```
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

print("[INFO] accessing MNIST...")
```

```
[INFO] accessing MNIST...
```

```
((trainX, trainY), (testX, testY)) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

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


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


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


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

```
model.add(Dense(10, activation="softmax"))


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=100, batch_size=128)
```

```
 [INFO] training network...
 Epoch 1/100
 469/469 [==============================] - 6s 12ms/step - loss: 2.2731 - accuracy:
 Epoch 2/100
 469/469 [==============================] - 4s 8ms/step - loss: 2.2046 - accuracy: (
 Epoch 3/100
 469/469 [==============================] - 4s 8ms/step - loss: 2.1174 - accuracy: (
 Epoch 4/100
 469/469 [==============================] - 3s 7ms/step - loss: 1.9922 - accuracy: (
 Epoch 5/100
 469/469 [==============================] - 3s 7ms/step - loss: 1.8154 - accuracy: (
 Epoch 6/100
 469/469 [==============================] - 4s 8ms/step - loss: 1.5958 - accuracy: (
 Epoch 7/100
 469/469 [==============================] - 3s 7ms/step - loss: 1.3709 - accuracy: (
 Epoch 8/100
 469/469 [==============================] - 3s 7ms/step - loss: 1.1774 - accuracy: (
 Epoch 9/100
 469/469 [==============================] - 3s 7ms/step - loss: 1.0269 - accuracy: (
 Epoch 10/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.9132 - accuracy: (
 Epoch 11/100
 469/469 [==============================] - 4s 8ms/step - loss: 0.8261 - accuracy: (
 Epoch 12/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.7580 - accuracy: (
 Epoch 13/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.7035 - accuracy: (
 Epoch 14/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.6590 - accuracy: (
 Epoch 15/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.6221 - accuracy: (
 Epoch 16/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.5912 - accuracy: (
 Epoch 17/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.5648 - accuracy: (
 Epoch 18/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.5422 - accuracy: (
 Epoch 19/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.5226 - accuracy: (
 Epoch 20/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.5052 - accuracy: (
 Epoch 21/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.4900 - accuracy: (
 Epoch 22/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.4763 - accuracy: (
 Epoch 23/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.4640 - accuracy: (
 Epoch 24/100
 469/469 [==============================] - 3s 7ms/step - loss: 0.4530 - accuracy: (
```

```
Epoch 25/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4431 - accuracy: 
Epoch 26/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4340 - accuracy: 
Epoch 27/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4257 - accuracy: 
Epoch 28/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4181 - accuracy:
```

```
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...
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       980
           1       0.97      0.98      0.97      1135
           2       0.93      0.90      0.91      1032
           3       0.90      0.91      0.91      1010
           4       0.92      0.93      0.93       982
           5       0.91      0.86      0.88       892
           6       0.93      0.95      0.94       958
           7       0.93      0.92      0.93      1028
           8       0.89      0.89      0.89       974
           9       0.90      0.91      0.91      1009

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 100), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 100), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 100), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 100), 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 0x7fc2e8febfd0>
```