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

```
[INFO] accessing MNIST...
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 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 [==============================] - 5s 9ms/step - loss: 2.2745 - accuracy: 0.2052 - val_loss: 2.2362 - val_accuracy: 0.
Epoch 2/100
469/469 [==============================] - 6s 13ms/step - loss: 2.2033 - accuracy: 0.4202 - val_loss: 2.1610 - val_accuracy: 0
Epoch 3/100
469/469 [==============================] - 7s 14ms/step - loss: 2.1168 - accuracy: 0.5341 - val_loss: 2.0564 - val_accuracy: 0
Epoch 4/100
469/469 [==============================] - 6s 13ms/step - loss: 1.9923 - accuracy: 0.5907 - val_loss: 1.9051 - val_accuracy: 0
Epoch 5/100
469/469 [==============================] - 5s 10ms/step - loss: 1.8196 - accuracy: 0.6321 - val_loss: 1.7064 - val_accuracy: 0
Epoch 6/100
469/469 [==============================] - 3s 7ms/step - loss: 1.6107 - accuracy: 0.6739 - val_loss: 1.4864 - val_accuracy: 0.
Epoch 7/100
469/469 [==============================] - 4s 8ms/step - loss: 1.3978 - accuracy: 0.7168 - val_loss: 1.2801 - val_accuracy: 0.
Epoch 8/100
469/469 [==============================] - 4s 8ms/step - loss: 1.2090 - accuracy: 0.7477 - val_loss: 1.1074 - val_accuracy: 0.
Epoch 9/100
469/469 [==============================] - 4s 8ms/step - loss: 1.0551 - accuracy: 0.7740 - val_loss: 0.9720 - val_accuracy: 0.
Epoch 10/100
469/469 [==============================] - 3s 7ms/step - loss: 0.9344 - accuracy: 0.7927 - val_loss: 0.8656 - val_accuracy: 0.
Epoch 11/100
469/469 [==============================] - 4s 7ms/step - loss: 0.8409 - accuracy: 0.8070 - val_loss: 0.7834 - val_accuracy: 0.
Epoch 12/100
469/469 [==============================] - 4s 8ms/step - loss: 0.7681 - accuracy: 0.8188 - val_loss: 0.7191 - val_accuracy: 0.
Epoch 13/100
469/469 [==============================] - 5s 10ms/step - loss: 0.7104 - accuracy: 0.8274 - val_loss: 0.6676 - val_accuracy: 0
Epoch 14/100
469/469 [==============================] - 5s 11ms/step - loss: 0.6638 - accuracy: 0.8348 - val_loss: 0.6260 - val_accuracy: 0
Epoch 15/100
469/469 [==============================] - 4s 8ms/step - loss: 0.6259 - accuracy: 0.8415 - val_loss: 0.5916 - val_accuracy: 0.
Epoch 16/100
469/469 [==============================] - 3s 7ms/step - loss: 0.5943 - accuracy: 0.8474 - val_loss: 0.5630 - val_accuracy: 0.
Epoch 17/100
469/469 [==============================] - 4s 8ms/step - loss: 0.5674 - accuracy: 0.8530 - val_loss: 0.5386 - val_accuracy: 0.
Epoch 18/100
469/469 [==============================] - 3s 7ms/step - loss: 0.5443 - accuracy: 0.8570 - val_loss: 0.5180 - val_accuracy: 0.
Epoch 19/100
469/469 [==============================] - 4s 8ms/step - loss: 0.5243 - accuracy: 0.8613 - val_loss: 0.4988 - val_accuracy: 0.
Epoch 20/100
469/469 [==============================] - 4s 7ms/step - loss: 0.5066 - accuracy: 0.8653 - val_loss: 0.4827 - val_accuracy: 0.
Epoch 21/100
```

```
469/469 [==============================] - 3s 7ms/step - loss: 0.4911 - accuracy: 0.8689 - val_loss: 0.4681 - val_accuracy: 0.
Epoch 22/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4770 - accuracy: 0.8716 - val_loss: 0.4550 - val_accuracy: 0.
Epoch 23/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4645 - accuracy: 0.8747 - val_loss: 0.4431 - val_accuracy: 0.
Epoch 24/100
469/469 [==============================] - 4s 8ms/step - loss: 0.4533 - accuracy: 0.8775 - val_loss: 0.4328 - val_accuracy: 0.
Epoch 25/100
469/469 [==============================] - 4s 8ms/step - loss: 0.4431 - accuracy: 0.8801 - val_loss: 0.4235 - val_accuracy: 0.
Epoch 26/100
469/469 [==============================] - 4s 7ms/step - loss: 0.4337 - accuracy: 0.8824 - val_loss: 0.4150 - val_accuracy: 0.
Epoch 27/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4253 - accuracy: 0.8843 - val_loss: 0.4070 - val_accuracy: 0.
Epoch 28/100
469/469 [==============================] - 4s 8ms/step - loss: 0.4175 - accuracy: 0.8859 - val_loss: 0.3993 - val_accuracy: 0
```

```
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_]))
```

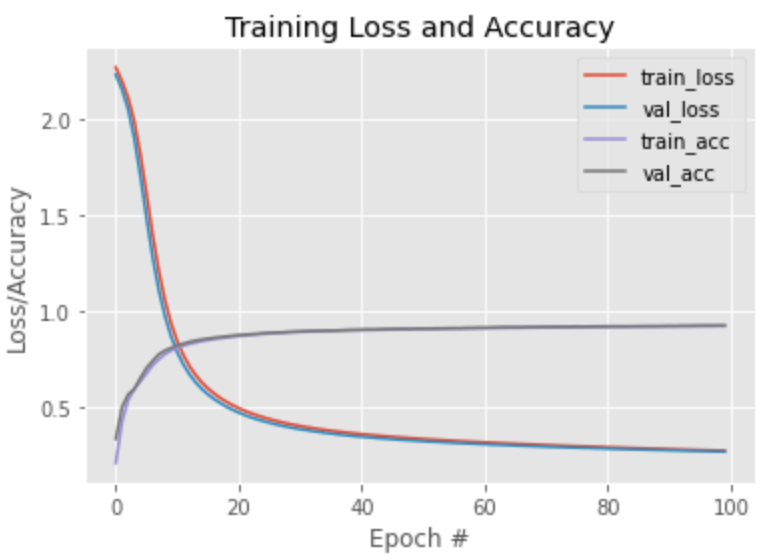
```
[INFO] evaluating network...
79/79 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       980
           1       0.97      0.98      0.97      1135
           2       0.92      0.91      0.92      1032
           3       0.90      0.91      0.91      1010
           4       0.92      0.93      0.93       982
           5       0.90      0.87      0.88       892
           6       0.93      0.94      0.94       958
           7       0.93      0.92      0.93      1028
           8       0.90      0.88      0.89       974
           9       0.91      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 0x7f8ccf8b1950>
```



Colab paid products - Cancel contracts here