МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографии»

Студент гр. 7383 _____ Левкович Д.В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

Распознавание объектов на фотографиях (Object Recognition in Photographs) CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

Задачи.

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Ход работы.

1. Изначально была построена и обучена базовая сеть (код представлен в приложении А) с размером batch_size = 256, количеством эпох epochs = 12 и размером ядра свертки 3x3. Результаты представлены на рис. 1-2.

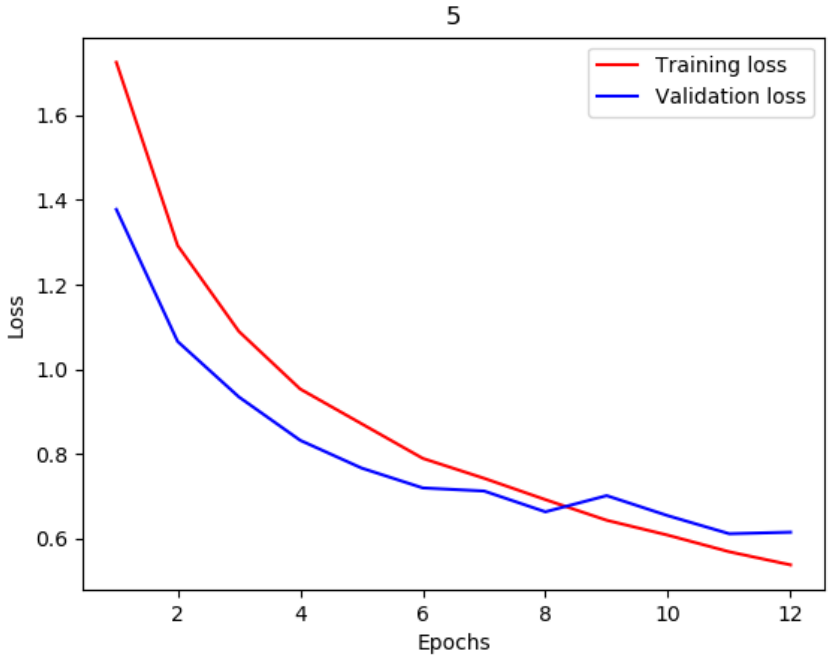

Рисунок 1 - График потерь на базовой модели

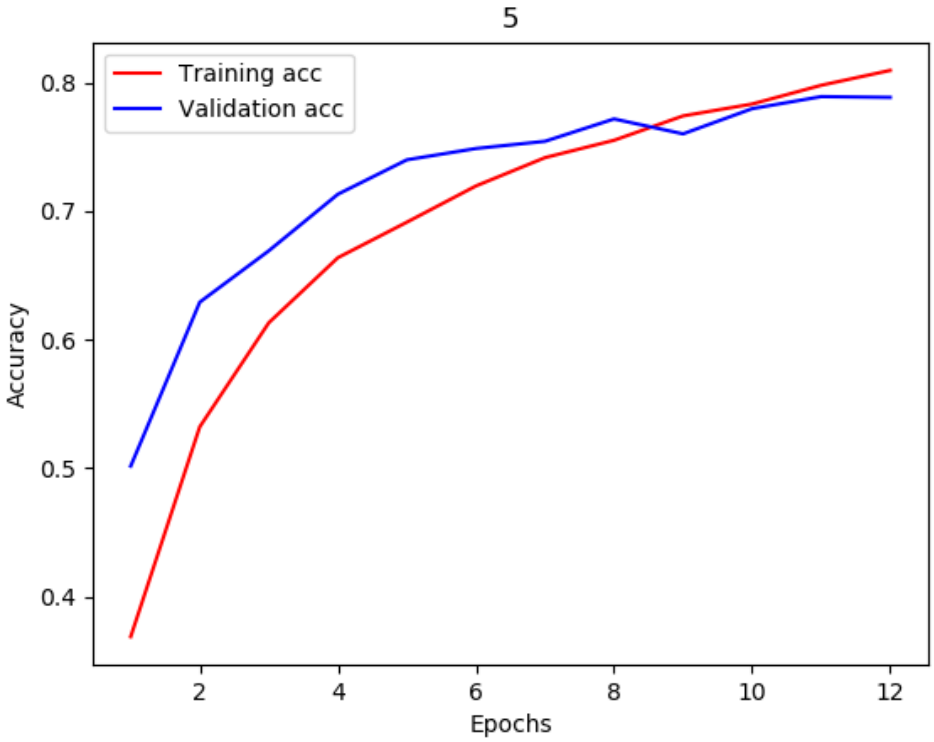

Рисунок 2 - График точности на базовой модели

Как видно из графиков, точность составила около 0.78.

2. Было проведено тестирование влияния слоя Dropout на результат обучения нейронной сети. Результаты обучения сети с выключенным слоем Dropout и размером ядра свертки 3x3 представлен на рис. 3-4.

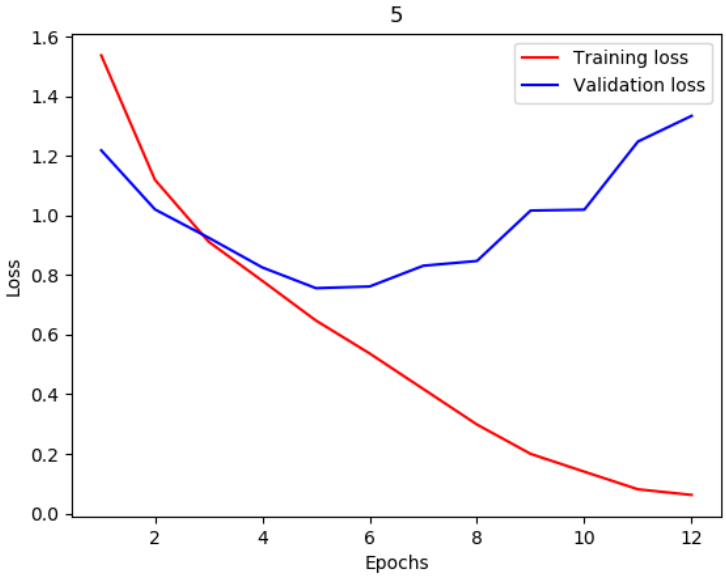

Рисунок 3 - График потерь на базовой модели без слоя Dropout

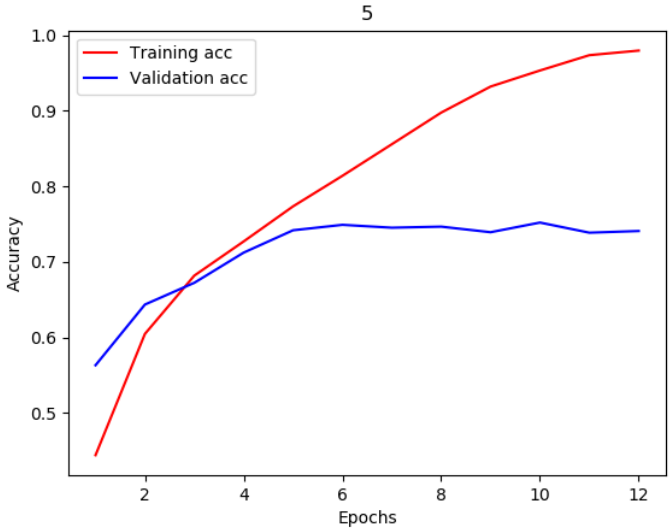

Рисунок 4 - График точности на базовой модели без слоя Dropout

Как видно из графиков, точность на тренировочных данных растет, однако на тестовых данных начиная с пятой эпохи перестает улучшаться, поэтому было принято решение в дальнейшем использовать слой Dropout.

3. Далее исследуем работы сети при различных размерах ядра свертки. Все остальные параметры идентичны модели из первого пункта.

На рис. 5-6 представлены точность и потери сети с размером ядра свертки 2x2.

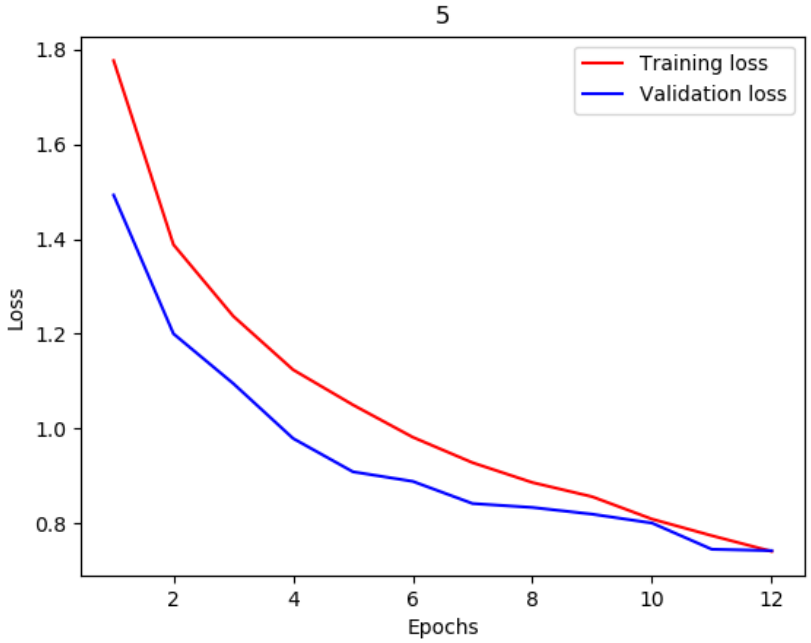

Рисунок 5 - График потерь с размером ядра свертки 2x2

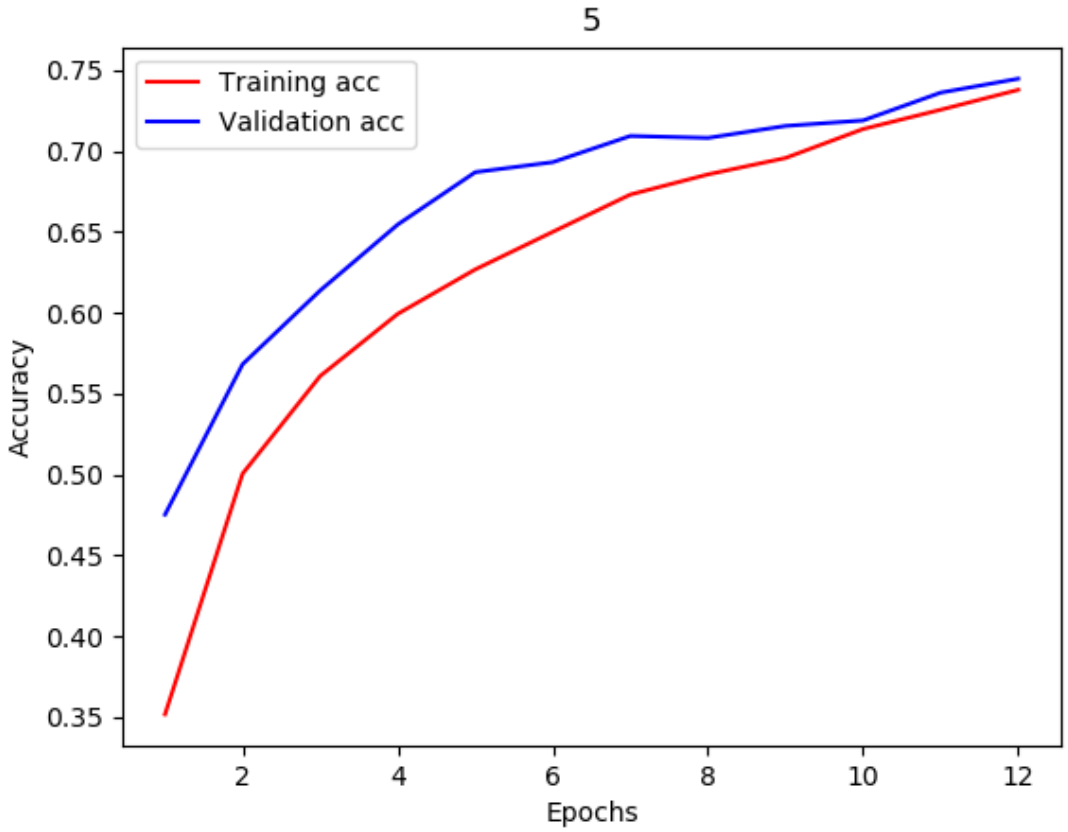

Рисунок 6 - График точности с размером ядра свертки 2x2

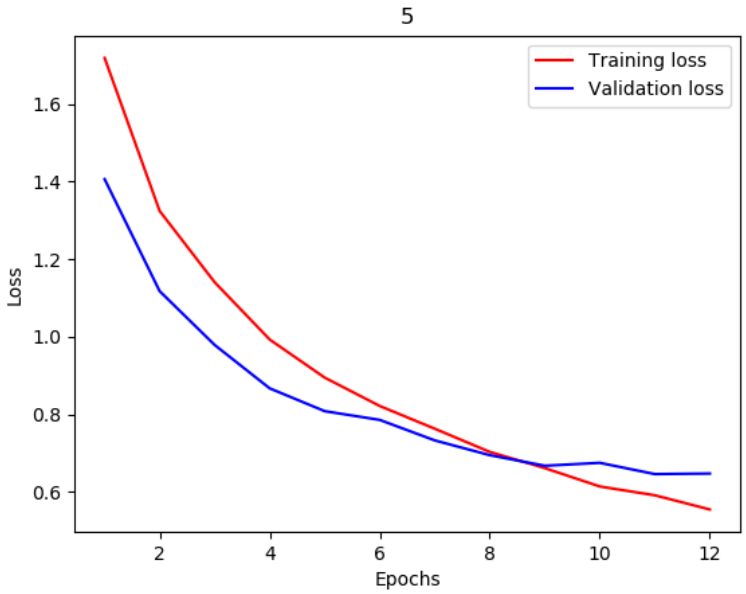

Рисунок 7 - График потерь с размером ядра свертки 5x5

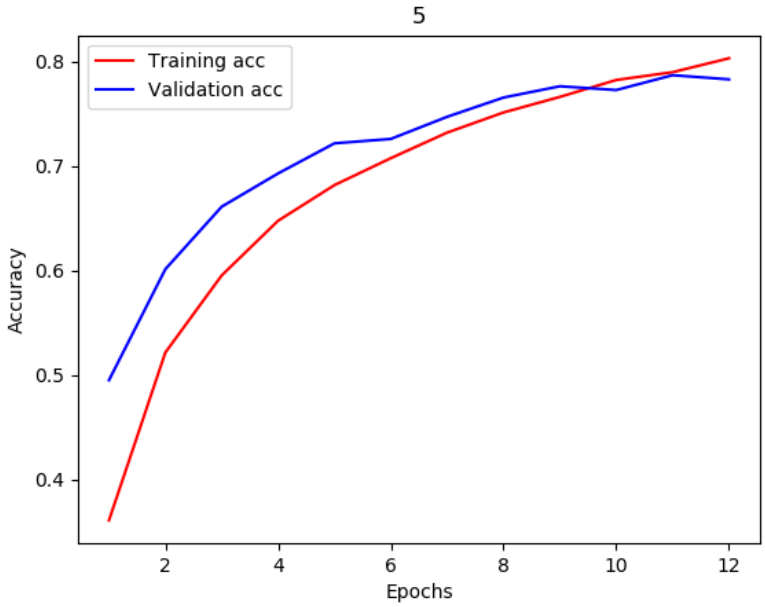

Рисунок 7 - График точности с размером ядра свертки 5x5

Далее проверим с размерами ядра свертки 7x7. Результаты представлены на рис. 9-10.

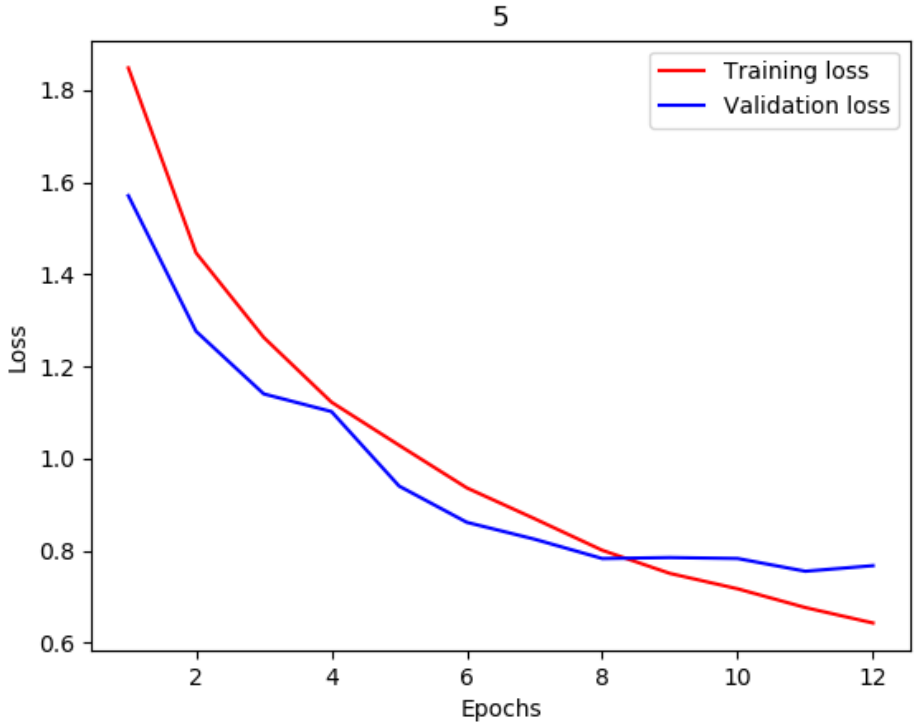

Рисунок 9 - График потерь с размером ядра свертки 7x7

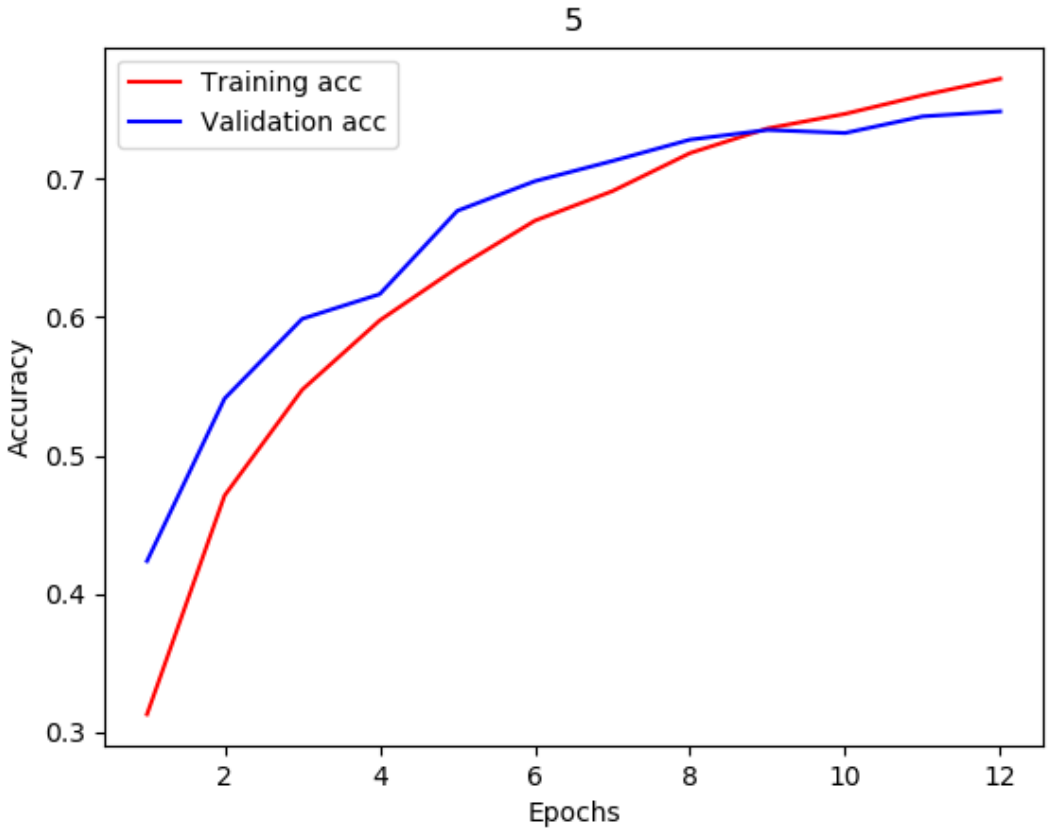

Рисунок 10 - График потерь с размером ядра свертки 7x7

С увеличением размера (7x7) ядра свертки точность упала несильно (на ~3-5%), а ошибка несильно увеличилась (на > 3-5%), относительно свертки 3x3. Можно отметить, что при увеличении размеров ядра свертки увеличивается время обучения.

Выводы.

В ходе выполнения данной лабораторной работы была построена модель искусственной нейронной сети, распознающей объекты на изображениях, было изучено влияние слоя Dropout на обучение сети. Данный слой помогает бороться с переобучением. Было изучено влияние размера слоя свертки на результат обучения сети. При увеличении размера слоя свертки возрастает время обучения сети.

ПРИЛОЖЕНИЕ А

```
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D,
Dense, Dropout, Flatten
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt

batch_size = 256
num_epochs = 12
kernel_size = 3
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

(X_train, y_train), (X_test, y_test) = cifar10.load_data() #
fetch CIFAR-10 data
num_train, depth, height, width = X_train.shape # there are
50000 training examples in CIFAR-10
num_test = X_test.shape[0] # there are 10000 test examples in
CIFAR-10
num_classes = np.unique(y_train).shape[0] # there are 10 image
classes

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range
Y_train = np_utils.to_categorical(y_train, num_classes) # One-
hot encode the labels
Y_test = np_utils.to_categorical(y_test, num_classes) # One-hot
encode the labels


inp = Input(shape=(depth, height, width)) # N.B. depth goes
first in Keras


# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling
layer)

conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
```

```
drop_1 = Dropout(drop_prob_1)(pool_1)
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling
layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply Dense -> ReLU (with dropout) ->
softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(input=inp, output=out) # To define a model, just
specify its input and output layers
model.compile(loss='categorical_crossentropy', # using the
cross-entropy loss function
              optimizer='adam', # using the Adam optimiser
              metrics=['accuracy']) # reporting the accuracy
H = model.fit(X_train, Y_train, # Train the model using the
training set...
          batch_size=batch_size, nb_epoch=num_epochs,
          verbose=1, validation_split=0.1) # ...holding out 10%
of the data for validation
model.evaluate(X_test, Y_test, verbose=1) # Evaluate the
trained model on the test set!

loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title("5")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title("5")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```