```
#import the necessary packages
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 cifar10
import matplotlib.pyplot as plt
import numpy as np
```

```
# load the training and testing data, scale it into the range [0, 1],
# then reshape the design matrix
print("[INFO] loading CIFAR-10 data...")
((trainX, trainY), (testX, testY)) = cifar10.load_data()

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

trainX = trainX.reshape((trainX.shape[0], 3072))
testX = testX.reshape((testX.shape[0], 3072))
```

```
[INFO] loading CIFAR-10 data...
```

```
# convert the labels from integers to vectors
lb = LabelBinarizer()
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)
# initialize the label names for the CIFAR-10 dataset
labelNames = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
model = Sequential()
model.add(Dense(1024, input_shape=(3072,), activation="relu"))
model.add(Dense(512, activation="relu"))
model.add(Dense(10, activation="softmax"))
```

```
# train the model using SGD
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=50, batch_size=32)
```

```
1563/1563 [==============================] - 40s 26ms/step - loss: 0.6257 - accuracy: 0.7863 - val_loss: 1.4827 - val_accuracy: 0
Epoch 42/50
1563/1563 [==============================] - 41s 27ms/step - loss: 0.6044 - accuracy: 0.7946 - val_loss: 1.4171 - val_accuracy: 0
Epoch 43/50
1563/1563 [==============================] - 42s 27ms/step - loss: 0.5904 - accuracy: 0.7988 - val_loss: 1.5315 - val_accuracy: 0
Epoch 44/50
1563/1563 [==============================] - 44s 28ms/step - loss: 0.5666 - accuracy: 0.8083 - val_loss: 1.4514 - val_accuracy: 0
Epoch 45/50
1563/1563 [==============================] - 43s 27ms/step - loss: 0.5514 - accuracy: 0.8143 - val_loss: 1.6072 - val_accuracy: 0
Epoch 46/50
1563/1563 [==============================] - 43s 27ms/step - loss: 0.5311 - accuracy: 0.8199 - val_loss: 1.5450 - val_accuracy: 0
Epoch 47/50
1563/1563 [==============================] - 40s 26ms/step - loss: 0.5134 - accuracy: 0.8284 - val_loss: 1.5363 - val_accuracy: 0
Epoch 48/50
1563/1563 [==============================] - 45s 29ms/step - loss: 0.4946 - accuracy: 0.8350 - val_loss: 1.4619 - val_accuracy: 0
Epoch 49/50
1563/1563 [==============================] - 41s 26ms/step - loss: 0.4772 - accuracy: 0.8398 - val_loss: 1.7001 - val_accuracy: 0
Epoch 50/50
1563/1563 [==============================] - 40s 26ms/step - loss: 0.4610 - accuracy: 0.8451 - val_loss: 1.6969 - val_accuracy: 0
```

```
# evaluate the network
print("[INFO] evaluating network...")
predictions = model.predict(testX, batch_size=32)
print(classification_report(testY.argmax(axis=1),predictions.argmax(axis=1), target_names=labelNames))
```

```
[INFO] evaluating network...
313/313 [==============================] - 3s 11ms/step
              precision    recall  f1-score   support

    airplane       0.61      0.65      0.63      1000
  automobile       0.81      0.49      0.61      1000
        bird       0.55      0.29      0.38      1000
         cat       0.40      0.32      0.35      1000
        deer       0.36      0.71      0.47      1000
         dog       0.47      0.39      0.42      1000
        frog       0.74      0.45      0.56      1000
       horse       0.48      0.72      0.58      1000
        ship       0.53      0.79      0.63      1000
       truck       0.73      0.44      0.55      1000

    accuracy                           0.52     10000
   macro avg       0.57      0.52      0.52     10000
weighted avg       0.57      0.52      0.52     10000
```

```
# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(50, 100), H.history["loss"], label="train_loss")
plt.plot(np.arange(50, 100), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(50, 100), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(50, 100), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig(args["output.png"])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-b4335a7655ed> in <cell line: 12>()
     10 plt.ylabel("Loss/Accuracy")
     11 plt.legend()
---> 12 plt.savefig(args["output.png"])

NameError: name 'args' is not defined
```

SEARCH STACK OVERFLOW