In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras import layers

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt # plotting library
%matplotlib inline

from tensorflow.keras.models import Sequential
from keras.layers import Dense , Activation, Dropout
from tensorflow.keras.optimizers import Adam ,RMSprop, SGD
from keras import  backend as K
```

In [2]:
```
num_classes = 10
input_shape = (28, 28, 1)

# Load the data and split it between train and test sets
path = 'C:\\Users\\mourya\\Downloads\\dl\\mnist.npz'
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data(path)
len(x_train)


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

# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255
# Make sure 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)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")


# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [3]:
```
model = keras.Sequential(
    [
        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()
```

```
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 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 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
_________________________________________________________________
```

In [9]:

```
batch_size = 128
epochs = 10

model.compile(loss="categorical_crossentropy", optimizer="SGD", metrics=["accuracy"]

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0
```

```
Epoch 1/10
422/422 [==============================] - 31s 72ms/step - loss: 0.1674 - accuracy:
0.9490 - val_loss: 0.0970 - val_accuracy: 0.9750
Epoch 2/10
422/422 [==============================] - 30s 72ms/step - loss: 0.1595 - accuracy:
0.9521 - val_loss: 0.0925 - val_accuracy: 0.9757
Epoch 3/10
422/422 [==============================] - 30s 71ms/step - loss: 0.1515 - accuracy:
0.9539 - val_loss: 0.0891 - val_accuracy: 0.9757
Epoch 4/10
422/422 [==============================] - 30s 72ms/step - loss: 0.1475 - accuracy:
0.9556 - val_loss: 0.0864 - val_accuracy: 0.9767
Epoch 5/10
422/422 [==============================] - 29s 69ms/step - loss: 0.1430 - accuracy:
0.9575 - val_loss: 0.0828 - val_accuracy: 0.9775
Epoch 6/10
422/422 [==============================] - 29s 70ms/step - loss: 0.1381 - accuracy:
0.9575 - val_loss: 0.0810 - val_accuracy: 0.9775
Epoch 7/10
422/422 [==============================] - 30s 71ms/step - loss: 0.1332 - accuracy:
0.9600 - val_loss: 0.0783 - val_accuracy: 0.9783
Epoch 8/10
422/422 [==============================] - 29s 70ms/step - loss: 0.1271 - accuracy:
0.9616 - val_loss: 0.0762 - val_accuracy: 0.9795
Epoch 9/10
422/422 [==============================] - 30s 72ms/step - loss: 0.1254 - accuracy:
0.9624 - val_loss: 0.0738 - val_accuracy: 0.9805
Epoch 10/10
422/422 [==============================] - 31s 74ms/step - loss: 0.1203 - accuracy:
0.9636 - val_loss: 0.0728 - val_accuracy: 0.9802
```

Out[9]: <keras.callbacks.History at 0x2ac4b8281c0>

In [10]:

```
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.07545896619558334
Test accuracy: 0.9767000079154968
```

In [ ]:

In [ ]: