Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Лабораторная работа №4 по дисциплине**

«Проектирование интеллектуальных систем»

**ИСПОЛНИТЕЛЬ:**

Чечнев А.А.

Группа ИУ5-23М

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

"\_\_"\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Москва 2020

1. **Задание**

Модифицировать программный код лабораторной №3 с добавлением сохранения модели и сохранения сводных статистик для изучения Tensorboard. Написать дополнительный код, который покажет демонстрацию восстановления модели из файла с расширением ckpt.

1. **CIFAR10**

**Импорт библиотек**

%tensorflow\_version 1.x

import tensorflow as tf

tf.\_\_version\_\_

from tensorflow.examples.tutorials.mnist import input\_data

import numpy as np

import time

from tensorflow.keras.datasets import cifar10

from sklearn.preprocessing import OneHotEncoder

**Класс загрузки cifar**

class cif10:

    \_cursor = 0

    def \_\_init\_\_(self):

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

        self.X\_train = self.X\_train / 255

        self.X\_test = self.X\_test / 255

        self.X\_train = self.X\_train.reshape([-1, 3072]) #32\*32\*3

        self.X\_test = self.X\_test.reshape([-1, 3072])

        ohe = OneHotEncoder()

        self.y\_train = ohe.fit\_transform(self.y\_train).toarray()

        self.y\_test = ohe.fit\_transform(self.y\_test).toarray()

    def next\_batch\_train(self, batch\_size=50):

        res = [self.X\_train[self.\_cursor : self.\_cursor + batch\_size],

               self.y\_train[self.\_cursor : self.\_cursor + batch\_size]]

        self.\_cursor = self.\_cursor + batch\_size

        if (self.\_cursor+batch\_size > self.X\_train.shape[0]):

            self.\_cursor = 0

        return res

**Функции, требуемые для создания слоев**

def weight\_variable(shape):

  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):

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

def max\_pool\_2x2(x):

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

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

def conv\_layer(input, shape):

  W = weight\_variable(shape)

  b = bias\_variable([shape[3]])

  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

def variable\_summaries(var):  # переменные, записываемые в summary для tensorboard

    with tf.name\_scope('summaries'):

        mean = tf.reduce\_mean(var)

        tf.summary.scalar('mean', mean)

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

            stddev = tf.sqrt(tf.reduce\_mean(tf.square(var - mean)))

        tf.summary.scalar('stddev', stddev)

        tf.summary.scalar('max', tf.reduce\_max(var))

        tf.summary.scalar('min', tf.reduce\_min(var))

        tf.summary.histogram('histogram', var)

**Изменяемые гиперпараметры**

epochs = 14

n\_samples = 50000

batch\_size = 25

learning\_rate = 1e-2

neurons\_flat = 512

#MAIN\_DIR = '/content/lab4'

#LOG\_DIR = MAIN\_DIR + '/logs/summaries'

#MAIN\_DIR = 'drive/My Drive/Colab Notebooks/data/saved\_models/lab4\_model'

#LOG\_DIR = '/logs'

MAIN\_DIR = '/content'

LOG\_DIR = MAIN\_DIR + '/logs'

**Создание вычислительного графа**

cif = cif10()

tf.reset\_default\_graph()

#x = tf.placeholder(tf.float32, shape = [None, 1024])       #для мниста

x = tf.placeholder(tf.float32, shape = [None, 3072])        #cifar

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

keep\_prob = tf.placeholder(tf.float32)

x\_image = tf.reshape(x, [-1,32,32,3])

with tf.name\_scope('conv\_1'):

  conv1 = conv\_layer(x\_image, shape = [3,3,3,32])

  conv1\_pool = max\_pool\_2x2(conv1)

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\_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, neurons\_flat))

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)

  variable\_summaries(y\_conv)

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

  #mnist = input\_data.read\_data\_sets('tmp\ data', one\_hot=True)

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.AdagradOptimizer(learning\_rate=learning\_rate).minimize(cross\_entropy)

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

with tf.name\_scope('accuracy'):   # определение точности

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

    tf.summary.scalar('accuracy', accuracy) # запись в summary

merged = tf.summary.merge\_all() # Слияние

# Сохранение логов в коллекции

train\_var = [x, y\_, accuracy, keep\_prob]

tf.add\_to\_collection('train\_var', train\_var[0])

tf.add\_to\_collection('train\_var', train\_var[1])

tf.add\_to\_collection('train\_var', train\_var[2])

tf.add\_to\_collection('train\_var', train\_var[3])

saver = tf.train.Saver(max\_to\_keep=7, keep\_checkpoint\_every\_n\_hours=0.5)  # сохранение последних 7 точек каждые полчаса

saver.export\_meta\_graph(MAIN\_DIR + LOG\_DIR + "/model\_ckpt.meta", # сохранение в файл .meta

                        collection\_list=['train\_var']);

**Запуск сессии**

with tf.Session() as sess :

  train\_writer = tf.summary.FileWriter(MAIN\_DIR + LOG\_DIR + '/summary/train',

                                         graph=tf.get\_default\_graph())

  test\_writer = tf.summary.FileWriter(MAIN\_DIR + LOG\_DIR + '/summary/test',

                                        graph=tf.get\_default\_graph())

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

  start\_time = time.time()

  for e in range(2):

    print('Epoch ', e, ':')

    for i in range(n\_iters):

      batch = cif.next\_batch\_train(batch\_size)

      log\_step = 200

      if i % log\_step == 0:

        summary,train\_accuracy = sess.run([merged,accuracy], feed\_dict={

                          x: batch[0],

                          y\_: batch[1],

                          keep\_prob: 1})

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

                                                                i, train\_accuracy))

        #print(i+e\*100)

        train\_writer.add\_summary(summary, global\_step= (i + e\*n\_iters))

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

  saver.save(sess, MAIN\_DIR + LOG\_DIR + '/session/model')

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

  Y = cif.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))

  test\_accuracy = []  # проверка точности модели

  for i in range(10):

      summary, test\_accur = sess.run([merged,accuracy], feed\_dict={x:X[i], y\_:Y[i], keep\_prob:0.5})

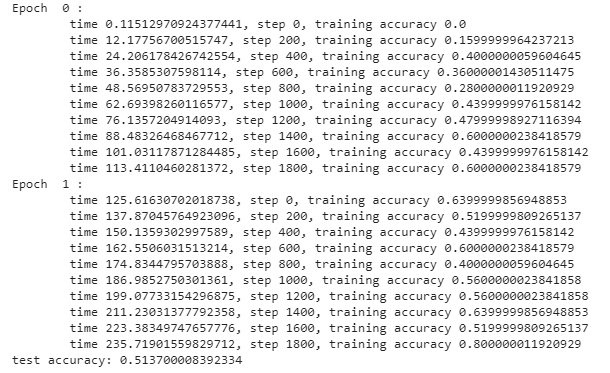
      test\_accuracy.append(test\_accur)

      test\_writer.add\_summary(summary, i)

  test\_accuracy = np.mean(test\_accuracy)

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

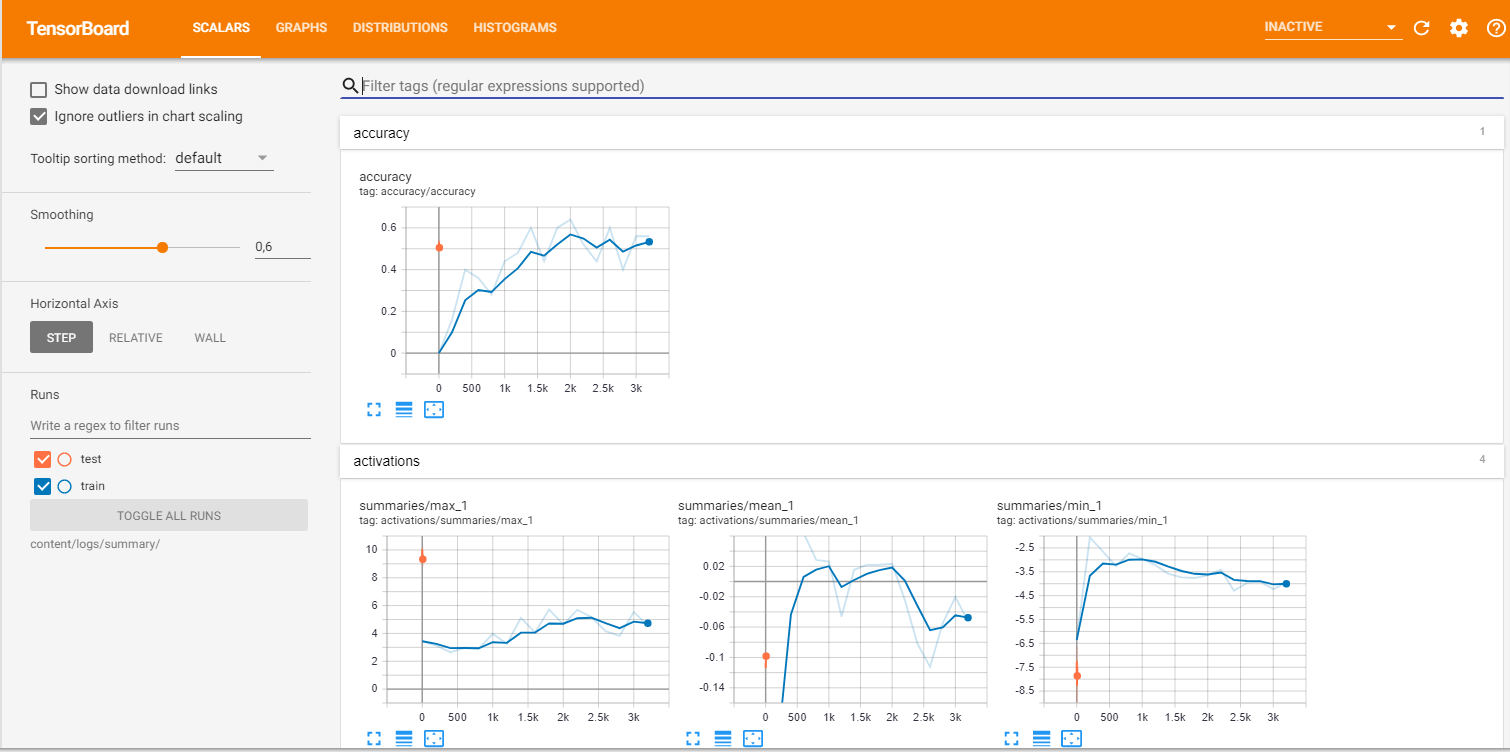
**Результат работы (2 эпохи)**



**TensorBoard**

%reload\_ext tensorboard

%tensorboard --logdir 'content/logs/'



**Восстановление модели**

tf.reset\_default\_graph()

nb\_classes = 10

cif = cif10()

with tf.Session() as sess:

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

  saver = tf.train.import\_meta\_graph("content/logs/model\_ckpt.meta")

  saver.restore(sess, "content/logs/session/model")

  x = tf.get\_collection('train\_var')[0]

  y\_ = tf.get\_collection('train\_var')[1]

  accuracy = tf.get\_collection('train\_var')[2]

  keep\_prob = tf.get\_collection('train\_var')[3]

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

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

  test\_accuracy = np.mean([sess.run(accuracy, feed\_dict={x:X[i], y\_:Y[i], keep\_prob:0.5}) for i in range(10)])

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

**Вывод**

INFO:tensorflow:Restoring parameters from content/logs/session/model

test accuracy: 0.51419997215271

1. **Контрольные вопросы**
2. Как включить TensorBoard?

Создать граф TensorFlow, из которого надо собрать сводные данные, и определить, какие узлы надо комментировать. (или объединить сводки командой tf.summary.merge\_all()) Чтобы генерировать сводки, нужно запустить все эти узлы. Затем запустить объединенный итоговый оператор op, который будет генерировать сериализованный объект со всеми сводными данными на данном этапе.

Для запуска Tensorboard в Google Colab необходимо запустить команды:

%load\_ext tensorboard

%tensorboard --logdir <LOG\_DIR> --port PORTID

1. Как сбросить граф?

tf.reset\_default\_graph()

1. Зачем нужны коллекции?

После импорта графа и весов у нас отсутствуют входные данные, и мы не имеем возможности продолжить обучение или пересчитать граф, поэтому мы сохраняем эти переменные в коллекции. Коллекция - это объект похожий на словарь, в котором мы храним элементы узлов графа.

1. Перечислите команды для добавления переменных в сводную статистику.

tf.summary.scalar('mean', mean)

tf.summary.scalar('stddev', stddev)

tf.summary.scalar('max', tf.reduce\_max(var))

tf.summary.scalar('min', tf.reduce\_min(var))

tf.summary.histogram('histogram', var)

# **Список литературы**

1. Терехов В.И., Черненький И.М., Методические указания к лабораторной работе №2 - М, 2020
2. TensorFlow [Электронный ресурс] - Режим доступа - <https://www.tensorflow.org/>