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

Студентка гр. 7382 _____ Петрова А.

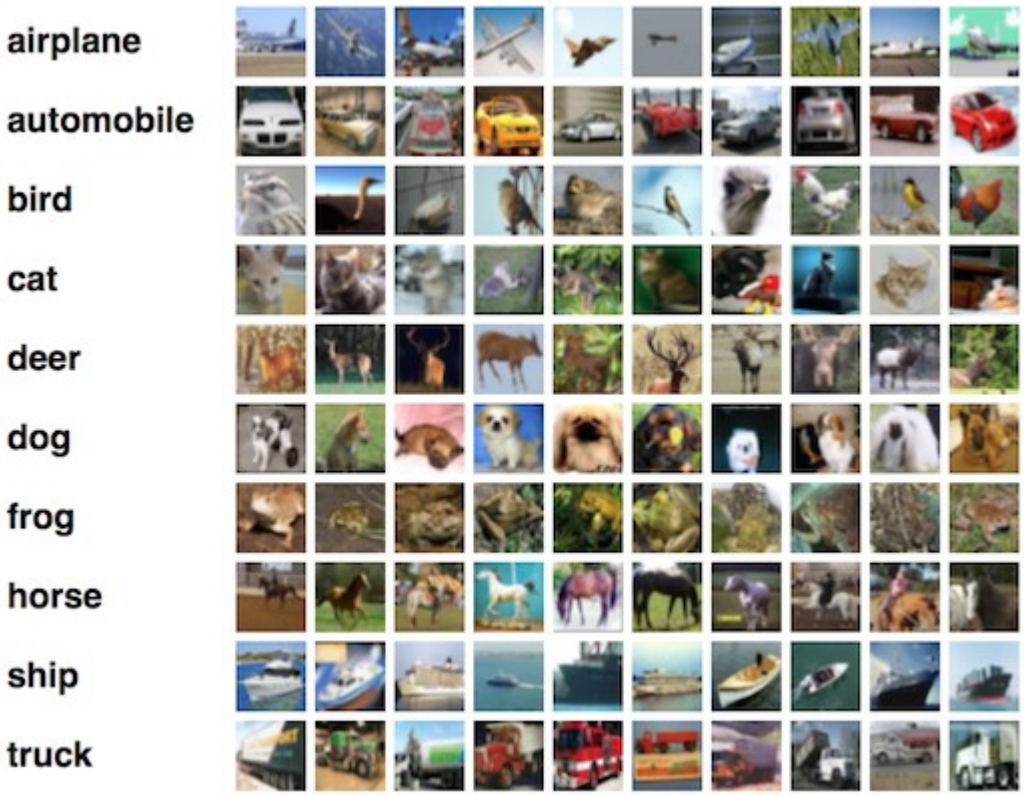
Преподаватель _____ Жукова Н.А.

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).



Требования к выполнению задания

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Задачи

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

Ход работы

Была создана и обучена модель искусственной нейронной сети, код которой представлен в Приложении А. Архитектура нейронной сети основана на исходных данных к лабораторной работы. Параметры:

batch_size = 256

num_epochs = 20

kernel_size = 3

pool_size = 2

conv_depth_1 = 32

conv_depth_2 = 64

drop_prob_1 = 0.25

drop_prob_2 = 0.5

hidden_size = 512

После обучения были получены результаты, представленные на рис. 1 и рис.2.

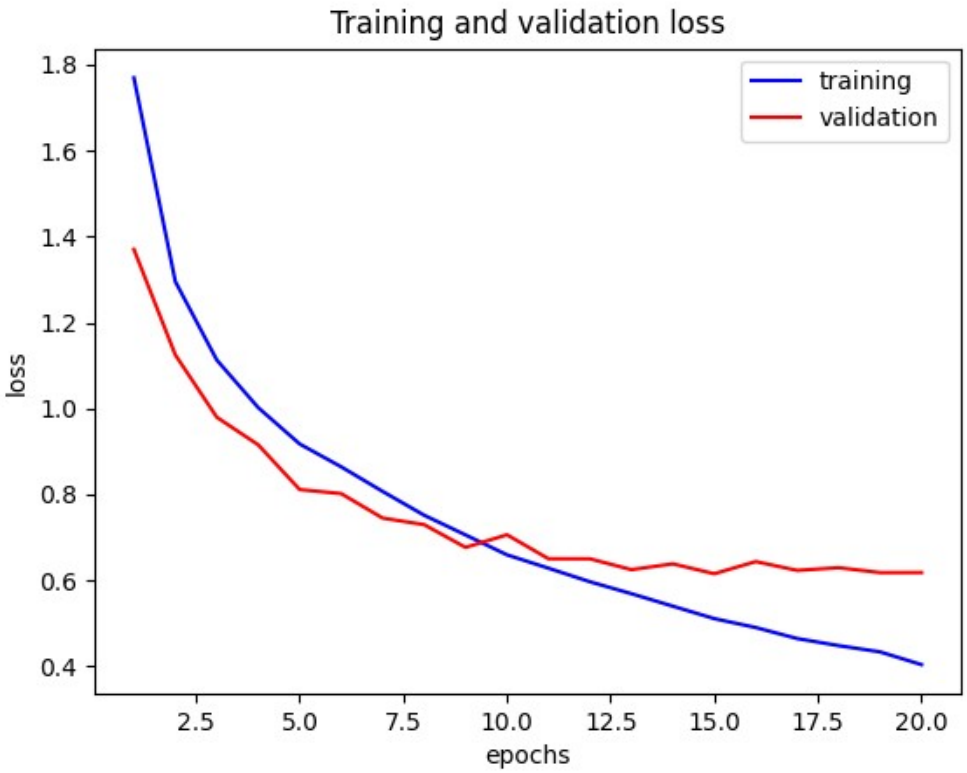


Рисунок 1 — График потерь для ядра 3x3

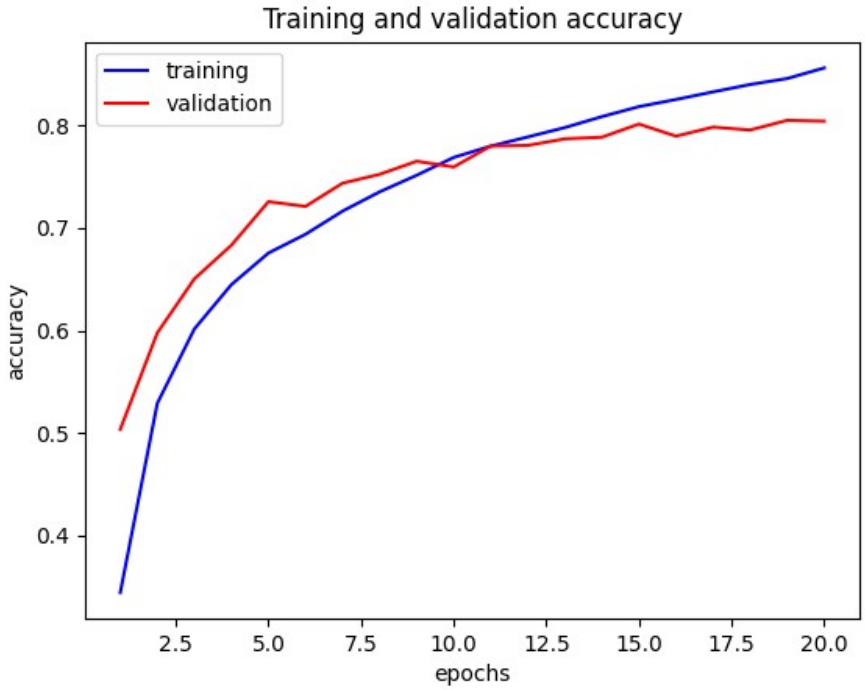


Рисунок 2 — График точности для ядра 3x3

Была получена ~80% точность

Был убран слой dropout, полученные результаты представлены на рисунках 3 и 4.

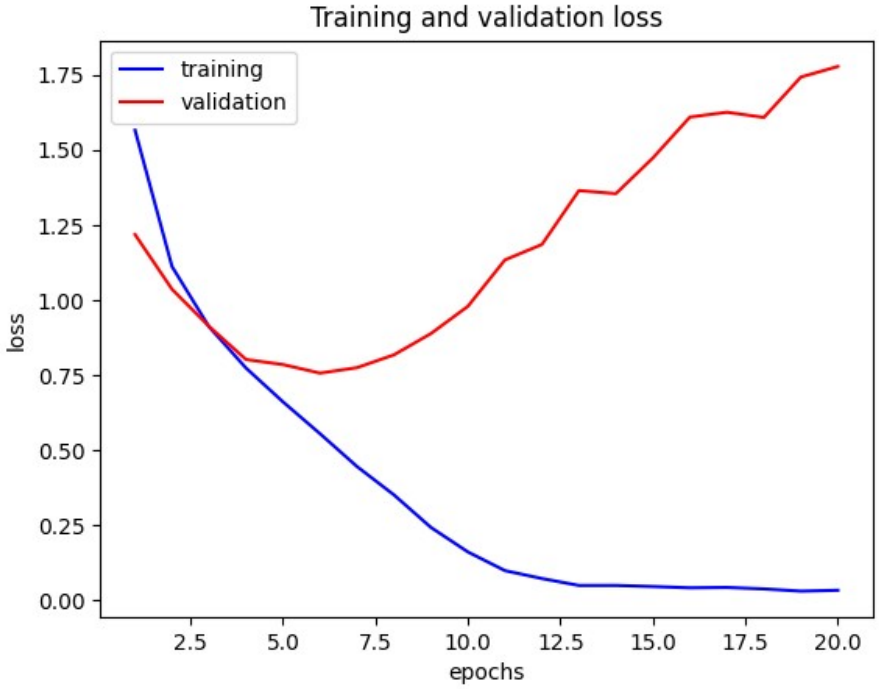


Рисунок 3 — График потерь без dropout

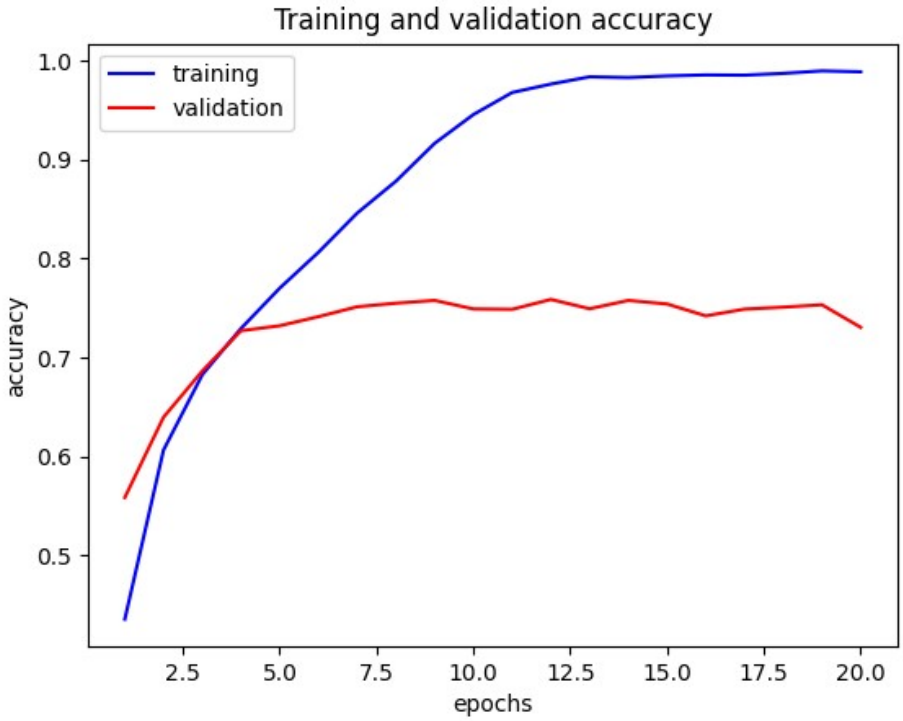


Рисунок 4 - График точности без dropout

Модель начала переобучаться примерно на 5 эпохе. Это происходит, потому что модель начинает обращать внимание на шум.

Теперь изменим размер свертки ядра с 3x3 на 7x7. Полученные результаты представлены на рисунках 5 и 6.

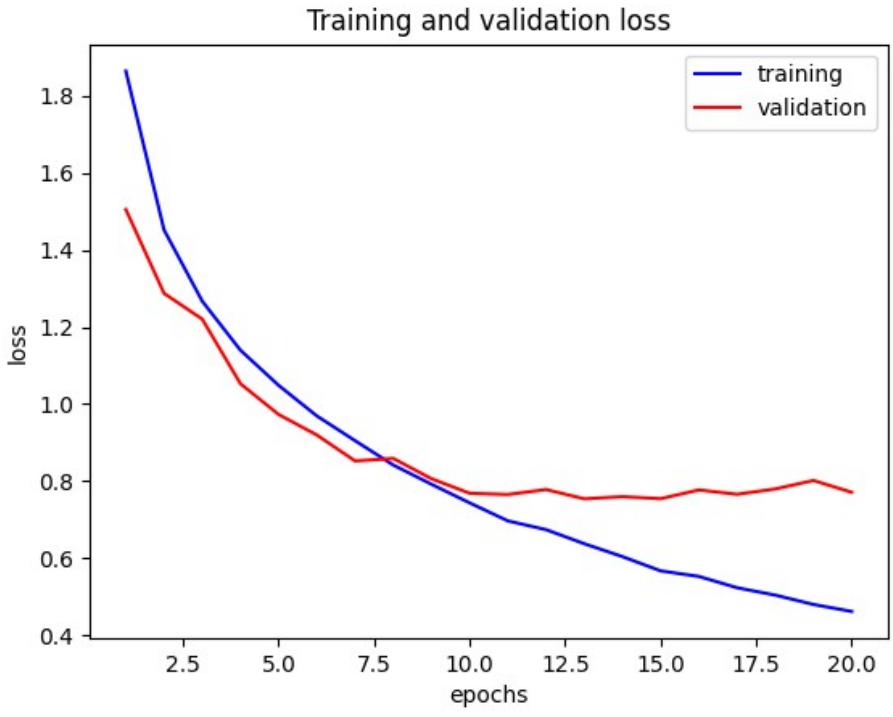


Рисунок 5 — График потерь для ядра 7x7

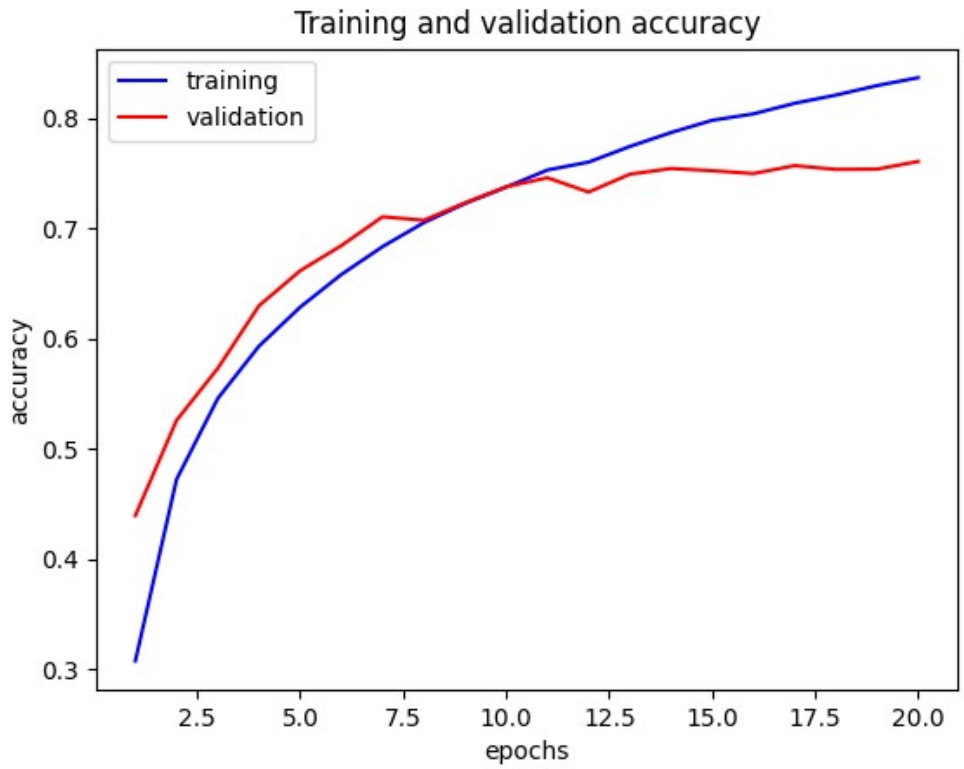


Рисунок 6 — График точности для ядра 7x7

Точность ~75%.

При увеличении размера ядра свертки заметно увеличилось время обучения нейронной сети, а так же уменьшается точность и возрастает ошибка.

Вывод.

В ходе выполнения лабораторной работы было реализовано распознавание объектов на фотографиях CIFAR-10 с помощью искусственных нейронных сетей. Было проверено влияние слоев dropout и изменение ядра свертки на результат.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
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='training')
plt.plot(x, val_loss, 'r', label='validation')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()

acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
x = range(1, 21)
plt.plot(x, acc, 'b', label='training')
plt.plot(x, val_acc, 'r', label='validation')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
```

```
plt.xlabel('epochs')
plt.legend()
plt.show()
```