

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 фотографии

Студент гр. 7381

Дорох С.В.

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, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

Порядок выполнения работы.

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

Требования.

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

Ход работы.

Для исследования была разработана и использована программа. Код программы приведен в приложении А.

Были рассмотрены модели со слоями Dropout и без них при размере ядра свертки 3x3. На рисунках 1-2 представлены результаты.

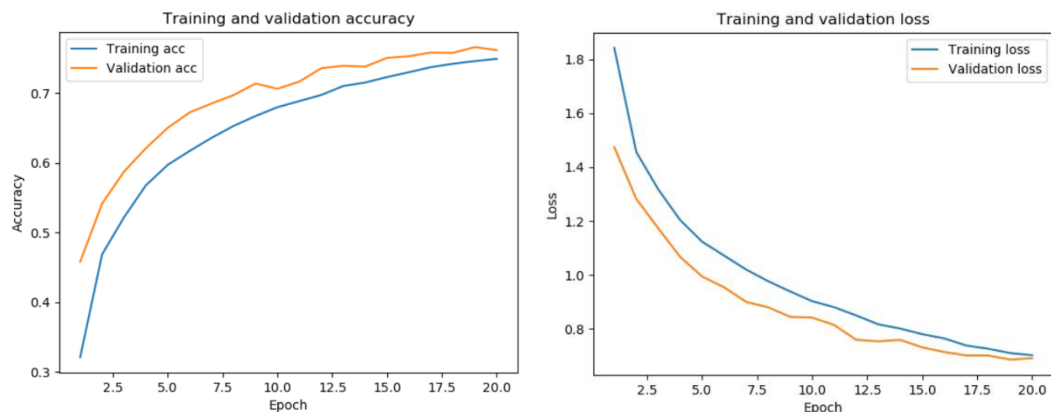


Рисунок 1 – Графики точности и потерь с Dropout и с размером ядра 3x3

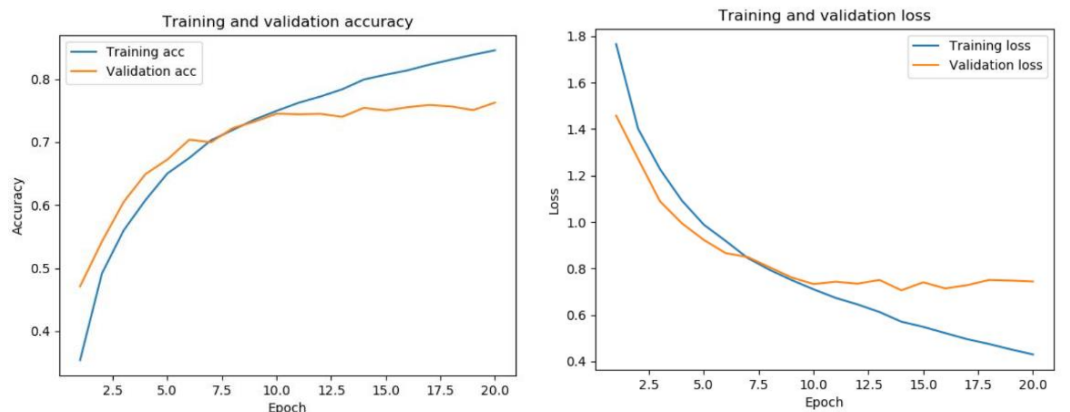


Рисунок 2 – Графики точности и потери без Dropout слоёв с размером ядра 3x3

На графиках видно, что после 10 эпох в модели без Dropout слоев начинается переобучение, чего не было обнаружено у модели со слоями прореживания. Следовательно, Dropout слои лучше использовать, далее, рассматривая модели с большими размерами ядра будем их включать.

Рассмотрим как будет вести себя модель с размерами ядра 5x5 и 7x7. Результаты работы показаны на рисунках 3-4.

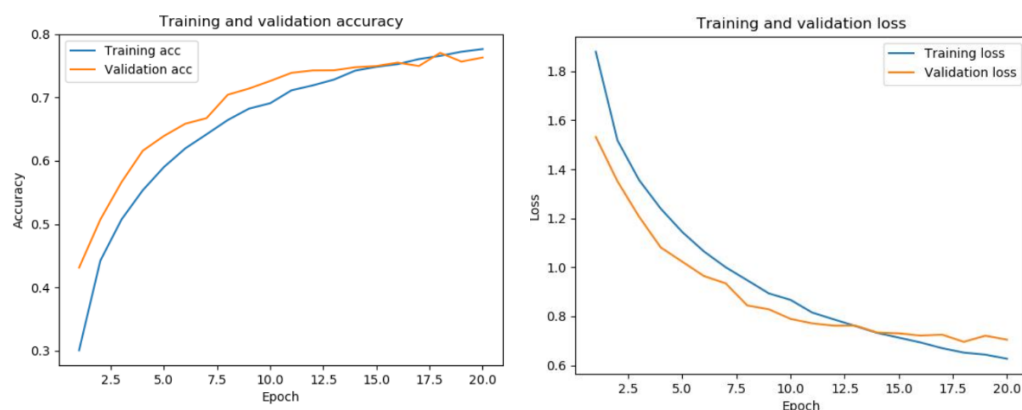


Рисунок 3 – Графики точности и потерь с размером ядра 5x5

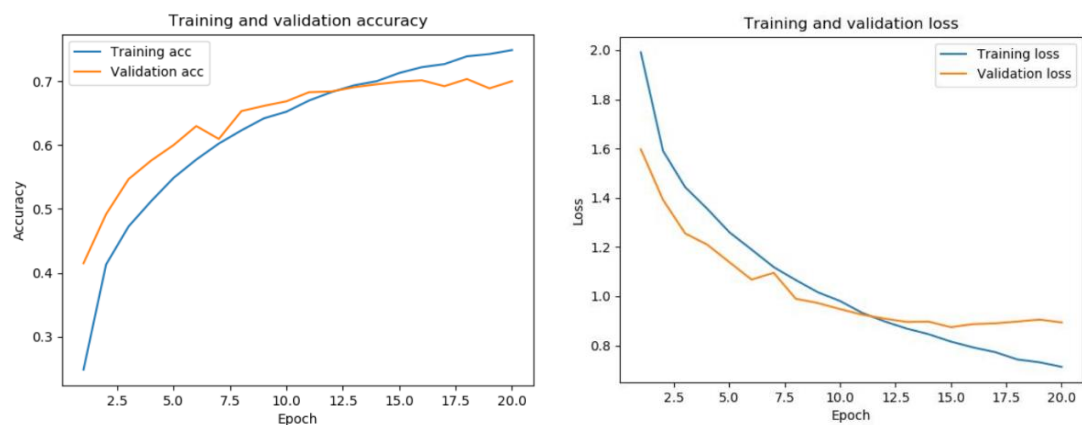


Рисунок 3 – Графики точности и потерь с размером ядра 7x7

Глядя на графики можно увидеть, что с увеличением ядра свёртки падает точность, а также увеличивается ошибка.

Выводы.

В ходе выполнения данной работы была создана сеть, которая может распознавать объекты на фотографиях. Было исследовано влияние наличия Dropout слоев в нейронной сети и зависимость от размера ядра свертки. Было выявлено, что при большем размере ядра свертки, процесс обучения проходит дольше.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

batch_size = 256
num_epochs = 20
kernel_size = 5
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()

num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train) #

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)

def run_model(dropout=True):
    inp = Input(shape=(depth, height, width))
    conv_1 = Convolution2D(conv_depth_1, (kernel_size,
kernel_size),
                        padding='same',
activation='relu')(inp)
    conv_2 = Convolution2D(conv_depth_1, (kernel_size,
kernel_size),
                        padding='same',
activation='relu')(conv_1)
    pool_1 = MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_2)
    drop_1 = Dropout(drop_prob_1)(pool_1) if dropout else pool_1
    conv_3 = Convolution2D(conv_depth_2, (kernel_size,
```

```

kernel_size),
                                padding='same',
activation='relu')(drop_1)
    conv_4 = Convolution2D(conv_depth_2, (kernel_size,
kernel_size),
                                padding='same',
activation='relu')(conv_3)
    pool_2 = MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_4)
    drop_2 = Dropout(drop_prob_1)(pool_2) if dropout else pool_2
    flat = Flatten()(drop_2)
    hidden = Dense(hidden_size, activation='relu')(flat)
    drop_3 = Dropout(drop_prob_2)(hidden) if dropout else hidden
    out = Dense(num_classes, activation='softmax')(drop_3)
    model = Model(input=inp, output=out)
    model.compile(loss='categorical_crossentropy',

                    optimizer='adam',

                    metrics=['accuracy'])
    history = model.fit(X_train, Y_train, batch_size=batch_size,
nb_epoch=num_epochs, verbose=1, validation_split=0.1)
    model.evaluate(X_test, Y_test, verbose=1)

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Training and validation accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Training and validation loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

run_model()

```