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

Ход работы.

Для исследования была разработана и использована программа, код которой приведен в приложении Б.

Для исследования будут рассмотрены модели со слоями Dropout и буз них, а также с размером ядра свертки: 2x2, 3x3 и 5x5. Все исследования будут проходить при 10 эпохах обучения.

Результаты тестирования модели с ядром свертки 2x2 со слоями Dropout и без них представлены в приложении А на рис. 1. Из графиков видно, обе модели достигли сопоставимого результата, однако модель без слоев Dropout начала переобучаться после 5 эпохи, тогда как модель со слоями Dropout еще не начала.

Результаты тестирования модели с ядром свертки 3x3 со слоями Dropout и без них представлены в приложении А на рис. 2. Из графиков видно, обе модели достигли сопоставимого результата, однако модель без слоев Dropout начала переобучаться после 5 эпохи, тогда как модель со слоями Dropout еще не начала.

Результаты тестирования модели с ядром свертки 5x5 со слоями Dropout и без них представлены в приложении А на рис. 3. Из графиков видно, обе модели достигли сопоставимого результата, однако модель без

слоев Dropout начала переобучаться после 5 эпохи, тогда как модель со слоями Dropout еще не начала.

В результате тестирования видно, что с увеличением размера ядра свертки результат улучшается незначительно, однако скорость обучения нейронной сети быстро снижается.

Также видно, модели со слоями Dropout и без них приходят к сопоставимым результатам после 10 эпох обучения, но модели без слоев Dropout начинают переобучаться после 5 эпох обучения.

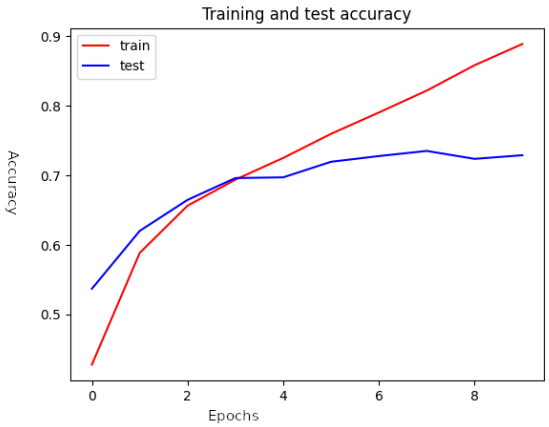
Таким образом, наиболее оптимальной является модель со слоями Dropout и ядром свертки 2x2.

Выводы.

В ходе выполнения лабораторной работы была создана сеть для распознавания объектов на изображениях. Было исследовано влияние размера ядра свертки и наличия слоев Dropout на результат обучения. Выявлено, что при увеличении размеров ядра свертки, скорость обучения значительно понижается, тогда как результат улучшается незначительно. Также выявлено, что при отсутствии слоев Dropout сеть быстро начинает переобучаться.

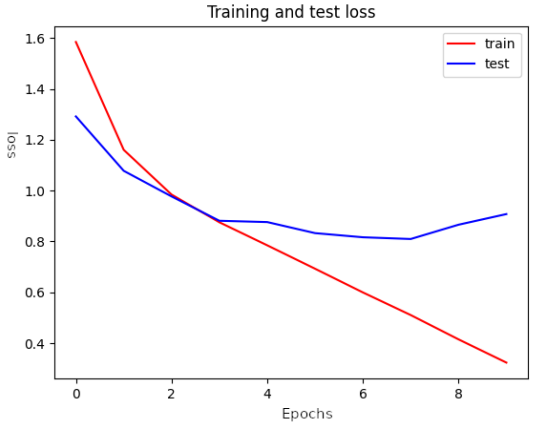
ПРИЛОЖЕНИЕ А

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

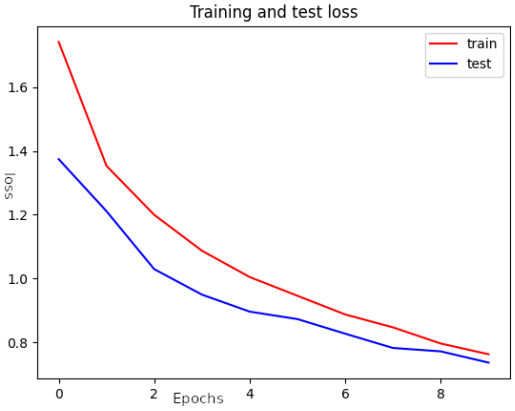


accuracy (no Dropout)

accuracy (with Dropout)



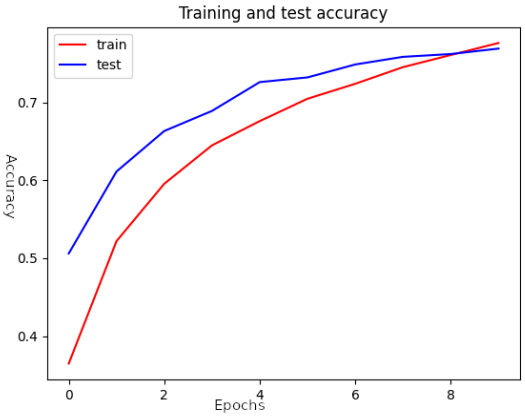
loss (no Dropout)



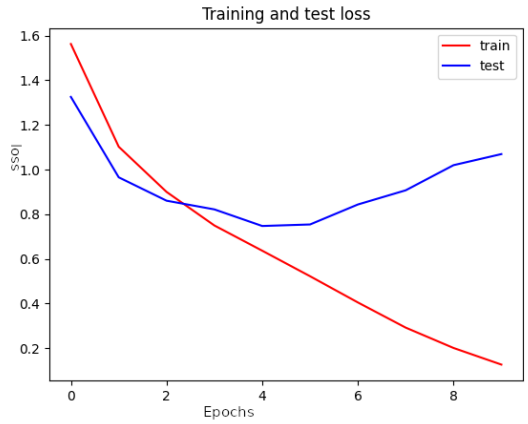
loss (with Dropout)

Рисунок 1 — Результаты тестирования с ядром свертки 2x2

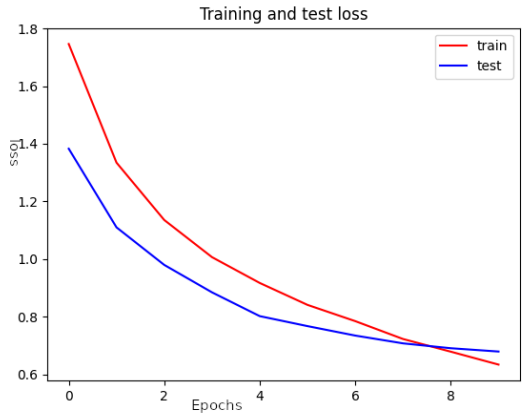
accuracy (no Dropout)



accuracy (with Dropout)



loss (no Dropout)

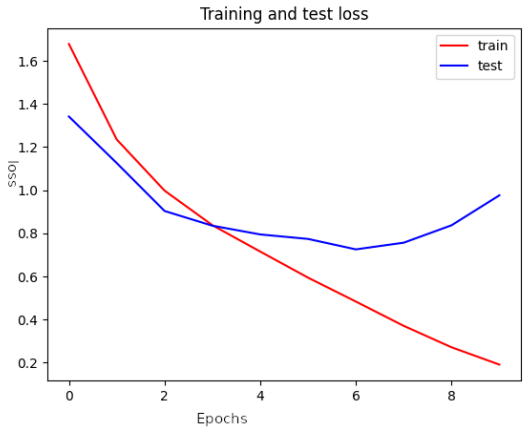


loss (with Dropout)

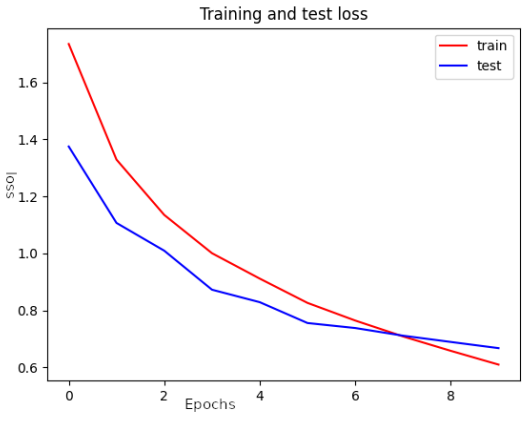
Рисунок 2 — Результаты тестирования с ядром свертки 3x3

accuracy (no Dropout)

accuracy (with Dropout)



loss (no Dropout)



loss (with Dropout)

Рисунок 3 — Результаты тестирования с ядром свертки 5x5

ПРИЛОЖЕНИЕ Б

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 = 250
num_epochs = 10
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512


def buildModel(kernel_size=3, 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)
    m = Model(inputs=inp, outputs=out)
    m.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return m


def testModel(ks, dropout):
    model = buildModel(kernel_size=ks, dropout=dropout)
```

```
    history = model.fit(X_train, Y_train, batch_size=batch_size,
epochs=num_epochs, verbose=1, validation_split=0.1)
    test_loss, test_acc = model.evaluate(X_test, Y_test,
verbose=1)

    plt.title('Training and test accuracy')
    plt.plot(history.history['accuracy'], 'r', label='train')
    plt.plot(history.history['val_accuracy'], 'b', label='test')
    plt.legend()
    plt.savefig("Graphics/%s%s_acc.png" % ("d" if d else "_",
ks), format='png')
    plt.clf()

    plt.title('Training and test loss')
    plt.plot(history.history['loss'], 'r', label='train')
    plt.plot(history.history['val_loss'], 'b', label='test')
    plt.legend()
    plt.savefig("Graphics/%s%s_loss.png" % ("d" if d else "_",
ks), format='png')
    plt.clf()

    result["%s%s" % ("d" if d else "_", ks)] = test_acc


(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)

result = dict()
for ks in [2, 3, 5]:
    for d in [True, False]:
        testModel(ks, d)

for res in result:
    print("%s: %s" % (res, result[res]))
```