Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Лабораторная работа №3 по дисциплине**

«Проектирование интеллектуальных систем»

**ИСПОЛНИТЕЛЬ:**

Березин И.С.

Группа ИУ5-23М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2019 г.

Москва, 2019

1. **Задание**

Обучить нейронную сеть на наборе данных CIFAR10. Точность модели должна достигать 70%. Сеть должна состоять из трех сверхточных слоев и полносвязной сети.

1. **Код программы**

import tensorflow as tf

from tensorflow.python.keras.datasets import cifar10

import numpy as np

import time

from keras.utils import np\_utils

from tensorflow.python import debug as tf\_debug

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

nb\_classes = 10

print(y\_train.shape)

y\_train = y\_train.reshape(y\_train.shape[0]) # somehow y\_train comes as a 2D nx1 matrix

y\_test = y\_test.reshape(y\_test.shape[0])

x\_train = X\_train.astype('float32')

x\_test = X\_test.astype('float32')

x\_train /= 255

x\_test /= 255

x\_train = np.reshape(x\_train, [-1, 3072])

x\_test = np.reshape(x\_test, [-1, 3072])

y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

import os

def weight\_variable(shape): # создание переменной весов слоя

initial = tf.truncated\_normal(shape, stddev=0.1)

return tf.Variable(initial)

def bias\_variable(shape):

initial = tf.constant(0.1, shape=shape)

return tf.Variable(initial)

def conv2d(x\_, W): # операция свертки

return tf.nn.conv2d(x\_, W, strides=[1, 1, 1, 1], padding='SAME')

def max\_pool\_2x2(x\_): # операция пулинга

return tf.nn.max\_pool(x\_, ksize=[1, 2, 2, 1],

strides=[1, 2, 2, 1], padding='SAME')

def conv\_layer(input, shape): # сверточный слой

W = weight\_variable(shape) # переменная для весов свертки

b = bias\_variable([shape[3]]) # свободные члены = количество фильтров

return tf.nn.relu(conv2d(input, W) + b)

def full\_layer(input, size):

in\_size = int(input.get\_shape()[1])

W = weight\_variable([in\_size, size])

b = bias\_variable([size])

return tf.matmul(input, W) + b

x = tf.placeholder(tf.float32, shape=[None, 3072]) # создаем переменную для x

y\_ = tf.placeholder(tf.float32, shape=[None, 10])

keep\_prob = tf.placeholder(tf.float32) # переменная для dropout

x\_image = tf.reshape(x, [-1, 32, 32, 3]) # преобразование в размеры картинки

with tf.name\_scope('conv\_1'):

conv1 = conv\_layer(x\_image, shape=[3, 3, 3, 32]) # создание сверточного слоя

conv1\_pool = max\_pool\_2x2(conv1) # слой пулинга

with tf.name\_scope('conv\_2'):

conv2 = conv\_layer(conv1\_pool, shape=[3, 3, 32, 64])

conv2\_pool = max\_pool\_2x2(conv2)

with tf.name\_scope('conv\_3'):

# conv3 = conv\_layer(conv2, shape=[3, 3, 64, 128])

conv3 = conv\_layer(conv2\_pool, shape=[3, 3, 64, 128])

conv3\_pool = max\_pool\_2x2(conv3)

conv3\_flat = tf.reshape(conv3\_pool, [-1, 4\*4\*128])

with tf.name\_scope('full\_1'):

full\_1 = tf.nn.relu(full\_layer(conv3\_flat, 512))

with tf.name\_scope('dropout'):

full1\_drop = tf.nn.dropout(full\_1, keep\_prob=keep\_prob)

with tf.name\_scope('activations'):

y\_conv = full\_layer(full1\_drop, 10)

tf.summary.scalar('cross\_entropy\_loss',y\_conv)

with tf.name\_scope('cross'):

cross\_entropy = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits\_v2(logits=y\_conv, labels=y\_))

# #SGD

train\_step = tf.train.AdamOptimizer(learning\_rate=0.0005).minimize(cross\_entropy)

correct\_prediction = tf.equal(tf.argmax(y\_conv, 1), tf.argmax(y\_, 1))

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction , tf.float32))

with tf.Session() as sess:

# write = tf.summary.FileWriter('./event\_logs\_1', sess.graph)

sess.run(tf.global\_variables\_initializer())

start\_time = time.time()

for epochs in range(5):

for i in range(2500):

batch = [x\_train[20\*i:20\*(i+1)] , y\_train[20\*i:20\*(i+1)]]

if i % 250 == 0:

train\_accuracy = sess.run(accuracy, feed\_dict={x: batch[0],

y\_: batch[1], keep\_prob: 1.0})

print("time {}, step {}, training accuracy {}".format(time.time() - start\_time,i, train\_accuracy))

sess.run(train\_step, feed\_dict={x: batch[0], y\_: batch[1], keep\_prob: 0.5})

print('-----------------------------------------------Эпоха№', epochs)

# saver.save(sess, os.path.join('logs', 'images.ckpt'))

X = x\_test.reshape(10, 1000, 3072)

Y = y\_test.reshape(10, 1000, 10)

test\_accuracy = np.mean([sess.run(accuracy, feed\_dict={x:X[i], y\_:Y[i], keep\_prob:1.0}) for i in range(10)])

print("test accuracy: {}".format(test\_accuracy))

**Результат выполнения:**

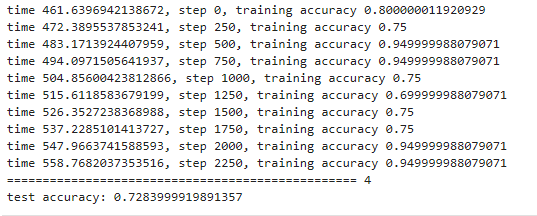


Рисунок 1. Результат выполнения программы распознавания набора данных CIFAR10

**Список литературы**

* Черненький И. М., Методические указания к лабораторной работе №3.
* Николенко С.И., Кадурин А.А., Архангельская Е.О. Глубокое обучение. – Издательский дом "Питер", 2019. — 480 с.: ил. — (Серия «Библиотека программиста»).