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.

2. 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 ten
sorboard
    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_v
2(logits=y_conv, labels=y_))

# #SGD
train_step = tf.train.AdagradOptimizer(learning_rate=learning_rate).minimi
ze(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", # сохране
ние в файл .meta
                        collection_list=['train_var']);
```

Запуск сессии

```
with tf.Session() as sess :
  train_writer = tf.summary.FileWriter(MAIN_DIR + LOG_DIR + '/summary/trai
n',
                                       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_a
ccuracy))
        #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)
```

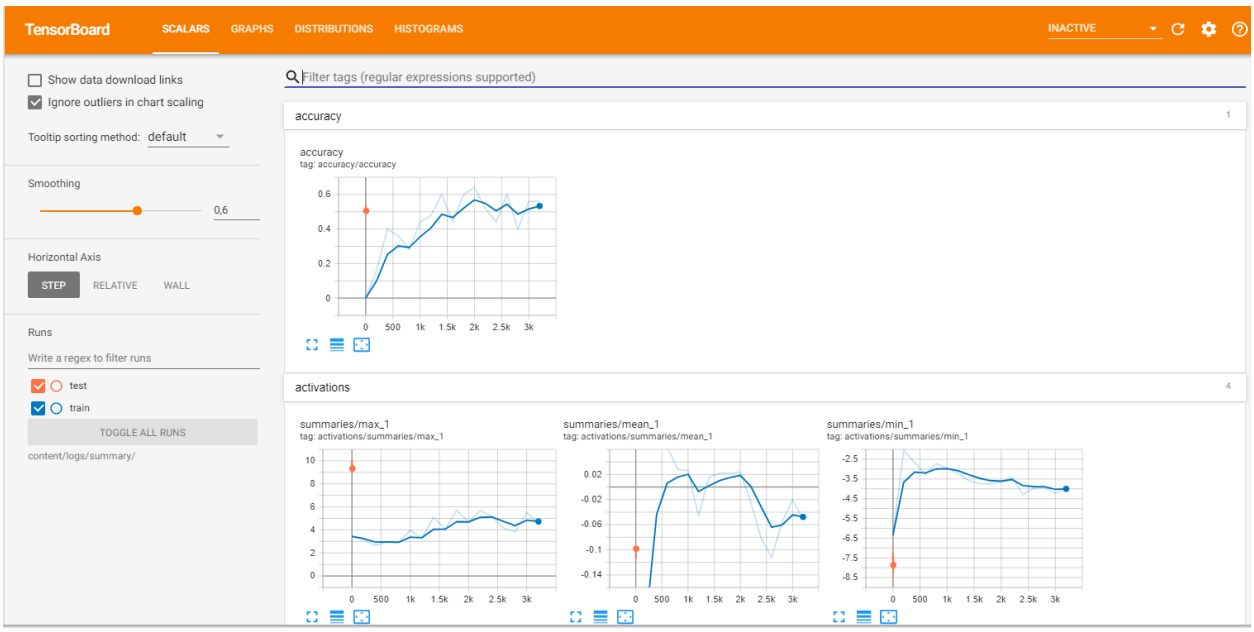
```
  Y = cif.y_test.reshape(10, 1000, 10)

  #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)

```
  Epoch  0 :
         time 0.11512970924377441, step 0, training accuracy 0.0
         time 12.17756700515747, step 200, training accuracy 0.1599999964237213
         time 24.206178426742554, step 400, training accuracy 0.4000000059604645
         time 36.3585307598114, step 600, training accuracy 0.36000001430511475
         time 48.56950783729553, step 800, training accuracy 0.2800000011920929
         time 62.69398260116577, step 1000, training accuracy 0.4399999976158142
         time 76.1357204914093, step 1200, training accuracy 0.47999998927116394
         time 88.48326468467712, step 1400, training accuracy 0.6000000238418579
         time 101.03117871284485, step 1600, training accuracy 0.4399999976158142
         time 113.4110460281372, step 1800, training accuracy 0.6000000238418579
  Epoch  1 :
         time 125.61630702018738, step 0, training accuracy 0.6399999856948853
         time 137.87045764923096, step 200, training accuracy 0.5199999809265137
         time 150.1359302997589, step 400, training accuracy 0.4399999976158142
         time 162.5506031513214, step 600, training accuracy 0.6000000238418579
         time 174.8344795703888, step 800, training accuracy 0.4000000059604645
         time 186.9852750301361, step 1000, training accuracy 0.5600000023841858
         time 199.07733154296875, step 1200, training accuracy 0.5600000023841858
         time 211.23031377792358, step 1400, training accuracy 0.6399999856948853
         time 223.38349747657776, step 1600, training accuracy 0.5199999809265137
         time 235.71901559829712, step 1800, training accuracy 0.800000011920929
  test accuracy: 0.513700008392334
```

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
```

3. Контрольные вопросы

1. 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
```

2. Как сбросить граф?

tf.reset_default_graph()

3. 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.

4. 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 №4 - М, 2020
2. TensorFlow [Электронный ресурс] - Режим доступа - https://www.tensorflow.org/