```
import numpy as np
import pickle
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD
from keras.datasets import cifar10
from keras.utils.np_utils import to_categorical
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

```
Using TensorFlow backend.
```

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 28s 0us/step
```

```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(50000, 32, 32, 3)
(50000, 10)
(10000, 32, 32, 3)
(10000, 10)
```

```
#We will use two convolutional layers, each with 32 filters a kernel size of (3,3) and ReLU activat
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=sgd)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/c
Instructions for updating:
Colocations handled automatically by placer.
```

```
history = model.fit(X_train, y_train, batch_size=32, epochs=15, verbose=2, validation_split=0.2)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_op
Instructions for updating:
Use tf.cast instead.
Train on 40000 samples, validate on 10000 samples
Epoch 1/15
 - 15s - loss: 1.5183 - acc: 0.4535 - val_loss: 1.2322 - val_acc: 0.5676
Epoch 2/15
 - 10s - loss: 1.0773 - acc: 0.6185 - val_loss: 1.0434 - val_acc: 0.6323
Epoch 3/15
 - 10s - loss: 0.8475 - acc: 0.6998 - val_loss: 1.0072 - val_acc: 0.6507
Epoch 4/15
 - 10s - loss: 0.6476 - acc: 0.7728 - val_loss: 1.0114 - val_acc: 0.6728
Epoch 5/15
 - 10s - loss: 0.4495 - acc: 0.8434 - val_loss: 1.1563 - val_acc: 0.6653
Epoch 6/15
 - 10s - loss: 0.2862 - acc: 0.9014 - val_loss: 1.3334 - val_acc: 0.6616
Epoch 7/15
 - 11s - loss: 0.1985 - acc: 0.9306 - val_loss: 1.5633 - val_acc: 0.6543
Epoch 8/15
 - 11s - loss: 0.1529 - acc: 0.9476 - val_loss: 1.7882 - val_acc: 0.6474
Epoch 9/15
 - 11s - loss: 0.1266 - acc: 0.9576 - val_loss: 1.8989 - val_acc: 0.6609
Epoch 10/15
 - 11s - loss: 0.1124 - acc: 0.9624 - val_loss: 2.0340 - val_acc: 0.6455
Epoch 11/15
 - 11s - loss: 0.0882 - acc: 0.9699 - val_loss: 2.0463 - val_acc: 0.6518
Epoch 12/15
 - 11s - loss: 0.0777 - acc: 0.9742 - val_loss: 2.2374 - val_acc: 0.6449
Epoch 13/15
 - 11s - loss: 0.0723 - acc: 0.9760 - val_loss: 2.3876 - val_acc: 0.6523
Epoch 14/15
```

```
score = model.evaluate(X_test, y_test, batch_size=128, verbose=0)
```

```
 - 11s - loss: 0.0711 - acc: 0.9768 - val_loss: 2.3769 - val_acc: 0.6554
```

```
print(model.metrics_names)
print(score)
```

```
['loss', 'acc']
[2.411338766479492, 0.6478]
```