

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»

Студентка гр. 7383

Иолшина 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, 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)

Требования.

- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

Ход работы.

Была создана и обучена модель искусственной нейронной сети (код программы представлен в приложении А).

Исследуем влияние наличия слоя Dropout, а также разных размеров ядра свертки на результат обучения нейронной сети.

Первоначальное размер ядра свертки 3x3, слой Dropout включен. Результаты обучения сети при данной архитектуре приведены на рис. 1-2.

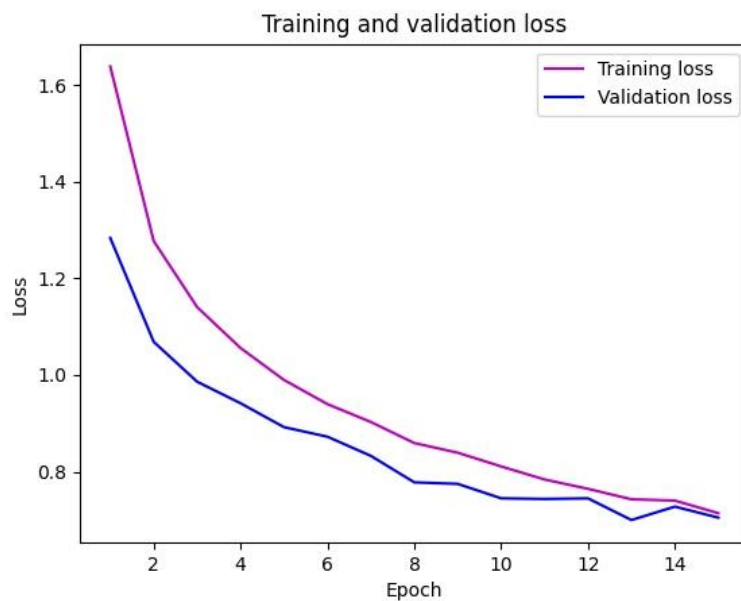


Рисунок 1 – График потери при включенном слое Dropout

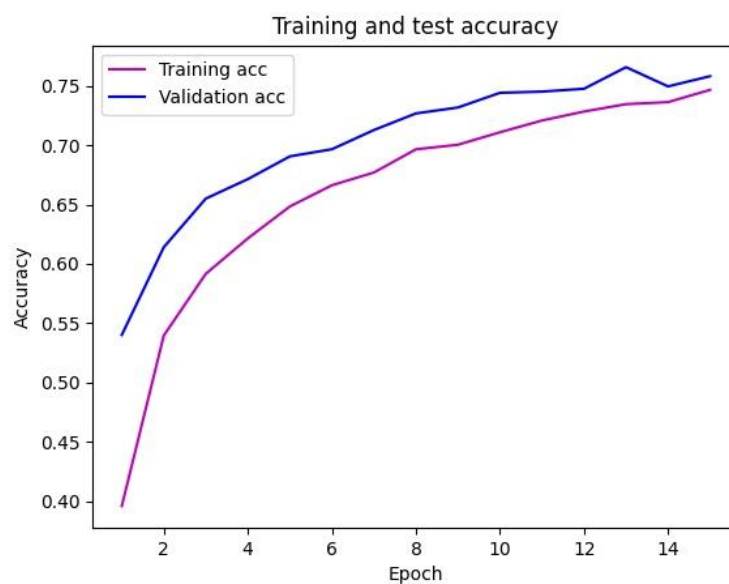


Рисунок 2 - График точности при включенном слое Dropout

По графикам видно, что точность при данной архитектуре сети составляет около 75%.

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

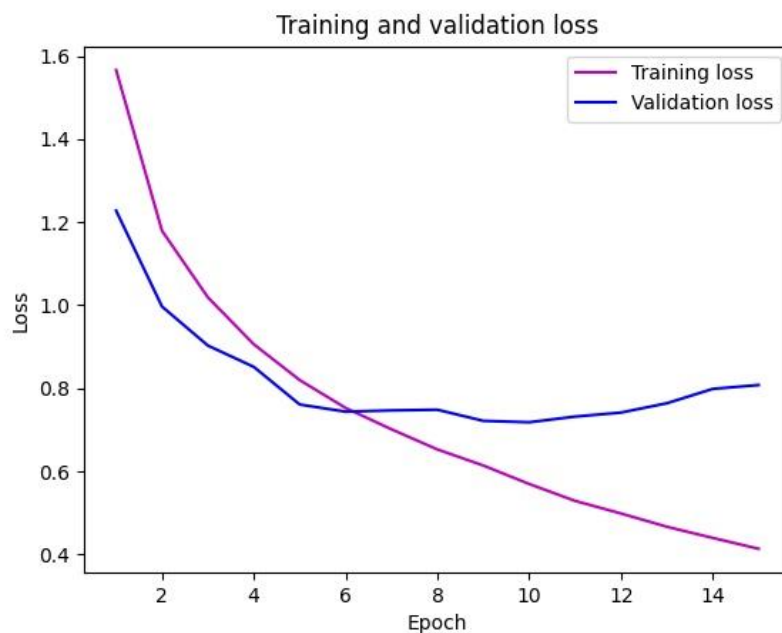


Рисунок 3 - График потери при выключенном слое Dropout

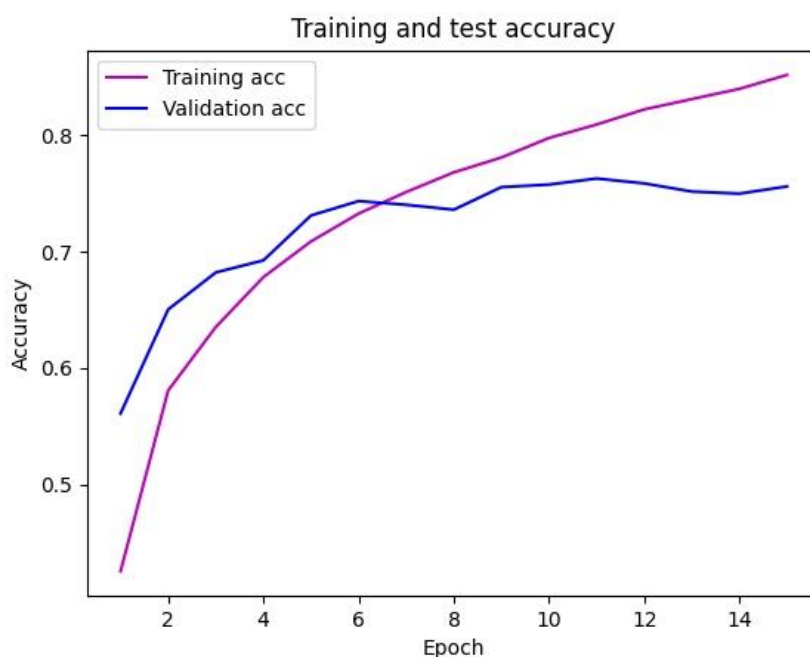


Рисунок 4 - График точности сети с выключенным слоем Dropout

По графикам можно заметить влияние слоя разреживания на результат обучения сети: при отключении слоя Dropout точность на обучающих данных возрастает, однако точность на тестовых данных увеличивается незначительно, или уменьшается, начиная с 6 эпохи. Это говорит о переобучении сети. Следовательно, использование слоя Dropout в данной задаче оправдано.

Затем, была исследована работа сети при разных размерах ядра свёртки.

Результаты обучения с ядрами размера 5×5 и 7×7 продемонстрированы на рис. 5 – 8.

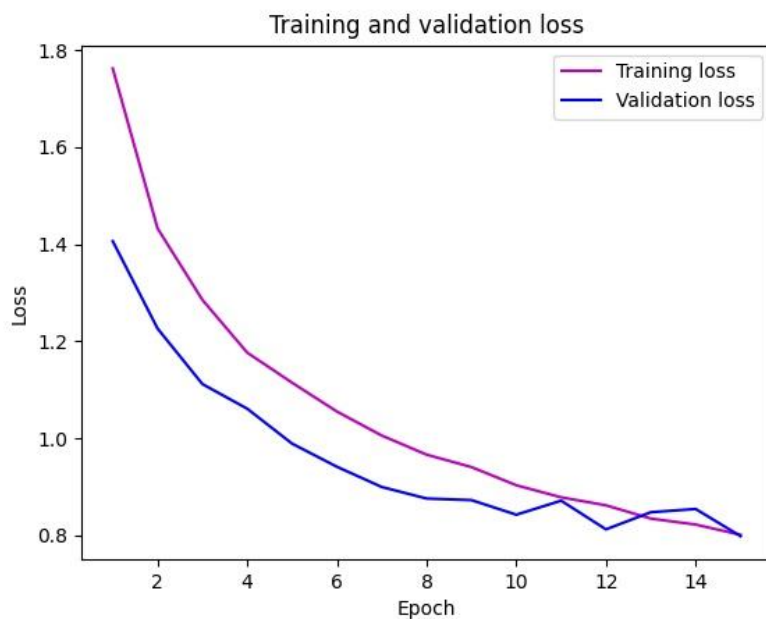


Рисунок 5 – График потери при размере ядра свёртки 5×5 .

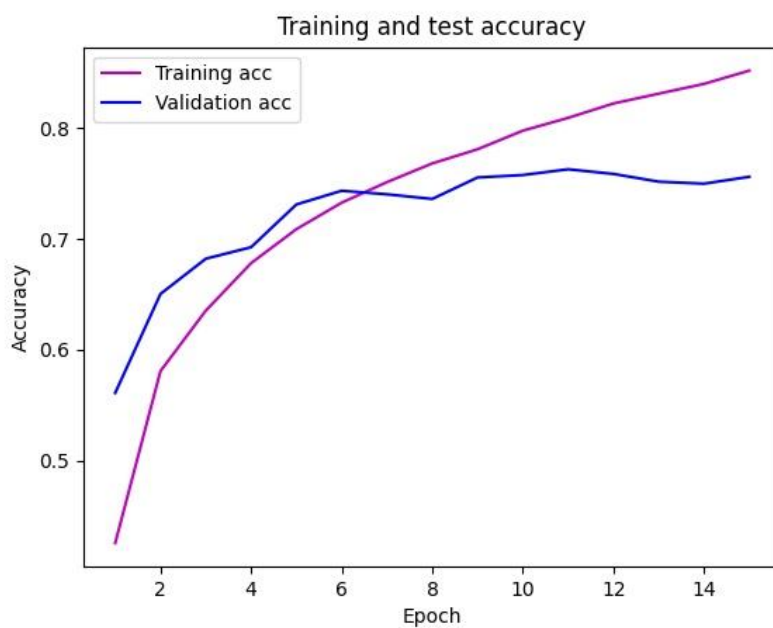


Рисунок 6 – График точности при размере ядра свёртки 5×5 .

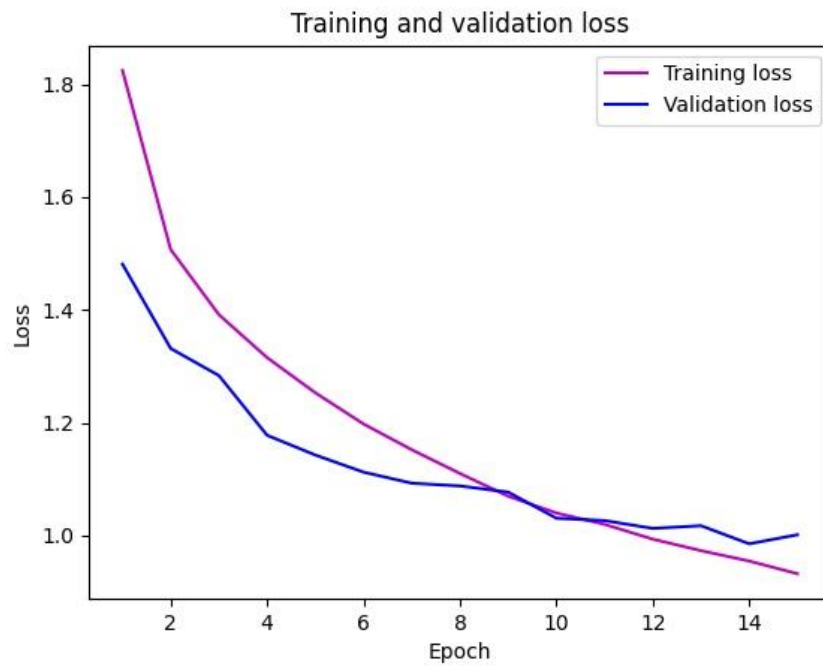


Рисунок 7 – График потери при размере ядра свёртки 7×7 .

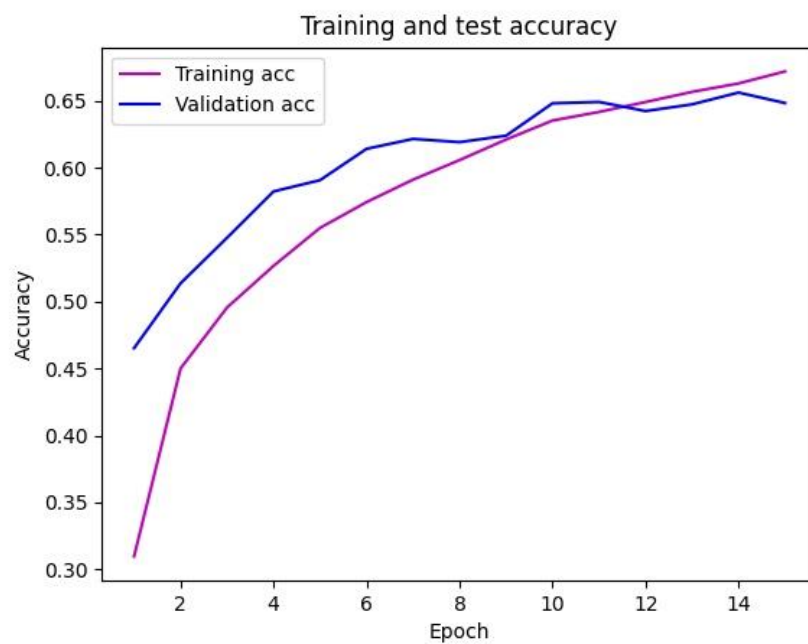


Рисунок 8 – График точности при размере ядра свёртки 7×7 .

Из графиков видно, что с увеличением размера ядра свёртки при неизменных других параметрах сети ошибка возрастает, а точность падает (74% и 64%).

Вывод.

В ходе выполнения лабораторной работы была построена модель искусственной нейронной сети, распознающей объекты на изображениях. Было изучено влияние слоя Dropout на обучение сети, а также влияние размера слоя свертки.

ПРИЛОЖЕНИЕ А

```
from tensorflow.keras.datasets import cifar10

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Convolution2D,
MaxPooling2D, Dense, Dropout, Flatten

from tensorflow.keras.utils import to_categorical

import matplotlib.pyplot as plt

import numpy as np


batch_size = 50

num_epochs = 15

kernel_size = 7

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 = to_categorical(y_train, num_classes)

Y_test = to_categorical(y_test, num_classes)


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)

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)

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)

model = Model(inp, out)

model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size,
epochs=num_epochs, verbose=1, validation_split=0.1)

```

```

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

print("Model accuracy: %s" % (ev[1]))

x = range(1, num_epochs + 1)

plt.title('Training and validation loss')

plt.plot(x, history.history['loss'], 'm', label='Training loss')

plt.plot(x, history.history['val_loss'], 'b', label='Validation
loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.savefig("./loss.png", format='png')

plt.show()

plt.clf()

plt.title('Training and test accuracy')

plt.plot(x, history.history['accuracy'], 'm', label='Training acc')

plt.plot(x, history.history['val_accuracy'], 'b', label='Validation
acc')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend()

plt.savefig("./acc.png", format='png')

plt.show()

plt.clf()

```