```
Assignment No.2
Name : Shivani Gaikwad
Roll No. 43315
Batch : P-11
```

1. Import the necessary packages

In [1]:
```
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import mnist
from tensorflow.keras import backend as K
import matplotlib.pyplot as plt
import numpy as np
import argparse
```

2. Grab the MNIST Dataset

In [2]:
```
print("Accessing mnist...")
((trainX, trainY), (testX, testY)) = mnist.load_data()

# Flatten the images
trainX = trainX.reshape((trainX.shape[0], 28 * 28 * 1))
testX = testX.reshape((testX.shape[0], 28 * 28 * 1))

# Scale data to the range of [0, 1]
trainX = trainX.astype("float32") / 255.0
testX = testX.astype("float32") / 255.0

# convert labels from integers to vectors
le = LabelBinarizer()
trainY = le.fit_transform(trainY)
testY = le.fit_transform(testY)
```

```
Accessing mnist...
```

3. Feed Forward Neural Network

In [3]:
```
model = Sequential()
model.add(Dense(256, input_shape = (784,), activation = "sigmoid"))
model.add(Dense(128, activation = "sigmoid"))
model.add(Dense(10, activation = "softmax"))
```

4. Adding SGD Optimizer and training

In [4]:
```
# Adding optimizer
# Learning rate --> 0.01
sgd = SGD(0.01)

# Fitting and training
model.compile(loss="categorical_crossentropy", optimizer=sgd, metrics=["accuracy"
H = model.fit(trainX, trainY, validation_data=(testX, testY), epochs=100, batch_s
```

```
Epoch 57/100
469/469 [==============================] - 2s 5ms/step - loss: 0.3169 - accur
acy: 0.9090 - val_loss: 0.3057 - val_accuracy: 0.9115
```

In [5]:
```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               200960

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0
_________________________________________________________________
None
```

5. Evaluating the model

In [6]:
```
predictions = model.predict(testX, batch_size=128)
print(classification_report(testY.argmax(axis=1), predictions.argmax(axis=1), tar
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       980
           1       0.97      0.98      0.97      1135
           2       0.92      0.91      0.91      1032
           3       0.90      0.91      0.91      1010
           4       0.92      0.93      0.92       982
           5       0.91      0.86      0.88       892
           6       0.93      0.94      0.94       958
           7       0.93      0.92      0.93      1028
           8       0.89      0.90      0.89       974
           9       0.91      0.90      0.91      1009

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

6. Plotting training loss and accuracy

In [7]:
```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 100), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 100), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 100), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 100), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
```

Out[7]: <matplotlib.legend.Legend at 0x1966024e850>