INTELLIGENT HANDWRITTEN DIGIT IDENTIFICATION SYSTEM FOR COMPUTER APPLICATIONS

Using IBM Watson Studio

Developed By: G. Rajeshwari, B. Sandeep, P. Sathwika, Mirza Imran Baig

Smart Bridge – Mini Project Report

1.INTRODUCTION

1.1 Overview

The handwritten digit identification is the capability of computer applications to recognize the human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit identification is the solution to this problem which uses the image of a digit and recognizes the digit present in the image**.** For this recognition process we have used MNIST dataset which has 70000 handwritten digits. This project contains the deep idea of image processing techniques and we have used Artificial Neural Network (ANN), Convolution Neural Network (CNN) to train the system with these images and built a deep learning model. We have created a web application where the user can upload an image of a handwritten digit, this image is analyzed by the model and the detected result is returned on to UI (User Interface).

1.2 Purpose

It is difficult for the humans to identify the handwritten digits because every individual has a unique way of writing. To rectify this issue we have build a deep learning model using ANN and CNN to train the machine with the handwritten digit images. The main aim of this project is to use the neural network approach for recognizing handwritten digits. Its applications are such as programmed bank checks, health, post offices, for education, etc.

2. LITERATURE SURVEY

2.1 Existing problem and Existing approaches or methods

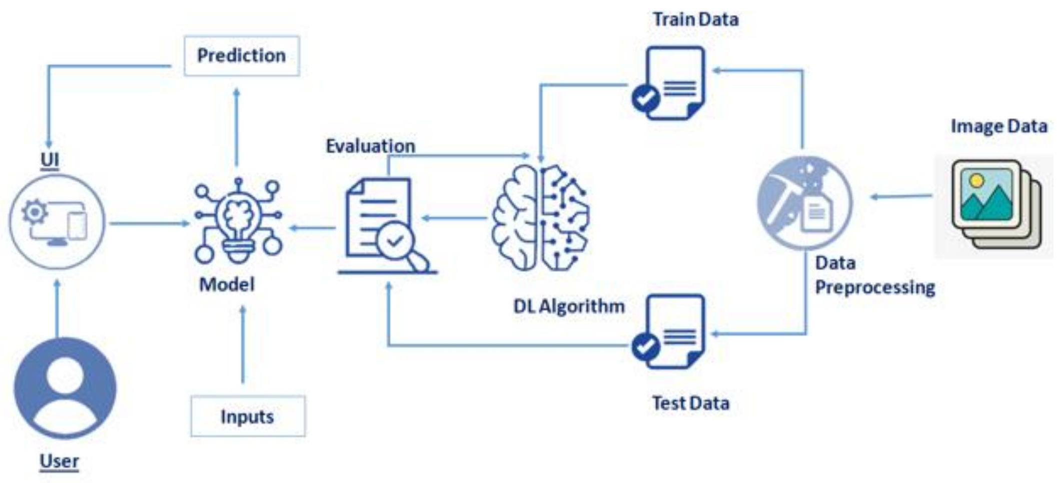
Nowadays we see that technology is increasing repeatedly and many options are available to perform Handwritten digit recognition. The algorithms to recognize the handwritten digits includes Deep Learning/CNN, SVM, DNN, Gaussian Naïve Bayes, KNN, Decision Trees, Random Forests etc. Many of the researchers prefer CNN because it gives high accuracy in image classification, video analysis etc. The CNN (Convolutional Neural Network) has brought a revolutionary change in the field of machine learning, particularly in character recognition. DNN gives accuracy of around 98.85% but fails against CNN approach in terms of time.

2.2 Proposed solution

In this project we have shown use of deep learning with others such as CNN using TensorFlow, Keras, OpenCV. These algorithms are used widely by researchers as experiments for theories of machine learning. For developing the deep learning model, we will be using Convolutional Neural Network. The dataset for this project is imported from the keras module. This dataset contains ten classes: Digits from 0 to 9. Each digit is taken as a class. The dataset contains no noise or major problems to be processed, so we were able to use it without any preprocessing operations. It has 70,000 images, all of which are grouped into one file. We split the data into training data and testing data. The test set contains an example of the pictures that must be compared to the images in the reference set to identify the handwritten digits. Using the training dataset, we train the model and the testing dataset is used to predict the results. We have divided the dataset as 60000 images to training data and 10000 images to testing data. To list out the dimensions of the data, we are finding out the shape of X_train and x_test. These images are not directly visible. To display these images, we have to use matplotlib. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware and software requirements

For running a machine learning model on the system you need a system with minimum of 16 GB RAM in it and you require a good processor for high performance of the model. In the list of software requirements you must have:

● Jupyter Notebook for programming, which can be installed by Anaconda IDE.

● Python packages.

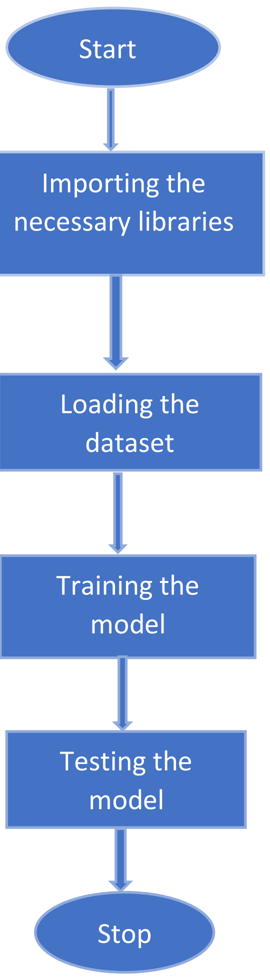
● A better software for running the html and css files for application building phase e.g. spyder.

4. EXPERIMENTAL INVESTIGATIONS

For developing the project the team has completed several tasks:

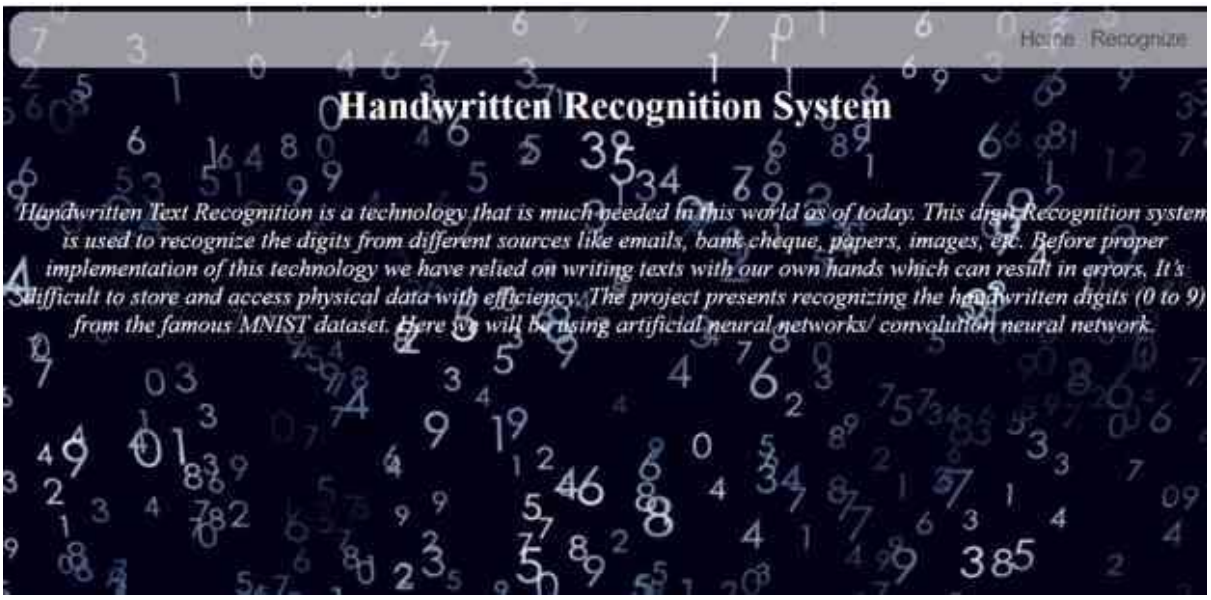
- **Understanding the data.**
    - ➢ Importing the required libraries.
    - ➢ Loading the data.
    - ➢ Analyzing the data.
    - ➢ Reshaping the data.
    - ➢ Applying One Hot Encoding.
- **Model Building**
    - ➢ Creating the model and adding the input, hidden and output layers to it.
    - ➢ Compiling the model.
    - ➢ Training the model.
    - ➢ Predicting the result.
    - ➢ Testing the model by taking image inputs.
    - ➢ Saving the model.
- **Application Building**
    - ➢ Create an HTML file.
    - ➢ Build Python code.

5. FLOW CHART


Start
Importing the necessary libraries
Loading the dataset
Training the model
Testing the model
Stop

6. RESULTS

This is the main page which describes about the project and summarizes it.

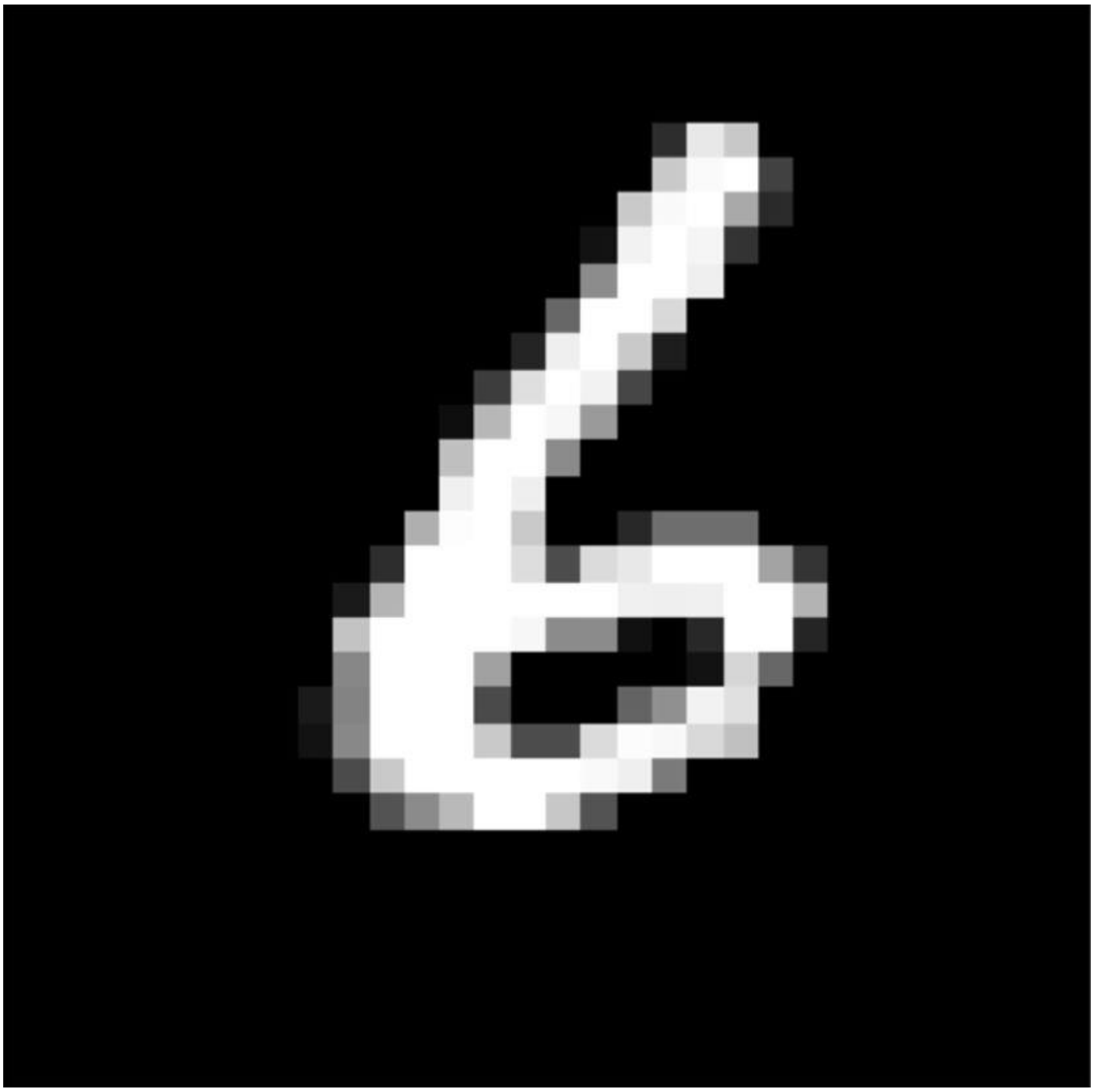


Handwritten Recognition System

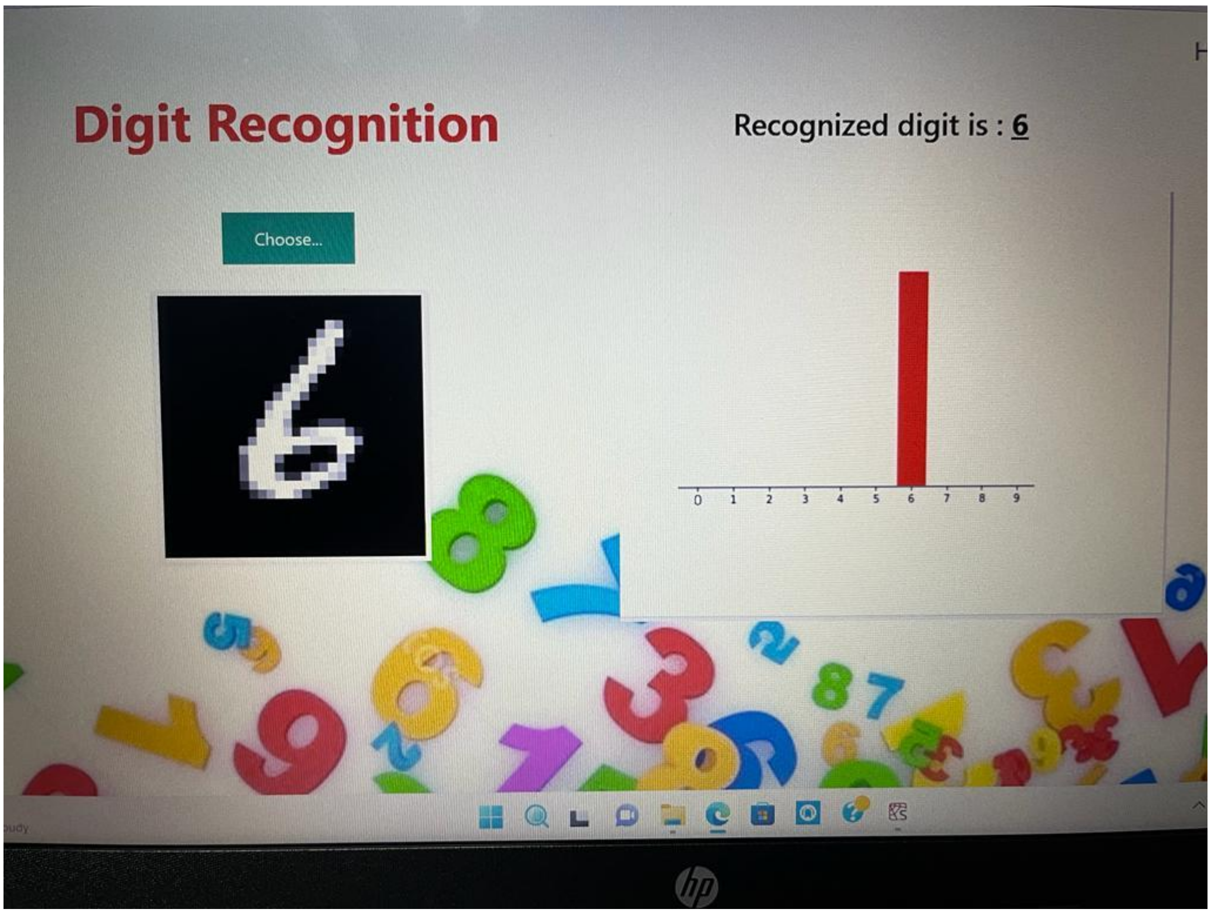
Handwritten Text Recognition is a technology that is much needed in this world as of today. This digit Recognition system is used to recognize the digits from different sources like emails, bank cheque, papers, images, etc. Before proper implementation of this technology we have relied on writing texts with our own hands which can result in errors. It's difficult to store and access physical data with efficiency. The project presents recognizing the handwritten digits (0 to 9) from the famous MNIST dataset. Here we will be using artificial neural networks/ convolution neural network.

This is the prediction page where we get to choose the image from our local system and predict output.

This is the input image, which should be submitted to the system by the user.



This is the output. The system has successfully recognised the input digit.



7. ADVANTAGES AND DISADVANTAGES

Advantages:

- The main aim of the proposed system is to make the path towards digitalization by providing the high accuracy and faster computational for recognizing the handwritten digits.
- In addition, the proposed system reduces computational time significantly for training and testing due to which algorithm becomes efficient.
- The main advantage of this project it will resolve many issues in banking, education, health etc.
- The results of this project are highly accurate because of the many no of convolution layers and hidden neurons.

Disadvantages:

- At present, the aim of this project is just to recognize the digits, but it can be extended to letters and then a person's handwriting.
- The input of the proposed system should be a handwritten digit, but it can be extended to a digital input.

8. APPLICATIONS

- The proposed system will work efficiently in Education field for recognizing the handwritten digits of the students.
- It will be the main application in Banking field, to recognize the digits in check processing.
- Another application of this project is Postal mail sorting.
- Form data entry is also an application of this project.

9. CONCLUSION

In this project, the Handwritten Digit Recognition using Deep learning methods has been implemented. The most widely used Machine learning algorithms CNN has been trained and tested on the MNIST dataset. With extensive testing using the MNIST data, the current function suggests the role of various hyper parameters. We also confirmed that a good adjustment of hyper parameters is important in improving the performance of Convolutional Neural Network. Utilizing this deep learning technique, a high amount of accuracy can be obtained. This model is able to achieve a recognition rate of 98.85% accuracy and is significantly identifying real world images as well. The effect of increasing the number of convolutional layers on CNN structure in the performance of handwritten digital recognition is clearly demonstrated by experiments. Later it can be expanded to identify the character and a real-time person's handwriting. Digital recognition is the first step in the larger field of Artificial Intelligence and Computer Vision. The results can be made more accurate with multiple layers of convolution and an additional number of hidden neurons.

10. FUTURE SCOPE

Identifying the handwritten digits is such a difficult issue for the system because every individual has a unique way of writting. The main goal of this project is to identify the handwritten digits efficiently. To obtain the efficiency, we have used Artificial Neural Network and Convolution Neural Networks. The main application of this project is to train a system to recognize the handwritten digits accurately. This project can be used in various applications such as Banking, Education, Health, Post office etc. At present, the aim of this project is just to recognize the digits, but it can be extended to letters and then a person’s handwriting. The input of the proposed system should be a handwritten digit, but it can be extended to a digital input.

10. BIBILOGRAPHY

https://www.academia.edu/62212425/Handwritten_Digit_Reco

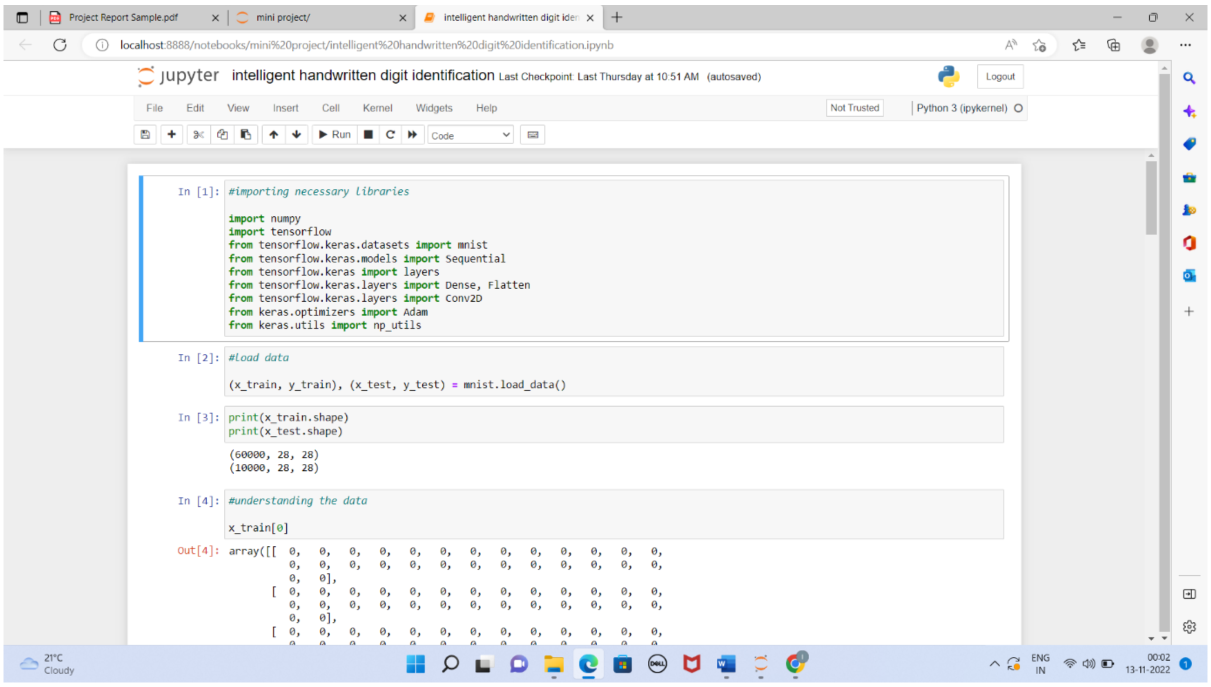
https://ieeexplore.ieee.org/document/7478642/references#references

https://projectworlds.in/tag/application-of-handwritten-digit-recognition/

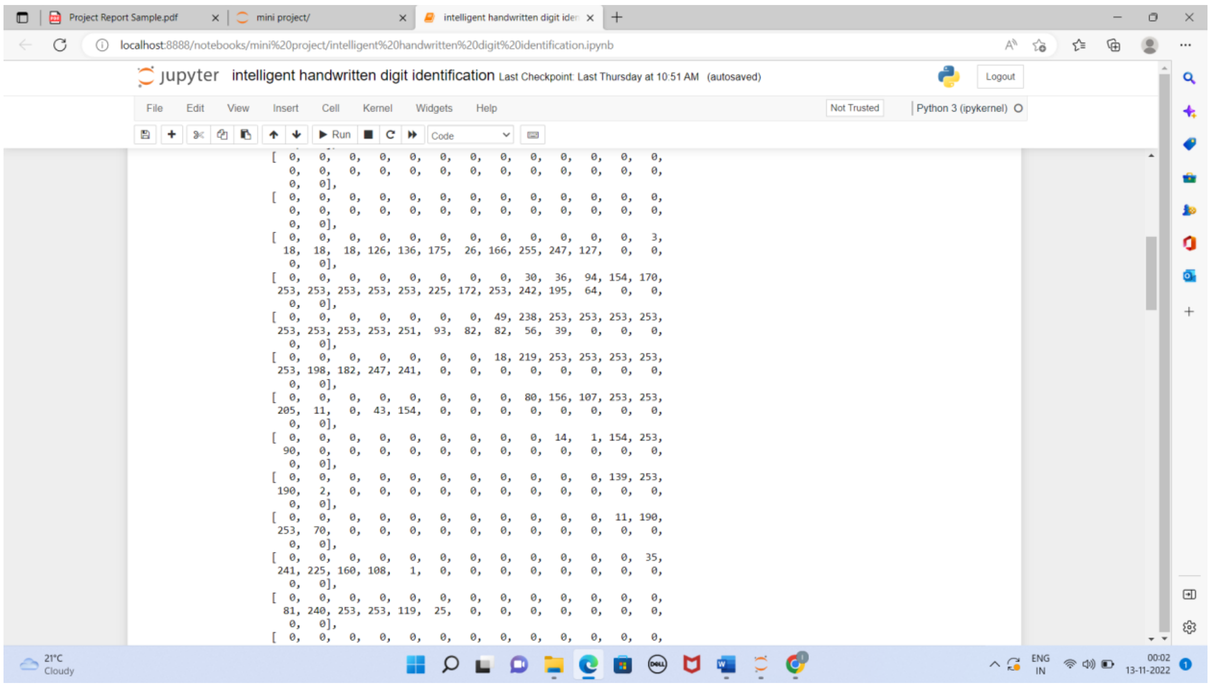
https://www.sciencedirect.com/topics/computer-science/handwriting-recognition

11. APPENDIX

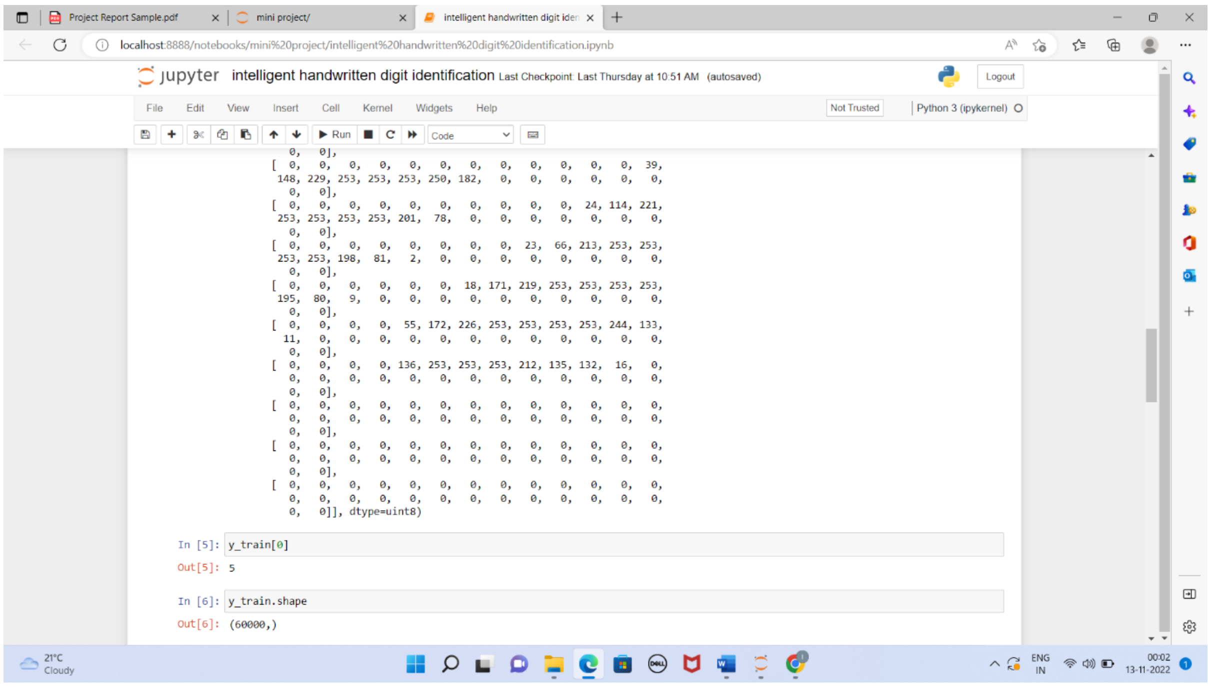
Understanding the data:



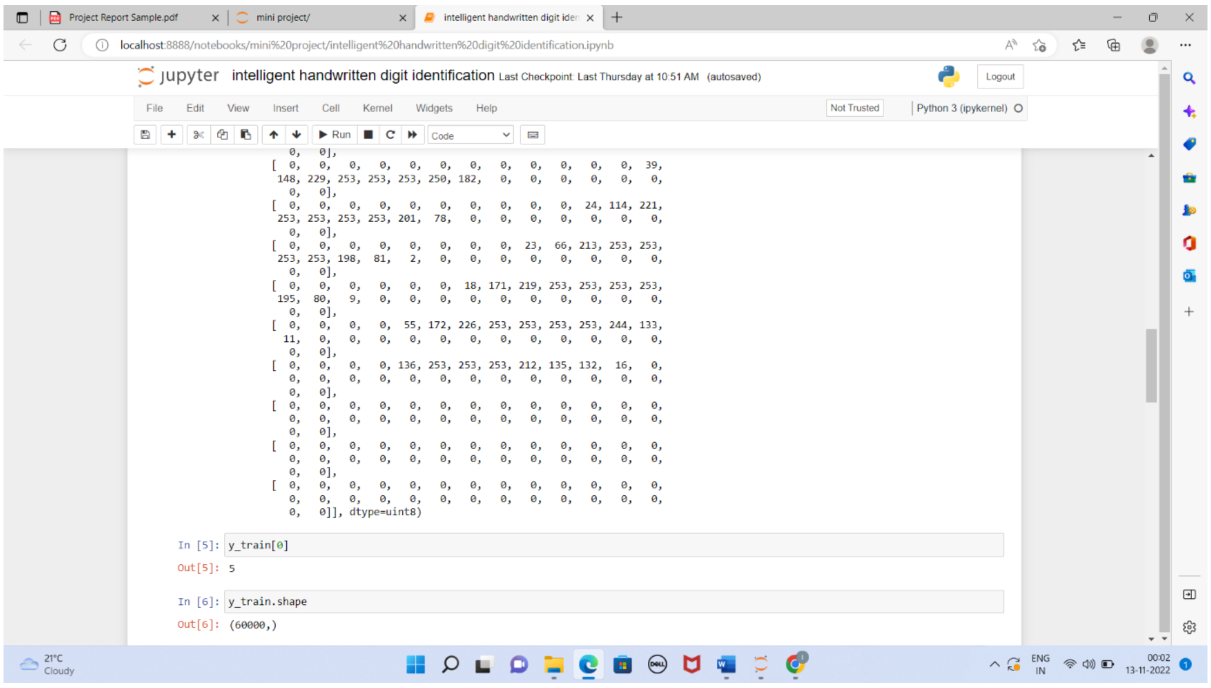
```
In [1]: #importing necessary libraries

        import numpy
        import tensorflow
        from tensorflow.keras.datasets import mnist
        from tensorflow.keras.models import Sequential
        from tensorflow.keras import layers
        from tensorflow.keras.layers import Dense, Flatten
        from tensorflow.keras.layers import Conv2D
        from keras.optimizers import Adam
        from keras.utils import np_utils

In [2]: #load data

        (x_train, y_train), (x_test, y_test) = mnist.load_data()

In [3]: print(x_train.shape)
        print(x_test.shape)

        (60000, 28, 28)
        (10000, 28, 28)

In [4]: #understanding the data

        x_train[0]

Out[4]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```



```
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
                 18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
                253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
                253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
                205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
                 90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
                190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
                253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
                241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                 81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

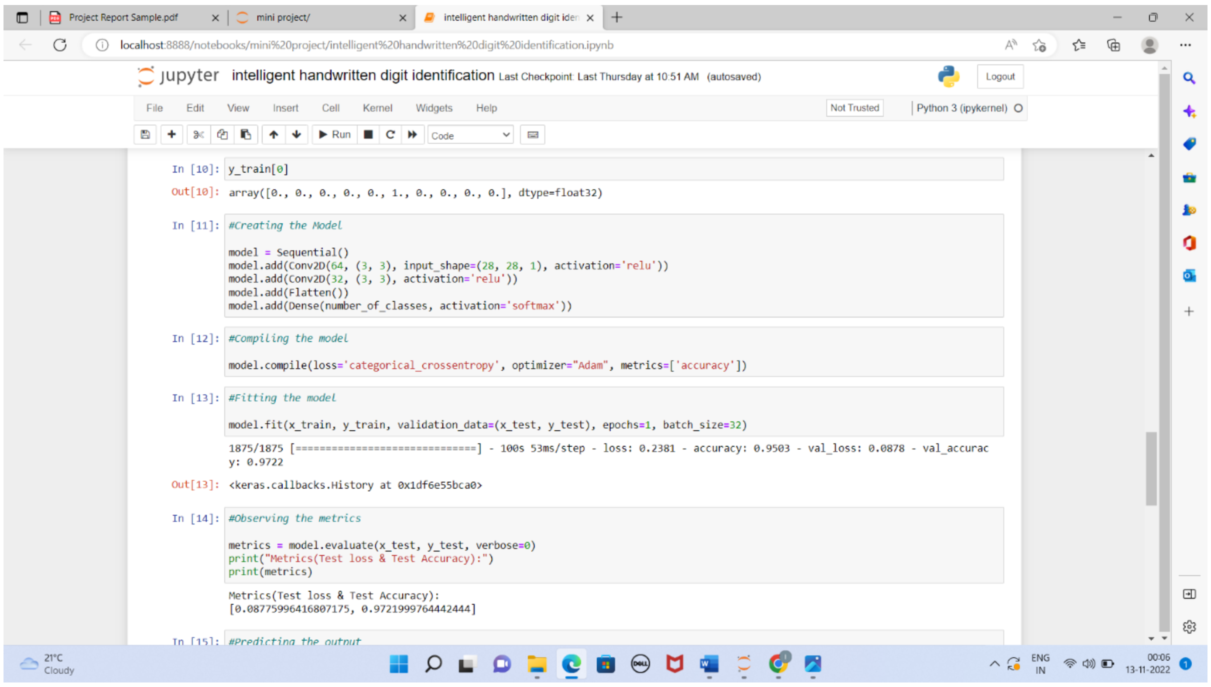


```
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)

In [5]: y_train[0]

Out[5]: 5

In [6]: y_train.shape

Out[6]: (60000,)
```

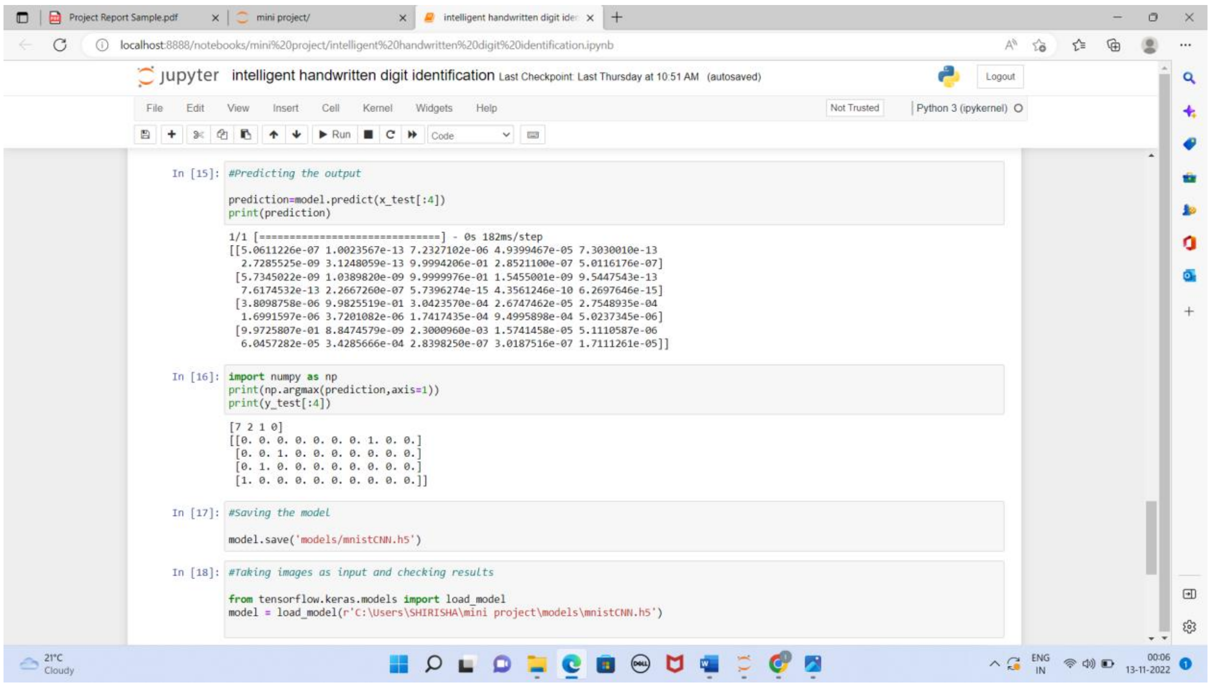


```
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)

In [5]: y_train[0]

Out[5]: 5

In [6]: y_train.shape

Out[6]: (60000,)
```

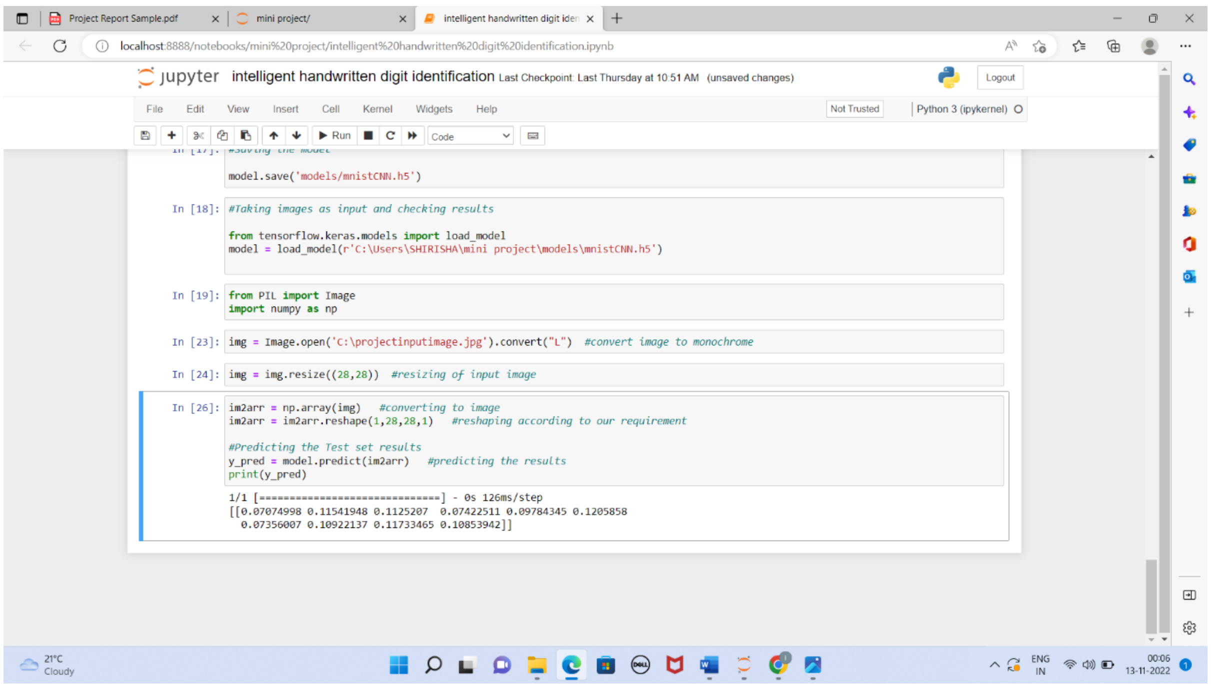
Model Building:



```
In [10]: y_train[0]

Out[10]: array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

In [11]: #Creating the Model

         model = Sequential()
         model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
         model.add(Conv2D(32, (3, 3), activation='relu'))
         model.add(Flatten())
         model.add(Dense(number_of_classes, activation='softmax'))

In [12]: #Compiling the model

         model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])

In [13]: #Fitting the model

         model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=1, batch_size=32)

         1875/1875 [==============================] - 100s 53ms/step - loss: 0.2381 - accuracy: 0.9503 - val_loss: 0.0878 - val_accurac
         y: 0.9722

Out[13]: <keras.callbacks.History at 0x1df6e55bca0>

In [14]: #Observing the metrics

         metrics = model.evaluate(x_test, y_test, verbose=0)
         print("Metrics(Test loss & Test Accuracy):")
         print(metrics)

         Metrics(Test loss & Test Accuracy):
         [0.08775996416807175, 0.9721999764442444]

In [15]: #Predicting the output
```



```
In [15]: #Predicting the output

         prediction=model.predict(x_test[:4])
         print(prediction)

         1/1 [==============================] - 0s 182ms/step
         [[5.0611226e-07 1.0023567e-13 7.2327102e-06 4.9399467e-05 7.3030010e-13
           2.7285525e-09 3.1248059e-13 9.9994206e-01 2.8521100e-07 5.0116176e-07]
          [5.7345022e-09 1.0389820e-09 9.9999976e-01 1.5455001e-09 9.5447543e-13
           7.6174532e-13 2.2667260e-07 5.7396274e-15 4.3561246e-10 6.2697646e-15]
          [3.8098758e-06 9.9825519e-01 3.0423570e-04 2.6747462e-05 2.7548935e-04
           1.6991597e-06 3.7201082e-06 1.7417435e-04 9.4995898e-04 5.0237345e-06]
          [9.9725807e-01 8.8474579e-09 2.3000960e-03 1.5741458e-05 5.1110587e-06
           6.0457282e-05 3.4285666e-04 2.8398250e-07 3.0187516e-07 1.7111261e-05]]

In [16]: import numpy as np
         print(np.argmax(prediction,axis=1))
         print(y_test[:4])

         [7 2 1 0]
         [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
          [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
          [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
          [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

In [17]: #Saving the model

         model.save('models/mnistCNN.h5')

In [18]: #Taking images as input and checking results

         from tensorflow.keras.models import load_model
         model = load_model(r'C:\Users\SHIRISHA\mini project\models\mnistCNN.h5')
```



intelligent handwritten digit identification Last Checkpoint: Last Thursday at 10:51 AM (unsaved changes)

```
In [17]: #saving the model

         model.save('models/mnistCNN.h5')

In [18]: #Taking images as input and checking results

         from tensorflow.keras.models import load_model
         model = load_model(r'C:\Users\SHIRISHA\mini project\models\mnistCNN.h5')

In [19]: from PIL import Image
         import numpy as np

In [23]: img = Image.open('C:\projectinputimage.jpg').convert("L")  #convert image to monochrome

In [24]: img = img.resize((28,28))  #resizing of input image

In [26]: im2arr = np.array(img)   #converting to image
         im2arr = im2arr.reshape(1,28,28,1)   #reshaping according to our requirement

         #Predicting the Test set results
         y_pred = model.predict(im2arr)   #predicting the results
         print(y_pred)

         1/1 [==============================] - 0s 126ms/step
         [[0.07074998 0.11541948 0.1125207  0.07422511 0.09784345 0.1205858
           0.07356007 0.10922137 0.11733465 0.10853942]]
```

app.py

#Importing libraries

```
from flask import Flask,render_template,request
from PIL import Image
import numpy as np
from tensorflow.keras.models import load_model
import tensorflow as tf

app=Flask(__name__)
model=load_model("mnistCNN.h5")
#Routing to html page

@app.route('/')
def upload_file():
    return render_template('main.html')
@app.route('/about')
def upload_file1():
    return render_template('main.html')
@app.route('/upload',methods=["POST","GET"])
def upload_file2():
    return render_template('index6.html')
```

#Returning the prediction on UI

```
@app.route('/predict',methods=["POST","GET"])
def upload_image_file():
    if request.method=="POST":
        img=Image.open(request.files['file'].stream).convert("L")
        img=img.resize((28,28),Image.ANTIALIAS)
        im2arr=np.array(img)
        im2arr=im2arr.reshape(1,28,28,1)
        y_pred=model.predict(im2arr)
        print(y_pred)

        if(y_pred==0):
            return render_template("0.html",showcase=str(y_pred))
        elif(y_pred==1):
            return render_template("1.html",showcase=str(y_pred))
        elif(y_pred==2):
            return render_template("2.html",showcase=str(y_pred))
        elif(y_pred==3):
            return render_template("3.html",showcase=str(y_pred))
        elif(y_pred==4):
            return render_template("4.html",showcase=str(y_pred))
        elif(y_pred==5):
            return render_template("5.html",showcase=str(y_pred))
        elif(y_pred==6):
            return render_template("6.html",showcase=str(y_pred))
        elif(y_pred==7):
            return render_template("7.html",showcase=str(y_pred))
        elif(y_pred==8):
            return render_template("8.html",showcase=str(y_pred))
        else:
            return render_template("9.html",showcase=str(y_pred))
    else:
        return None
if __name__=='__main__':
```

```
app.run(host='0.0.0.0',port=3000,debug=True)
```