Let's import Library

In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import tensorflow as tf

from tensorflow.keras import models,layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization,Activation,Flatten
from tensorflow.keras.optimizers import Adam
```

In [2]:

```
num_classes = 10
input_shape = (28,28,1)

#load MNIST dataset

(x_train,y_train),(x_test,y_test) = tf.keras.datasets.mnist.load_data()
```

Observation : We have taken input shape (28,28,1) and loaded the MNIST dataset

In [3]:

```
print(x_train.shape)
```

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

Observation : There are 60000 rows and image pixels size is 28*28

In [4]:

```
print(x_test.shape)
```

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

In [5]:

```
print(y_train.shape)
```

```
(60000,)
```

In [6]:

```
print(y_test.shape)
```

```
(10000,)
```

In [7]:

```
#scale images to the [0,1] range

x_train = x_train.astype("float32")/255
x_test = x_test.astype("float32")/255

#images have shape (28,28,1)
x_train = np.expand_dims(x_train,-1)
x_test = np.expand_dims(x_test,-1)
print("x_train shape",x_train.shape)
```

```
x_train_shape (60000, 28, 28, 1)
```

In [8]:

```
print(x_train.shape[0],"train_samples")
print(x_test.shape[0],"test_samples")
```

```
60000 train_samples
10000 test_samples
```

One hot encoding

In [9]:

```
y_train = tf.keras.utils.to_categorical(y_train,num_classes)
y_test = tf.keras.utils.to_categorical(y_test,num_classes)
```

In [10]:

```
print(y_train.shape)
print(y_test.shape)
```

```
(60000, 10)
(10000, 10)
```

In [11]:

```
model = tf.keras.Sequential(
[
    tf.keras.Input(shape=input_shape),
    layers.Conv2D(32,kernel_size=(3,3),activation = "relu"),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(64,kernel_size=(3,3),activation = "relu"),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes,activation="softmax")

])

model.summary()
```

```
WARNING:tensorflow:From C:\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: cal
ling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and w
ill be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0         
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0         
_________________________________________________________________
dense (Dense)                (None, 10)                16010     
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Train Model

In [12]:

```
batch_size=128
epochs= 15

model.compile(loss = "categorical_crossentropy",optimizer = "adam",metrics= ["accuracy"])
```

In [13]:

```
model.fit(x_train,y_train,batch_size = batch_size,epochs = 15)
```

```
Epoch 1/15
60000/60000 [==============================] - 21s 355us/sample - loss: 0.3482 - acc: 0.8942
Epoch 2/15
60000/60000 [==============================] - 22s 361us/sample - loss: 0.1068 - acc: 0.9685
Epoch 3/15
60000/60000 [==============================] - 21s 357us/sample - loss: 0.0816 - acc: 0.9755
Epoch 4/15
60000/60000 [==============================] - 25s 410us/sample - loss: 0.0691 - acc: 0.9786
Epoch 5/15
60000/60000 [==============================] - 24s 407us/sample - loss: 0.0587 - acc: 0.9824
Epoch 6/15
60000/60000 [==============================] - 26s 425us/sample - loss: 0.0536 - acc: 0.9833
Epoch 7/15
60000/60000 [==============================] - 24s 397us/sample - loss: 0.0496 - acc: 0.9842
Epoch 8/15
60000/60000 [==============================] - 23s 388us/sample - loss: 0.0467 - acc: 0.9850
Epoch 9/15
60000/60000 [==============================] - 23s 387us/sample - loss: 0.0425 - acc: 0.9868
Epoch 10/15
60000/60000 [==============================] - 23s 387us/sample - loss: 0.0413 - acc: 0.9866
Epoch 11/15
60000/60000 [==============================] - 23s 388us/sample - loss: 0.0378 - acc: 0.9877
Epoch 12/15
60000/60000 [==============================] - 23s 387us/sample - loss: 0.0367 - acc: 0.9878
Epoch 13/15
60000/60000 [==============================] - 24s 400us/sample - loss: 0.0338 - acc: 0.9890
Epoch 14/15
60000/60000 [==============================] - 24s 400us/sample - loss: 0.0336 - acc: 0.9895
Epoch 15/15
60000/60000 [==============================] - 25s 416us/sample - loss: 0.0319 - acc: 0.9897
```

Out[13]:

```
<tensorflow.python.keras.callbacks.History at 0x1bb3f7bd7f0>
```

Observation : Found trained accuracy 98%

In [16]:

```
score = model.evaluate(x_test,y_test)
print("Test Loss",score[0])
print("Test accuracy",score[1])
```

```
10000/10000 [==============================] - 1s 95us/sample - loss: 0.0238 - acc: 0.9918
Test Loss 0.02377357642118004
Test accuracy 0.9918
```

Observation : WE found test accuracy : 99%

In [ ]:

```

```