Project Development Phase

Sprint - 2

Team ID	PNT2022TMID18280
Project Name	A Novel Method for Handwritten Digit Recognition

Sprint - 2

Team Id : PNT2022TMID18280

Importing Packages

In [2]:
```
import numpy
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten
from tensorflow.keras.optimizers import Adam
```

Loading the data

In [3]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Analysing the data

In [4]:
```
print(X_train.shape)
print(X_test.shape)
```

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

In [5]:
```
X_train[2]
```

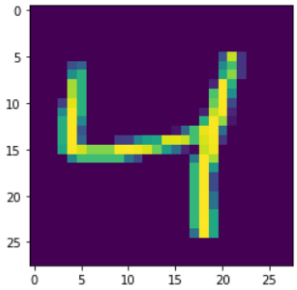
Out[5]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   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,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,  67, 232,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  62,  81,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0, 120, 180,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 126, 163,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   2, 153, 210,  40,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 220, 163,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,  27, 254, 162,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 222, 163,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0, 183, 254, 125,   0,   0,   0,   0,
          0,   0],
```

In [6]:
```
y_train[2]
```

Out[6]: 4

In [7]:
```
plt.imshow(X_train[2])
```

Out[7]: <matplotlib.image.AxesImage at 0x20e3cd0bac0>



Data Preprocessing

In [8]:
```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

In [9]:
```
number_of_classes = 10
Y_train = np_utils.to_categorical(y_train, number_of_classes)
Y_test = np_utils.to_categorical(y_test, number_of_classes)
```

In [10]:
```
Y_train[2]
```

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

Create the Model

In [11]:
```
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"))
```

Compile the Model

In [12]:
```
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["accuracy"])
```

Train the Model

In [13]:
```
model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 245s 125ms/step - loss: 0.2357 - accuracy: 0.9507 - val_loss: 0.0820 - val_accurac
y: 0.9741
Epoch 2/5
1875/1875 [==============================] - 123s 66ms/step - loss: 0.0679 - accuracy: 0.9787 - val_loss: 0.1080 - val_accurac
y: 0.9707
Epoch 3/5
1875/1875 [==============================] - 110s 59ms/step - loss: 0.0480 - accuracy: 0.9847 - val_loss: 0.0740 - val_accurac
y: 0.9787
Epoch 4/5
1875/1875 [==============================] - 111s 59ms/step - loss: 0.0373 - accuracy: 0.9883 - val_loss: 0.0922 - val_accurac
y: 0.9747
Epoch 5/5
1875/1875 [==============================] - 153s 82ms/step - loss: 0.0288 - accuracy: 0.9912 - val_loss: 0.1193 - val_accurac
y: 0.9757
```

Out[13]: <keras.callbacks.History at 0x20e3c9f37f0>

Test the Model

In [14]:
```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.11928823590278625, 0.9757000207901001]
```

In [15]:
```
prediction = model.predict(X_test[:5])
print(prediction)
```

```
1/1 [==============================] - 8s 8s/step
[[1.98845407e-09 2.53684687e-13 5.25401900e-09 2.03659079e-06
  4.40039061e-10 3.57320044e-15 2.47841625e-17 9.99986053e-01
  8.53644710e-09 1.19455672e-05]
 [3.59238993e-11 1.60101763e-08 9.99999404e-01 1.22668133e-08
  7.41926958e-16 8.29145623e-16 6.16015882e-07 2.35153823e-11
  6.87689281e-11 2.78174328e-16]
 [2.26069632e-11 9.99945521e-01 1.92773841e-05 7.22188206e-13
  8.31564739e-06 9.44725098e-09 2.58532729e-09 2.39214569e-05
  2.84418707e-06 1.08004850e-08]
 [1.00000000e+00 2.52734939e-20 8.90307561e-09 3.92685074e-19
  1.31294670e-15 4.15755517e-14 3.17433052e-10 3.82893409e-15
  2.22973942e-13 8.94497698e-10]
 [1.18800011e-12 1.20145148e-12 1.31972875e-14 2.86097066e-11
  1.00000000e+00 1.27235741e-15 3.66013524e-16 1.25803716e-16
  4.67583617e-15 1.86367545e-12]]
```

In [16]:
```
print(numpy.argmax(prediction, axis=1))
print(Y_test[:5])
```

```
[7 2 1 0 4]
[[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.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
```