МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5
"Распознавание объектов на фотографиях"
по дисциплине «Искусственные нейронные сети»

Студентка гр. 8382 _____ Ивлева О.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

Реализовать классификацию небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик.

Задание.

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

Ход работы.

1. Была создана модель сверточной нейронной сети и настроены параметры обучения.

2. Для исследования влияния слоев разреживания на работу сети выберем 2 конфигурации моделей: конфигурацию без слоя Dropout и конфигурацию с этим слоем. Обучим две модели при одинаковых параметрах обучения и оценим нейронные сети. Графики ошибок и точностей для моделей представлены на рис. 1-4.

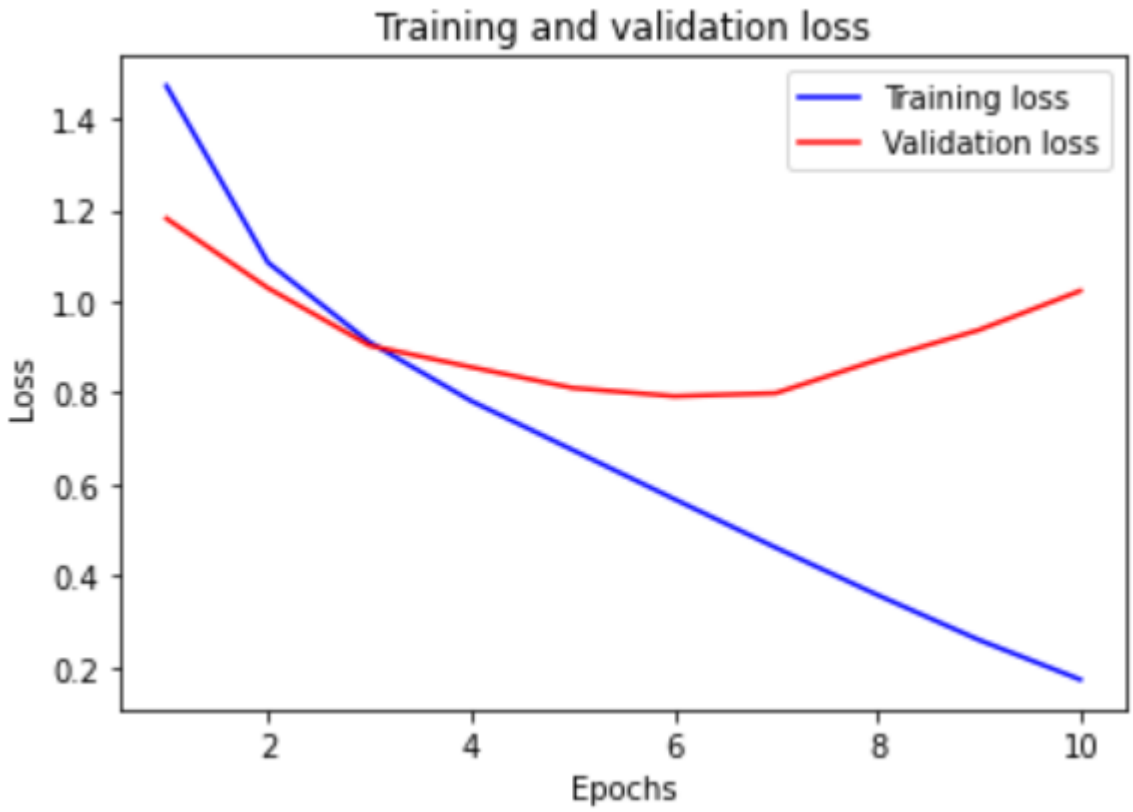

Рисунок 1 – График ошибок модели без слоев Dropout

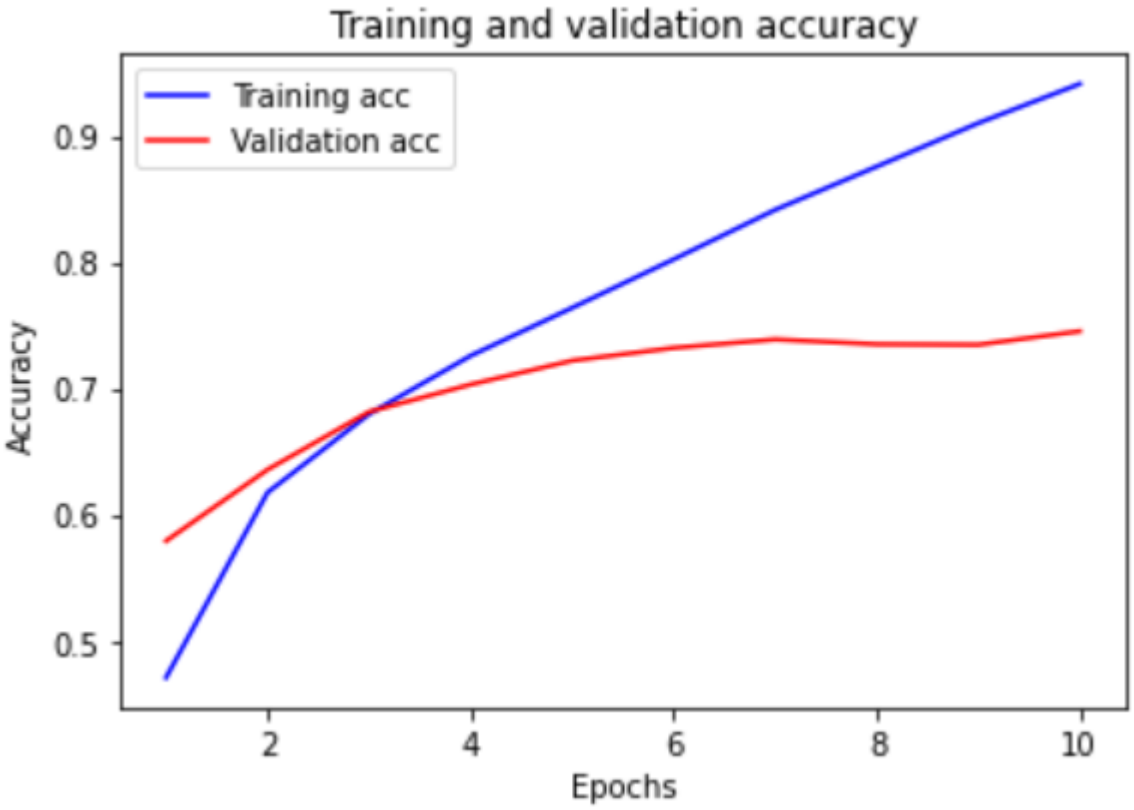

Рисунок 2 – График точности модели без слоев Dropout

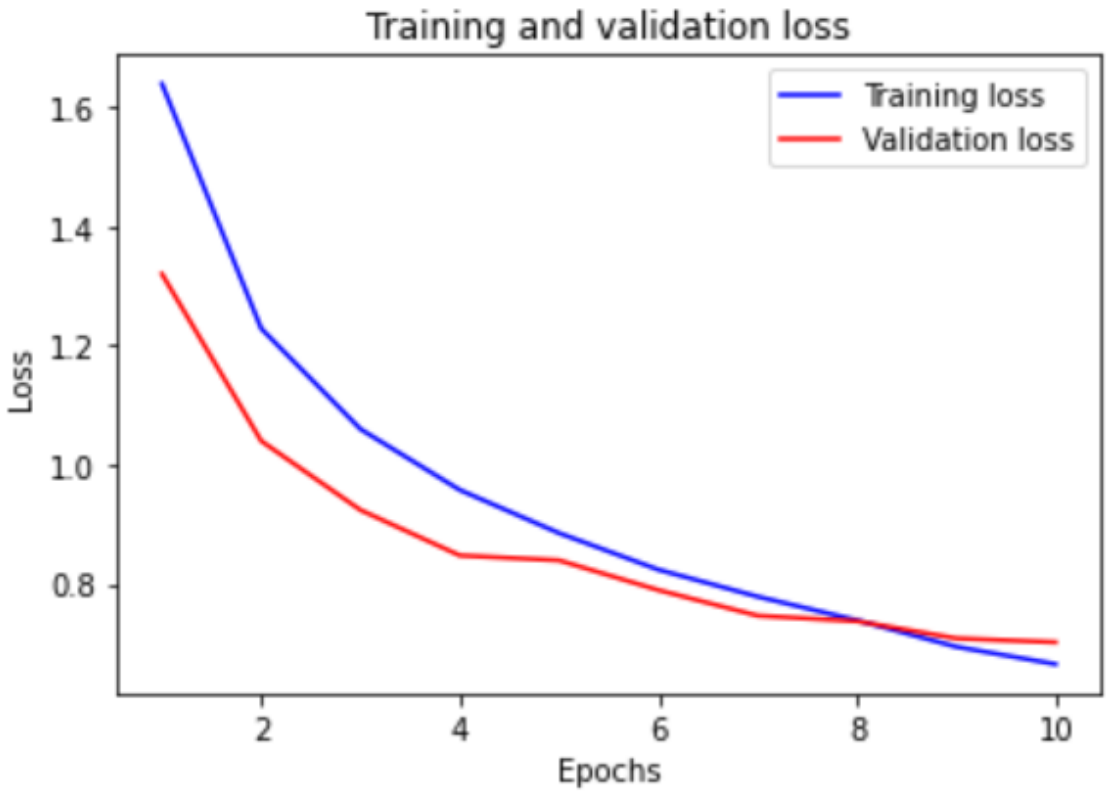

Рисунок 3 – График ошибок модели со слоями Dropout

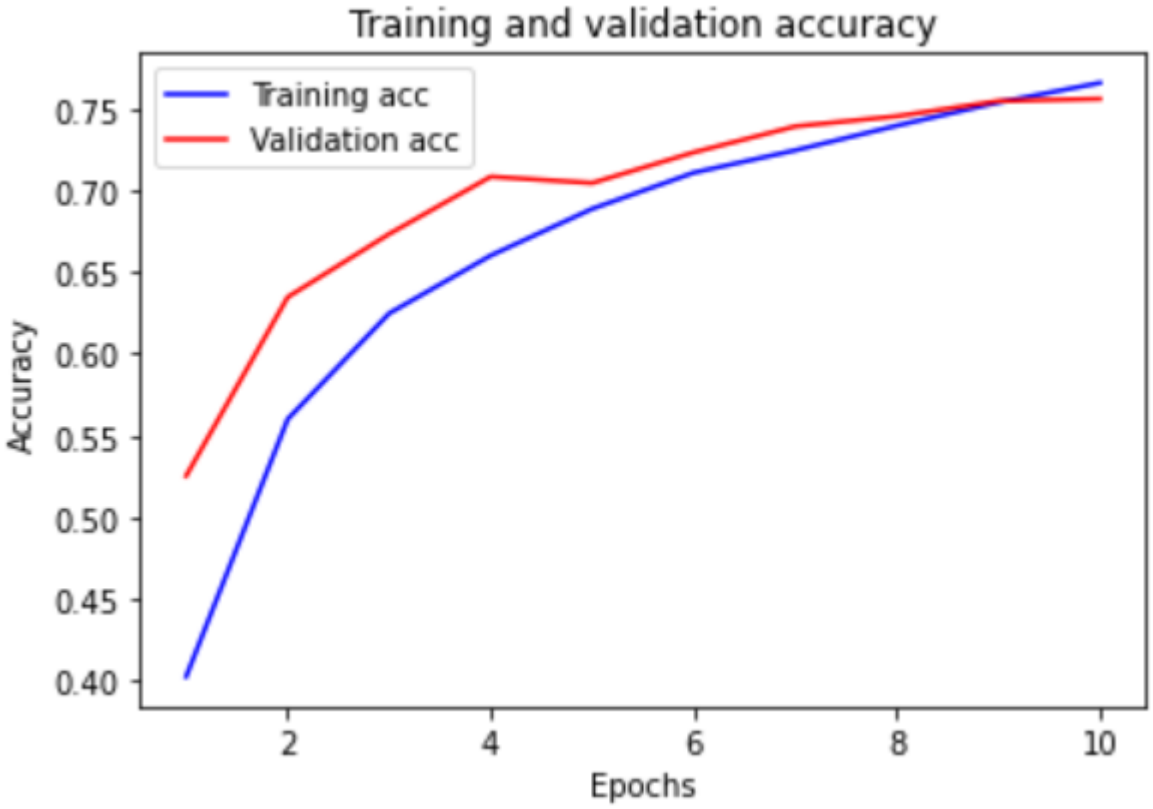

Рисунок 4 – График точности модели со слоями Dropout

Как видно из графиков в обучении модели без слоев разрежения, начиная с 5-ей эпохи, началось переобучение, в то время как со слоями разрежения значения потерь на контрольной выборке не возрастали, что говорит о необходимости добавления слоев разрежения в конфигурацию модели.

Для исследования работы сети при разных размерах ядра свертки будем рассматривать конфигурации моделей с размерами ядер 2x2, 3x3, 4x4. Графики ошибок и точностей для моделей представлены на рис. 5-10.

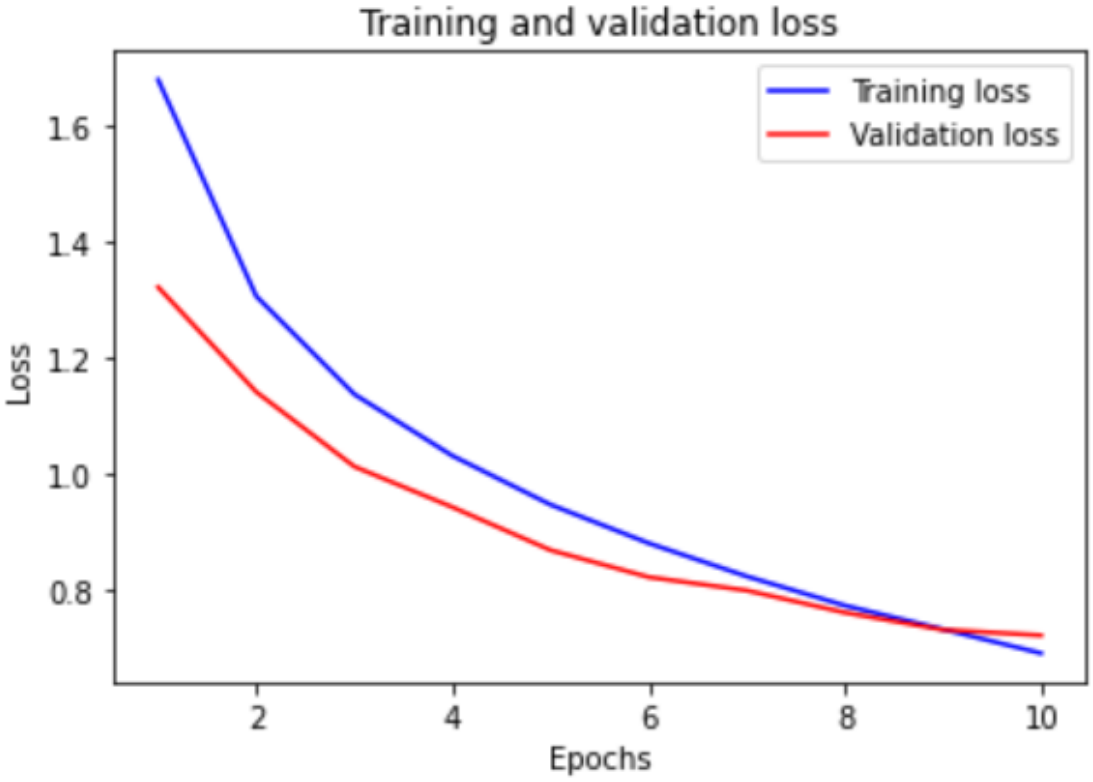

Рисунок 5 – График ошибок модели с размером ядра 2x2

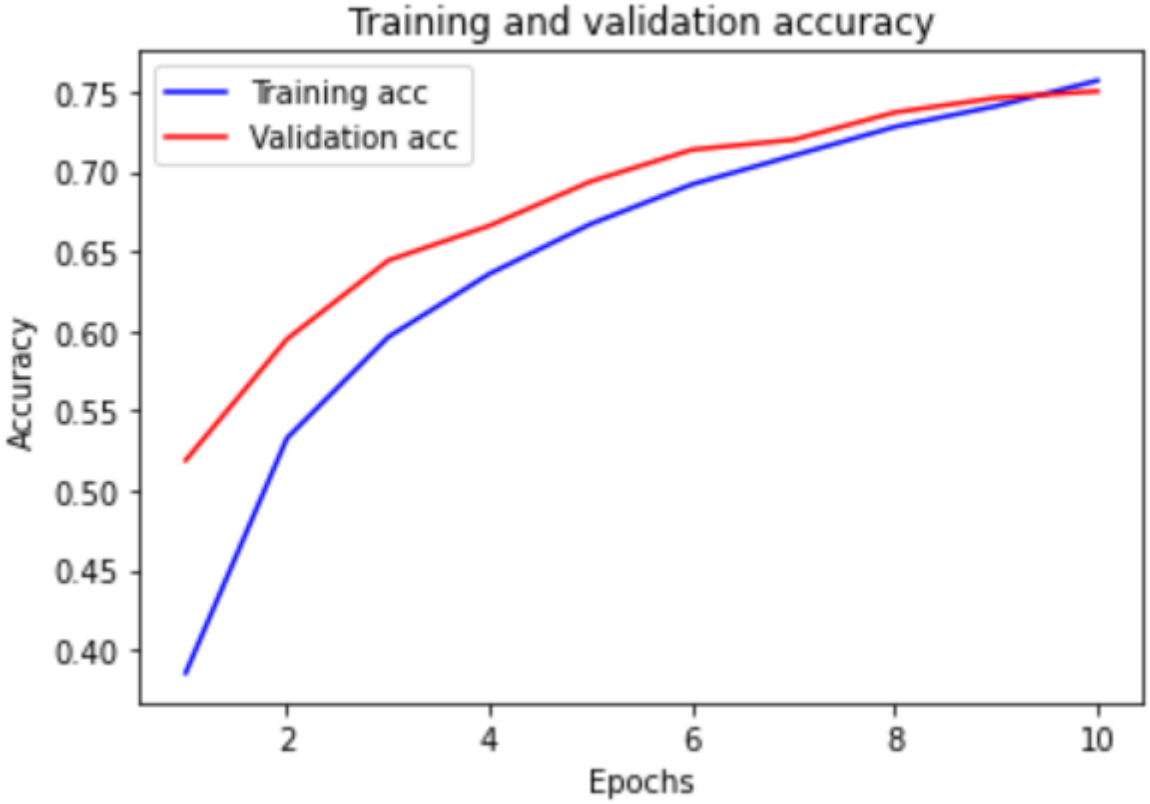

Рисунок 6 – График точности модели с размером ядра 2x2

Точность на контрольной выборке – 0.7505.

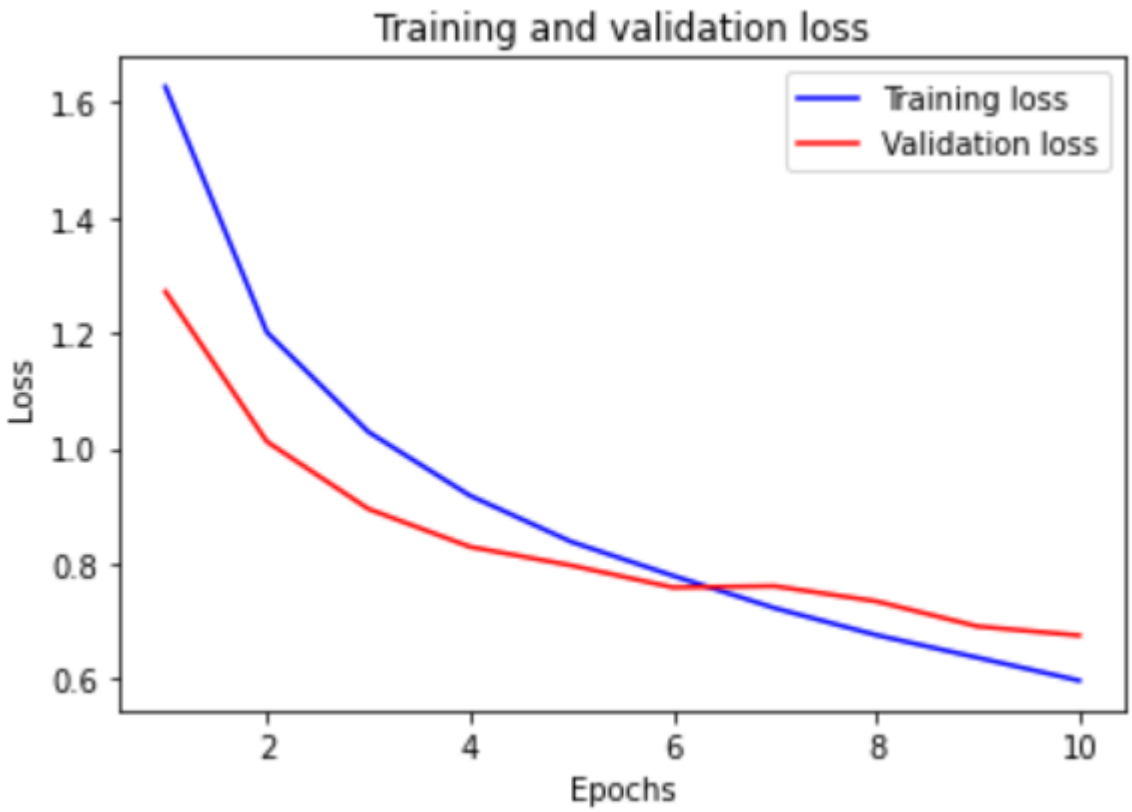

Рисунок 7 – График ошибок модели с размером ядра 3x3

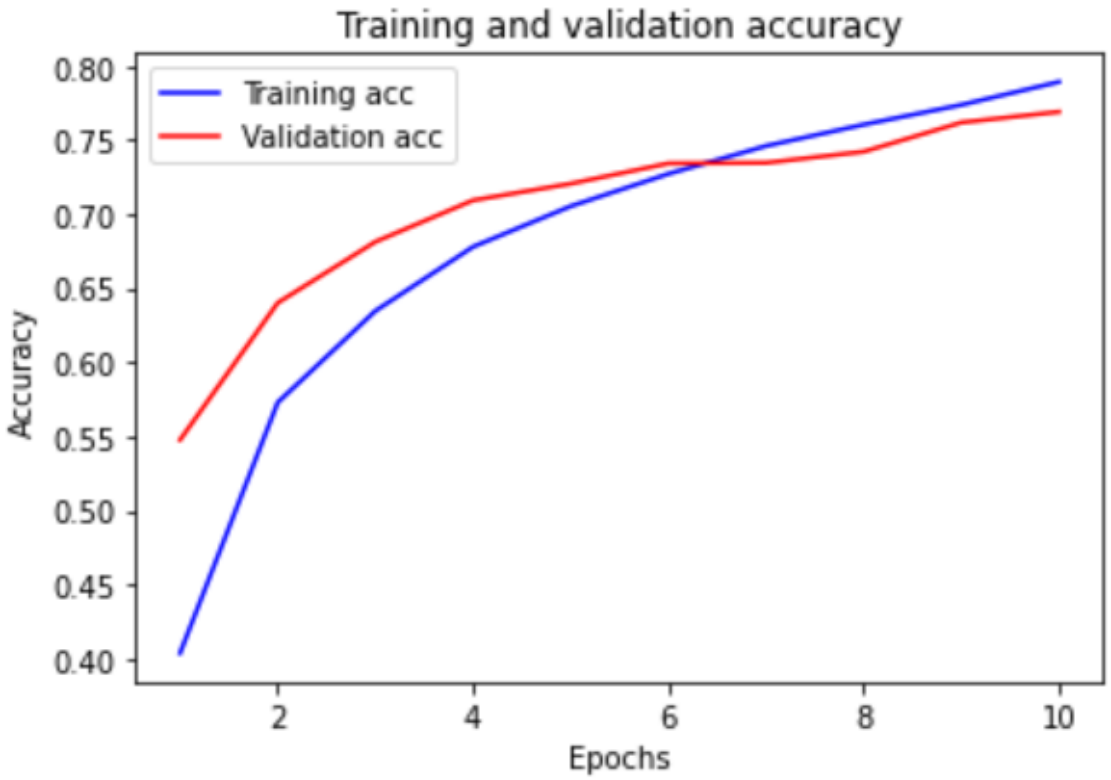

Рисунок 8 – График точности модели с размером ядра 3x3

Точность на контрольной выборке – 0.7689.

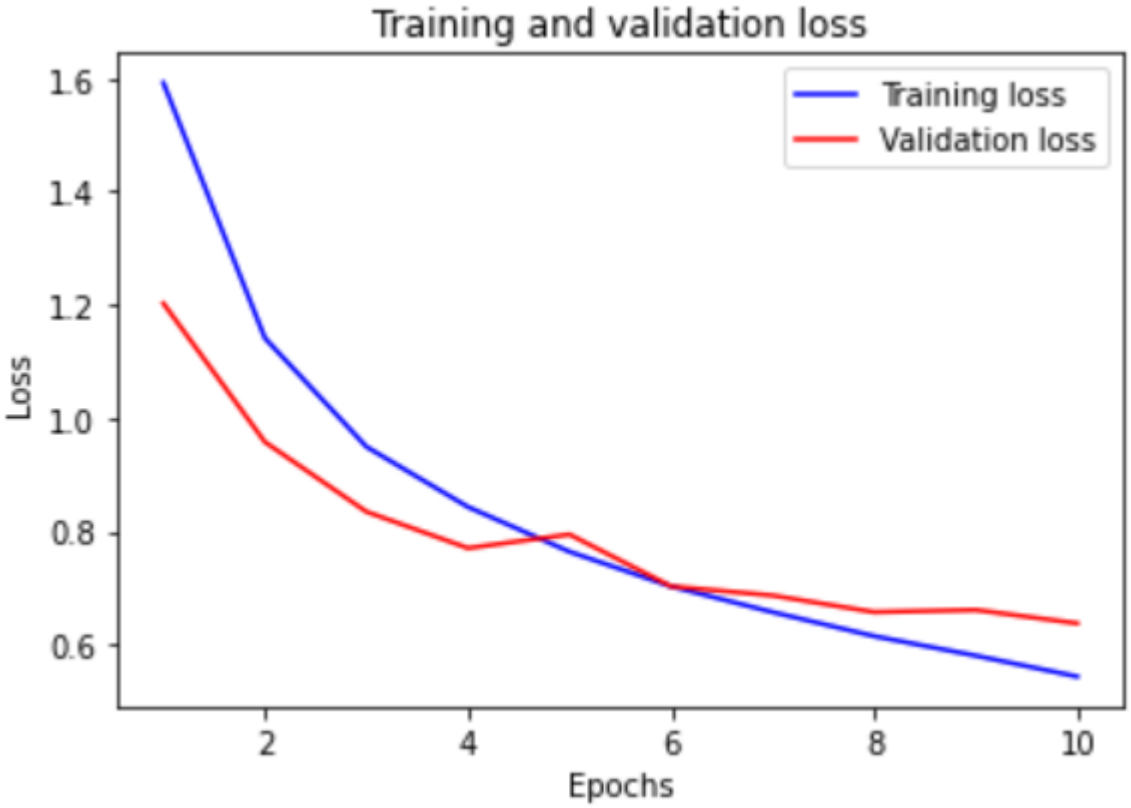

Рисунок 9 – График ошибок модели с размером ядра 4x4

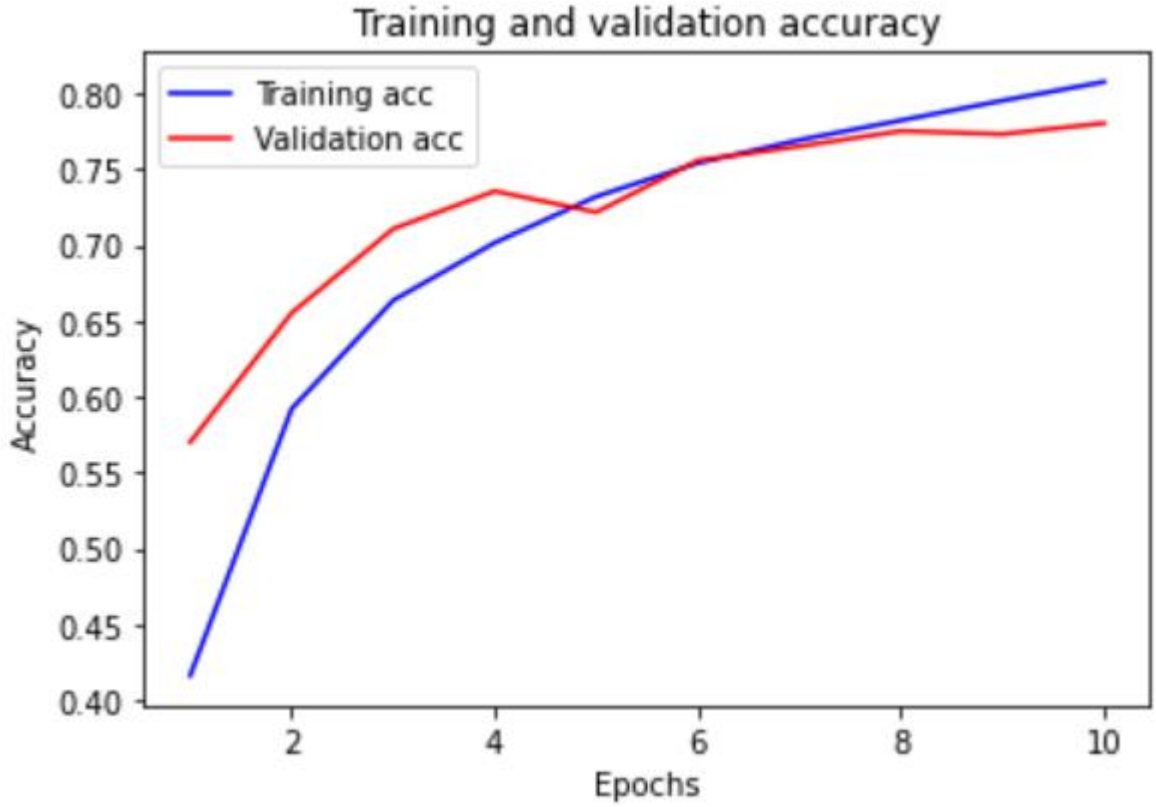

Рисунок 10 – График точности модели с размером ядра 4x4

Точность на контрольной выборке – 0.7802.

По графикам и результатам точности на контрольных выборках можно сделать вывод о том, что для заданных параметров обучения и текущей конфигурации наилучшее значение размерности ядра – 4x4, так как при этой размерности достигается наибольшая итоговая точность сети при обучении и наименьшее значение потерь. Дальнейшее увеличение размерности приводит к уменьшению итоговой точности сети и возрастанию потерь сети.

Выводы.

В ходе выполнения данной работы была создана сверточная нейронная сеть для распознавания объектов на фотографиях, а также исследовано влияние слоев разрежения и размерность ядра на работу сети.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr4.py

```
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 = 150
num_epochs = 10
kernel_size = 4
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 /= 255.0
X_test /= 255.0

# 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), 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)
if with_dropout:
    drop_1 = Dropout(drop_prob_1)(pool_1)
conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
activation='relu')(drop_1 if with_dropout else pool_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)
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(inp, out)
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Fitting model:
h = model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs,
verbose=1, validation_data=(X_test, Y_test))

print(model.evaluate(X_test, Y_test))

# 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()
```