In [1]:
```
#Assignment 2: Implementing Feedforward neural networks with Keras and TensorFlo
from sklearn.preprocessing import LabelBinarize
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 [2]:
```
((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 [3]:
```
lb = LabelBinarizer()
Y_train = lb.fit_transform(Y_train)
Y_test = lb.transform(Y_test)
```

In [4]:
```
model = Sequential()
model.add(Dense(128, input_shape=(784,), activation="sigmoid"))
model.add(Dense(64, activation="sigmoid"))
model.add(Dense(10, activation="softmax"))
```

In [8]:
```
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 [==============================] - 4s 7ms/step - loss: 2.2846 - accurac
y: 0.1874 - val_loss: 2.2474 - val_accuracy: 0.2559
Epoch 2/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2204 - accurac
y: 0.3637 - val_loss: 2.1847 - val_accuracy: 0.4210
Epoch 3/10
469/469 [==============================] - 3s 6ms/step - loss: 2.1479 - accurac
y: 0.4922 - val_loss: 2.0960 - val_accuracy: 0.5758
Epoch 4/10
469/469 [==============================] - 3s 6ms/step - loss: 2.0428 - accurac
y: 0.5624 - val_loss: 1.9677 - val_accuracy: 0.6247
Epoch 5/10
469/469 [==============================] - 3s 6ms/step - loss: 1.8940 - accurac
y: 0.6043 - val_loss: 1.7939 - val_accuracy: 0.6364
Epoch 6/10
469/469 [==============================] - 3s 6ms/step - loss: 1.7091 - accurac
y: 0.6344 - val_loss: 1.5973 - val_accuracy: 0.6702
Epoch 7/10
469/469 [==============================] - 3s 6ms/step - loss: 1.5184 - accurac
y: 0.6687 - val_loss: 1.4123 - val_accuracy: 0.6941
Epoch 8/10
469/469 [==============================] - 3s 6ms/step - loss: 1.3480 - accurac
y: 0.6986 - val_loss: 1.2550 - val_accuracy: 0.7204
Epoch 9/10
469/469 [==============================] - 3s 6ms/step - loss: 1.2052 - accurac
y: 0.7268 - val_loss: 1.1259 - val_accuracy: 0.7414
Epoch 10/10
469/469 [==============================] - 3s 6ms/step - loss: 1.0875 - accurac
y: 0.7487 - val_loss: 1.0191 - val_accuracy: 0.7692
```

In [10]:
```
predictions = model.predict(X_test, batch_size=128)
print(classification_report(Y_test.argmax(axis=1),predictions.argmax(axis=1),tar
```

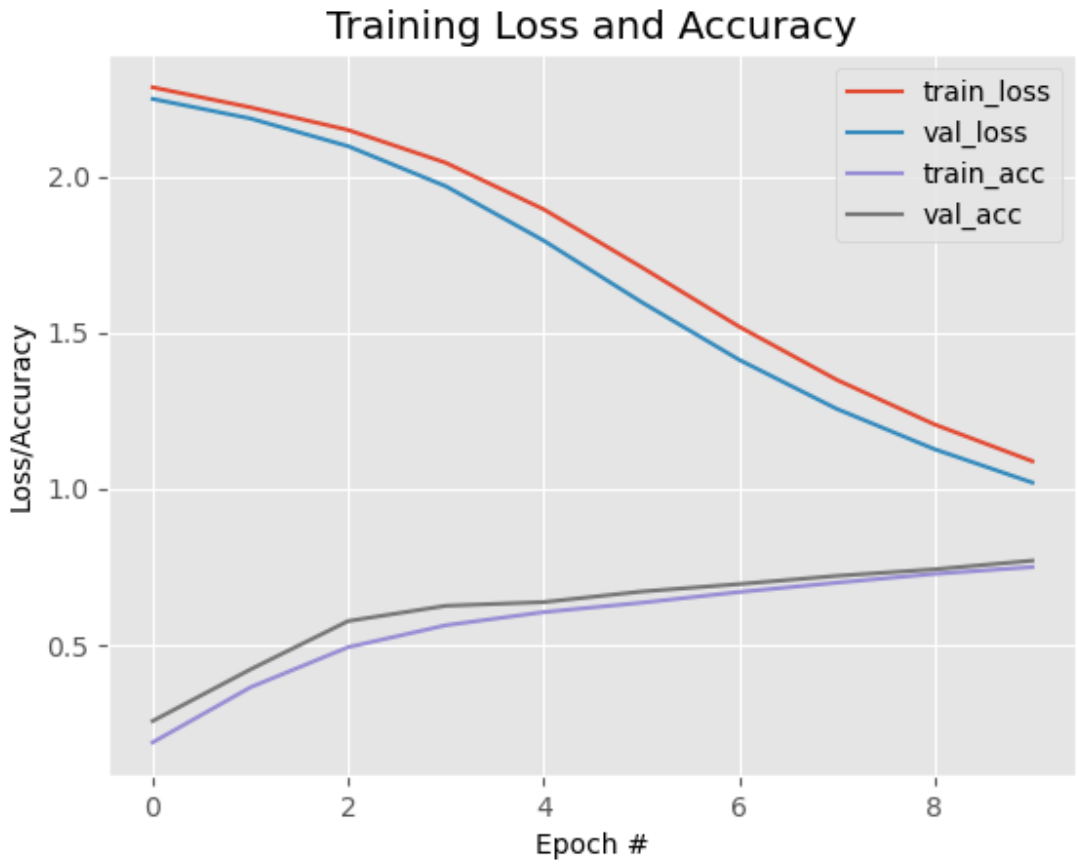
```
79/79 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

           0       0.82      0.97      0.89       980
           1       0.77      0.99      0.87      1135
           2       0.84      0.70      0.76      1032
           3       0.67      0.83      0.74      1010
           4       0.71      0.88      0.78       982
           5       0.78      0.41      0.54       892
           6       0.81      0.87      0.84       958
           7       0.76      0.88      0.81      1028
           8       0.79      0.54      0.64       974
           9       0.79      0.56      0.65      1009

    accuracy                           0.77     10000
   macro avg       0.77      0.76      0.75     10000
weighted avg       0.77      0.77      0.76     10000
```

In [11]:
```
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[11]: <matplotlib.legend.Legend at 0x1ee51a68d90>



In [ ]: