МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Домашнее задание №\_\_1\_\_**

по дисциплине«Проектирование интеллектуальных систем»

Тема: «Создание набора данных для обучения нейронной сети»

ИСПОЛНИТЕЛЬ: Чечнев А.А.

ФИО

группа ИУ5-23М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2020 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО

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

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2020 г.

Москва - 2020

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

**Оглавление**

[**Цель работы** 3](#_Toc41782557)

[**Основная часть** 3](#_Toc41782558)

[**1.** **Сбор датасета** 3](#_Toc41782559)

[**2.** **Импорт библиотек** 3](#_Toc41782560)

[**3.** **Загрузка данных** 3](#_Toc41782561)

[**3.1.** **Функции загрузки данных** 3](#_Toc41782562)

[**3.2.** **Загрузка и предобработка** 5](#_Toc41782563)

[**4.** **Загрузка предобработанного датасета** 7](#_Toc41782564)

[**4.1.** **Функция загрузки numpy файлов** 7](#_Toc41782565)

[**4.2.** **Загрузка** 8](#_Toc41782566)

[**4.3.** **Предобработка** 8](#_Toc41782567)

[**5.** **Визуализация датасета** 9](#_Toc41782568)

[**Заключение** 10](#_Toc41782569)

[**Список использованных источников** 11](#_Toc41782570)

**Цель работы**

Необходимо подготовить собственный набор данных для использования в нейронной сети.

**Основная часть**

1. **Сбор датасета**

Перед написанием программного кода были сформированы 2 директории для тренировочных и тестовых наборов данных. В каждой директории находилось множество папок, соответствующих классам и содержащих изображения в форматах png, jpg, jpeg.

1. **Импорт библиотек**

import os

import glob

import cv2

import time

import numpy as np

import matplotlib.pyplot as plt

1. **Загрузка данных**

GLOBAL\_PATH = '/content/drive/My Drive/Colab Notebooks/hw/data/'

* 1. **Функции загрузки данных**

Определим функцию создания словаря классов и функцию загрузки и предобработки.

Функции *map\_chars* позволяет найди все вложенные папки по абсолютному пути к ним. Полезна для случаев, когда нужен список всех возможных классов.

Функция *load\_train\_set* загружает и масштабирует фотографии, возвращая многомерный.

def map\_chars(path):

  '''Возвращает все названия папок в дирректории path

    Parameters:

      path (string): должен заканчиваться "/"'''

  map\_characters = {}

  i = 0

  for fold in glob.glob(path + '\*', recursive=False):

    map\_characters[i] = fold.split('/')[-1]

    i += 1

  return(map\_characters)

def load\_train\_set(path, map\_characters=None, size=(64, 64),

                   n\_samples=None, verbose=1, print\_step = 50):

  '''Загружает датасет из картинок

        Классы должны быть разбиты по папкам

  Parameters:

     path (string): директория с папками, в которых картинки класса

                                            должен заканчиваться "/"

    map\_characters (dict): словарь названий классов и их лейблов.

            Функция будет загружать только классы из этого словаря

            Если None то соберет словарь самостоятельно map\_char(path), включив

              все классы (папки)

    size (tuple(n,m)): Скалирует фото к размеру [m, n]

    n\_samples (int): Максимальное кол-во загружаемых фото в каждом классе

    verbose (0, 1, 2): 0 - не выводить логи

                       1 - выводить логи по классам и итерациям с частотой print\_step

                       2 - выводит логи только по классам

    print\_step (int): Частота вывода логов по итерациям (Только если verbose=1)

  '''

  if (map\_characters == None):

    map\_characters = map\_chars(path)

  pics, labels = [], []

  time\_start = time.time()

  for (i, char) in map\_characters.items():

    if(verbose > 0):

      print('Loading {}, time {}'.format(char, time.time() - time\_start))

    for j, image\_path in enumerate(glob.glob(path + char + '/\*.\*')):

      try:

        temp = cv2.imread(image\_path)

        temp = cv2.cvtColor(temp, cv2.COLOR\_BGR2RGB)

        temp = cv2.resize(temp,(size[0],size[1])).astype('float32') / 255.

        if ((verbose == 1) and (j % print\_step == 0)):

          print('\tStep {}:, time {}'.format(j, time.time() - time\_start))

        if (j == n\_samples):

          break

        pics.append(temp)

        labels.append(i)

      except:

        print('\tStep {}: Error'.format(j))

        continue

  print('Done. Total time: {}'.format(time.time() - time\_start))

  print('Done. Total imgs: {}'.format(len(pics)))

  return(np.array(pics), np.array(labels))

* 1. **Загрузка и предобработка**

Преобразуем фотографии класса *homer*\_*simpson* в *numpy* файл:

homer = {0:'homer\_simpson'}

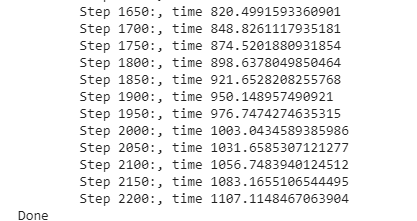
X, y = load\_train\_set(path=(GLOBAL\_PATH + 'simpsons\_dataset/'),

                      map\_characters=homer,

                      print\_step=50)

np.save(GLOBAL\_PATH + 'np\_images/train/homer.npy', X)

Вывод:



Преобразуем остальные классы, которые будем использовать в нейронной сети:

char\_list1 = ['bart\_simpson',  'abraham\_grampa\_simpson', 'fat\_tony', 'krusty\_the\_clown']

char\_list2 = ['marge\_simpson', 'maggie\_simpson', 'lisa\_simpson']

for character in char\_list1:

  X\_temp, \_ = load\_train\_set(path=(GLOBAL\_PATH + 'simpsons\_dataset/'),

                      map\_characters={0:character},

                      print\_step=500)

  np.save(GLOBAL\_PATH + 'np\_images/train/'+ character +'.npy', X\_temp)

for character in char\_list2:

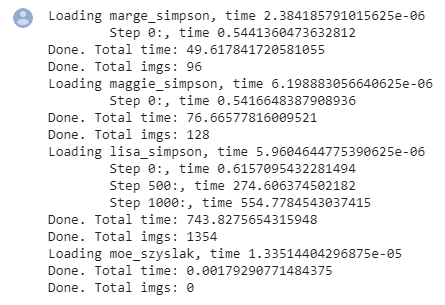
  X\_temp, \_ = load\_train\_set(path=(GLOBAL\_PATH + 'simpsons\_dataset/'),

                      map\_characters={0:character},

                      print\_step=500)

  np.save(GLOBAL\_PATH + 'np\_images/train/'+ character +'.npy', X\_temp)

Вывод:



1. **Загрузка предобработанного датасета**
   1. **Функция загрузки numpy файлов**

def load\_data\_from\_np(folder\_path, classes = None, verbose = 1):

  '''

  Загрузит множество npy файлов и объединит. (X, y)

  Parameters:

    folder\_path (string): Путь до директории с файлами (Заканчивается именем папки)

    classes (list[string]): Названия классов, которые необходимо загружать

          Если None то загрузит все

    verbose (int): Если 1 то выводит логи. 0 - иначе

  '''

  X, y = None, []

  start\_time = time.time()

  for file\_path in glob.glob(folder\_path + '/\*.\*'):

    class\_name = file\_path.split('/')[-1].split('.')[0]

    if ((classes == None) or (class\_name in classes)):

      if (X is None):

        X = np.load(file\_path)

        y = np.array([class\_name]\*X.shape[0])

      else:

        X\_temp = np.load(file\_path)

        X = np.concatenate((X, X\_temp))

        y = np.concatenate((y, np.array([class\_name]\*X\_temp.shape[0])))

      if (verbose == 1):

        #print('{} loaded. Total time {}'.format(class\_name, time.time() - start\_time))

        print('%-25s Total time: %-4f'%(class\_name, time.time() - start\_time))

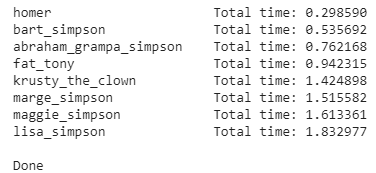
  print('\nDone')

  return (X, np.array(y))

* 1. **Загрузка**

Загрузим датасет и подготовим к использованию в нейросети

X, y = load\_data\_from\_np(GLOBAL\_PATH + 'np\_images/train')



* 1. **Предобработка**

Бинаризуем целевую переменную для использования в нейросети:

from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(sparse=False)

y = ohe.fit\_transform(y.reshape(-1, 1))

map\_characters = {i : ohe.categories\_[0][i] for i in range(7) }

Разобьем выборку на тренировочный и валидационный сет:

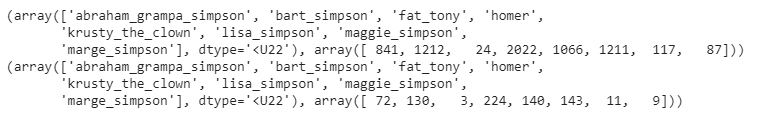
from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, shuffle=True)

Проверим что каждый в оба сета попали все классы

print(np.unique(ohe.inverse\_transform(y\_train), return\_counts=True))

print(np.unique(ohe.inverse\_transform(y\_test), return\_counts=True))



1. **Визуализация датасета**

plt.figure(figsize=(20,10))

for i in range(25):

    plt.subplot(5, 5, i+1)

    n = np.random.choice(X.shape[0])

    plt.imshow(X[n])

    #plt.title(ohe.inverse\_transform(y[n].reshape(1, -1))[0][0])

    plt.title(map\_characters[np.argmax(y[n])])



Рисунок 1 – Загруженный набор данных

**Заключение**

Таким образом в ходе домашнего задания 1 мы научились подготовлять и предобрабатывать изображения для дальнейшего использования в нейронной сети.

**Список использованных источников**

1. Николенко С.И., Кадурин А.А., Архангельская Е.О. Глубокое обучение. – Издательский дом "Питер", 2017. — 480 с.: ил. — (Серия «Библиотека программиста»). [http://ebooks.bmstu.ru/catalog](http://www.google.com/url?q=http%3A%2F%2Febooks.bmstu.ru%2Fcatalog&sa=D&sntz=1&usg=AFQjCNFUQRkK3kBZ5yVzL0YiD4OF9-bSNA) (дата обращения: 21.05.2018) ISBN 978-5-496-02536-2.
2. TensorFlow: Библиотека программного обеспечения с открытым исходным кодом для Machine Intelligence <https://www.tensorflow.org/>
3. Keras: библиотека глубокого обучения Python [https://keras.io/](https://www.google.com/url?q=https%3A%2F%2Fkeras.io%2F&sa=D&sntz=1&usg=AFQjCNGgqzMRRUmljoJs4ZhlAJh_PcgWcw)