

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

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Искусственные нейронные сети»
Тема: Распознавание объектов на фотографиях

Студентка гр. 7382

Еременко А.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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).

Порядок выполнения работы.

1. Ознакомиться со сверточными нейронными сетями.
2. Изучить построение модели в Keras в функциональном виде.
3. Изучить работу слоя разреживания (Dropout).

Требования к выполнению задания.

1. Построить и обучить сверточную нейронную сеть.
2. Исследовать работу сеть без слоя Dropout.
3. Исследовать работу сети при разных размерах ядра свертки.

Основные теоретические положения.

Проблема автоматической идентификации объектов на фотографиях является сложной из-за почти бесконечного количества перестановок объектов, положений, освещения и так далее.

Набор данных CIFAR-10 состоит из 60000 фотографий, разделенных на 10 классов (отсюда и название CIFAR-10). Классы включают в себя общие объекты, такие как самолеты, автомобили, птицы, кошки и так далее. Набор данных разделяется стандартным способом, где 50 000 изображений используются для обучения модели, а остальные 10 000 - для оценки ее производительности.

Фотографии цветные, с красными, зелеными и синими компонентами, но маленькие, размером 32 на 32 пикселя.

Ход работы.

Была построена сверточная нейронная сеть. Код предоставлен в приложении А.

1. Архитектура:
 - Оптимизатор adam.
 - Скорость обучения = 0.001.
 - Epochs = 15, batch_size = 100, loss = categorical_crossentropy
 - Модель:

```
inp = Input(shape=(depth, height, width)) # N.B. depth goes first in Keras
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_3 = Convolution2D(CONV_DEPTH_2, 3, 3, border_mode='same', activation='relu')(drop_1)
conv_4 = Convolution2D(CONV_DEPTH_2, 3, 3, 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)
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)
```

Данная архитектура дает точность ~ 85%. Графики точности и ошибки предоставлены на рис. 1 и рис. 2 соответственно.

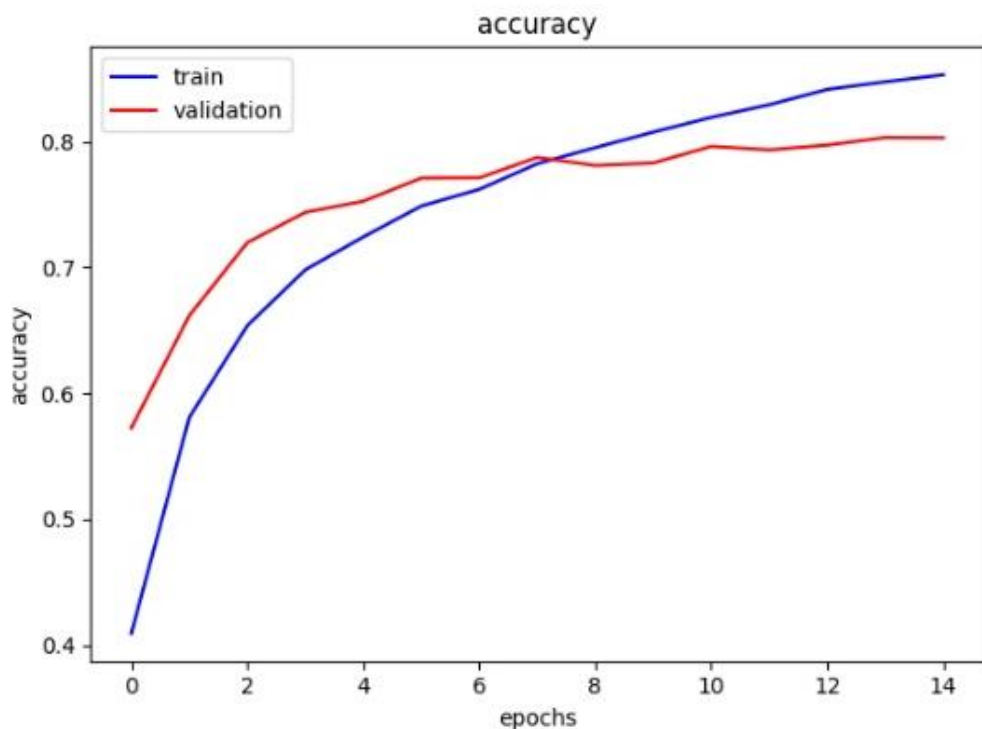


Рисунок 1 – График точности

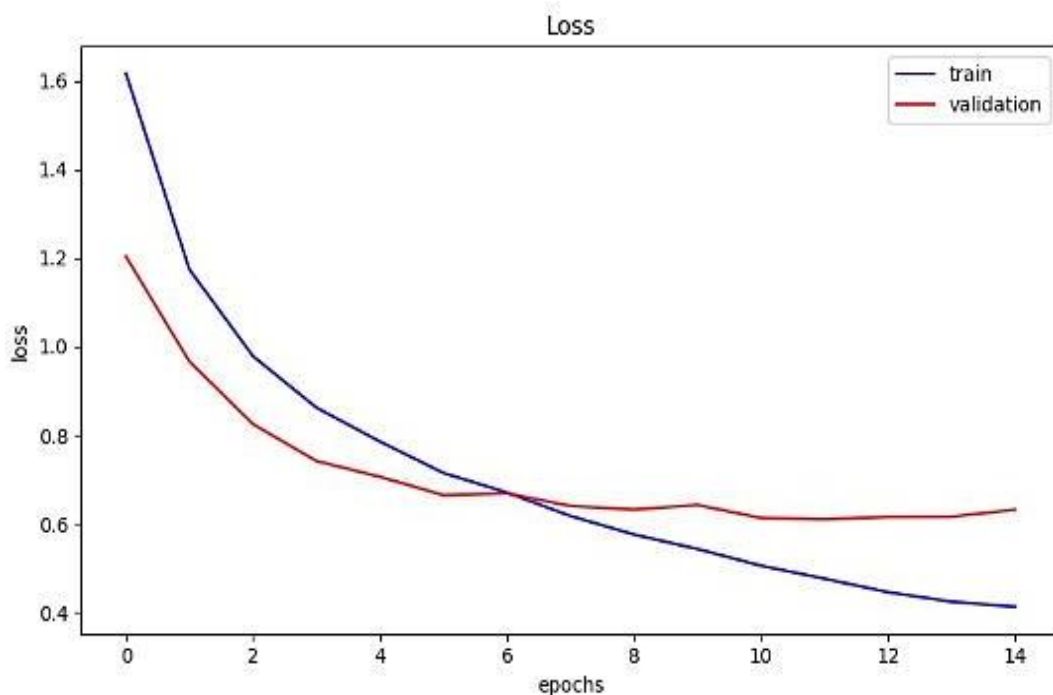


Рисунок 2 – График потерь для оптимизатора adam

2. Уберем из моделей слои dropout.

Ошибка на валидации пошла вверх, что говорит о переобучении. Dropout используется для устранения переобучения, путем случайного отключения связей или нейронов, таким образом, что либо связь выдает нулевой сигнал на выход, либо нейрон выдает на все свои выходы нулевой сигнал. Графики точности и ошибки предоставлены на рис. 3 и рис. 4 соответственно.

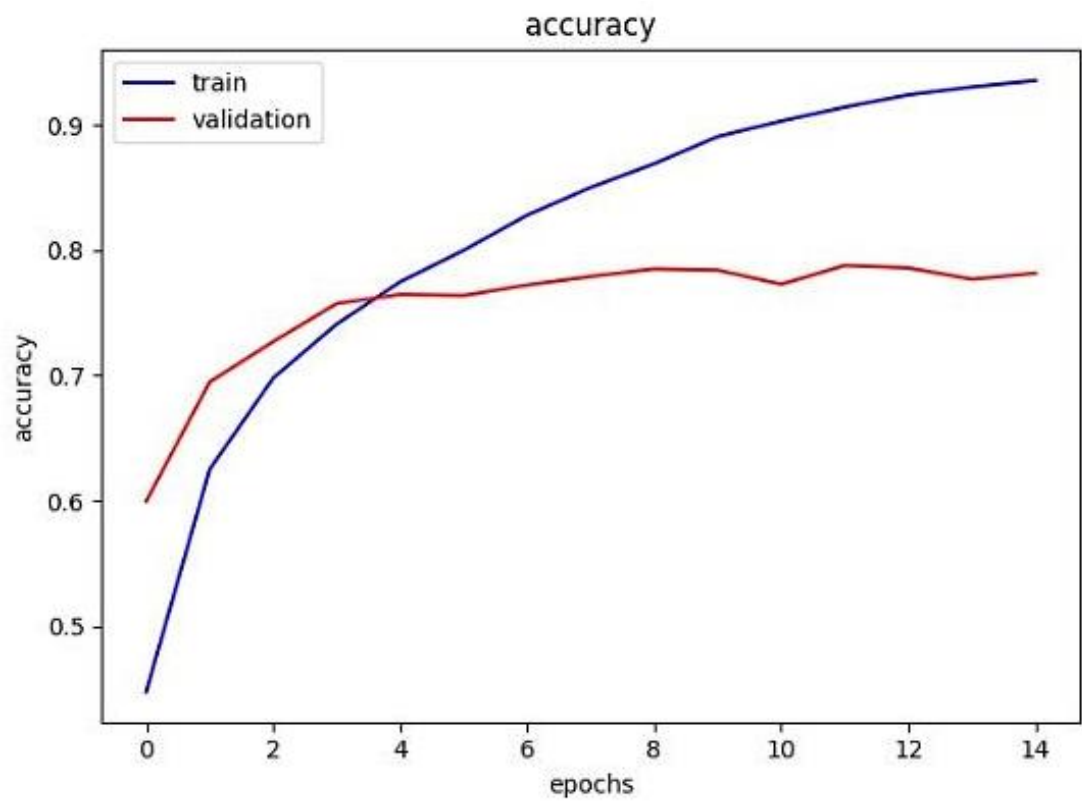


Рисунок 3 – График точности без dropout

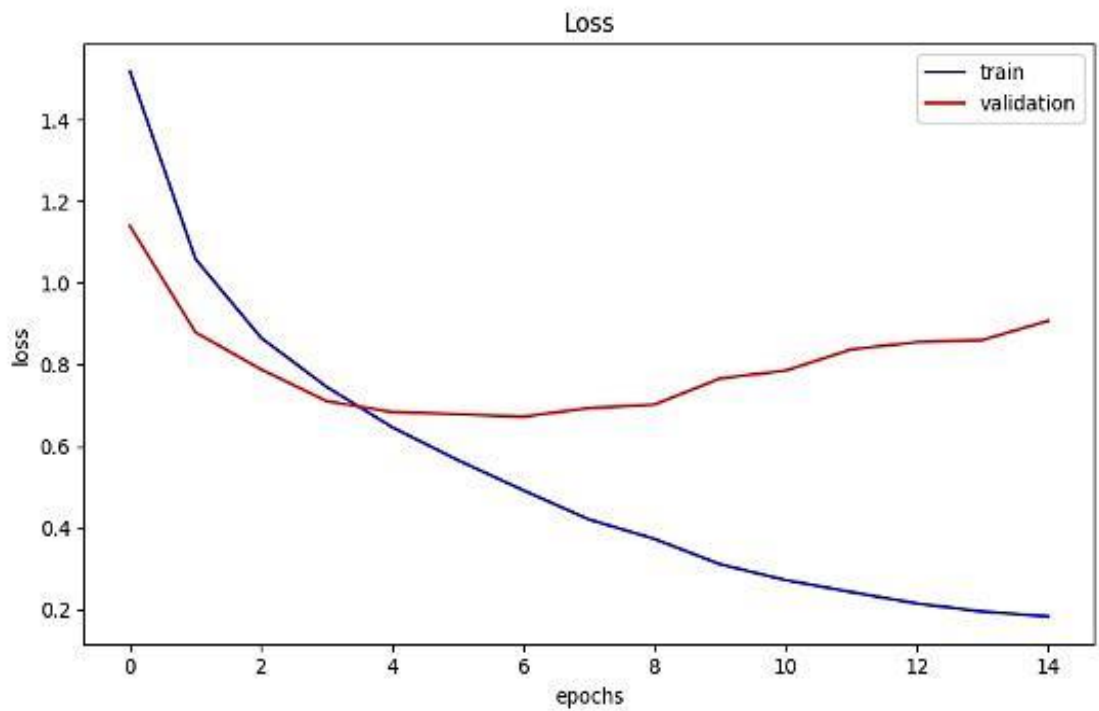


Рисунок 4 – График потерь без dropout

3. Исследуем работу сети при разных размерах ядра свертки.

Поменяем размер ядра свертки на среднем и последнем сверточном слое с 3x3 на 5x5. Графики точности и ошибки предоставлены на рис. 5 и рис. 6 соответственно.

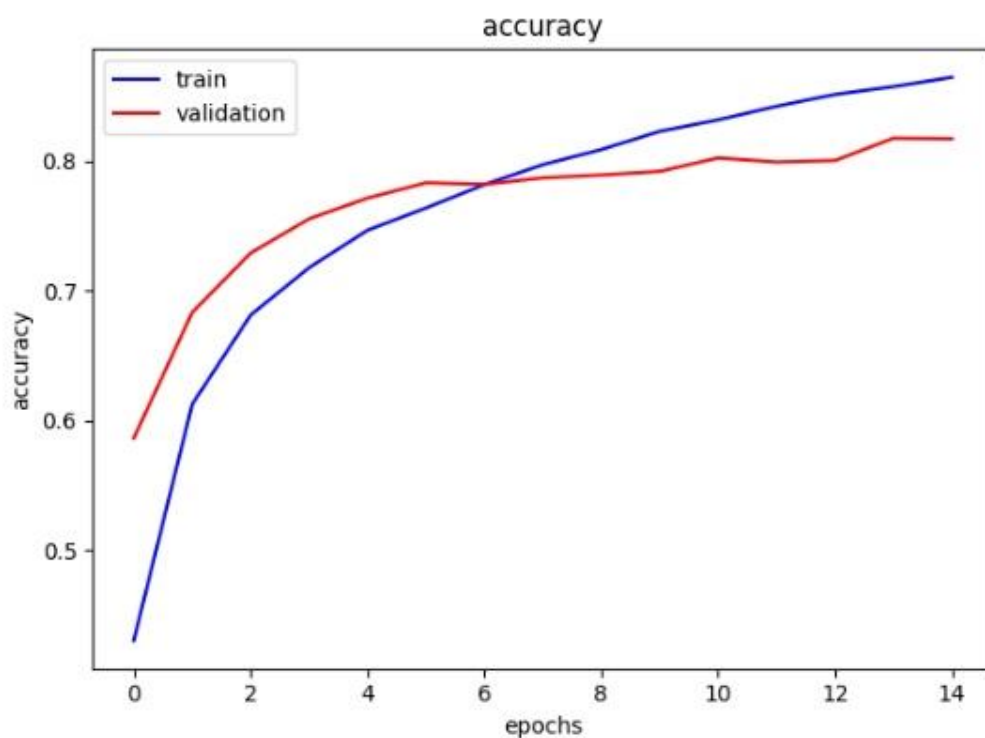


Рисунок 5 – График точности с размером ядра на 5x5

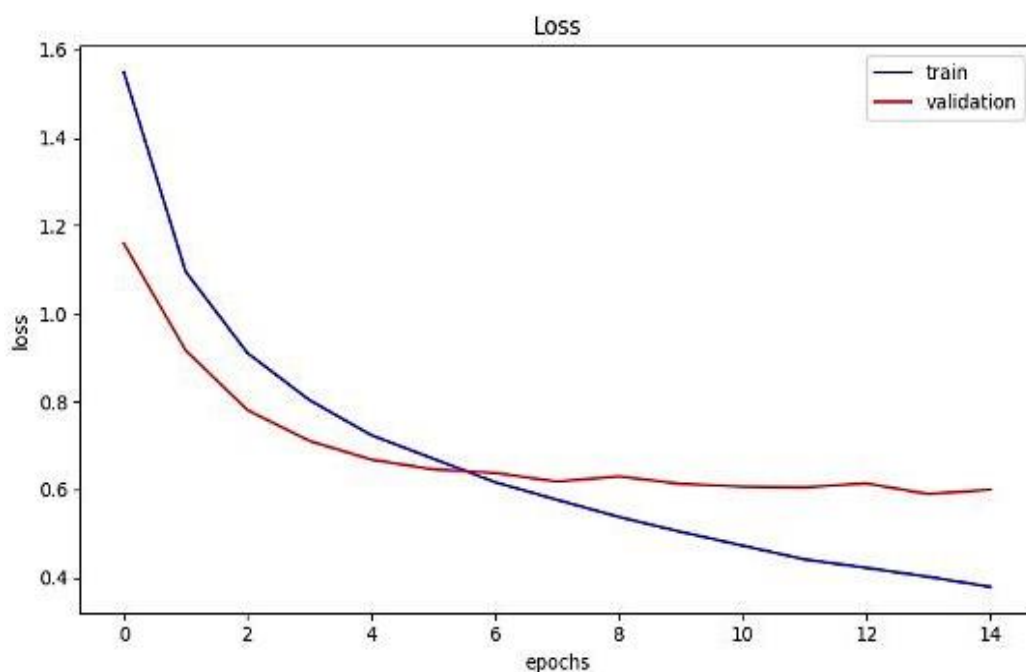


Рисунок 6 – График потерь с размером ядра на 5x5

Изменим размер ядра свертки на одном сверточном слое с 3x3 на 5x5. Графики точности и ошибки предоставлены на рис. 7 и рис. 8 соответственно.

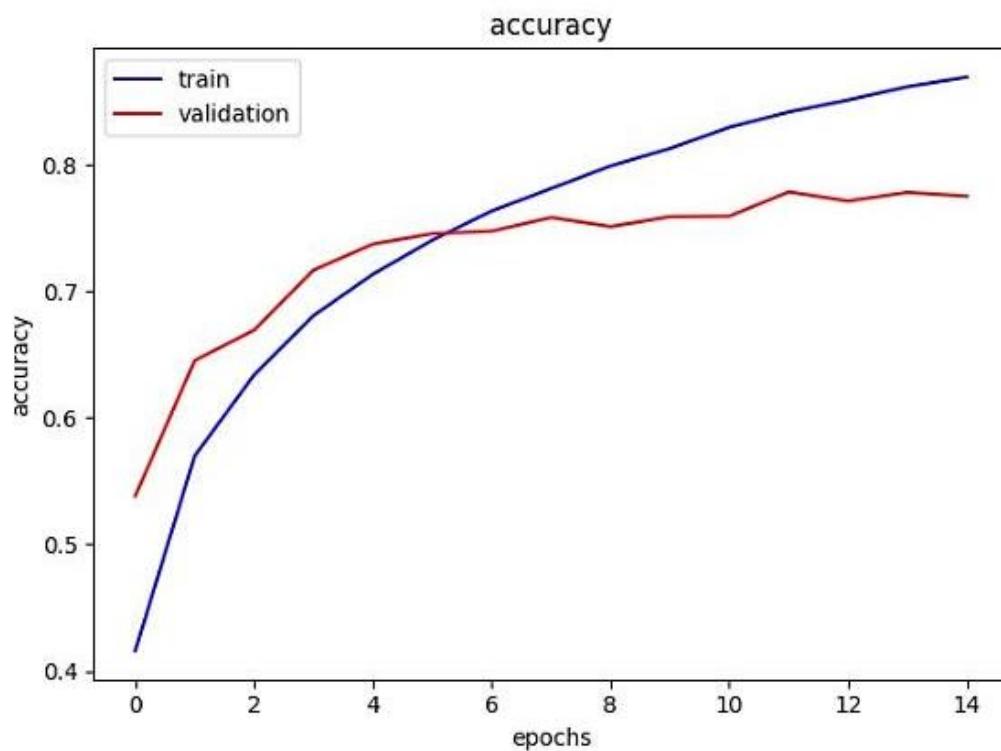


Рисунок 7 – График точности с размером ядра 5x5 на одном слое

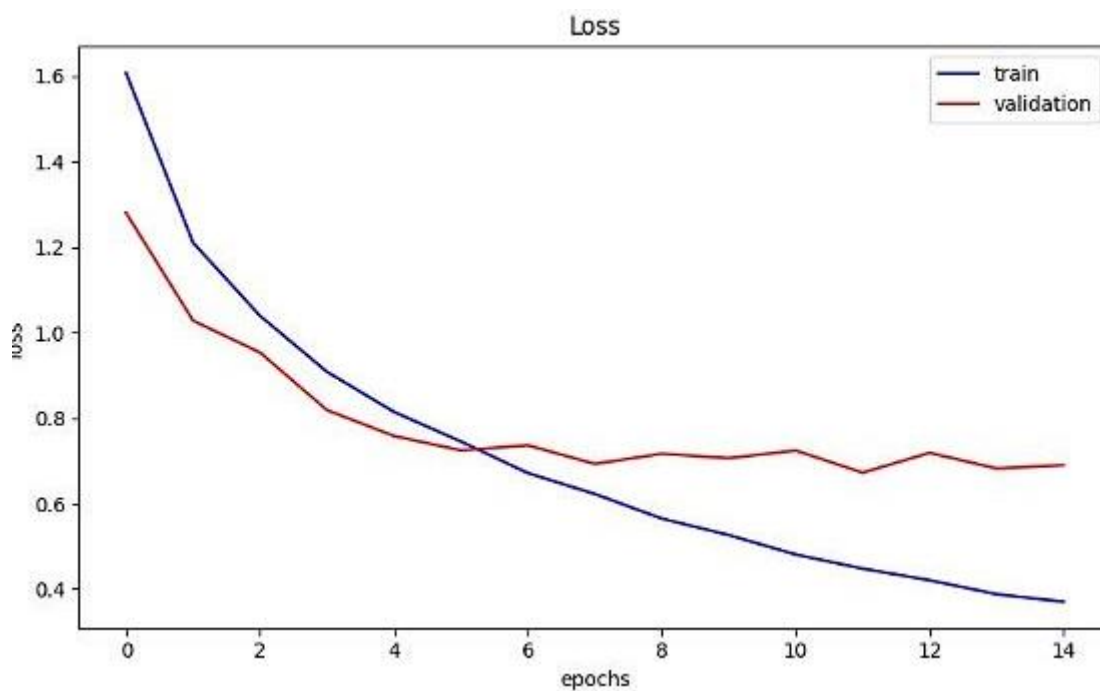


Рисунок 8 – График потерь с размером ядра 5x5 на одном слое

Выводы.

В ходе работы была изучена задача классификация изображений из датасета CIFAR-10. Подобрана архитектура, дающая точность 85%. Показано, что Dropout увеличивает устойчивость сети к отклонению частей связи и к переобучению. Смена размера ядра свертки в двух слоях практически не повлияла на конечную точность, но при установке в первом слое kernel size 5x5 ошибка увеличилась. Это связано с тем, что при таком ядре свертке признаки, которые вывела НС оказались неудачными.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

def plot_loss(loss, v_loss):
    plt.figure(1, figsize=(8, 5))
    plt.plot(loss, 'b', label='train')
    plt.plot(v_loss, 'r', label='validation')
    plt.title('Loss')
    plt.ylabel('loss')
    plt.xlabel('epochs')
    plt.legend()
    plt.show()
    plt.clf()

def plot_acc(acc, val_acc):
    plt.plot(acc, 'b', label='train')
    plt.plot(val_acc, 'r', label='validation')
    plt.title('accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epochs')
    plt.legend()
    plt.show()
    plt.clf()

(X_train, y_train), (X_test, y_test) = cifar10.load_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_1 = Convolution2D(32, 3, 3, border_mode='same',
activation='relu')(inp)
conv_2 = Convolution2D(32, 3, 3, border_mode='same',
activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(2, 2))(conv_2)
drop_1 = Dropout(32)(pool_1)
conv_3 = Convolution2D(64, 3, 3, border_mode='same',
activation='relu')(drop_1)
conv_4 = Convolution2D(64, 3, 3, border_mode='same',
activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(2, 2))(conv_4)
drop_2 = Dropout(0.25)(pool_2)
flat = Flatten()(drop_2)
hidden = Dense(512, activation='relu')(flat)
drop_3 = Dropout(0.5)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(input=inp, output=out)

model.compile(Adam(), loss=CategoricalCrossentropy(),
metrics=['accuracy'])

```

```
H = model.fit(X_train, Y_train, batch_size=100, epochs=15, verbose=1,  
validation_split=.1)
```

```
l, accuracy = model.evaluate(X_test, Y_test)  
print('test data', accuracy)
```

```
plot_loss(H.history['loss'], H.history['val_loss'])  
plot_acc(H.history['accuracy'], H.history['val_accuracy'])
```