BIRLA INSTITUTE OF TECHNOLOGY, MESRA

(Documentation of the Assignment Project:)

Submitted To:

Rashmi Rathi Upadhyay

Submitted By:

Rupa Kumari
MCA/10021/19

Handwritten Digit Recognition

Description:- The handwritten digit recognition is the ability of computers to recognize 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 recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

Technology and Tools are used:-

- Python (3.9)
- Numpy (version 1.16.5)
- Keras (version 2.3.1)
- Tensorflow (as keras uses tensorflow in backend and for image preprocessing) (version 2.0.0)

The MNIST dataset:- This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The MNIST dataset contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. So, the MNIST dataset has 10 different classes. The handwritten digits images are represented as a 28×28 matrix where each cell contains grayscale pixel value.

Building Project:

Import the libraries and load the dataset:- First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it. The mnist.load_data() method returns us the training data, its labels and also the testing data and its labels.

Preprocess the data:- The image data cannot be fed directly into the model so we need to perform some operations and process the data to make it ready for our neural network. The dimension of the training data is (60000,28,28). The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1).

Create the model:- Now we will create our CNN model in Python data science project. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces offer fitting of the model. We will then compile the model with the Adadelta optimizer.

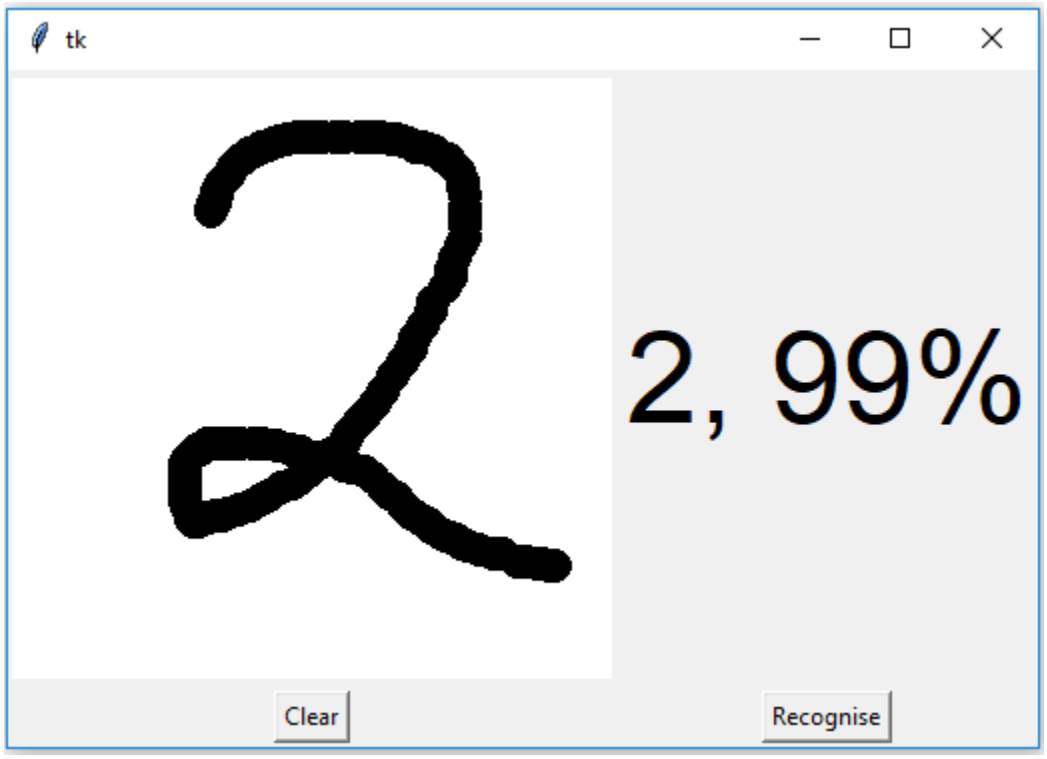
Train the model:- The model.fit() function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size.
It takes some time to train the model. After training, we save the weights and model definition in the 'mnist.h5' file.

Evaluate the model:- We have 10,000 images in our dataset which will be used to evaluate how good our model works**.** The testing data was not involved in the training of the data therefore, it is new data for our model. The MNIST dataset is well balanced so we can get around 99% accuracy.

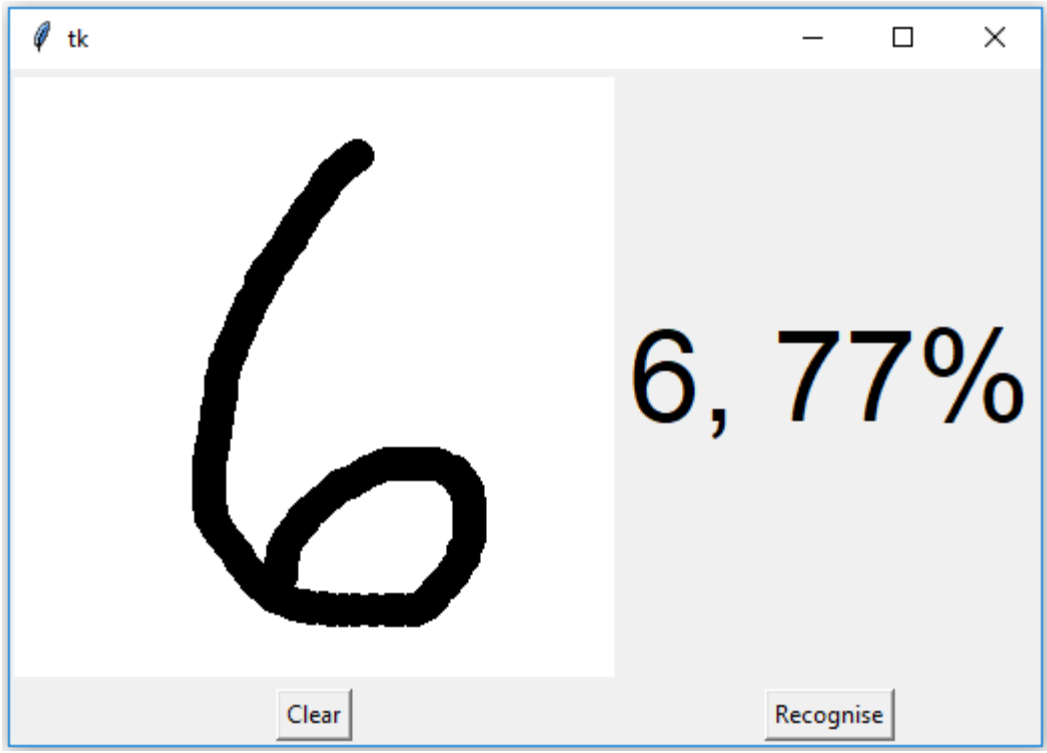
Create GUI to predict digits:- Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function predict_digit() that takes the image as input and then uses the trained model to predict the digit.

Then we create the App class which is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict_digit() function and display the results.

Screenshots:-




tk
5, 70%
Clear
Recognise


tk
6, 77%
Clear
Recognise

Challenges and Limitations:-

This project was my first encounter with Optical Character Recognition (OCR). That is why, while working on this project I was faced with many challenges and issues. First of all, it took me a long while to understand all the concepts and get familiar with them, from image processing to OCR, to all of the techniques and algorithms used in it. Furthermore, the data we were dealing with was very problematic in terms of the way the digits are written. Since some of the digits were rotated by an angle, some of them were thicker or thinner than the rest, and some of the digits were not well centered or were written in confusing ways. In addition to that, overcoming these challenges was not easy since we were only using basic image correlation techniques.

Conclusion and Future Work:-

Optical Character Recognition is a very broad field concerned with turning an image or a scanned document containing a set of characters into an encoded text that could be read by machines. In this project, we have attempted to build a recognizer for handwritten digits using the MNIST dataset. The challenge of this project was to be able to come up with some basic image correlation techniques, instead of some sophisticated algorithms, and see to what extent we can make this mechanism accurate. We have tried several versions and kept trying to improve each one in order to reach a higher performance rate.

The future steps that to go for would be having a closer look at the results of all the versions in order to find new rules. By extracting and implementing them, we will be able to enhance the performance of these versions. Moreover, it would be good if we could make some modifications to both the reference set and the rules in order to make our program more general and able to identify both typed and handwritten digits.

Furthermore, in the future, we could make a great use of the matrices that indicate the first maximum overlap of each test image with the reference images, along with the number of pixels left out from both. These matrices could be used with some clustering algorithms to build a program able to recognize handwritten digits with a very high efficiency.

Last but not least, we thought about using linear or high level regression in the versions we have developed in order to create more rules. As regression could be

used for binary classification and is not very suitable to classify a digit out of ten, this technique could be used in order to tell which digit is the most suitable, the first maximum or second maximum, which will enable us to generate more rules; thus, reach a higher efficiency.