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. 7381 _____ Трушников А.П.

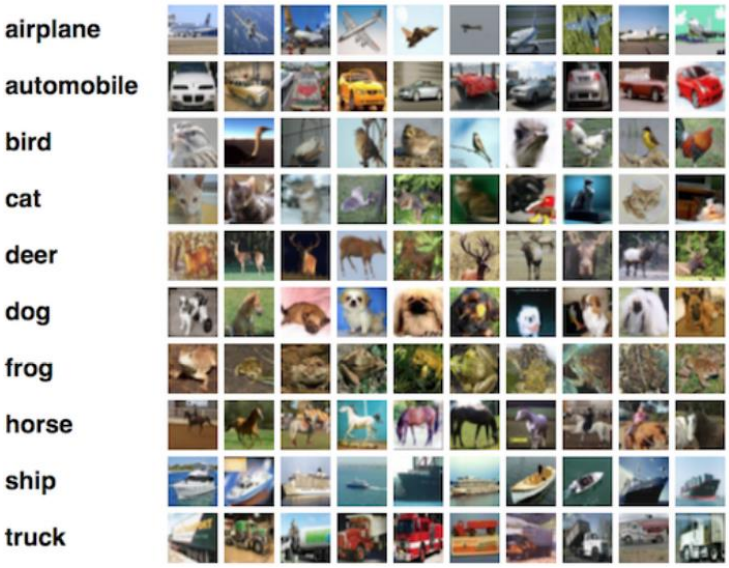
Преподаватель _____ Жукова Н.А..

Санкт-Петербург

2020

Цель работы.

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. Построить и обучить сверточную нейронную сеть

   Была найдена архитектура, которая даёт точность $\approx 60\%$, параметры которой представлены в таблице 1–2.

Таблица 1

Оптимизатор	Функция потерь	Метрика качества обучения сети	Число эпох	batch_size
Adam	Categorical_crossentropy	accuracy	20	32

Таблица 2

Номер слоя	Описание слоя
1	inp = Input(shape=(depth, height, width))
2	conv_1 = Convolution2D(32, 3, 3, border_mode='same', activation='relu')(inp)
3	conv_2 = Convolution2D(32, 3, 3, border_mode='same', activation='relu')(conv_1)
4	pool_1 = MaxPooling2D(pool_size=(2, 2))(conv_2)
5	drop_1 = Dropout(0.35)(pool_1)
6	conv_3 = Convolution2D(64, 3, 3, border_mode='same', activation='relu')(drop_1)
7	conv_4 = Convolution2D(64, 3, 3, border_mode='same', activation='relu')(conv_3)
8	pool_2 = MaxPooling2D(pool_size=(2, 2))(conv_4)
9	drop_2 = Dropout(0.35)(pool_2)
10	flat = Flatten()(drop_2)
11	hidden = Dense(512, activation='relu')(flat)
12	drop_3 = Dropout(0.5)(hidden)
13	out = Dense(num_classes, activation='softmax')(drop_3)

Графики точности и ошибки представлены на рис.1–2.

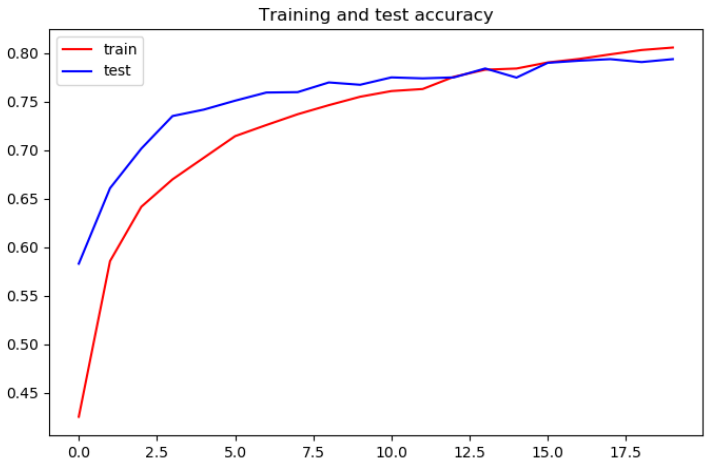


Рисунок 1 – График точности построенной сети

Рисунок 2 – График ошибок построенной сети

2. Исследовать работу сеть без слоя Dropout. Результаты представлены на рис.3–4

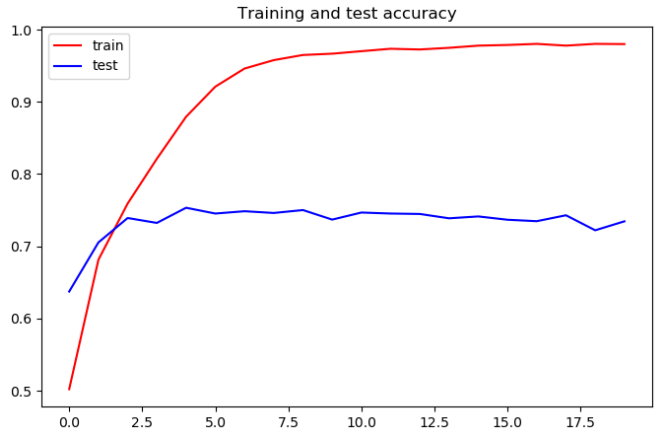


Рисунок 3 – График точности построенной сети без dropout

Рисунок 4 – График ошибок построенной сети без dropout

Как видно из графиков переобучение начинается после 2 эпохи.

Следовательно использование слоёв dropout необходимо.

3. Исследовать работу сети при разных размерах ядра свертки. Результаты для размеров $2\times2, 3\times3, 5\times5$ представлены на рис. 5–10.

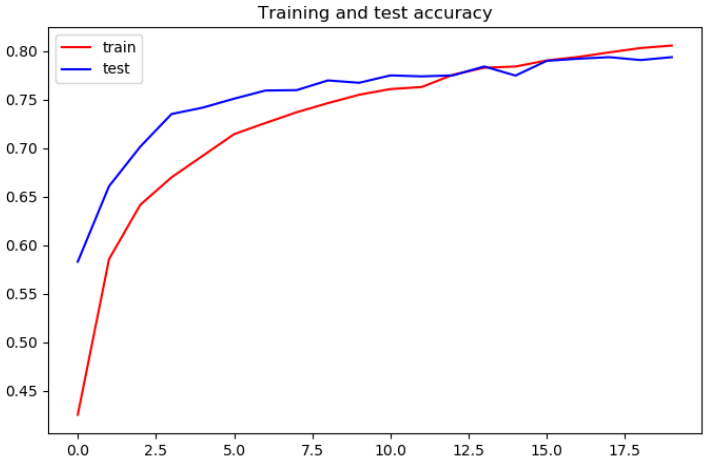


Рисунок 5 – График точности построенной сети 2x2

Рисунок 6 – График ошибок построенной сети 2x2

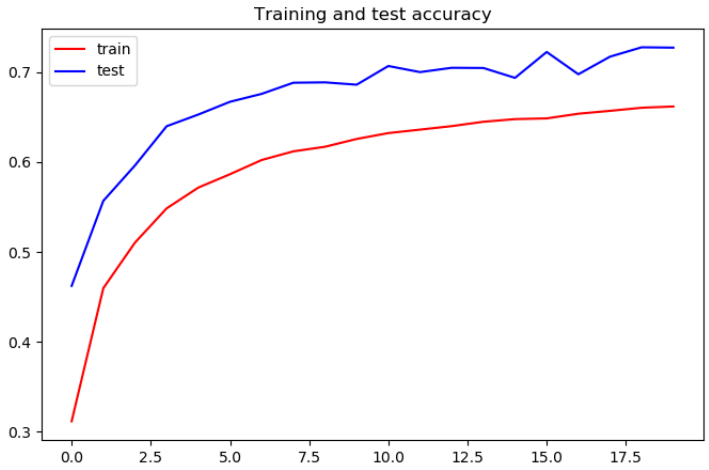


Рисунок 7 – График точности построенной сети 5x5

Рисунок 8 – График ошибок построенной сети 5x5

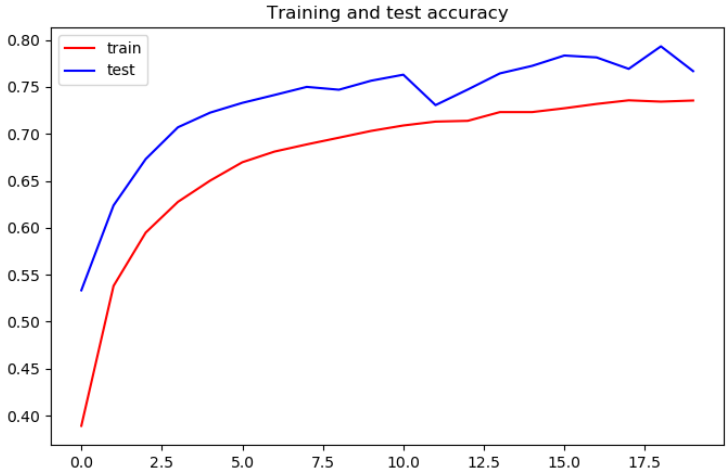


Рисунок 9 – График точности построенной сети 3x3

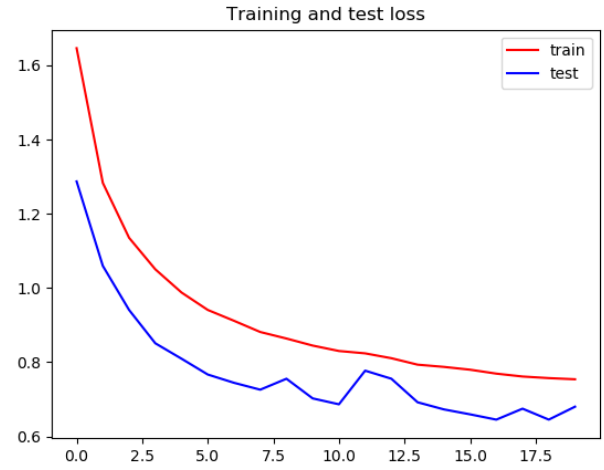


Рисунок 10 – График ошибок построенной сети 3x3

При размере ядер $2\times2,3\times3,5\times5$ точность составила 60%, 55%. 37% соответственно. Следовательно выбранная архитектура при размере ядер 2×2 является наилучшей.

Выводы.

В ходе выполнения данной работы ознакомились со сверточными нейронными сетями, также изучена работа слоя разреживания Dropout, изучено влияние размера ядра на архитектуру сети.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout,
Flatten
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)
pool_size = 3 # we will use 2x2 pooling throughout

(X_train, y_train), (X_test, y_test) = cifar10.load_data() # fetch CIFAR-10
data


num_train, depth, height, width = 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=(depth, height, width)) # N.B. depth goes first in Keras


# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(32, 3, 3, border_mode='same', activation='relu')(inp)
conv_2 = Convolution2D(32, 3, 3, border_mode='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(0.35)(pool_1)


# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(64, 3, 3, border_mode='same', activation='relu')(drop_1)
conv_4 = Convolution2D(64, 3, 3, border_mode='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(0.35)(pool_2)

# Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(512, activation='relu')(flat)
drop_3 = Dropout(0.5)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
```

```
model = Model(input=inp, output=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=32, nb_epoch=20,
          verbose=0, validation_split=0.1) # ...holding out 10% of the data for
validation

score = model.evaluate(X_test, Y_test, verbose=0) # Evaluate the trained model
on the test set!

print('Test loss:', score[0])
print('Test accuracy:', score[1])

plt.figure(1, figsize=(8, 5))
plt.title("Training and test accuracy")
plt.plot(h.history['acc'], 'r', label='train')
plt.plot(h.history['val_acc'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()

plt.figure(1, figsize=(8, 5))
plt.title("Training and test loss")
plt.plot(h.history['loss'], 'r', label='train')
plt.plot(h.history['val_loss'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()
```