**TEAM ID :PNT2022TMID43906**

**Add CNN Layers**

import cv2

import numpy as np

from keras.datasets import mnist

from keras.layers import Dense, Flatten, MaxPooling2D, Dropout

from keras.layers.convolutional import Conv2D

from keras.models import Sequential

from tensorflow.keras.utils import to\_categorical

import matplotlib.pyplot as plt

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

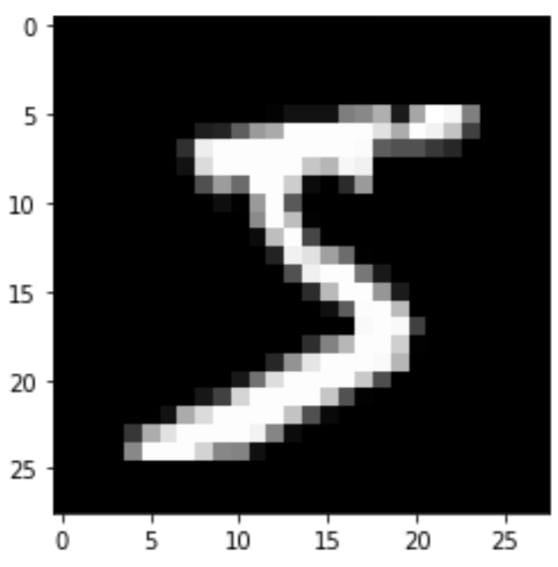
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

11490434/11490434 [==============================] - 0s 0us/step

plt.imshow(X\_train[0], cmap="gray")

plt.show()

print (y\_train[0])



print ("Shape of X\_train: {}".format(X\_train.shape))

print ("Shape of y\_train: {}".format(y\_train.shape))

print ("Shape of X\_test: {}".format(X\_test.shape))

print ("Shape of y\_test: {}".format(y\_test.shape))

Shape of X\_train: (60000, 28, 28)

Shape of y\_train: (60000,)

Shape of X\_test: (10000, 28, 28)

Shape of y\_test: (10000,)

# Reshaping so as to convert images for our model

X\_train = X\_train.reshape(60000, 28, 28, 1)

X\_test = X\_test.reshape(10000, 28, 28, 1)

print ("Shape of X\_train: {}".format(X\_train.shape))

print ("Shape of y\_train: {}".format(y\_train.shape))

print ("Shape of X\_test: {}".format(X\_test.shape))

print ("Shape of y\_test: {}".format(y\_test.shape))

Shape of X\_train: (60000, 28, 28, 1)

Shape of y\_train: (60000,)

Shape of X\_test: (10000, 28, 28, 1)

Shape of y\_test: (10000,)

#one hot encoding

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

model = Sequential()

## Declare the layers

layer\_1 = Conv2D(64, kernel\_size=3, activation='relu', input\_shape=(28, 28, 1))

layer\_2 = MaxPooling2D(pool\_size=2)

layer\_3 = Conv2D(32, kernel\_size=3, activation='relu')

layer\_4 = MaxPooling2D(pool\_size=2)

layer\_5 = Dropout(0.5)

layer\_6 = Flatten()

layer\_7 = Dense(128, activation="relu")

layer\_8 = Dropout(0.5)

layer\_9 = Dense(10, activation='softmax')

## Add the layers to the model

model.add(layer\_1)

model.add(layer\_2)

model.add(layer\_3)

model.add(layer\_4)

model.add(layer\_5)

model.add(layer\_6)

model.add(layer\_7)

model.add(layer\_8)

model.add(layer\_9)

**Compiling The Model**

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

**Train The Model**

model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=3)

Epoch 1/3

1875/1875 [==============================] - 58s 31ms/step - loss: 0.8654 - accuracy: 0.7801 - val\_loss: 0.1307 - val\_accuracy: 0.9630

Epoch 2/3

1875/1875 [==============================] - 58s 31ms/step - loss: 0.2703 - accuracy: 0.9201 - val\_loss: 0.0750 - val\_accuracy: 0.9757

Epoch 3/3

1875/1875 [==============================] - 56s 30ms/step - loss: 0.2055 - accuracy: 0.9385 - val\_loss: 0.0746 - val\_accuracy: 0.9772