Assignment 2

Problem statement

Implementing Feedforward neural networks with Keras and TensorFlow

Details

1. Name : Akash Kulkarni
2. Branch : Information Technology
3. Division : BE 10
4. Batch : R-10
5. Roll Number : 43241
6. Course : Laboratory Practice 4 (Deep Learning)

```
#installations
import tensorflow as tf
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
```

```
#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
```

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

```
#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"))
```

```
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, ba
```

```
Epoch 1/10
469/469 [==============================] - 3s 5ms/step - loss: 2.2911 - accur
Epoch 2/10
469/469 [==============================] - 2s 5ms/step - loss: 2.2339 - accur
Epoch 3/10
469/469 [==============================] - 2s 5ms/step - loss: 2.1731 - accur
Epoch 4/10
469/469 [==============================] - 2s 5ms/step - loss: 2.0895 - accur
Epoch 5/10
469/469 [==============================] - 2s 5ms/step - loss: 1.9721 - accur
Epoch 6/10
469/469 [==============================] - 2s 5ms/step - loss: 1.8177 - accur
Epoch 7/10
469/469 [==============================] - 2s 5ms/step - loss: 1.6385 - accur
Epoch 8/10
469/469 [==============================] - 2s 5ms/step - loss: 1.4558 - accur
Epoch 9/10
469/469 [==============================] - 2s 5ms/step - loss: 1.2882 - accur
Epoch 10/10
469/469 [==============================] - 2s 5ms/step - loss: 1.1450 - accur
```

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

```
79/79 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.83      0.96      0.89       980
           1       0.78      0.98      0.87      1135
           2       0.83      0.80      0.82      1032
           3       0.69      0.76      0.73      1010
           4       0.71      0.78      0.74       982
           5       0.87      0.32      0.47       892
           6       0.82      0.90      0.86       958
           7       0.79      0.88      0.83      1028
           8       0.76      0.67      0.71       974
           9       0.69      0.59      0.64      1009

    accuracy                           0.77     10000
   macro avg       0.78      0.76      0.75     10000
weighted avg       0.78      0.77      0.76     10000
```

```
#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()
```

```
<matplotlib.legend.Legend at 0x7ff3310f5a50>
```