```python
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

```

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

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
```

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

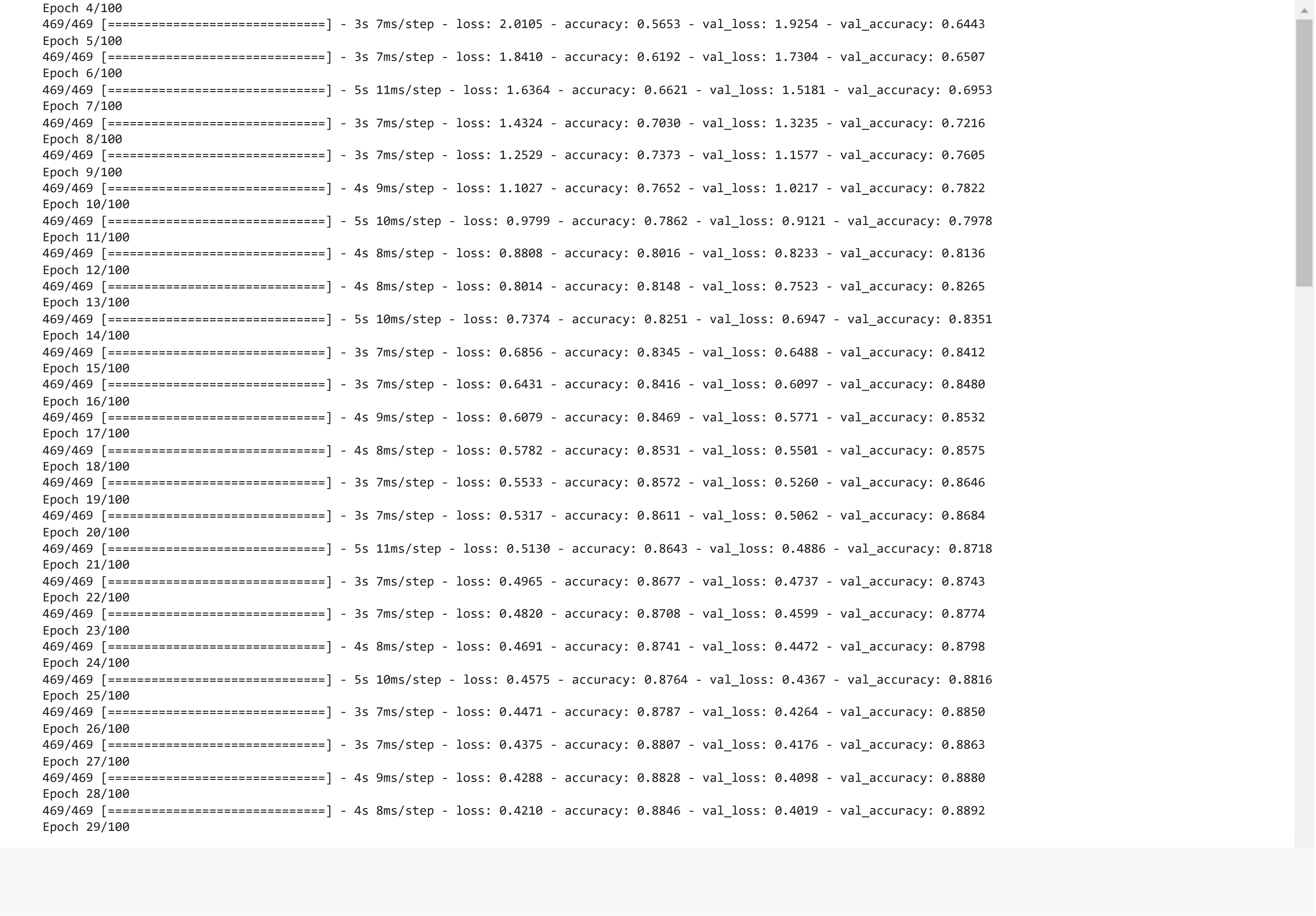
```
Start coding or generate with AI.
```

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

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

```python
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 11ms/step - loss: 2.2901 - accuracy: 0.1797 - val_loss: 2.2501 - val_accuracy: 0.2269
Epoch 2/100
469/469 [==============================] - 5s 10ms/step - loss: 2.2163 - accuracy: 0.3625 - val_loss: 2.1756 - val_accuracy: 0.4378
Epoch 3/100
469/469 [==============================] - 5s 10ms/step - loss: 2.1317 - accuracy: 0.4981 - val_loss: 2.0731 - val_accuracy: 0.4991
```



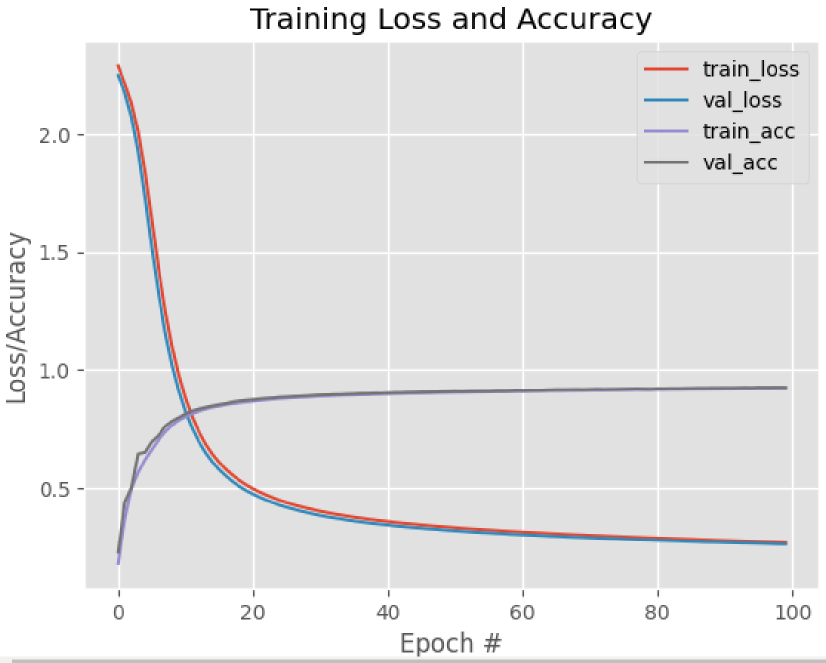
```
Epoch 4/100
469/469 [==============================] - 3s 7ms/step - loss: 2.0105 - accuracy: 0.5653 - val_loss: 1.9254 - val_accuracy: 0.6443
Epoch 5/100
469/469 [==============================] - 3s 7ms/step - loss: 1.8410 - accuracy: 0.6192 - val_loss: 1.7304 - val_accuracy: 0.6507
Epoch 6/100
469/469 [==============================] - 5s 11ms/step - loss: 1.6364 - accuracy: 0.6621 - val_loss: 1.5181 - val_accuracy: 0.6953
Epoch 7/100
469/469 [==============================] - 3s 7ms/step - loss: 1.4324 - accuracy: 0.7030 - val_loss: 1.3235 - val_accuracy: 0.7216
Epoch 8/100
469/469 [==============================] - 3s 7ms/step - loss: 1.2529 - accuracy: 0.7373 - val_loss: 1.1577 - val_accuracy: 0.7605
Epoch 9/100
469/469 [==============================] - 4s 9ms/step - loss: 1.1027 - accuracy: 0.7652 - val_loss: 1.0217 - val_accuracy: 0.7822
Epoch 10/100
469/469 [==============================] - 5s 10ms/step - loss: 0.9799 - accuracy: 0.7862 - val_loss: 0.9121 - val_accuracy: 0.7978
Epoch 11/100
469/469 [==============================] - 4s 8ms/step - loss: 0.8808 - accuracy: 0.8016 - val_loss: 0.8233 - val_accuracy: 0.8136
Epoch 12/100
469/469 [==============================] - 4s 8ms/step - loss: 0.8014 - accuracy: 0.8148 - val_loss: 0.7523 - val_accuracy: 0.8265
Epoch 13/100
469/469 [==============================] - 5s 10ms/step - loss: 0.7374 - accuracy: 0.8251 - val_loss: 0.6947 - val_accuracy: 0.8351
Epoch 14/100
469/469 [==============================] - 3s 7ms/step - loss: 0.6856 - accuracy: 0.8345 - val_loss: 0.6488 - val_accuracy: 0.8412
Epoch 15/100
469/469 [==============================] - 3s 7ms/step - loss: 0.6431 - accuracy: 0.8416 - val_loss: 0.6097 - val_accuracy: 0.8480
Epoch 16/100
469/469 [==============================] - 4s 9ms/step - loss: 0.6079 - accuracy: 0.8469 - val_loss: 0.5771 - val_accuracy: 0.8532
Epoch 17/100
469/469 [==============================] - 4s 8ms/step - loss: 0.5782 - accuracy: 0.8531 - val_loss: 0.5501 - val_accuracy: 0.8575
Epoch 18/100
469/469 [==============================] - 3s 7ms/step - loss: 0.5533 - accuracy: 0.8572 - val_loss: 0.5260 - val_accuracy: 0.8646
Epoch 19/100
469/469 [==============================] - 3s 7ms/step - loss: 0.5317 - accuracy: 0.8611 - val_loss: 0.5062 - val_accuracy: 0.8684
Epoch 20/100
469/469 [==============================] - 5s 11ms/step - loss: 0.5130 - accuracy: 0.8643 - val_loss: 0.4886 - val_accuracy: 0.8718
Epoch 21/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4965 - accuracy: 0.8677 - val_loss: 0.4737 - val_accuracy: 0.8743
Epoch 22/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4820 - accuracy: 0.8708 - val_loss: 0.4599 - val_accuracy: 0.8774
Epoch 23/100
469/469 [==============================] - 4s 8ms/step - loss: 0.4691 - accuracy: 0.8741 - val_loss: 0.4472 - val_accuracy: 0.8798
Epoch 24/100
469/469 [==============================] - 5s 10ms/step - loss: 0.4575 - accuracy: 0.8764 - val_loss: 0.4367 - val_accuracy: 0.8816
Epoch 25/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4471 - accuracy: 0.8787 - val_loss: 0.4264 - val_accuracy: 0.8850
Epoch 26/100
469/469 [==============================] - 3s 7ms/step - loss: 0.4375 - accuracy: 0.8807 - val_loss: 0.4176 - val_accuracy: 0.8863
Epoch 27/100
469/469 [==============================] - 4s 9ms/step - loss: 0.4288 - accuracy: 0.8828 - val_loss: 0.4098 - val_accuracy: 0.8880
Epoch 28/100
469/469 [==============================] - 4s 8ms/step - loss: 0.4210 - accuracy: 0.8846 - val_loss: 0.4019 - val_accuracy: 0.8892
Epoch 29/100
```

```
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       0.93      0.90      0.92      1032
           3       0.90      0.91      0.91      1010
           4       0.92      0.94      0.93       982
           5       0.90      0.87      0.89       892
           6       0.93      0.95      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                           0.93     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.93      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 0x7d677d9e8af0>
```



```
1 Start coding or generate with AI.
```