IDENTIFICATION OF HANDWRITTEN DIGITS USING THE MNIST DATASET

Group Members:

Full Name	UNT ID	E-Mail
Bharath Kumar Sunkari	11504574	bharathkumarsunkari@my.unt.edu
Neeraj Mothukuri	11549636	dhamodarneerajmothukuri@my.unt.edu
Sai Krishna Dasineni	11549138	saikrishnadasineni@my.unt.edu
Sampath Potluri	11516629	sampathpotluri@my.unt.edu

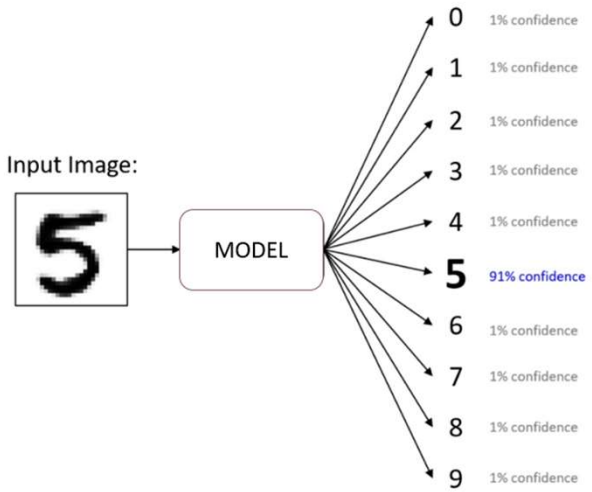
Table of Contents:

ABSTRACT

- Humans detect and visually experience the world around them using their eyes and brains. Computer vision aims to make computers capable of observing and evaluating images in the same way that humans do. One of the difficult approaches in pattern recognition applications is recognizing handwritten numbers.
- Digit recognition applications include postal codes or data form filling, to mention a few. The primary goal is to create a model that can better recognize and determine a handwritten number based on its appearance. We're using an MNIST, which is a dataset of the MNIST database for our project.

PROJECT GOAL

- To create a machine learning model which predicts the hand-written digits collected from the MNIST dataset.
- The goal is to recognize handwritten digits, which means we have 10 classes to predict (0 to 9).



MAJOR REQUIREMENTS

For creating a model:

- **Keras-** Keras is an api from tensorflow used for loading dataset and developing model
- **Scikit-Learn –** It is used for train_test_split, and for evaluating model performance
- **CNN Algorithm -** best accuracy on image classification

For developing a User Interface:

- **React JS -** It's a free JavaScript library for creating single-page user interfaces.
- **Flask -** Flask is a simple and lightweight Python web framework that provides essential tools and capabilities for developing online applications in Python.

Software's Used

- **Visual Studio Code –** In our project VS Code is used to develop the single page web application with the help of React.JS and Flask web frameworks. As a result, it's much easier to demonstrate the results simply by uploading an image of a handwritten digit and clicking Predict to see if we're obtaining the exact value as intended.
- **Anaconda –** we used anaconda as we know we will in the need of numerous data science packages for computing, management of the development and deployment of the project and in anaconda we used Jupiter notebook for writing the code and to develop the models.

STAGES OF PROJECT

Loading the dataset

```
[18] from keras.datasets import mnist
```

```
(trainX, trainy), (testX, testy) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
[3] train = np.concatenate((trainX,testX))
    test = np.concatenate((trainy,testy))
    print(len(train))
    print(len(test))
```

```
70000
70000
```

- Here we have done loading the dataset from Keras

STAGES OF PROJECT

Loading the dataset

```
from sklearn.model_selection import train_test_split
```

```
[4]  data_train, data_test, target_train, target_test =  train_test_split(train,test,stratify=test, test_size=0.25,random_state=42)
```

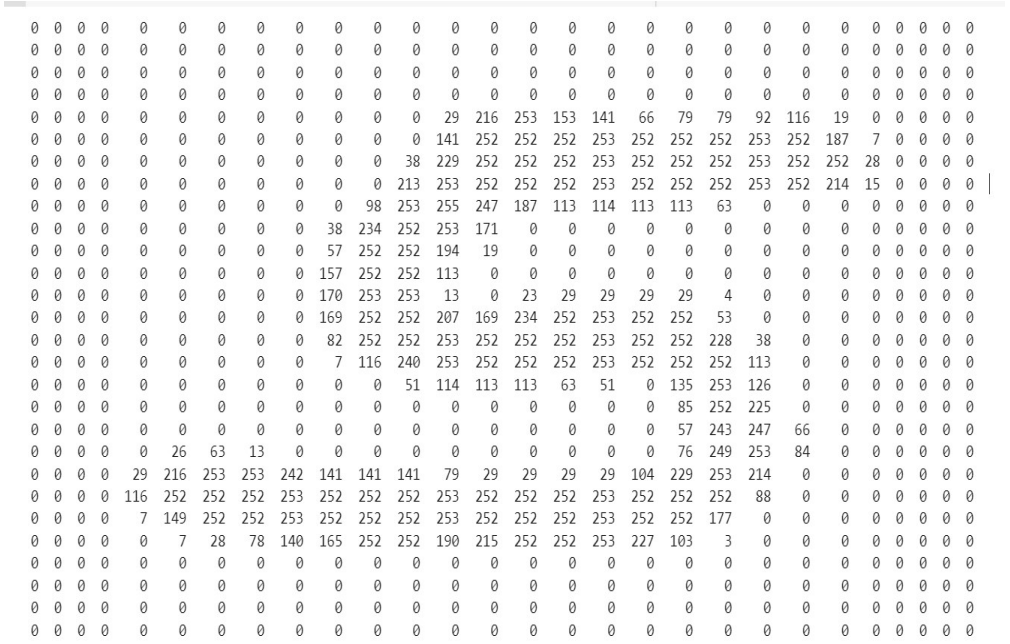
```
[5]  print(len(data_train),len(target_train))
     print(len(data_test),len(target_test))

     52500 52500
     17500 17500
```

- Using scikit-learn, we split the data into Training and Testing for the model. which is 52,500 images for training and 17,500 images for testing

STAGES OF PROJECT

About the dataset

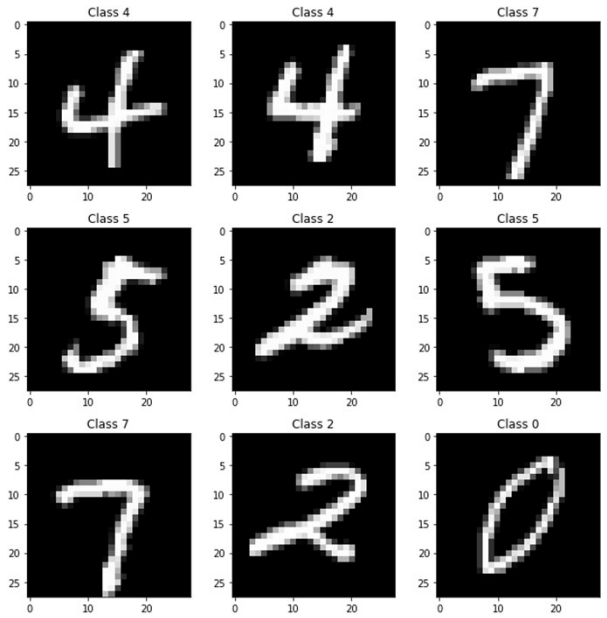
- Images are kept in the form of a matrix of integers, which are referred to as pixel values.



STAGES OF PROJECT

About the dataset

- Each image in the data set is a grey-scale image represented as a 28x28 pixel square, that means each image has total of 784 pixels.



STAGES OF PROJECT

Preprocessing

There are 4 steps in preprocessing:

- **Reshaping:** Reshaping both training and testing image arrays from (52500,28,28), (17500,28,28) to (52500,28,28,1), (17500,28,28,1) where Number 1 indicates gray scale image.
- **Type Conversion:** Converting type of both training and testing image arrays from int to float32
- **Normalization:** Normalize the pixel values to lie between 0 and 1 by dividing with 255.0
- **One-hot encoding :** We do encode the labels for both train and test images, where in an array of ten 0's the label number will be stored in that position.

  For example, label 7 will be stored as [0,0,0,0,0,0,0,1,0,0]

STAGES OF PROJECT

Defining the model

- Created model using convolutional neural network (CNN) machine learning algorithm
- Pixel array serves as input layer which has 28*28=784 pixels and then there are 2 hidden 512 node layers and 10 different category classes serves as output layer
- Keras Sequential Model is used as model architecture
- In last layer, softmax activation is used which converts the input values of an neuron into a probability distribution.

STAGES OF PROJECT

Model performance

- Displaying first 10 predicted and actual labels

```
[ ]  print("predicted values  Actual values")  # comparing first 10 predicted and actual values
     for i in range(10):
       print(prediction[i],"\t\t\t",target_test[i])
```

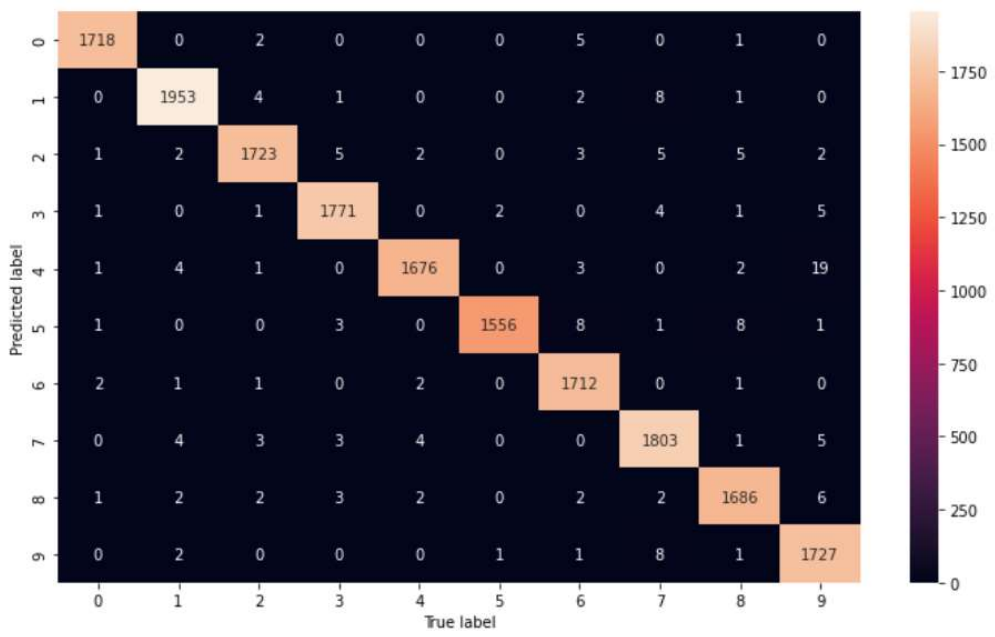
```
predicted values  Actual values
4                         4
1                         1
4                         4
7                         7
7                         7
6                         6
4                         4
7                         7
7                         7
3                         3
```

STAGES OF PROJECT

Confusion matrix:

It is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes.

Model performance



STAGES OF PROJECT

Model performance

Accuracy : It is used in classification problems used to tell the percentage of accurate predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

```
accuracy=accuracy_score(target_test, prediction)
print('accuracy: %.2f' % accuracy)
```

```
accuracy: 0.99
```

STAGES OF PROJECT

Model performance

Precision : Of all the labels that model predicted, what is the percentage of them are correct

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

```
precision = precision_score(target_test, prediction, average='micro')
print('precision: %.3f' % precision)
```

```
precision: 0.990
```

STAGES OF PROJECT

Model performance

Recall : Of all the actual labels, what is the percentage of them are predicted correctly

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

```
recall=recall_score(target_test, prediction, average='micro')
print('Recall: %.3f' % recall)
```

```
Recall: 0.990
```

STAGES OF PROJECT

Model performance

F1 Score: It combines the precision and recall of a classifier into a single metric by taking their harmonic mean

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

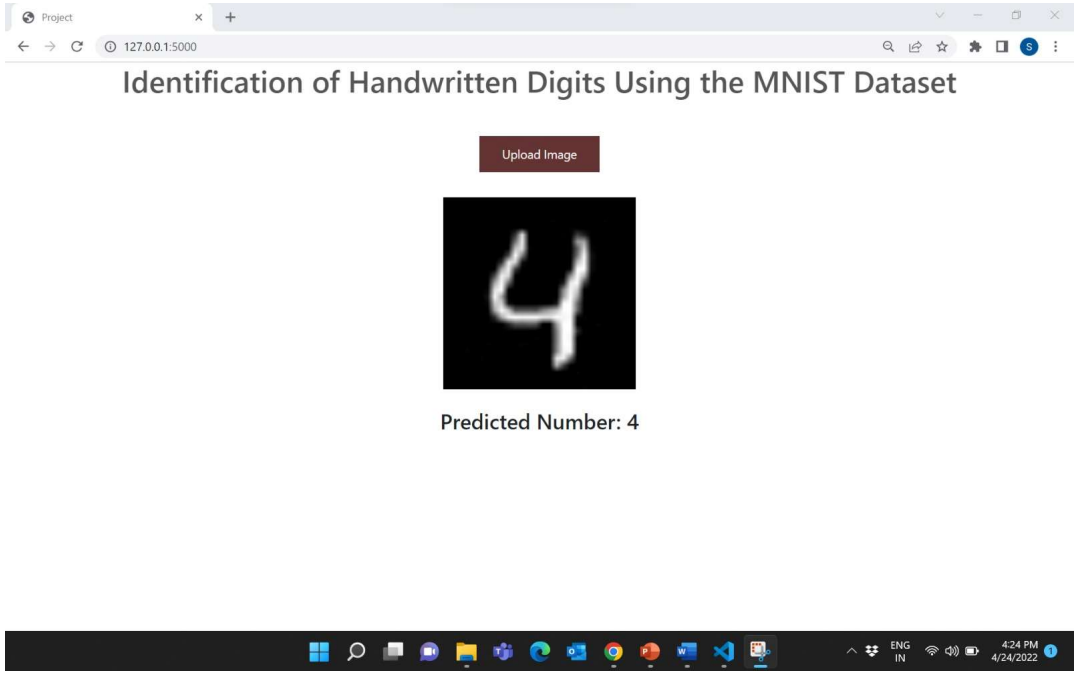
```
score = f1_score(target_test, prediction, average='micro')
print('F-Measure: %.3f' % score)
```

```
F-Measure: 0.990
```

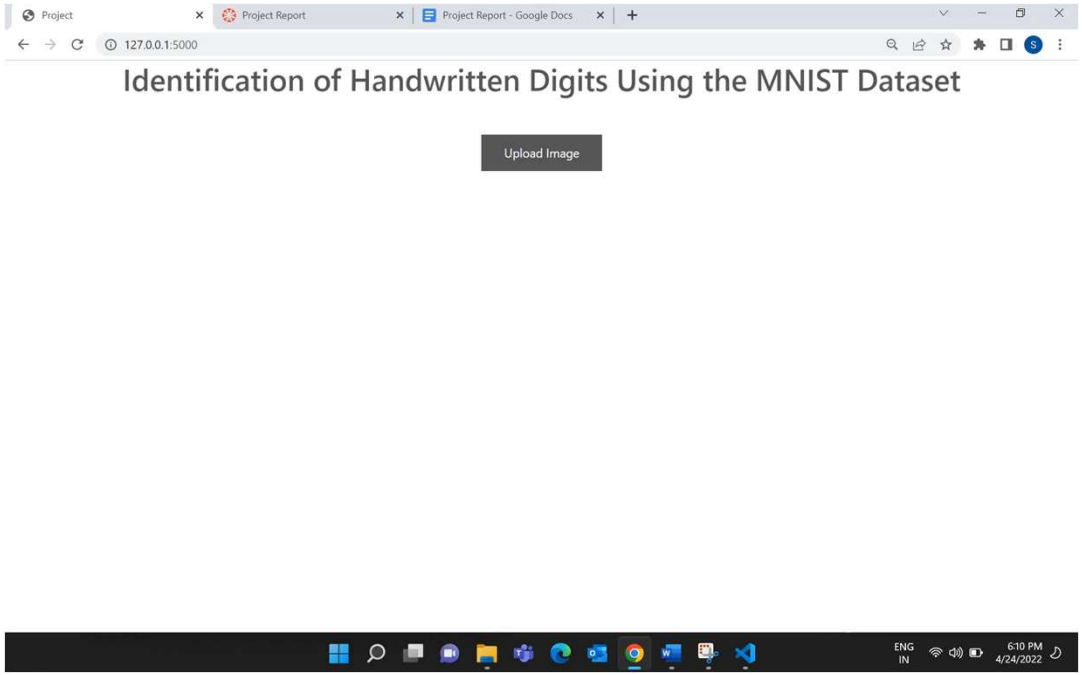
USER INTERFACE

- We have taken of advantage of developing the User-Interface where we can present the project easily understandable.
- To develop the web page, we have used **React.js** which is well known for developing the single page web applications and **Flask** web-framework, because the react and flask is having very good response for their combination in developing the web applications.

USER INTERFACE OUTPUT



USER INTERFACE OUTPUT (SCREEN RECORDED)



WHAT'S NEXT?

- As our current design can take the input of one image at a time and predict a single digit at a time, we think our design can take an image that consists of more than 1 digit and predict all the digits in the image.

TEAM COLLABORATION

- **Bharat Kumar Sunkari:** Worked effectively in choosing the dataset and pre-processing the data and involved while developing the model.
- **Neeraj Mothukuri:** Took initiative to develop the machine learning model from the stage of the choosing the right algorithm and training and testing the data for the best results.
- **Sai Krishna Dasineni**: Involved in developing the model and in addition developed a User Interface using web-frameworks, where the user can upload the image and predict the hand-written digit.
- **Sampath Potluri:** Worked on evaluating the model performance like Plotting the Confusion Matrix, Accuracy Score, Precision, Recall and F1 Score.
- In Overall, all the members contributed to documentation for the project and created a GIT Repository to access the project files.

THANK YOU