CNN - Image classification model

In [20]:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
import matplotlib.pyplot as plt
import numpy as np
```

1) Loading and preprocessing the Image data

In [21]:

```
mnist = tf.keras.datasets.mnist
```

In [22]:

```
(x_train, y_train) , (x_test, y_test) = mnist.load_data()
input_shape = (28 , 28 , 1)
```

Reshape changes the shape of the image without changing the total size. For example, you can reshape image from 100x100 to 10x1000 or to 1x100x100.

In [23]:

```
x_train = x_train.reshape(x_train.shape[0] , 28 , 28 , 1)
x_test = x_test.reshape(x_test.shape[0] , 28 , 28 , 1)
```

Python astype() method enables us to set or convert the data type of an existing data column in a dataset or a data frame. By this, we can change or transform the type of the data values or single or multiple columns to altogether another form using astype() function.

In [24]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

2) Training the Model

In [25]:

```
x_train = x_train/255
x_test = x_test/255
```

```
print("Shape of Training : " , x_train.shape)
print("Shae of Testing : " , x_test.shape)
```

```
Shape of Training :  (60000, 28, 28, 1)
Shae of Testing :  (10000, 28, 28, 1)
```

2)Defining Model Architecture

Flatten layers => are used when you got a multidimensional output and you want to make it linear to pass it onto a Dense layer. se.

Dense layers => are used when association can exist among any feature to any other feature in data point .Since between two layers of size n1 and n2, there can n1*n2 connections and these are referred to as Dense

conv layers => these are important when nearby associations among the features matter, example object detection. Neighborhoods matter to classify or detect.

Dropout is a way of cutting too much association among features by dropping the weights (edges) at a probability.

In [26]:

```
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(200, activation = "relu"))
model.add(Dropout(0.3))
model.add(Dense(10, activation="softmax"))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 28)        280       
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 13, 13, 28)        0         
 g2D)                                                            
                                                                 
 flatten_2 (Flatten)         (None, 4732)              0         
                                                                 
 dense_4 (Dense)             (None, 200)               946600    
                                                                 
 dropout_2 (Dropout)         (None, 200)               0         
                                                                 
 dense_5 (Dense)             (None, 10)                2010      
                                                                 
=================================================================
Total params: 948890 (3.62 MB)
Trainable params: 948890 (3.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

4) Estimating Model Performance

LOSS => In machine learning, Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process.

METRICS => In machine learning, Metrics is used to evaluate the performance of your model.

Optimization => is an important process which optimize the input weights by comparing the prediction and the loss function.

In [27]:

```
model.compile(optimizer='adam', loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(x_train, y_train, epochs = 2)
```

```
Epoch 1/2
1875/1875 [==============================] - 56s 29ms/step - loss: 0.2009 - accuracy: 0.9402
Epoch 2/2
1875/1875 [==============================] - 49s 26ms/step - loss: 0.0808 - accuracy: 0.9754
```

```
Out[27]: <keras.src.callbacks.History at 0x1a24ac5f3d0>
```

Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.

In [28]:

```python
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

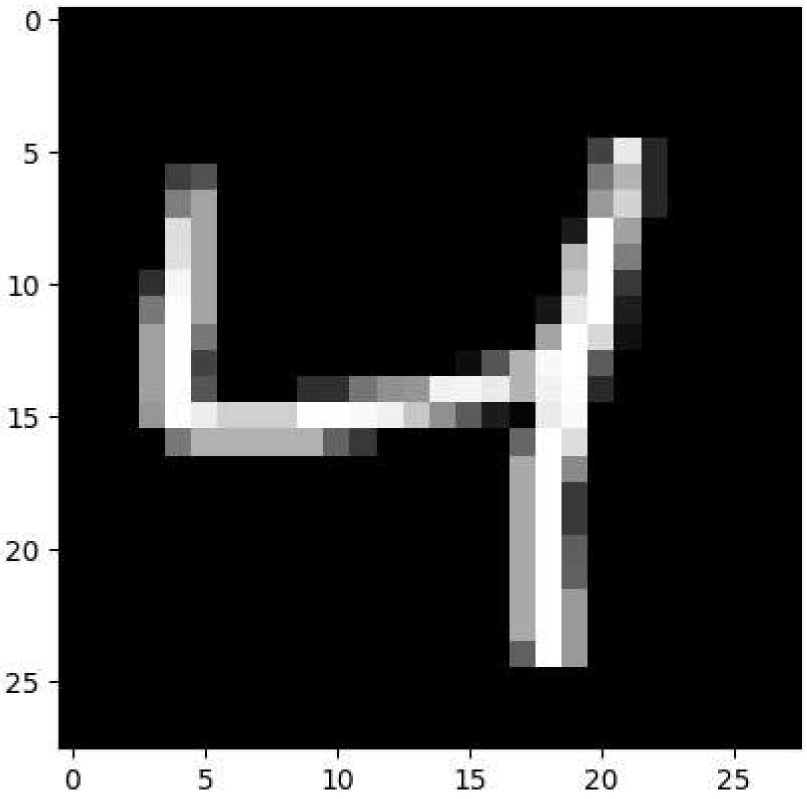
```
313/313 [==============================] - 2s 6ms/step - loss: 0.0529 - accuracy: 0.9837
Loss=0.053
Accuracy=0.984
```

The imshow() function in pyplot module of matplotlib library is used to display data as an image; i.e. on a 2D regular raster.

The squeeze() function in NumPy is used to remove an axis of length 1 from an input array. Axes in NumPy are defined for arrays having more than one dimension.

In [29]:

```python
image = x_train[2]
plt.imshow(np.squeeze(image), cmap='gray')
plt.show()
```



In [30]:

```
image = image.reshape(1, image.shape[0], image.shape[1], image.shape[2])
predict_model = model.predict([image])
print("Predicted class: {}".format(np.argmax(predict_model)))
```

```
1/1 [==============================] - 0s 76ms/step
Predicted class: 4
```