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:

Чечнев А.А.

Группа ИУ5-23М

_____

"__"_____ 2020 г.

Москва 2020

1. Задание

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

2. CIFAR10

CIFAR10 - это еще один набор данных с большим количеством использования в исследованиях машинного и глубокого обучения. В нем 60000 цветных изображений размером 32x32 пикселя, каждый из которых принадлежит одной из 10 категорий: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик.
Для загрузки набора данных предлагается использовать модуль keras.datasets.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
```

Изменяемые гиперпараметры

```
epochs = 15
n_samples = 50000
batch_size = 25
learning_rate = 1e-2
neurons_flat = 512
```

Класс подгружающий и обрабатывающий набор cifar10

```
class cif10:
    _cursor = 0

    def __init__(self):
        (self.X_train, self.y_train), (self.X_test, self.y_test) = cifar10.lo
ad_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()

    #Возвращает следующий батч тренировочного датасета
    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
```

Создание вычислительного графа

```
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)
  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(l
ogits=y_conv,
                                                                          l
abels=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))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Загрузка датасета

```
cif = cif10()
```

Запуск сессии

```
with tf.Session() as sess :
  sess.run(tf.global_variables_initializer())
  start_time = time.time()
  for e in range(epochs):
    print('Epoch ', e, ':')

    for i in range(int(n_samples / batch_size)):
      #batch = mnist.train.next_batch(50)
      batch = cif.next_batch_train(batch_size)

      if i % 100 == 0:
        train_accuracy = sess.run(accuracy, feed_dict={x: batch[0],
                                                       y_: batch[1],
                                                       keep_prob: 1.0})
        print("\ttime {}, step {}, training accuracy {}".format(time.time() -
 start_time,
                                                               i, train_accu
racy))
      sess.run(train_step, feed_dict={x: batch[0], y_: batch[1], keep_prob: 0
.5})

  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))
```

Результат работы

```
Epoch  14 :
        time 1625.513590335846, step 0, training accuracy 0.9599999785423279
        time 1631.311733007431, step 100, training accuracy 0.9200000166893005
        time 1637.1109476089478, step 200, training accuracy 0.7200000286102295
        time 1642.864307641983, step 300, training accuracy 0.7599999904632568
        time 1648.6093928813934, step 400, training accuracy 0.9200000166893005
        time 1654.3950219154358, step 500, training accuracy 0.6800000071525574
        time 1660.1820185184479, step 600, training accuracy 0.7599999904632568
        time 1665.951340675354, step 700, training accuracy 0.7599999904632568
        time 1671.7147064208984, step 800, training accuracy 0.7200000286102295
        time 1677.4866795539856, step 900, training accuracy 0.6800000071525574
        time 1683.2518157958984, step 1000, training accuracy 0.7200000286102295
        time 1689.0799627304077, step 1100, training accuracy 0.800000011920929
        time 1694.9216532707214, step 1200, training accuracy 0.6800000071525574
        time 1700.7179825305939, step 1300, training accuracy 0.6800000071525574
        time 1706.518023967743, step 1400, training accuracy 0.800000011920929
        time 1712.2979176044464, step 1500, training accuracy 0.5600000023841858
        time 1718.1150159835815, step 1600, training accuracy 0.8799999952316284
        time 1723.8811535835266, step 1700, training accuracy 0.5199999809265137
        time 1729.6619091033936, step 1800, training accuracy 1.0
        time 1735.445603609085, step 1900, training accuracy 0.7200000286102295
test accuracy: 0.7026000022888184
```

3. Контрольные вопросы

1) Что такое свертка?

Свертка - это специализированный вид линейной операции.

Алгоритм свертки представляет собой сумму результатов перемножения элементов ядра с элементами входной матрицы. Каждый результат этой операции записывается в соответствующий элемент карты признаков, затем осуществляется сдвиг, и операция проводится повторно.

2) Напишите математическую операцию свертки?

$$c_{ij} = \sum_{r=1}^{m} a_{ir} b_{rj} \quad (i = 1, 2, \ldots l;\ j = 1, 2, \ldots n)\,.$$

где i,j – ширина и длина фильтра (окна свертки)
a – значение входного нейрона
b – значение соответсвующего элемента фильтра

3) Какие свойства сверточного слоя?

1. Разреженные взаимодействия
2. Разделение параметров – относится к использованию одного и того же параметра для более чем одной функции в модели

3. Эквивариантность для изменений
   Эквариантность – свойство функции, при котором ее вход и выход изменяются одинаково. Т. е. верно равенство

$$f(g(x)) = g(f(x))$$

   При изменени входных данных на небольшую величину, значения большинства выходов на этапе пулинга не изменится.

4) Сколько этапов в сверточном слое? Какие?

Этапы светрочного слова:
1. Этап свертки (применение фильтра)
2. Применение нелинейности (Например функции RelU)
3. Этап дискретизации (pooling)

5) Что такое регуляризация? Зачем она нужна?

Регуляризация нужна для предотвращения явления переобучения модели, когда модель показывает хорошие результаты на тренировочном сете и плохие на тестовом. По сути, когда модель запоминает все входные данные и сопоставляет каждый с результатом. Регуляризация решает эту проблему, путем наложения штрафа, ограничивая сложность вычисления.

6) Какой вид регуляризации использовался в лабораторной?

Dropout.

Данный слой с определенной вероятностью "выключает" один из нейронов в слое. При обучении с дропаутом можно представлять себе процесс обучения ансамбля нейронных сетей. Данный слой используется только во время обучения. На этапе тестирования вероятность присваивается единице, и слой не влияет на качество работы.

4. Список литературы

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