Name: **Krishna Gadam**

Div: **BE9-Q9**

Roll no: **43124**

Title: **Assignment 2: Implementing Feedforward neural networks with Keras and TensorFlow**

In [8]:

```
#installations
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
```

In [9]:

```
#grabbing the mnist dataset
((X_train, Y_train), (X_test, Y_test)) = mnist.load_data()
X_train = X_train.reshape((X_train.shape[0], 28 * 28 * 1))
X_test = X_test.reshape((X_test.shape[0], 28 * 28 * 1))
X_train = X_train.astype("float32") / 255.0
X_test = X_test.astype("float32") / 255.0
```

In [10]:

```
lb = LabelBinarizer()
Y_train = lb.fit_transform(Y_train)
Y_test = lb.transform(Y_test)
```

In [11]:

```
#building the model
model = Sequential()
model.add(Dense(128, input_shape=(784,), activation="sigmoid"))
model.add(Dense(64, activation="sigmoid"))
model.add(Dense(10, activation="softmax"))
```

In [15]:
```
sgd = SGD(0.01)
epochs=10
model.compile(loss="categorical_crossentropy", optimizer=sgd,metrics=["accuracy"]
H = model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=epochs, b
```

```
Epoch 1/10
469/469 [==============================] - 3s 6ms/step - loss: 1.7774 - accurac
y: 0.5913 - val_loss: 1.6768 - val_accuracy: 0.6130
Epoch 2/10
469/469 [==============================] - 2s 5ms/step - loss: 1.5965 - accurac
y: 0.6282 - val_loss: 1.4964 - val_accuracy: 0.6528
Epoch 3/10
469/469 [==============================] - 2s 5ms/step - loss: 1.4255 - accurac
y: 0.6666 - val_loss: 1.3359 - val_accuracy: 0.6909
Epoch 4/10
469/469 [==============================] - 2s 5ms/step - loss: 1.2772 - accurac
y: 0.7010 - val_loss: 1.2009 - val_accuracy: 0.7123
Epoch 5/10
469/469 [==============================] - 2s 5ms/step - loss: 1.1540 - accurac
y: 0.7259 - val_loss: 1.0890 - val_accuracy: 0.7391
Epoch 6/10
469/469 [==============================] - 3s 5ms/step - loss: 1.0531 - accurac
y: 0.7453 - val_loss: 0.9984 - val_accuracy: 0.7653
Epoch 7/10
469/469 [==============================] - 2s 5ms/step - loss: 0.9705 - accurac
y: 0.7646 - val_loss: 0.9232 - val_accuracy: 0.7744
Epoch 8/10
469/469 [==============================] - 2s 5ms/step - loss: 0.9016 - accurac
y: 0.7781 - val_loss: 0.8599 - val_accuracy: 0.7926
Epoch 9/10
469/469 [==============================] - 2s 5ms/step - loss: 0.8431 - accurac
y: 0.7917 - val_loss: 0.8062 - val_accuracy: 0.8018
Epoch 10/10
469/469 [==============================] - 2s 5ms/step - loss: 0.7926 - accurac
y: 0.8022 - val_loss: 0.7584 - val_accuracy: 0.8154
```

In [16]:
```
#making the predictions
predictions = model.predict(X_test, batch_size=128)
print(classification_report(Y_test.argmax(axis=1),predictions.argmax(axis=1),targ
```

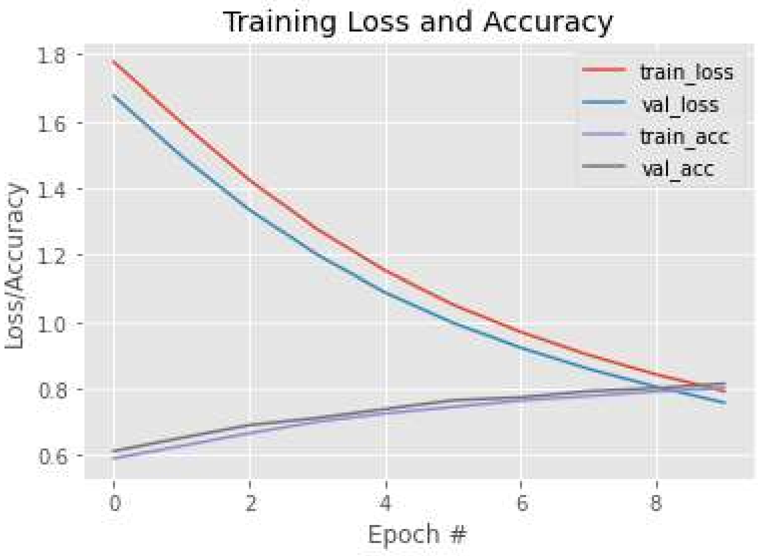
```
79/79 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       980
           1       0.87      0.98      0.92      1135
           2       0.85      0.77      0.81      1032
           3       0.76      0.81      0.79      1010
           4       0.73      0.84      0.78       982
           5       0.81      0.61      0.70       892
           6       0.87      0.90      0.89       958
           7       0.84      0.86      0.85      1028
           8       0.78      0.71      0.74       974
           9       0.76      0.67      0.71      1009

    accuracy                           0.82     10000
   macro avg       0.81      0.81      0.81     10000
weighted avg       0.81      0.82      0.81     10000
```

In [17]:
```
#plotting the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, epochs), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, epochs), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, epochs), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, epochs), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
```

Out[17]:
```
<matplotlib.legend.Legend at 0x7f55a707da10>
```


Training Loss and Accuracy
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch #