Assignment 2: Implementing Feedforward neural networks with Keras and TensorFlow

In [ ]:
```
#Tushar Santosh Kokane
#B511066
```

In [2]:
```
#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=["accu
H = model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=epo
```

```
Epoch 1/10
469/469 [==============================] - 9s 16ms/step - loss: 2.2897 - a
ccuracy: 0.1692 - val_loss: 2.2521 - val_accuracy: 0.2864
Epoch 2/10
469/469 [==============================] - 6s 12ms/step - loss: 2.2256 - a
ccuracy: 0.3262 - val_loss: 2.1920 - val_accuracy: 0.4748
Epoch 3/10
469/469 [==============================] - 5s 11ms/step - loss: 2.1567 - a
ccuracy: 0.4901 - val_loss: 2.1095 - val_accuracy: 0.5230
Epoch 4/10
469/469 [==============================] - 5s 11ms/step - loss: 2.0593 - a
ccuracy: 0.5760 - val_loss: 1.9911 - val_accuracy: 0.6251
Epoch 5/10
469/469 [==============================] - 5s 12ms/step - loss: 1.9214 - a
ccuracy: 0.6266 - val_loss: 1.8291 - val_accuracy: 0.6580
Epoch 6/10
469/469 [==============================] - 5s 11ms/step - loss: 1.7453 - a
ccuracy: 0.6672 - val_loss: 1.6372 - val_accuracy: 0.6906
Epoch 7/10
469/469 [==============================] - 5s 11ms/step - loss: 1.5541 - a
ccuracy: 0.6961 - val_loss: 1.4464 - val_accuracy: 0.7194
Epoch 8/10
469/469 [==============================] - 5s 12ms/step - loss: 1.3752 - a
ccuracy: 0.7232 - val_loss: 1.2776 - val_accuracy: 0.7429
Epoch 9/10
469/469 [==============================] - 5s 12ms/step - loss: 1.2225 - a
ccuracy: 0.7451 - val_loss: 1.1384 - val_accuracy: 0.7587
Epoch 10/10
469/469 [==============================] - 5s 11ms/step - loss: 1.0970 - a
ccuracy: 0.7626 - val_loss: 1.0251 - val_accuracy: 0.7720
```

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
```

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

           0       0.83      0.96      0.89       980
           1       0.76      0.99      0.86      1135
           2       0.82      0.64      0.72      1032
           3       0.65      0.85      0.74      1010
           4       0.71      0.83      0.76       982
           5       0.83      0.38      0.52       892
           6       0.85      0.89      0.87       958
           7       0.82      0.88      0.85      1028
           8       0.79      0.63      0.70       974
           9       0.75      0.61      0.67      1009

    accuracy                           0.77     10000
   macro avg       0.78      0.77      0.76     10000
weighted avg       0.78      0.77      0.76     10000
```

In [8]:
```
#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[8]:
```
<matplotlib.legend.Legend at 0x245819a3e50>
```