Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа 3
по курсу
«Проектирование интеллектуальных систем»
Тема:
«Сверточные нейронные сети»

Выполнил:
студент группы ИУ5-24М
Зубаиров В. А.

Проверил:

Москва — 2020 г.

1. Задание на лабораторную работу

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

2. Выполнение

2.1. 1. Импорт библиотек

In [27]:
```
import tensorflow as tf
print("TensorFlow version is ", tf.__version__)
from tensorflow.examples.tutorials.mnist import input_data

import numpy as np
import time

from keras.utils import to_categorical
```

```
TensorFlow version is  1.15.0


Using TensorFlow backend.
```

2.2. 2. Импорт датасета Cifar10

In [15]:
```
from tensorflow.python.keras.datasets import cifar10
```

2.3. 3. Реализация функций

In [17]:
```
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')
```

In [18]:
```
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
```

```
In [ ]: x = tf.placeholder(tf.float32, shape=[None, 784])
        y_ = tf.placeholder(tf.float32, shape=[None, 10])
        keep_prob = tf.placeholder(tf.float32)
        x_image = tf.reshape(x,[ -1, 28, 28, 1])

        with tf.name_scope('conv_1'):
            conv1 = conv_layer ( x_image , shape =[5 , 5 , 1 , 32])
            conv1_pool = max_pool_2x2(conv1)

        with tf.name_scope ('conv_2'):
            conv2 = conv_layer(conv1_pool, shape=[5, 5, 32, 64])
            conv2_pool = max_pool_2x2(conv2)
            conv2_flat = tf.reshape(conv2_pool, [ -1, 7*7*64])

        with tf.name_scope('full_1'):
            full_1 = tf.nn.relu(full_layer(conv2_flat, 1024))

        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


        train_step = tf.train.AdagradOptimizer(learning_rate=1e-4).minimize(cross_

        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:
            sess.run(tf.global_variables_initializer())
            start_time = time.time()

            for i in range (3000):
                batch = mnist . train . next_batch (50)
                if i % 1000 == 0:
                    train_accuracy = sess.run(accuracy, feed_dict={x: batch[0],
                                                                   y_: batch[1], k
                    print(" time {} , step {} , training accuracy {}".format(time
                sess.run(train_step, feed_dict={x: batch[0], y_: batch[1], keep_pr

            (x_train, y_train), (x_test, y_test) = cifar10.load_data()


            X_train = (x_train.astype('float32')) / 255
            X = (x_test.astype('float32')) / 255
```

```
Y = to_categorical(y_test)

test_accuracy = np.mean([sess.run(accuracy, feed_dict={x: X[i], y_: Y
                                                       keep_prob :1.0}) fo
print (" test accuracy : {}". format ( test_accuracy ))
```

2.3.1. Достигнутая точность - 76 процентов

2.4. Ответы на вопросы

2.4.1. 1. Что такое свертка?

- Свертка - это специализированный вид линейной операции. операция в функциональном анализе, которая при применении к двум функциям f и g возвращает третью функцию, соответствующую взаимнокорреляционной функции

2.4.2. 2. Напишите математическую операцию свертки.

$s(t) = \int x(a)w(t⊠a)da$ * w(a) – весовая функция * a – возраст измерения
$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, i-n)$

2.4.3. 3. Какие свойства сверточного слоя?

- разреженные взаимодействия
- разделение параметров (В сверточной нейронной сети каждый член ядра используется в каждой позиции ввода.)
- эквивариантные изменения. (вход и выход изменяются одинаково.)

2.4.4. 4. Сколько этапов в сверточном слое? Какие?

- Применение фильтра
- Применение нелинейности
- Дискретизация

2.4.5. 5. Что такое регуляризация? Зачем она нужна?

В машинном обучении и статистике регуляризация в основном используется для обозначения ограничения оптимизации путем наложения штрафа на сложность решения в попытке предотвратить переобучение на обучающей выборке. Переобучение происходит тогда, когда модель обучилась таким образом, что показывает идеальные результаты на обучающем наборе, но с плохим результатом на новых данных, не используемых ранее (т.е. систему нельзя использовать в продакшне).

2.4.6. 6. Как 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.

3. Список литературы

[1] Google. Tensorflow. 2018. Feb. url - https://www.tensorflow.org/install/install_windows.
[2] url - https://virtualenv.pypa.io/en/stable/userguide/.
[3] Microsoft. about_Execution_Policies. 2018. url - https://technet.microsoft.com/en-us/library/dd347641.aspx.
[4] Jupyter Project. Installing Jupyter. 2018. url - http://jupyter.org/install.