Deep Learning Assignment No:2

Title:Implementing Feedforward neural network with Keras and Tensorflow

Name:Pranit Dilip Menkar

BEIT

Roll no: 49

In [ ]:

```
```

In [ ]:

```
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
import argparse as ap
```

In [ ]:

```
print("[INFO] accessing MNIST...")
((trainX, trainY), (testX, testY)) = mnist.load_data()
```

```
[INFO] accessing MNIST...
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.n
pz
11490434/11490434 [==============================] - 0s 0us/step
```

In [ ]:

```
trainX = trainX.reshape((trainX.shape[0], 28 * 28 * 1))
testX = testX.reshape((testX.shape[0], 28 * 28 * 1))
```

In [ ]:

```
trainX = trainX.astype("float32") / 255.0
testX = testX.astype("float32") / 255.0
```

In [ ]:

```
lb = LabelBinarizer()
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)
```

In [ ]:

```
model = Sequential()
model.add(Dense(256, input_shape=(784,), activation="sigmoid"))
model.add(Dense(128, activation="sigmoid"))
model.add(Dense(10, activation="softmax"))
```

In [ ]:

```
print("[INFO] training network...")
sgd = SGD(0.01)
model.compile(loss="categorical_crossentropy", optimizer=sgd,
 metrics=["accuracy"])
H = model.fit(trainX, trainY, validation_data=(testX, testY),
```

```
 epochs=30, batch_size=128)
```

```
[INFO] training network...
Epoch 1/30
469/469 [==============================] - 4s 8ms/step - loss: 2.2853 - accuracy: 0.1891
- val_loss: 2.2428 - val_accuracy: 0.2862
Epoch 2/30
469/469 [==============================] - 3s 7ms/step - loss: 2.2073 - accuracy: 0.4050
- val_loss: 2.1639 - val_accuracy: 0.3989
Epoch 3/30
469/469 [==============================] - 3s 7ms/step - loss: 2.1146 - accuracy: 0.5338
- val_loss: 2.0511 - val_accuracy: 0.5430
Epoch 4/30
469/469 [==============================] - 3s 7ms/step - loss: 1.9803 - accuracy: 0.5965
- val_loss: 1.8871 - val_accuracy: 0.6196
Epoch 5/30
469/469 [==============================] - 3s 7ms/step - loss: 1.7940 - accuracy: 0.6371
- val_loss: 1.6754 - val_accuracy: 0.6695
Epoch 6/30
469/469 [==============================] - 4s 8ms/step - loss: 1.5750 - accuracy: 0.6759
- val_loss: 1.4509 - val_accuracy: 0.6994
Epoch 7/30
469/469 [==============================] - 5s 10ms/step - loss: 1.3637 - accuracy: 0.7108
- val_loss: 1.2545 - val_accuracy: 0.7314
Epoch 8/30
469/469 [==============================] - 3s 7ms/step - loss: 1.1871 - accuracy: 0.7398
- val_loss: 1.0970 - val_accuracy: 0.7593
Epoch 9/30
469/469 [==============================] - 3s 7ms/step - loss: 1.0490 - accuracy: 0.7647
- val_loss: 0.9761 - val_accuracy: 0.7812
Epoch 10/30
469/469 [==============================] - 3s 7ms/step - loss: 0.9416 - accuracy: 0.7848
- val_loss: 0.8805 - val_accuracy: 0.7981
Epoch 11/30
469/469 [==============================] - 3s 7ms/step - loss: 0.8560 - accuracy: 0.7990
- val_loss: 0.8037 - val_accuracy: 0.8067
Epoch 12/30
469/469 [==============================] - 3s 7ms/step - loss: 0.7866 - accuracy: 0.8116
- val_loss: 0.7405 - val_accuracy: 0.8259
Epoch 13/30
469/469 [==============================] - 3s 7ms/step - loss: 0.7292 - accuracy: 0.8224
- val_loss: 0.6878 - val_accuracy: 0.8321
Epoch 14/30
469/469 [==============================] - 3s 7ms/step - loss: 0.6815 - accuracy: 0.8307
- val_loss: 0.6440 - val_accuracy: 0.8398
Epoch 15/30
469/469 [==============================] - 3s 7ms/step - loss: 0.6415 - accuracy: 0.8382
- val_loss: 0.6078 - val_accuracy: 0.8476
Epoch 16/30
469/469 [==============================] - 3s 7ms/step - loss: 0.6078 - accuracy: 0.8450
- val_loss: 0.5770 - val_accuracy: 0.8533
Epoch 17/30
469/469 [==============================] - 3s 7ms/step - loss: 0.5791 - accuracy: 0.8513
- val_loss: 0.5496 - val_accuracy: 0.8575
Epoch 18/30
469/469 [==============================] - 3s 7ms/step - loss: 0.5543 - accuracy: 0.8564
- val_loss: 0.5273 - val_accuracy: 0.8613
Epoch 19/30
469/469 [==============================] - 3s 7ms/step - loss: 0.5331 - accuracy: 0.8602
- val_loss: 0.5065 - val_accuracy: 0.8671
Epoch 20/30
469/469 [==============================] - 3s 7ms/step - loss: 0.5141 - accuracy: 0.8644
- val_loss: 0.4890 - val_accuracy: 0.8711
Epoch 21/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4976 - accuracy: 0.8677
- val_loss: 0.4734 - val_accuracy: 0.8755
Epoch 22/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4828 - accuracy: 0.8713
- val_loss: 0.4593 - val_accuracy: 0.8768
Epoch 23/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4696 - accuracy: 0.8740
- val_loss: 0.4470 - val_accuracy: 0.8806
```

```
Epoch 24/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4577 - accuracy: 0.8769
- val_loss: 0.4360 - val_accuracy: 0.8828
Epoch 25/30
469/469 [==============================] - 4s 9ms/step - loss: 0.4468 - accuracy: 0.8795
- val_loss: 0.4260 - val_accuracy: 0.8855
Epoch 26/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4371 - accuracy: 0.8821
- val_loss: 0.4161 - val_accuracy: 0.8880
Epoch 27/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4281 - accuracy: 0.8844
- val_loss: 0.4079 - val_accuracy: 0.8889
Epoch 28/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4199 - accuracy: 0.8862
- val_loss: 0.4009 - val_accuracy: 0.8913
Epoch 29/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4124 - accuracy: 0.8877
- val_loss: 0.3937 - val_accuracy: 0.8921
Epoch 30/30
469/469 [==============================] - 3s 7ms/step - loss: 0.4055 - accuracy: 0.8893
- val_loss: 0.3867 - val_accuracy: 0.8942
```

In [ ]:

```
print("[INFO] evaluating network...")
predictions = model.predict(testX, batch_size=128)
print(classification_report(testY.argmax(axis=1),
 predictions.argmax(axis=1),
 target_names=[str(x) for x in lb.classes_]))
```

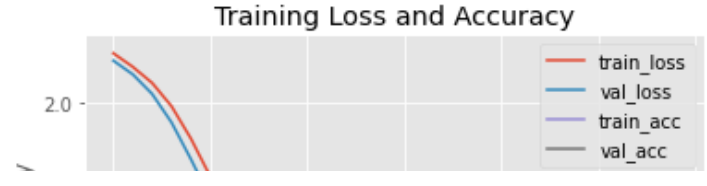
```
[INFO] evaluating network...
79/79 [==============================] - 1s 5ms/step
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       980
           1       0.94      0.97      0.96      1135
           2       0.91      0.86      0.88      1032
           3       0.88      0.89      0.89      1010
           4       0.87      0.91      0.89       982
           5       0.85      0.81      0.83       892
           6       0.91      0.92      0.91       958
           7       0.91      0.89      0.90      1028
           8       0.86      0.84      0.85       974
           9       0.87      0.85      0.86      1009

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

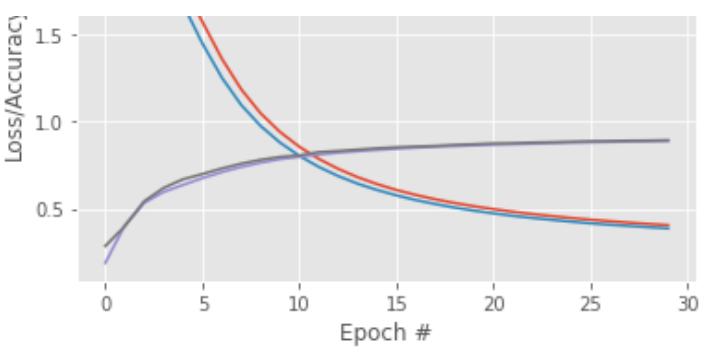
In [ ]:

```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 30), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 30), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 30), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 30), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7fabbfe114d0>
```





In [ ]: