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.

1. **CIFAR10**

CIFAR10 - это еще один набор данных с большим количеством использования в исследованиях машинного и глубокого обучения. В нем 60000 цветных изображений размером 32х32 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.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()

#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(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))

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\_accuracy))

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

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

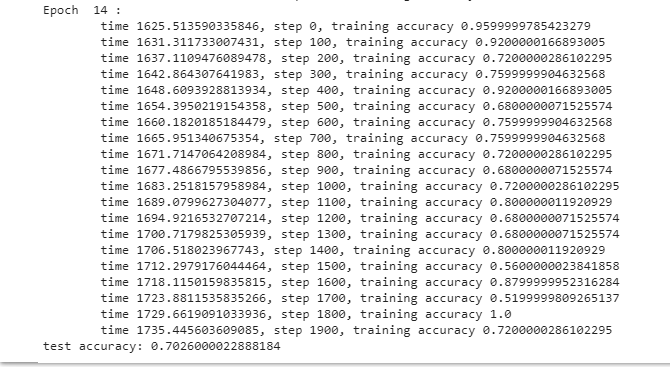
  Y = cif.y\_test.reshape(10, 1000, 10)

  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**

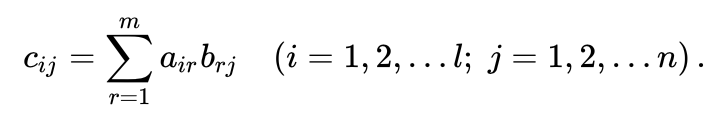


1. **Контрольные вопросы**
2. Что такое свертка?

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.

1. Напишите математическую операцию свертки?



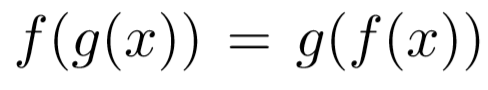
где i,j – ширина и длина фильтра (окна свертки)

a – значение входного нейрона

b – значение соответсвующего элемента фильтра

1. Какие свойства сверточного слоя?
2. Разреженные взаимодействия
3. Разделение параметров – относится к использованию одного и того же параметра для более чем одной функции в модели
4. Эквивариантность для изменений

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



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

1. Сколько этапов в сверточном слое? Какие?

Этапы светрочного слова:

1. Этап свертки (применение фильтра)
2. Применение нелинейности (Например функции RelU)
3. Этап дискретизации (pooling)
4. 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.

1. 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.

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