Assignment 2: Implementing Feedforward neural networks with Keras and TensorFlow

Name: Zahir Mulla

In [12]:
```
#RollnO: B512012
#Class: BE-IT(B)
```

In [1]:
```
#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 [2]:
```
#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 [3]:
```
lb = LabelBinarizer()
Y_train = lb.fit_transform(Y_train)
Y_test = lb.transform(Y_test)
```

In [4]:
```
#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 [5]:
```
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, bat
```

```
Epoch 1/10
469/469 [==============================] - 6s 8ms/step - loss: 2.2969 - accuracy:
0.1602 - val_loss: 2.2589 - val_accuracy: 0.2495
Epoch 2/10
469/469 [==============================] - 3s 7ms/step - loss: 2.2349 - accuracy:
0.3125 - val_loss: 2.2058 - val_accuracy: 0.4334
Epoch 3/10
469/469 [==============================] - 3s 7ms/step - loss: 2.1744 - accuracy:
0.4578 - val_loss: 2.1346 - val_accuracy: 0.5290
Epoch 4/10
469/469 [==============================] - 3s 7ms/step - loss: 2.0897 - accuracy:
0.5428 - val_loss: 2.0319 - val_accuracy: 0.5811
Epoch 5/10
469/469 [==============================] - 3s 7ms/step - loss: 1.9704 - accuracy:
0.5804 - val_loss: 1.8909 - val_accuracy: 0.6133
Epoch 6/10
469/469 [==============================] - 3s 7ms/step - loss: 1.8142 - accuracy:
0.6122 - val_loss: 1.7176 - val_accuracy: 0.6407
Epoch 7/10
469/469 [==============================] - 3s 7ms/step - loss: 1.6344 - accuracy:
0.6448 - val_loss: 1.5297 - val_accuracy: 0.6721
Epoch 8/10
469/469 [==============================] - 4s 8ms/step - loss: 1.4520 - accuracy:
0.6755 - val_loss: 1.3516 - val_accuracy: 0.7004
Epoch 9/10
469/469 [==============================] - 3s 7ms/step - loss: 1.2868 - accuracy:
0.7057 - val_loss: 1.1983 - val_accuracy: 0.7260
Epoch 10/10
469/469 [==============================] - 4s 8ms/step - loss: 1.1487 - accuracy:
0.7325 - val_loss: 1.0740 - val_accuracy: 0.7516
```

In [6]:

```
#making the predictions
predictions = model.predict(X_test, batch_size=128)
print(classification_report(Y_test.argmax(axis=1),predictions.argmax(axis=1),target
```

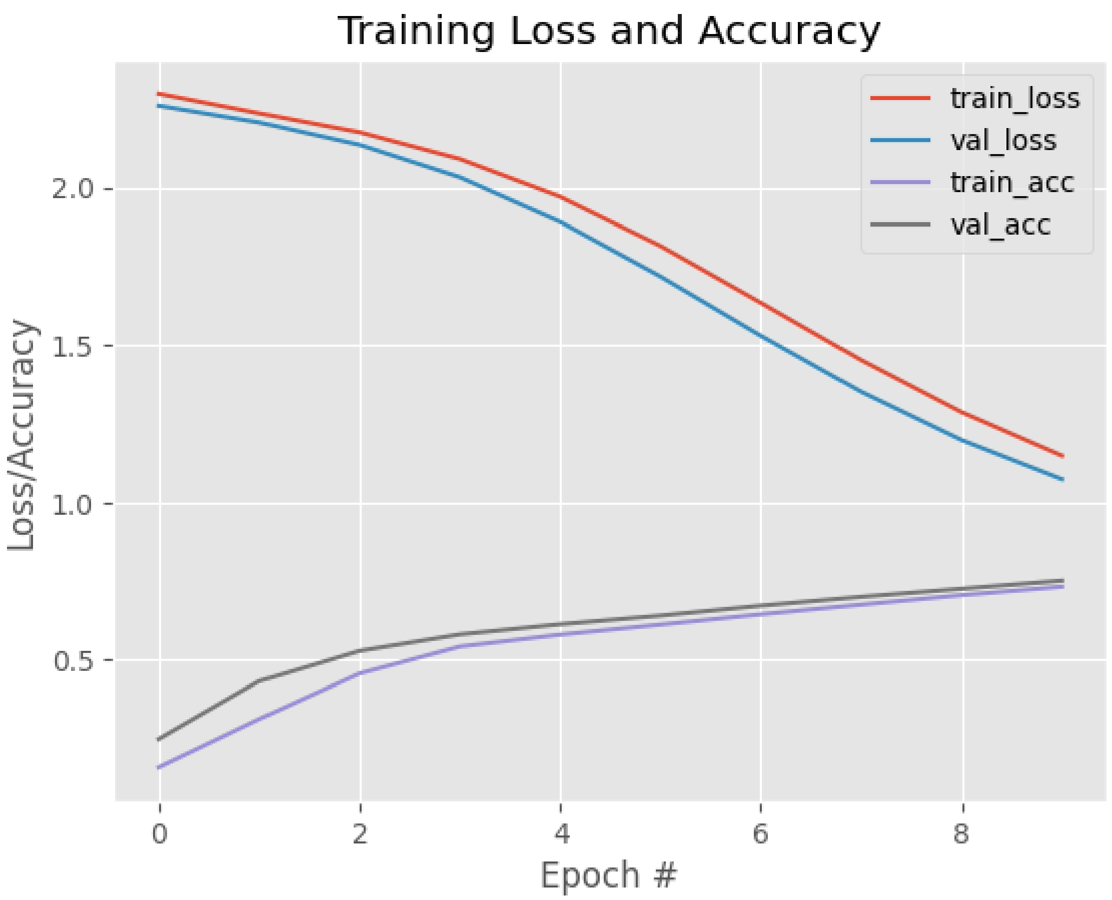
```
79/79 [==============================] - 1s 5ms/step
              precision    recall  f1-score   support

           0       0.81      0.97      0.88       980
           1       0.78      0.99      0.87      1135
           2       0.82      0.69      0.75      1032
           3       0.62      0.90      0.73      1010
           4       0.67      0.80      0.73       982
           5       0.81      0.27      0.41       892
           6       0.84      0.87      0.85       958
           7       0.78      0.87      0.83      1028
           8       0.82      0.55      0.66       974
           9       0.67      0.54      0.60      1009

    accuracy                           0.75     10000
   macro avg       0.76      0.74      0.73     10000
weighted avg       0.76      0.75      0.74     10000
```

In [7]:

```
#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[7]: <matplotlib.legend.Legend at 0x212d6c185e0>



In [ ]: