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 _____ Зуев 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. Была написана функция model_fit строящая модель и тренирующая ее искусственной нейронной сети, распознающей изображения из набора CIFAR-10, в зависимости от тестовых параметров, таких как размер ядра свертки и флаг включения слоев Dropout. Код представлен в приложении А.
2. Было проведено тестирование влияния слоя Dropout на результат обучения нейронной сети. Результаты обучения сети с включенным слоем Dropout и размером ядра свертки 3x3 представлен на рис. 1-2.

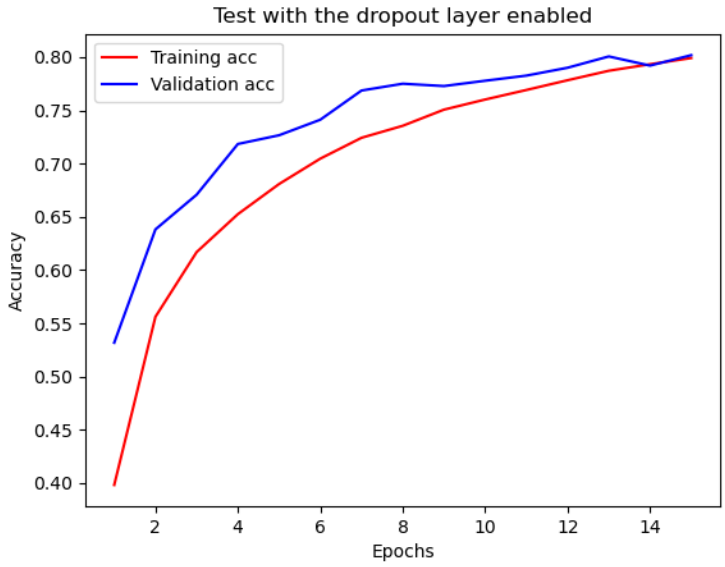

Рисунок 1 - Точность при включенном слое Dropout

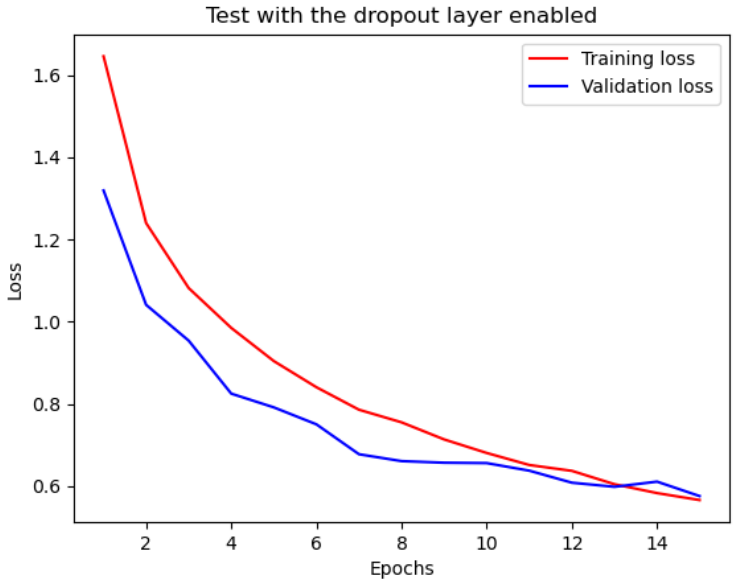

Рисунок 2 - Потери при включенном слое Dropout

Далее была протестирована та же архитектура за исключением слоя Dropout. Результаты обучения сети с выключенным слоем разреживания представлены на рис. 3-4.

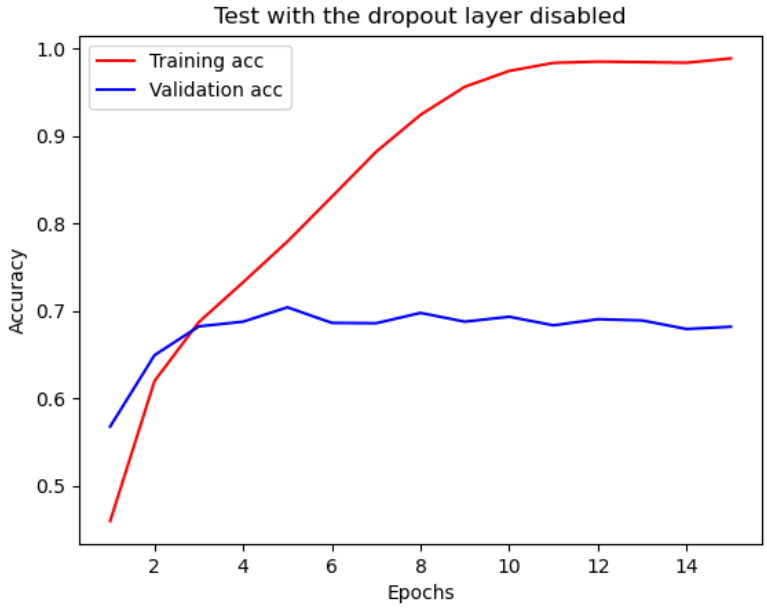

Рисунок 3 - Точность сети с выключенным слоем Dropout

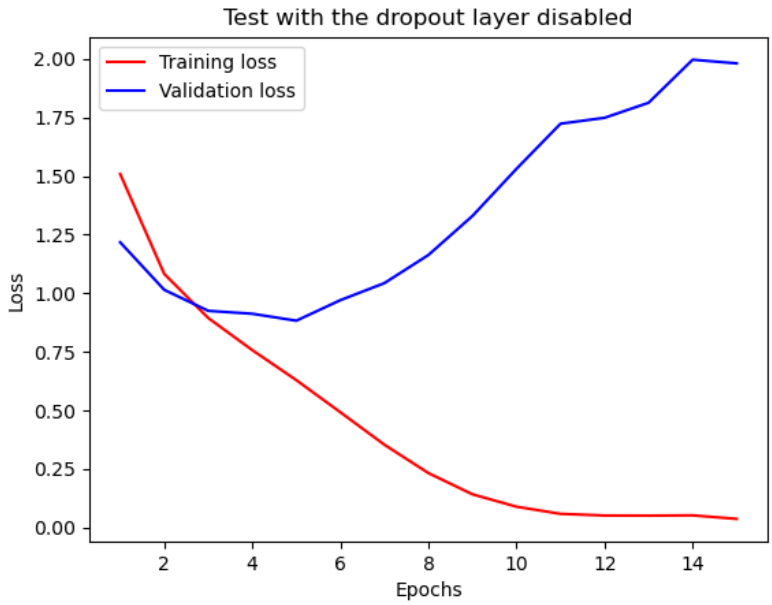

Рисунок 4 - Потери при выключенном слое Dropout

По результатам тестирования влияния слоя разреживания на результат обучения сети видно, что при отключении слоя Dropout точность на обучающих данных значительно возрастает, однако точность на тестовых данных в тоже время на каждой новой эпохе обучения либо не увеличивается, либо уменьшается начиная с пятой эпохи. Это говорит о том, что сеть начала переобучаться.

3. Было проведено тестирование влияния размера слоя свертки на результат обучения сети. Обучение при различных размерах ядра проводилось при включенном слое Dropout. Сравнивать будем с начальной архитектурой, результаты обучения которой представлены на рис. 1-2. Результат обучения при размере слоя свертки 2x2 представлен на рис. 5-6.

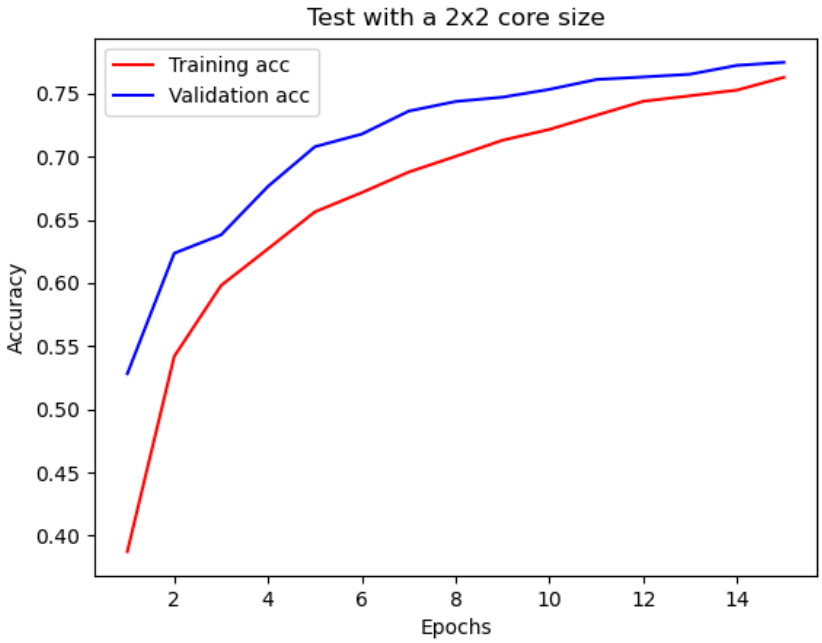

Рисунок 5 - Точность при размере свертки 2x2

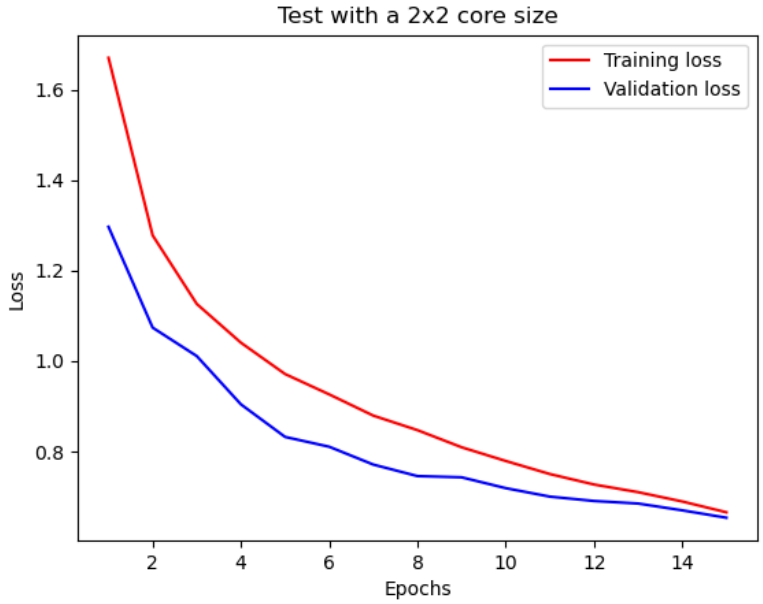

Рисунок 6 - Потери при размере свертки 2x2

Данная архитектура показывает результат обучения хуже, чем в начальной архитектуре. Возможно, это связано с тем, что слой свертки недостаточно преобразует изображение, чтобы сеть могла выявить различные закономерности, между начальным изображением и классом, к которому оно относится.

Результат обучения при размере слоя свертки 4x4 представлен на рис. 7-8.

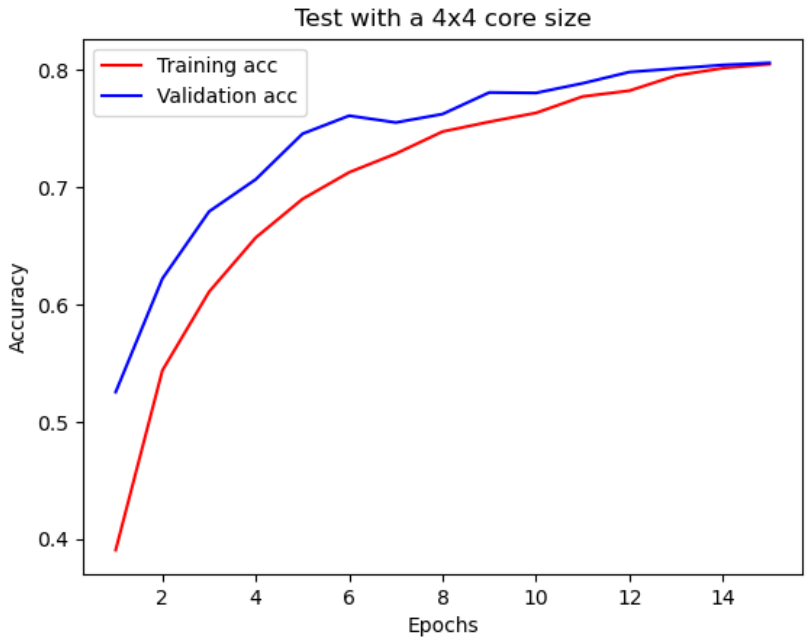

Рисунок 7 - Точность при размере свертки 4x4

Рисунок 8 - Потери при размере свертки 4x4

Результат обучения сети с данной архитектурой практически идентичен результату обучения с начальной архитектурой, однако обучение

проходило несколько дольше. Это связано с увеличением размера слоя свертки и, следовательно, увеличением числа вычислений.

Результат обучения при размере слоя свертки 6x6 представлен на рис. 9-10.

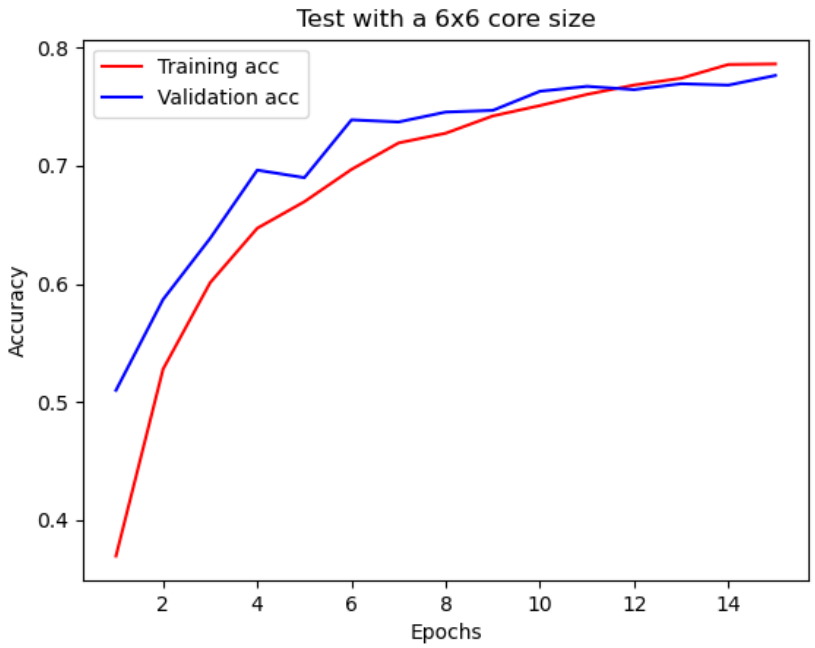

Рисунок 9 - Точность при размере свертки 6x6

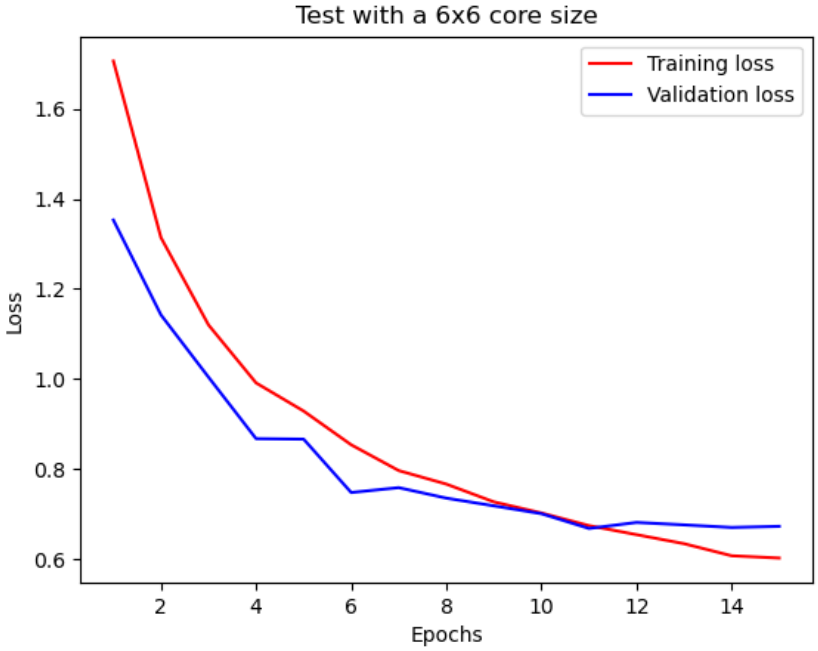

Рисунок 10 - Потери при размере свертки 6x6

По результатам обучения сети с данной архитектурой видно, что точность сети понизилась по сравнению с начальной архитектурой. Это может быть связано с тем, что при слишком большом размере слоя свертки при прохождении данных через этот слой контуры изображения идентифицируемого объекта слишком сильно «размываются» и его идентификация становится более затруднительной. Так же обучение проходило гораздо дольше, чем обучение начальной архитектуры, как и в случае размера слоя свертки 4x4 это может быть связано с увеличением числа расчетов.

Вывод.

В ходе выполнения данной лабораторной работы была построена модель искусственной нейронной сети, распознающей объекты на изображениях, было изучено влияние слоя Dropout на обучение сети. Выявлено, что при отсутствии слоя Dropout появляется эффект переобучения сети. Было изучено влияние размера слоя свертки на результат обучения сети. Выявлено, что при слишком малом, а также при слишком большом размере слоя свертки точность распознавания объектов на изображениях падает, так же 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, Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt

batch_size = 128
num_epochs = 15
kernel_sizes = [2, 4, 6]
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512
dropout_flags = [True, False]
count = 1

(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 fit_model(kernel_size, dropout_enabled):
    global count
    inp = Input(shape=(depth, height, width))

    conv_1 = Conv2D(conv_depth_1, (kernel_size,
kernel_size), activation='relu', padding='same')(inp)
    conv_2 = Conv2D(conv_depth_1, (kernel_size,
kernel_size), activation='relu', padding='same')(conv_1)
    pool_1 = MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_2)
    if dropout_enabled:
        drop_1 = Dropout(drop_prob_1)(pool_1)
    else:
        drop_1 = pool_1
```

```
    conv_3 = Conv2D(conv_depth_2, (kernel_size,
kernel_size), activation='relu', padding='same')(drop_1)
    conv_4 = Conv2D(conv_depth_2, (kernel_size,
kernel_size), activation='relu', padding='same')(conv_3)
    pool_2 = MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_4)
    if dropout_enabled:
        drop_2 = Dropout(drop_prob_2)(pool_2)
    else:
        drop_2 = pool_1

    flat = Flatten()(drop_2)
    hidden = Dense(hidden_size, activation='relu')(flat)
    if dropout_enabled:
        drop_3 = Dropout(drop_prob_2)(hidden)
    else:
        drop_3 = hidden
    out = Dense(num_classes, activation='softmax')(drop_3)

    model = Model(input=inp, output=out)
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    H = model.fit(X_train, Y_train,
                  batch_size=batch_size,
epochs=num_epochs,
                  verbose=1, validation_split=0.1)
    count = count + 1
    name = "model%i.h5" % count
    model.save(name)
    return H


def test_dropout_layer():
    for dropout_flag in dropout_flags:
        H = fit_model(3, dropout_flag)
        str = "enabled" if dropout_flag else "disabled"
        draw_test(H, "Test with the dropout layer "+str)


def test_kernel_size():
    for size in kernel_sizes:
        H = fit_model(size, True)
        draw_test(H, "Test with a %ix%i core size" %
(size, size))


def draw_test(H, title):
    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(title)
    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(title)
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


test_dropout_layer()
test_kernel_size()
```