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. 7383 _____ Рудоман В.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, 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 2x2. На рисунках 1-2 представлены результаты.

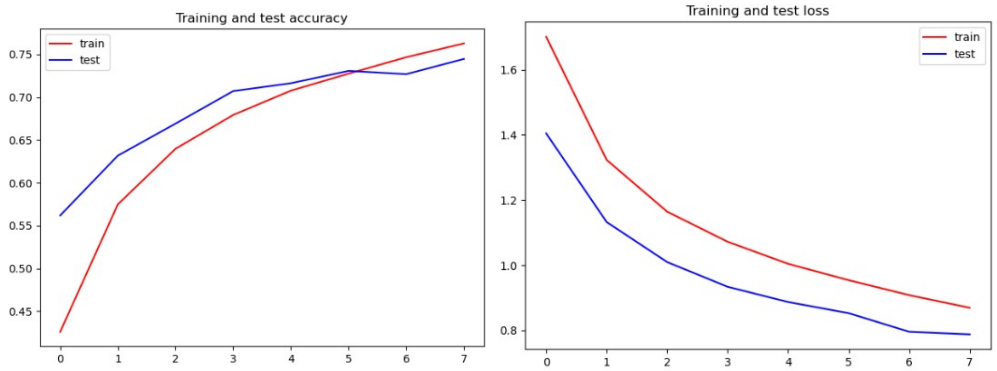

Рисунок 1 – Графики точности и потерь без Dropout и с размером ядра 2x2

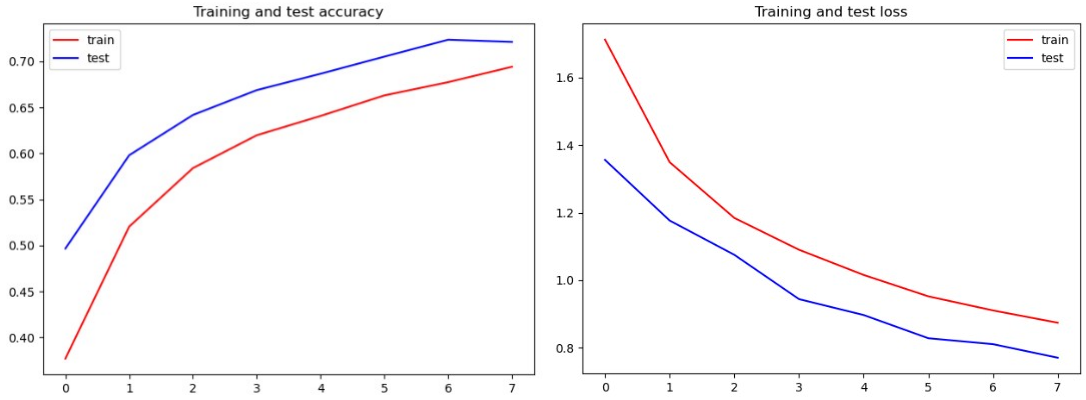

Рисунок 2 – Графики точности и потерь с Dropout и с размером ядра 2x2

На графиках видно, что после 5 эпох в модели без Dropout слоев начинается переобучение.

Рассмотрим как будет вести себя модель с размерами ядра 3x3 и 5x5. Результаты работы показаны на рисунках 3-4.

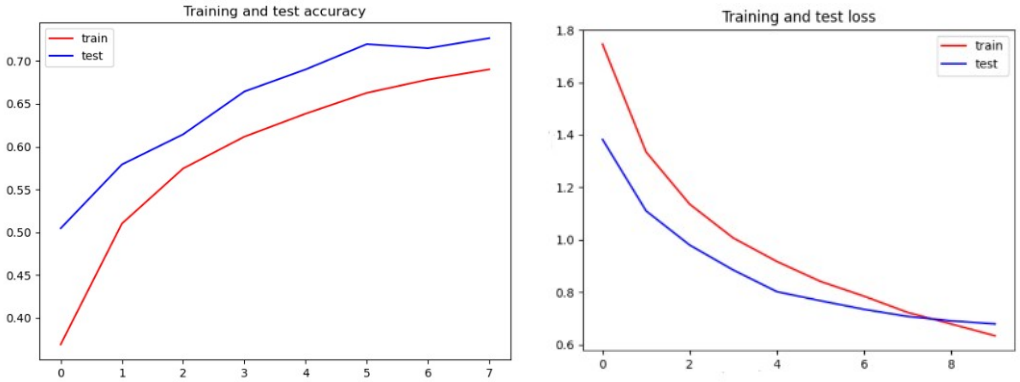

Рисунок 3 – Графики точности и потерь с размером ядра 3x3

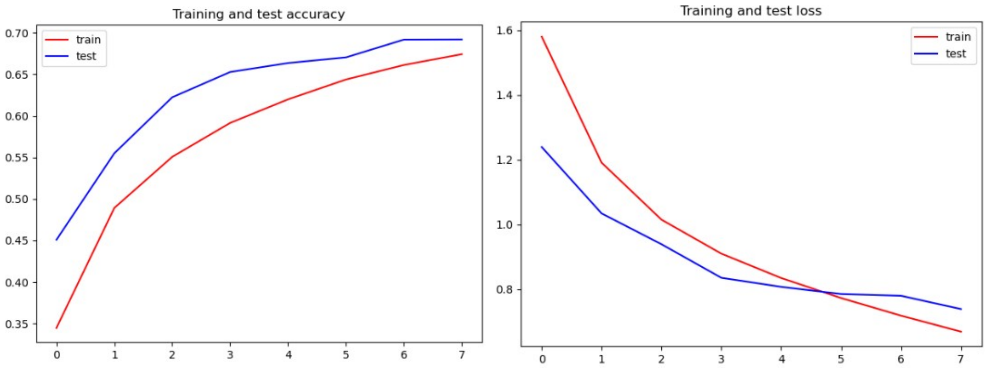

Рисунок 3 – Графики точности и потерь с размером ядра 5x5

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

```
import numpy as np
import matplotlib.pyplot as plt

from keras.datasets import cifar10
from keras.layers import Input, Convolution2D, MaxPooling2D,
Dense, Dropout, Flatten
from keras.models import Model
from keras.utils import np_utils

# Setting hyper-parameters:
batch_size = 300
num_epochs = 10
kernel_size = 5
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
hidden_size = 512
with_dropout = True
if with_dropout:
    drop_prob_1 = 0.25
    drop_prob_2 = 0.5

# Loading data:
(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]

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

# Converting labels:
Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)
# Building model:
inp = Input(shape=(depth, height, width))
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)
if with_dropout:
    drop_1 = Dropout(drop_prob_1)(pool_1)
```

```
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(drop_1 if with_dropout
else pool_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)
if with_dropout:
    drop_2 = Dropout(drop_prob_1)(pool_2)
flat = Flatten()(drop_2 if with_dropout else pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)
if with_dropout:
    drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3 if
with_dropout else hidden)

model = Model(input=inp, output=out)
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Fitting model:
h = model.fit(X_train, Y_train, batch_size=batch_size,
nb_epoch=num_epochs, verbose=1, validation_split=0.1)

print(model.evaluate(X_test, Y_test, verbose=1))

# Plotting:
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(range(1, num_epochs + 1), loss, 'b', label='Training
loss')
plt.plot(range(1, num_epochs + 1), val_loss, 'r',
label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
plt.plot(range(1, num_epochs + 1), acc, 'b', label='Training
acc')
plt.plot(range(1, num_epochs + 1), val_acc, 'r',
label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```