Handwritten Digit Classification Using Neural Networks

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.

**1)AIM:-** Implementation of a handwritten digit recognition app using the MNIST dataset by building a GUI in which you can draw the digit and recognize it straight away.

**2)TECHNIQUE USED:-** Convulation Neural Network(CNN)

**3)STEPS INVOLVED:-**

# (a) 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.

# (b)Preprocess the Data

### The image data cannot be fed directly into the model so we need to**performsome 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).

# (c)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.

# (d) 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.

# (e)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.

# (f) 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.

### 