibit significant variability due to differences in individual handwriting styles. This variability can pose challenges for automated systems trying to recognize and classify these digits. The objective is to develop a robust system capable of accurately identifying handwritten digits from images, despite the inherent variations in handwriting.

To achieve this, the project will focus on capturing handwritten digits through user interactions with a paint application, processing these images, and employing a machine learning model to classify the digits. The system must be capable of distinguishing between different digits even when written in diverse styles.

1.3 Objective

The primary objective of this project is to develop a system that can reliably recognize handwritten digits from 0 to 9. To accomplish this, the project will employ a Support Vector Machine (SVM) for classification. The SVM model will be trained to classify digits based on their pixel values, effectively solving a 10-class classification problem. The project will involve:

1. **Dataset Collection:** Gathering a diverse set of handwritten digit images.
2. **Model Training:** Using the collected dataset to train the SVM model.
3. **Digit Recognition:** Implementing a system that can predict the digit drawn by users in real-time.

The dataset will consist of 70,000 images of handwritten digits. Users will draw digits using a paint application, capture these images with the screenshot package, and store them in labelled folders.

1.4 System Development

1.4.1 Collecting Dataset: To build a robust digit recognition system, a diverse dataset of handwritten digits is required. The dataset will be collected by allowing users to draw digits in a paint application. The screenshot package will be used to capture these drawings and store them in a structured format, with images organized into folders labelled from 0 to 9. This process ensures that the dataset includes a variety of handwriting styles and digit representations.

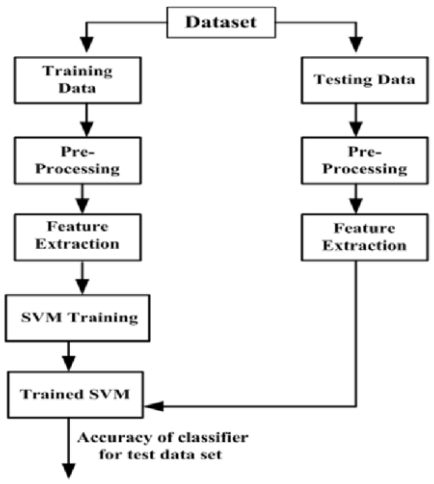


Fig.1 Flow chart

1.4.2 Model Training and Testing: The dataset will be divided into training and testing subsets. The training data will consist of pixel values (features) and their corresponding digit labels (targets). The model will learn to associate pixel patterns with specific digits during training. After training, the model will be tested on unseen data to evaluate its accuracy and generalization capability. The accuracy will be determined by comparing the model's predictions with the true labels of the test data.

1.4.3 Digit Prediction: The system will enable users to draw digits on a paint application. These drawings will be captured, processed, and passed to the trained model for prediction. The model's output will be displayed to the user in real-time, providing immediate feedback on the recognized digit.

Chapter-2

REQUIREMENT ANALYSIS

This chapter Discusses the limitations of current digit recognition systems and outlines the proposed system's improvements and requirements for software and hardware.

2.1 Existing System

Current methods for recognizing handwritten digits often use static models that aren't very adaptable. This results in several limitations, including limited flexibility in handling new or unusual handwriting styles and a lack of interactivity, which can be frustrating for users. Additionally, these systems often suffer from slow processing speeds, negatively impacting the user experience, and accuracy issues due to variations in handwriting. Setting up and adapting these systems can also be complex and cumbersome.

2.1.1 Disadvantages

The disadvantages of the existing systems are significant. Pre-trained models are rigid and don't adapt well to different handwriting styles. Slow response times make these systems unsuitable for real-time use, while outdated user interfaces may not support dynamic inputs effectively. Handwriting variations can lead to recognition errors, and the deployment process can be challenging, making it hard to set up and integrate these systems into different environments.

2.2 Proposed System

The new Dynamic Handwritten Digit Recognition System aims to solve these problems by offering several enhancements. It provides real-time recognition, allowing users to draw digits and get immediate feedback. The system's adaptable models learn and adjust to different handwriting styles, while its interactive design offers an easy-to-use interface for drawing and recognizing digits. This results in improved accuracy, better handling of various handwriting styles, and a simpler setup and deployment process.

Additional benefits of the proposed system include its ability to learn over time, becoming better at recognizing user handwriting with continued use. It is accessible on both mobile devices and web browsers, customizable to meet user needs, and includes options for users to provide feedback to improve the system.

2.2.1 Advantages

The proposed system offers numerous advantages, such as instant recognition, better accuracy in handling different handwriting styles, and a user-friendly interface that is simple and intuitive to use. It improves with use, learning and adapting based on feedback, and is versatile enough to work on various devices and platforms.

2.3 Software Requirements

- **Operating System**: Windows 10 or newer, Ubuntu 20.04 or newer
- **Languages**: Python
- **Tools**: TensorFlow or PyTorch, OpenCV, Flask or Django (for web apps)
- **Database**: SQLite or MySQL (if needed)
- **Development Tools**: Git, Docker (optional)

2.4 Hardware Requirements

- **Processor**: Intel i5 or equivalent, AMD Ryzen 5 or better
- **RAM**: 8 GB minimum, 16 GB recommended
- **Storage**: SSD with at least 256 GB
- **Graphics Card**: NVIDIA GTX 1050 or better (helps with faster processing)
- **Input Devices**: Digital drawing tablet or touchscreen (optional)
- **Output Devices**: Monitor screen, optional webcam for live input

Chapter 3

SYSTEM DEVELOPMENT

This chapter Explains the algorithms used, including SVM, neural networks, and CNN, and details the steps for model training, testing, and implementation.

3.1 Support Vector Machine Algorithm

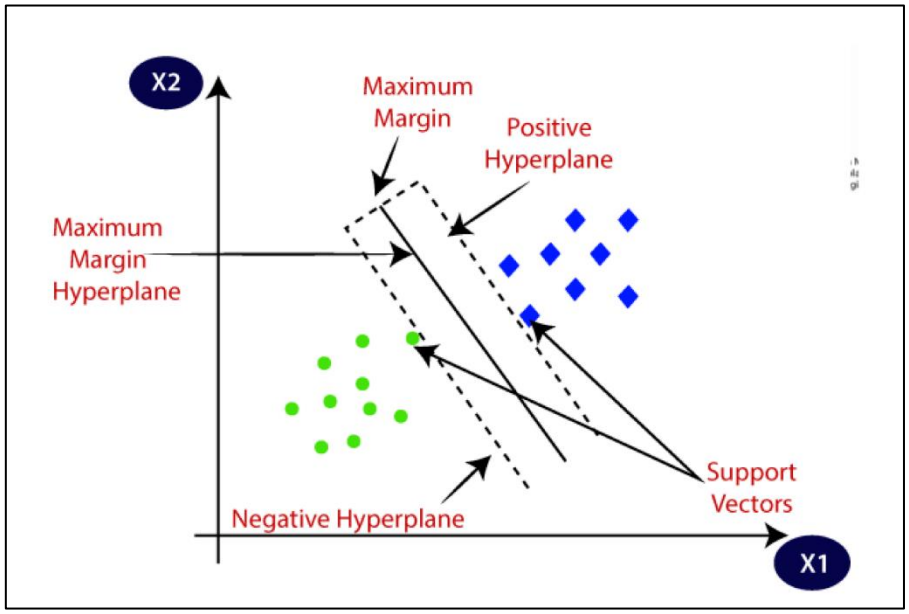


Fig 2. SVM classifier

The Support Vector Machine (SVM) algorithm is a powerful supervised learning method used for classification and regression tasks. The goal of SVM is to find the optimal hyperplane that separates different classes in the feature space. This hyperplane maximizes the margin between classes, with the data points closest to the hyperplane, known as support vectors, playing a crucial role in defining the decision boundary.

Techniques:

1. **Dataset Preparation:**
    - Obtain and preprocess the dataset of handwritten digit images.
    - Convert images into feature vectors suitable for SVM.
2. **Model Training:**
    - Train the SVM model using the prepared dataset.

- Optimize the model's parameters to enhance classification accuracy.

3. **Model Evaluation:**
   - Validate and test the model using separate subsets of the dataset.
   - Assess the model's performance using metrics such as accuracy, precision, recall, and F1-score.
4. **Digit Prediction:**
   - Implement the digit recognition system by integrating the trained SVM model.
   - Develop a user interface for drawing and predicting digits.

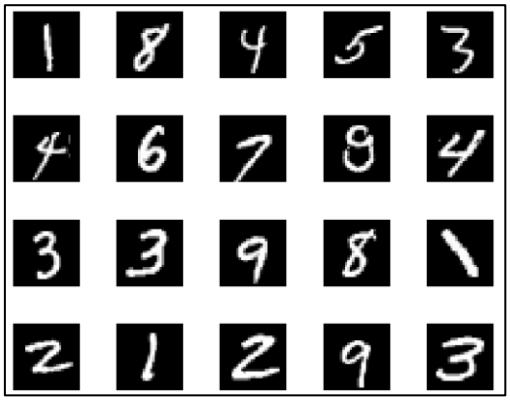


Fig 3. Train the model using a training set of images

3.2 Neural Networks

Neural networks are computational models inspired by the human brain's neural structure. They consist of interconnected nodes (neurons) organized in layers, including input, hidden, and output layers. Neural networks are highly effective at recognizing complex patterns and relationships in data.

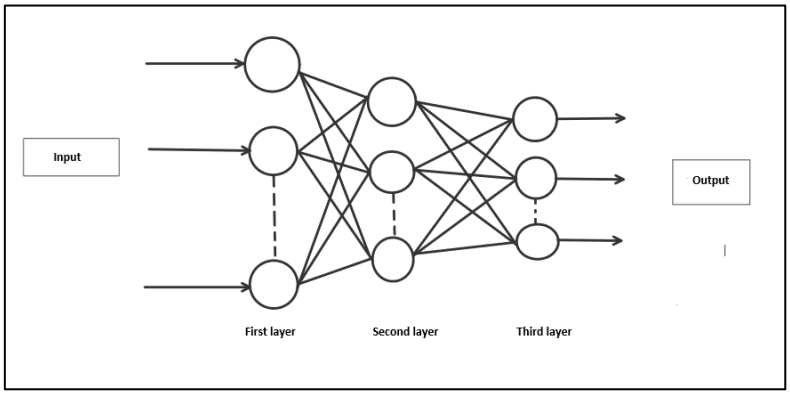
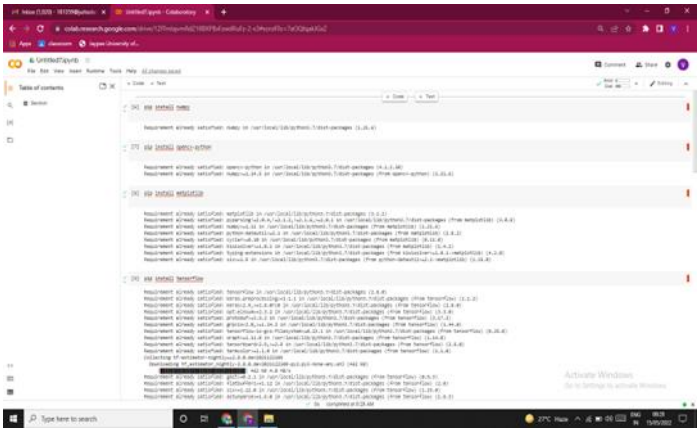


Fig 4. Neural Networks

Techniques:

1. **Dataset Loading:**
    - Import and preprocess the dataset of handwritten digit images.



2. **Model Building:**
    - Construct a neural network architecture suitable for digit recognition. This can involve defining the number of layers, neurons, activation functions, and other parameters.

```
[ ] import os
    import cv2
    import numpy as no
    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
```

3. **Model Training:**
    - Train the neural network using the dataset, adjusting weights and biases to minimize prediction errors.

```
[11] import tensorflow as tf
     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
```

4. **Model Evaluation:**
    - Evaluate the neural network's performance using metrics such as accuracy and loss.

```
[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
```

5. **User Interface Integration:**
    - Develop an interface to allow users to draw digits and obtain predictions from the trained neural network model.

3.3 Convolutional Neural Network (CNN)

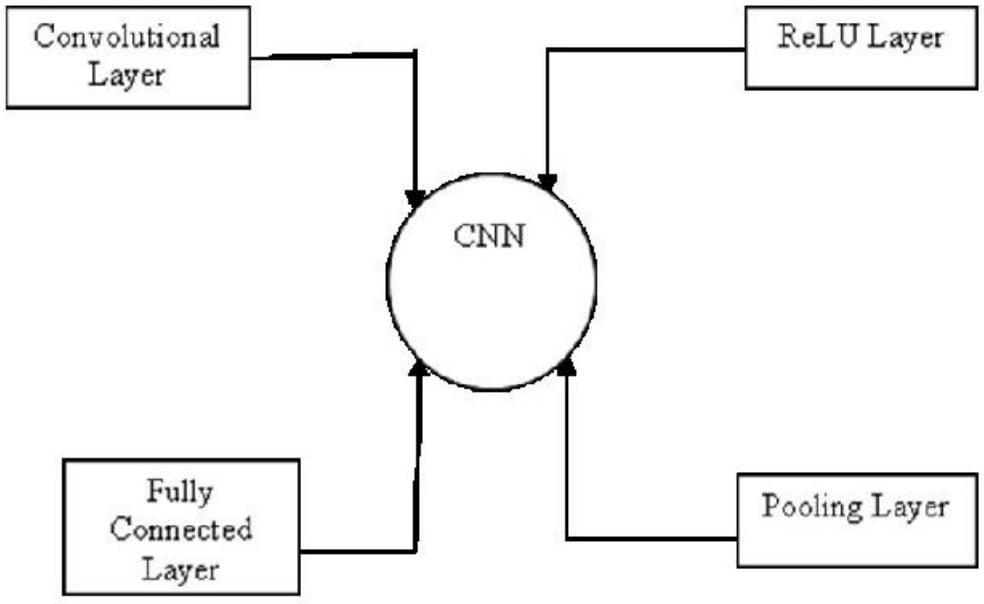


Fig 5. CNN

Convolutional Neural Networks (CNNs) are specialized neural networks designed for image processing tasks. They use convolutional layers to capture spatial hierarchies and patterns in images, making them highly effective for image classification.

Techniques:

1. **Dataset Preparation:**
    - Load and preprocess the handwritten digit images for CNN training.

```
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:**
    - Design a CNN architecture with convolutional, pooling, and fully connected layers. Fine-tune the network's parameters for optimal performance.

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (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. **Model Training:**
    - Train the CNN on the dataset, leveraging techniques such as data augmentation and dropout to prevent overfitting.

```
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)
```

4. **Model Evaluation:**

   - Assess the CNN's performance using accuracy and loss metrics. Visualize results to understand the model's classification capabilities.

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

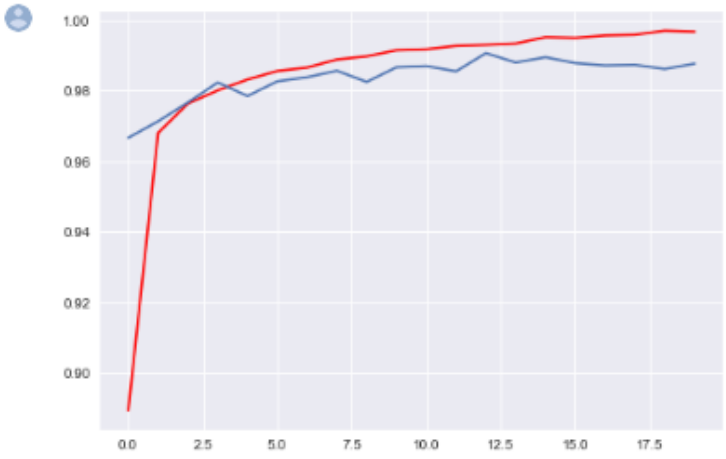
```
[ ] model.evaluate(xtest,ytest)

313/313 [==============================] - 2s 5ms/step - loss: 0.0659 - accuracy: 0.9868
[0.06588210910558 7, 0.9868000745773315]
```

5. **Implementation:**

   - Integrate the CNN model into the digit recognition system, allowing users to draw digits and receive real-time predictions.

```
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

This chapter Explains the algorithms used, including SVM, neural networks, and CNN, and details the steps for model training, testing, and implementation.

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>
```

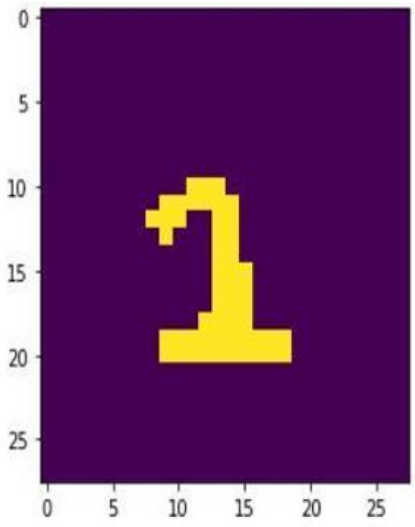


Fig 10

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
```

4.2 NEURAL NETWORK:

```
[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%

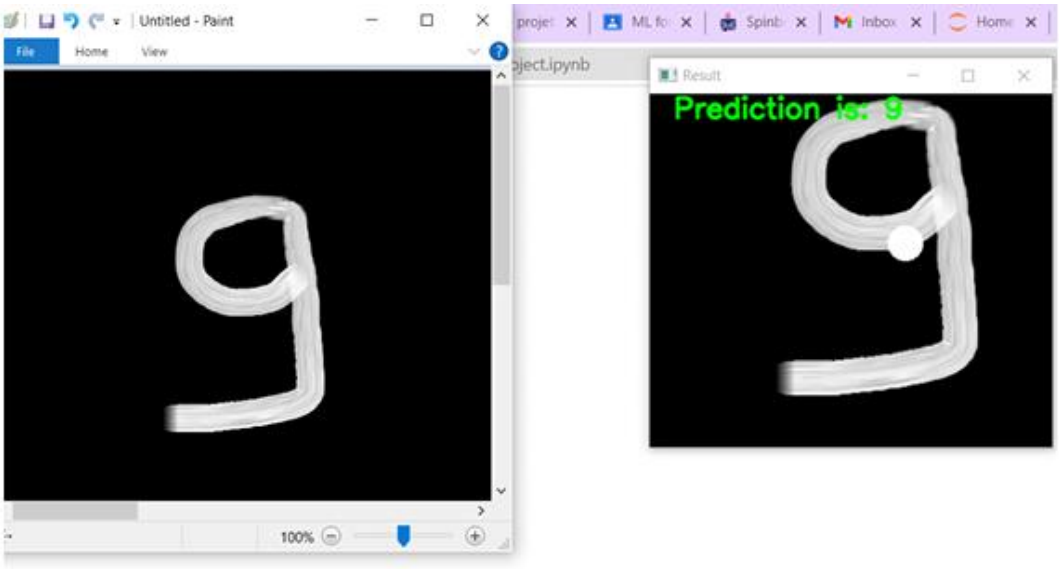
```
[ ] model.evaluate(xtest,ytest)

313/313 [==============================] - 2s 5ms/step - loss: 0.0659 - accuracy: 0.9868
[0.06588210105587, 0.9868000745773315]
```

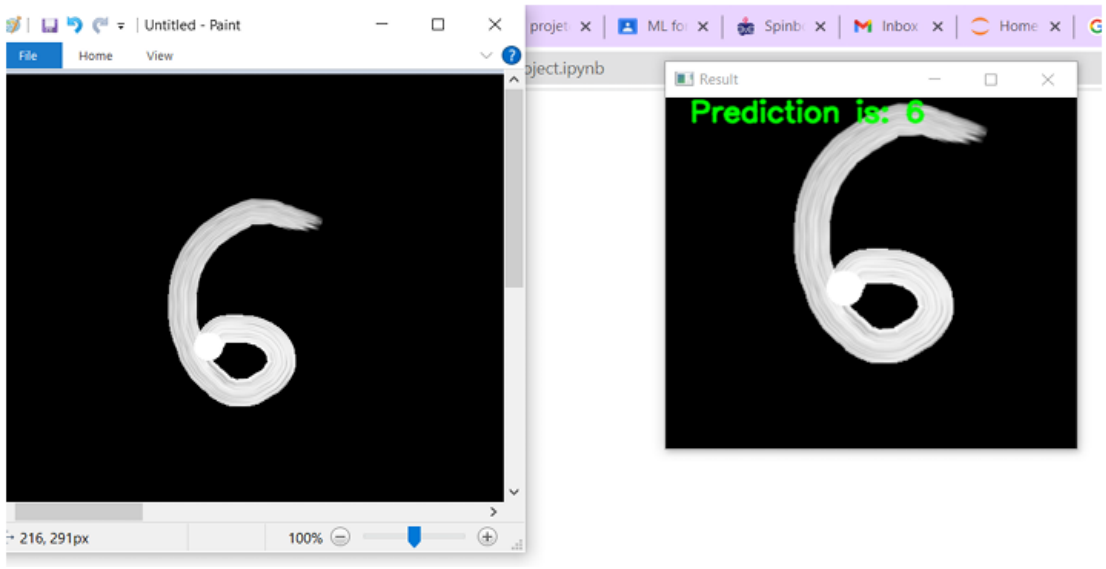
Chapter -5

RESULTS

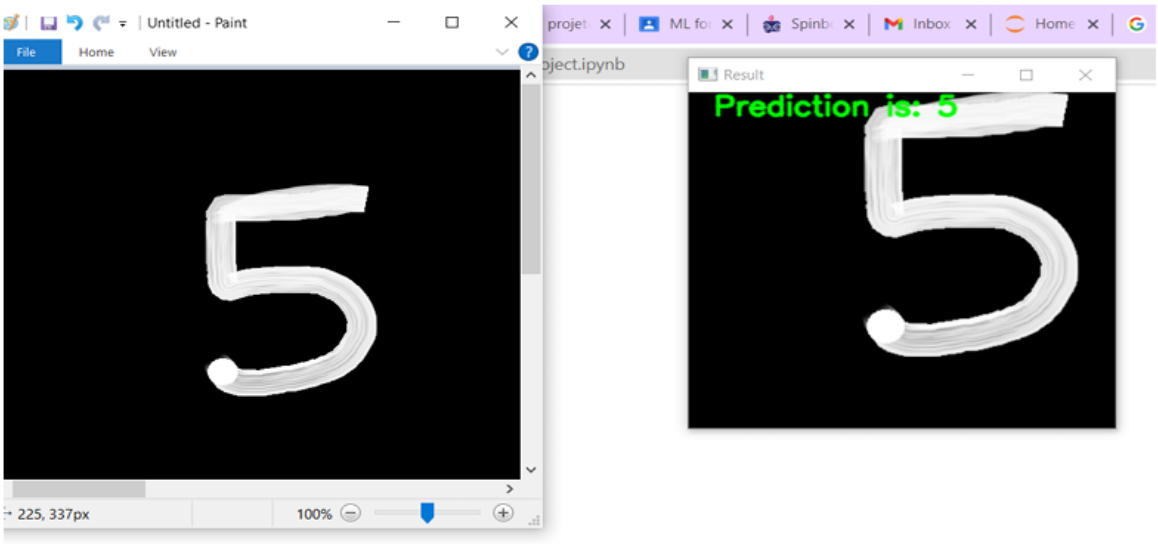
The following are the snippets through which we can analyze our result ,through the digit we have provided in input. Input digit is 9 and the predicted result is 9.



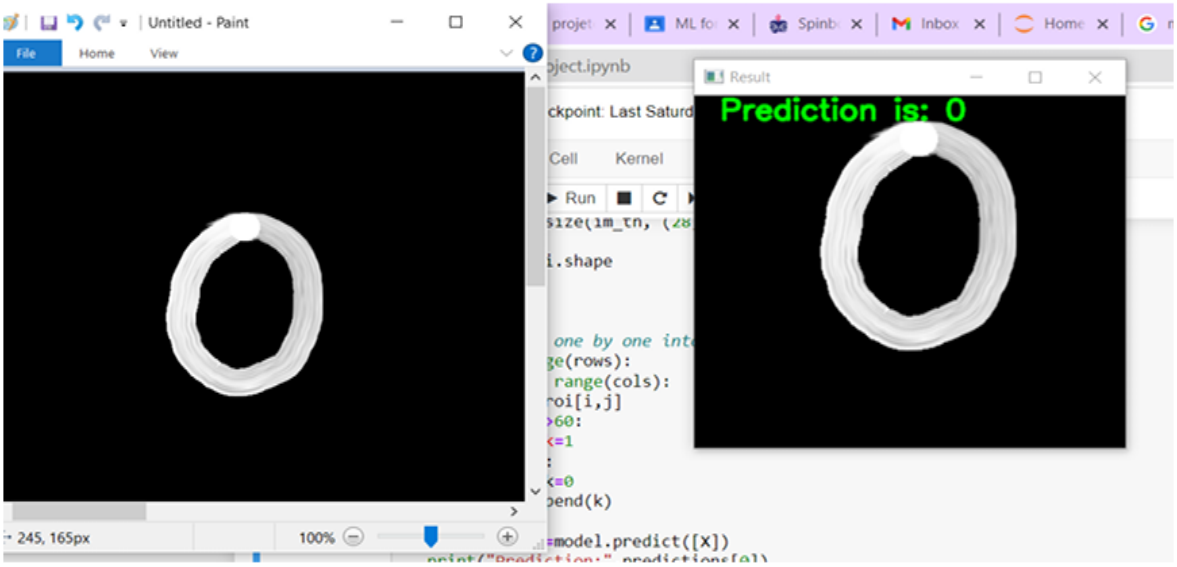
The input digit is 6 and the predicted result is 6.



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



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



CONCLUSION

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 Publisher: IEEE U. Markowska-Kaczmar; P. Kubacki
3. SVM based off-line handwritten digit recognition
4. Publisher: IEEE Gauri Katiyar; Shabana Mehfuz
5. 5.https://ieeexplore.ieee.org/document/8237400 Gu, J., Wang, G., Cai, J.,&Chen, T.
6. (2017). An empirical study of language cnn for image captioning. In Proceedings of
7. the IEEE International Conference on Computer Vision (pp. 1222-1231).
6. https://techvidvan.com/tutorials/handwritten-digit-recognition-with-python-cnn