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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр. 7382 _____ Гиззатов А.С.

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).

Основные теоретические положения.

Проблема автоматической идентификации объектов на фотографиях является сложной из-за почти бесконечного количества перестановок объектов, положений, освещения и так далее.

Набор данных CIFAR-10 состоит из 60000 фотографий, разделенных на 10 классов (отсюда и название CIFAR-10). Классы включают в себя общие объекты, такие как самолеты, автомобили, птицы, кошки и так далее. Набор данных разделяется стандартным способом, где 50 000 изображений используются для обучения модели, а остальные 10 000 - для оценки ее производительности.

Фотографии цветные, с красными, зелеными и синими компонентами, но маленькие, размером 32 на 32 пикселя.

Требования к выполнению задания.

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.

Была построена сверточная нейронная сеть.

1. Параметры слоев, компиляции и обучения:

```
2. 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 [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
   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)
# Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
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) # To define a model, just specify its input and
output layers
model.compile(loss='categorical_crossentropy', # using the cross-entropy
loss function
              optimizer='adam', # using the Adam optimiser
              metrics=['accuracy']) # reporting the accuracy
H = model.fit(X_train, Y_train, # Train the model using the training
set...
          batch_size=batch_size, nb_epoch=num_epochs,
          verbose=1, validation_split=0.1)
```

Данная нейронная сеть при 20 эпохах, и размером ядра 3x3 даетт точность ≈ 87% и ошибку ≈ 0.37, что можно увидеть на рис.1,2.

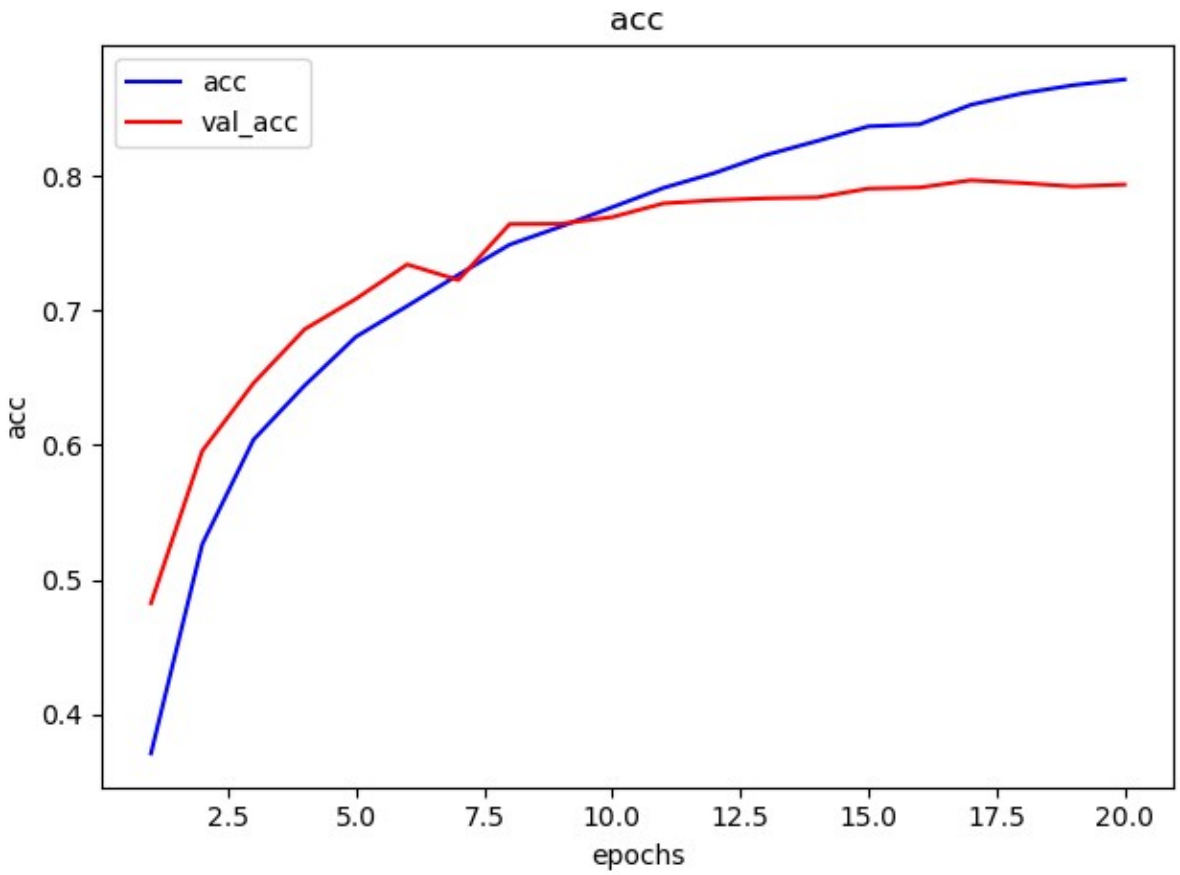

Рис.1 – Точность нейронной сети при размере ядра 3x3 с Dropout слоями.

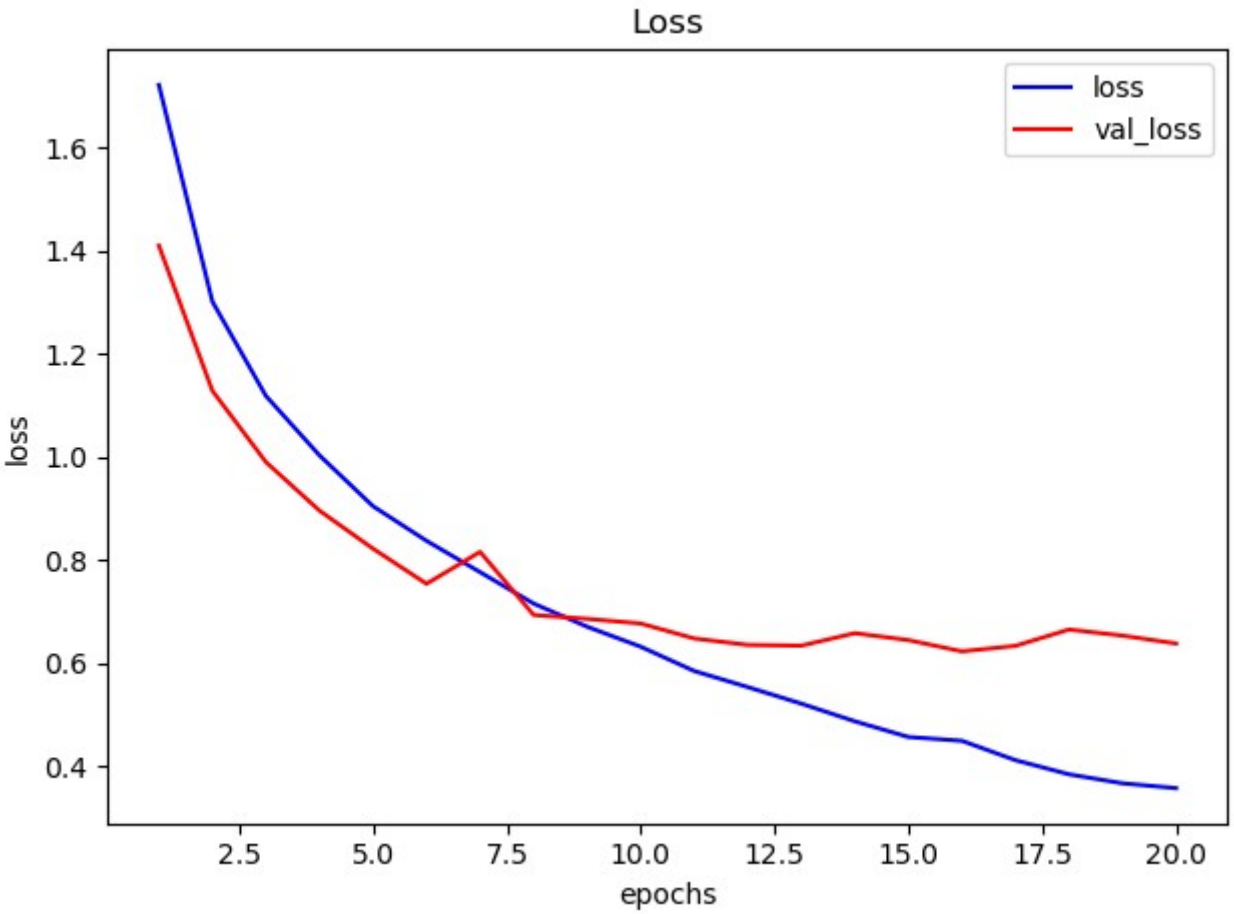

Рис.2 -Потери нейронной сети при размере ядра 3x3 с Dropout слоями.

Теперь уберем dropout слои и посмотрим как это отразится на точности и ошибке. Графики предоставлены на рис.3,4.

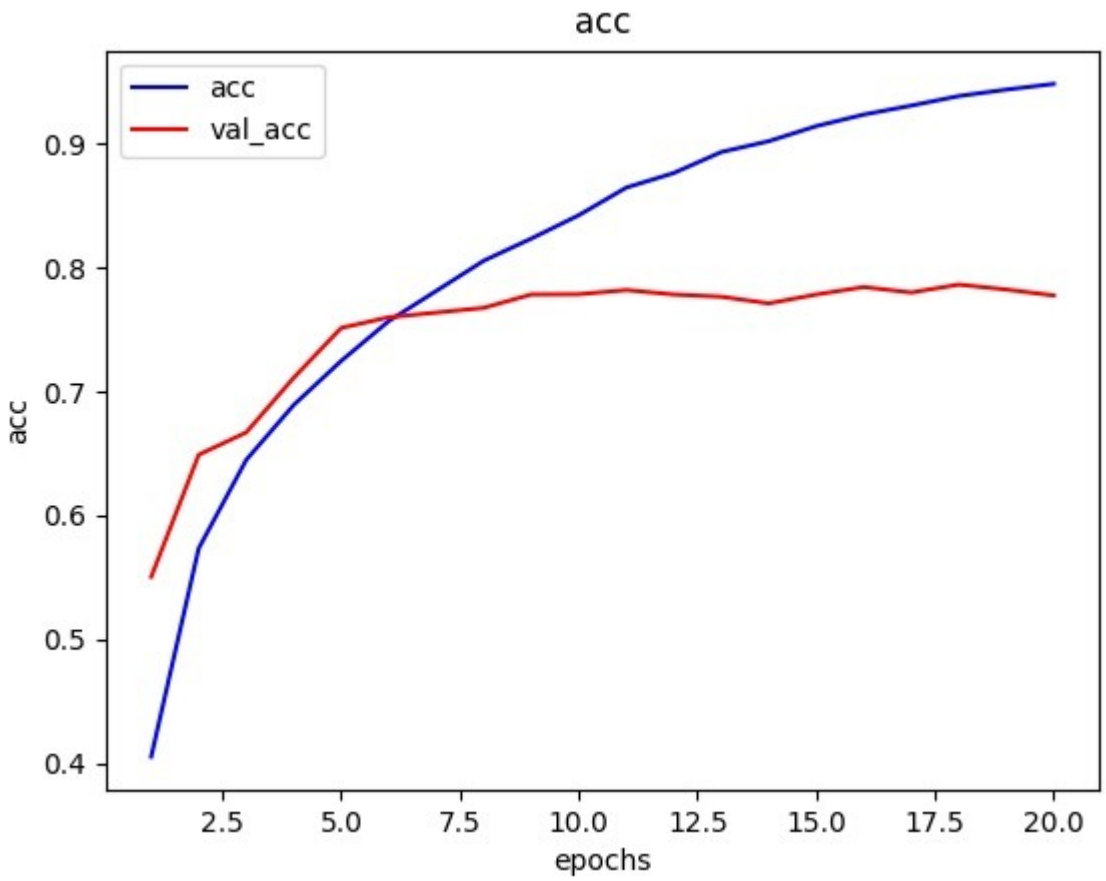

Рис.3 -Точность нейронной сети при размере ядра 3x3 без Dropout слоев.

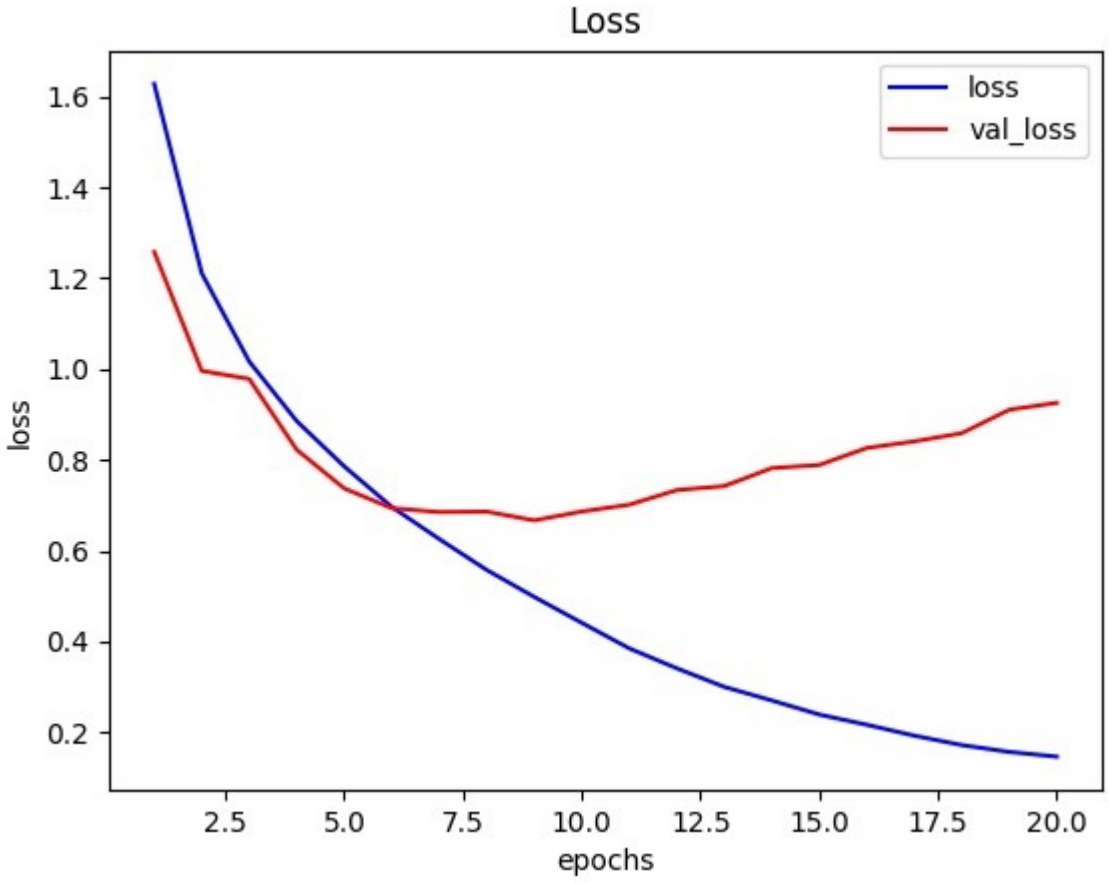

Рис.4 -Ошибка нейронной сети при размере ядра 3x3 без Dropout слоев.

Как можно заметить нейронная сеть выдает хорошую ошибку и точность для обучающей выборки, но маленькую точность и большую ошибку для тестовой выборки, что является последствием переобучения.

Теперь посмотрим как изменяются графики в зависимости от размера ядра свертки. Графики представлены на рис.5-8.

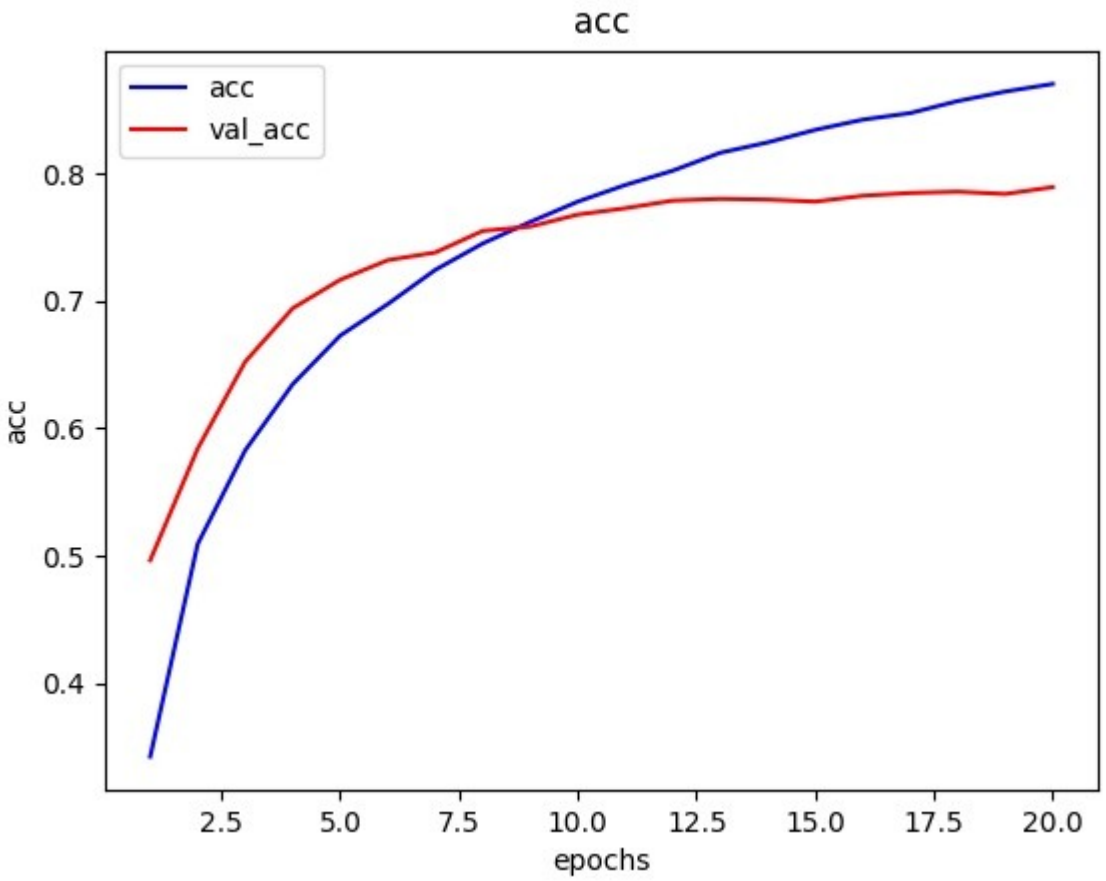

Рис.5 -Точность нейронной сети при размере ядра 5x5.

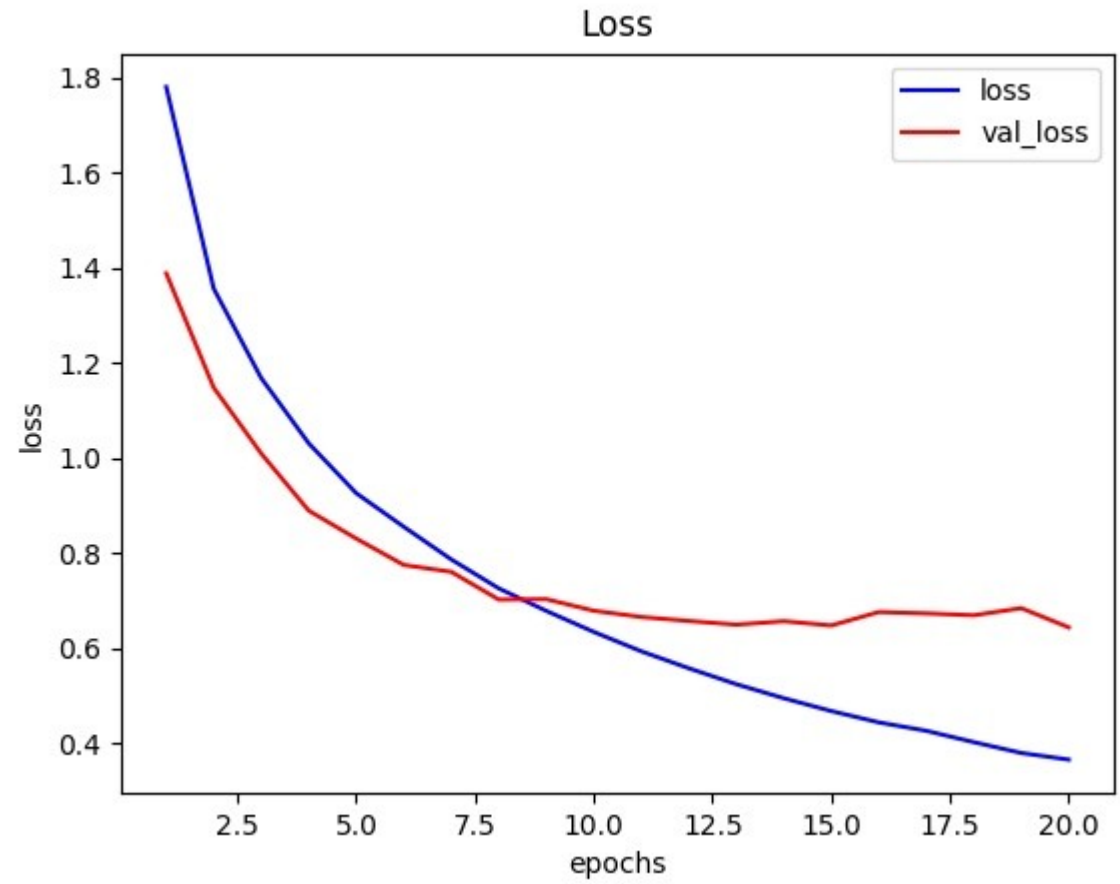

Рис.6 -Ошибка нейронной сети при размере ядра 5x5.

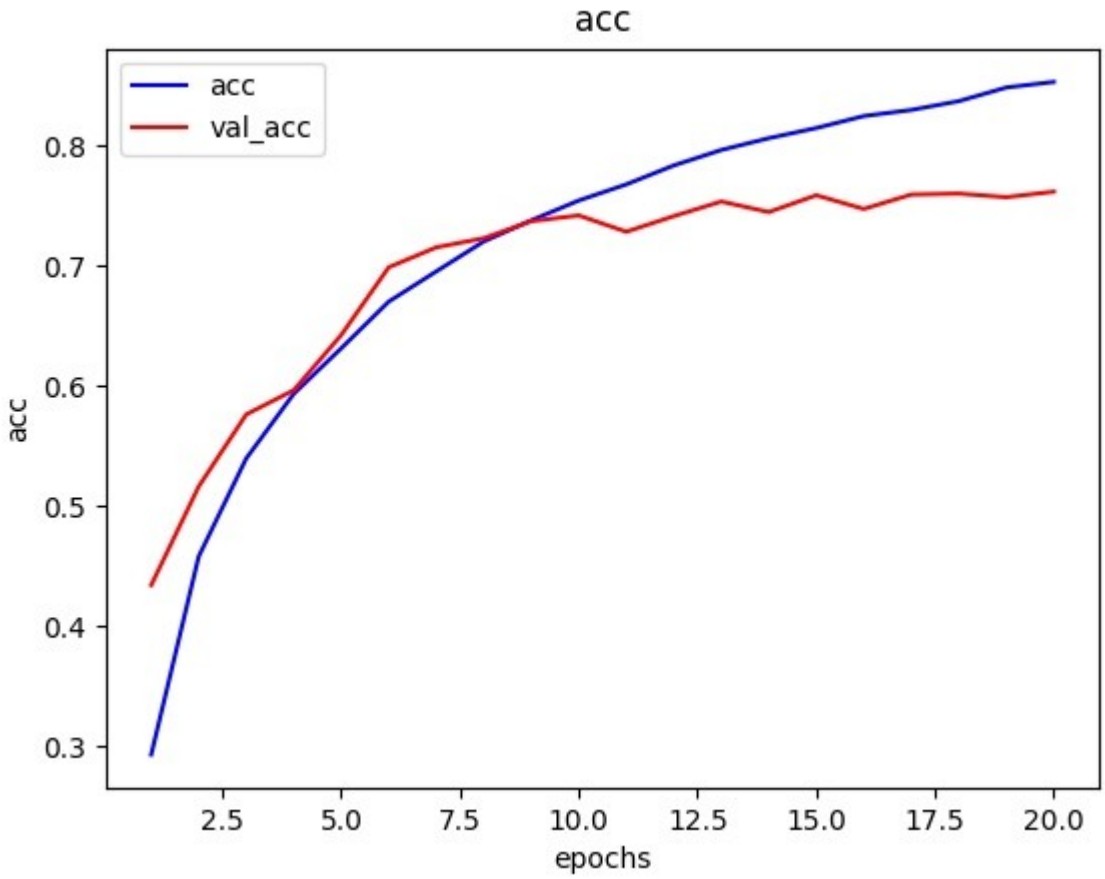

Рис.7 -Точность нейронной сети при размере ядра 7x7.

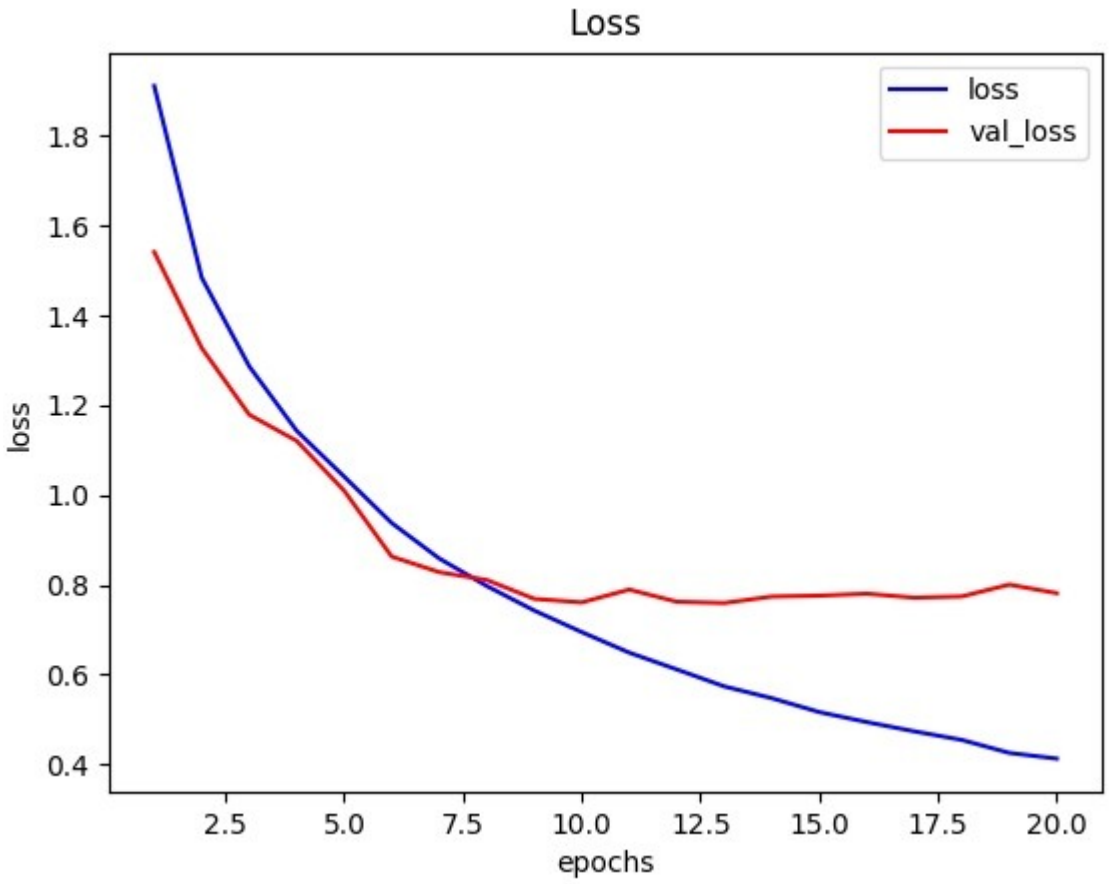

Рис.8 -Ошибка нейронной сети при размере ядра 7x7.

Из графиков можно увидеть, что при размере ядра 7x7 ошибка немного возросла, а точность уменьшилась. Это связано с неудачными признаками, которые были выданы нейронной сетью.

Выводы.

В ходе работы была изучена задача классификация изображений из датасета CIFAR-10. Также изучено влияние Dropout слоев и изменения размера ядра на точность и ошибку н.с.

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 keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt
batch_size = 256 # in each iteration, we consider 32 training examples at once
num_epochs = 20 # we iterate 200 times over the entire training set
kernel_size = 7 # we will use 3x3 kernels throughout
pool_size = 2 # we will use 2x2 pooling throughout
conv_depth_1 = 32 # we will initially have 32 kernels per conv. layer...
conv_depth_2 = 64 # ...switching to 64 after the first pooling layer
drop_prob_1 = 0.25 # dropout after pooling with probability 0.25
drop_prob_2 = 0.5 # dropout in the dense layer with probability 0.5
hidden_size = 512 # the dense layer will have 512 neurons
(X_train, y_train), (X_test, y_test) = cifar10.load_data() # fetch CIFAR-10 data
num_train, height, width,depth = X_train.shape # there are 50000 training examples in CIFAR-10
num_test = X_test.shape[0] # there are 10000 test examples in CIFAR-10
num_classes = np.unique(y_train).shape[0] # there are 10 image classes
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range
Y_train = np_utils.to_categorical(y_train, num_classes) # One-hot encode the labels
Y_test = np_utils.to_categorical(y_test, num_classes) # One-hot encode the labels
```

```
        inp = Input(shape=(height,width,depth)) # N.B. depth goes first in Keras
        # Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
        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 [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
        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)
        # Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
        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) # To define a model, just specify its input and output
layers
        model.compile(loss='categorical_crossentropy', # using the cross-entropy loss
function
                      optimizer='adam', # using the Adam optimiser
                      metrics=['accuracy']) # reporting the accuracy
        H = model.fit(X_train, Y_train, # Train the model using the training set...
                batch_size=batch_size, nb_epoch=num_epochs,
                verbose=1, validation_split=0.1) # ...holding out 10% of the data for
validation
```

model.evaluate(X_test, Y_test, verbose=1) # Evaluate the trained model on the test set!

```
loss = H.history['loss']
val_loss = H.history['val_loss']
x = range(1, 21)
plt.plot(x, loss, 'b', label='loss')
plt.plot(x, val_loss, 'r', label='val_loss')
plt.title('Loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend()
plt.show()
#plt.clf()

acc = H.history['acc']
val_acc = H.history['val_acc']
x = range(1, 21)
plt.plot(x, acc, 'b', label='acc')
plt.plot(x, val_acc, 'r', label='val_acc')
plt.title('acc')
plt.ylabel('acc')
plt.xlabel('epochs')
plt.legend()
plt.show()
#plt.clf()
```