МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Домашнее задание №\_\_2\_\_**

по дисциплине«Проектирование интеллектуальных систем»

Тема: «Обучение сверточной нейронной сети на собственном наборе данных»

ИСПОЛНИТЕЛЬ: Чечнев А.А.

ФИО

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

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2020 г.

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

ФИО

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

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2020 г.

Москва - 2020

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

**Оглавление**

[**Цель работы** 3](#_Toc41781696)

[**Основная часть** 3](#_Toc41781697)

[**1.** **Подготовка датасета** 3](#_Toc41781698)

[**1.1.** **Импорт библиотек** 3](#_Toc41781699)

[**1.2.** **Загрузка датасета** 3](#_Toc41781700)

[**1.3.** **Преобразуем датасет** 4](#_Toc41781701)

[**2.** **Изобразим загруженные данные** 5](#_Toc41781702)

[**3.** **Загрузка предобученной модели** 5](#_Toc41781703)

[**4.** **Создание нейросети на основе предобученной** 6](#_Toc41781704)

[**4.1.** **Уберем полносвязные слои и добавим свои** 6](#_Toc41781705)

[**5.** **Создадим генератор изображений для тренировочного датасета¶** 8](#_Toc41781706)

[**5.1.** **Изобразим работу датагенератора** 8](#_Toc41781707)

[**6.** **Обучение модели** 8](#_Toc41781708)

[**6.1.** **Проверка качества модели** 9](#_Toc41781709)

[**6.2.** **Сохрание весов** 10](#_Toc41781710)

[**7.** **Проверим модель на тестовом наборе данных** 10](#_Toc41781711)

[**7.1.** **Изобразим предсказания на тестовом сете** 11](#_Toc41781712)

[**Заключение** 12](#_Toc41781713)

[**Список использованных источников** 13](#_Toc41781714)

**Цель работы**

Необходимо обучить сверточную нейронную сеть на наборе данных из домашнего задания 1.

**Основная часть**

1. **Подготовка датасета**
   1. **Импорт библиотек**

import numpy as np

import glob

import numpy as np

import time

import matplotlib.pyplot as plt

* 1. **Загрузка датасета**

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))

DATA\_PATH = '../input/simpsons-train-numpy-my/np\_images/train'

INPUT\_PATH = '../input'

n\_classes = 7

X, y = load\_data\_from\_np(DATA\_PATH, classes=['bart\_simpson', 'marge\_simpson', 'lisa\_simpson', 'krusty\_the\_clown',

'homer', 'abraham\_grampa\_simpson', 'maggie\_simpson'])

* 1. **Преобразуем датасет**

from sklearn.preprocessing import OneHotEncoder

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

ohe = OneHotEncoder(sparse=False)

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

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

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

## Изобразим наши данные

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])])

1. **Изобразим загруженные данные**



Рисунок 2.1 – Датасет

1. **Загрузка предобученной модели**

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras.optimizers import SGD, Adam

def create\_model\_six\_conv(input\_shape):

*"""*

*CNN Keras model with 6 convolutions.*

*:param input\_shape: input shape, generally X\_train.shape[1:]*

*:return: Keras model, RMS prop optimizer*

*"""*

model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same', input\_shape=input\_shape, activation='relu'))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(256, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(18, activation='softmax'))

opt = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

return model, opt

def load\_model\_from\_checkpoint(weights\_path, input\_shape=(64,64,3)):

model, opt = create\_model\_six\_conv(input\_shape)

model.load\_weights(weights\_path)

model.compile(loss='categorical\_crossentropy',

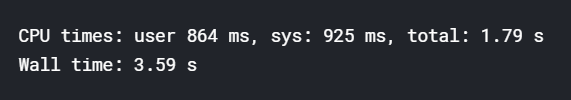
optimizer=opt,

metrics=['accuracy'])

return model

%time pretrained\_model = load\_model\_from\_checkpoint(INPUT\_PATH + '/simpsons-train-numpy-my/weights.best.hdf5')

Вывод:



1. **Создание нейросети на основе предобученной**
   1. **Уберем полносвязные слои и добавим свои**

model = Sequential()

for l **in** pretrained\_model.layers[:-3]:

model.add(l)

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(n\_classes, activation='softmax'))

opt = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical\_crossentropy',

optimizer=opt,

metrics=['accuracy'])

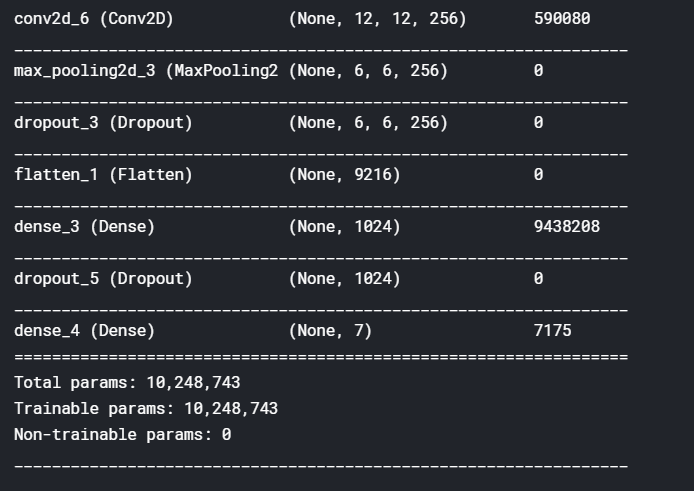
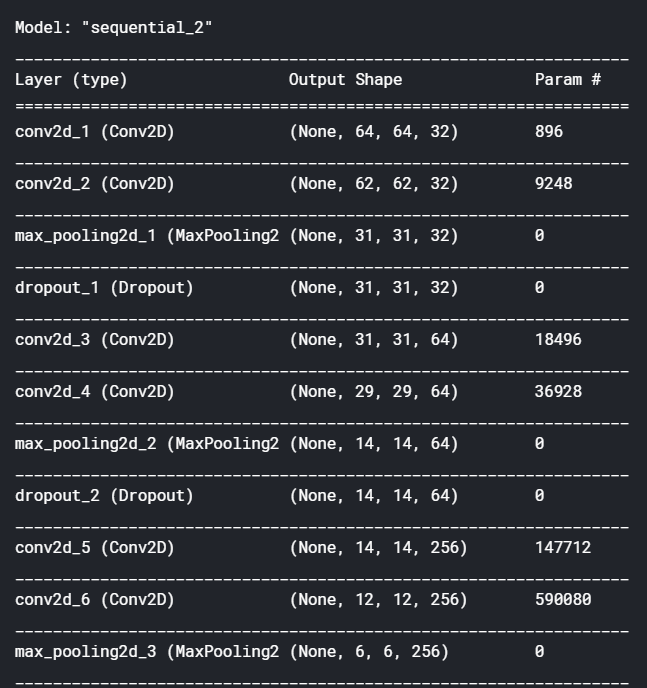


Рисунок 4.1 – Структура нейронной сети

1. **Создадим генератор изображений для тренировочного датасета[¶](https://www.kaggle.com/aksesssche/hw2-simpsons-classifier?scriptVersionId=35121175" \l "%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%B4%D0%B8%D0%BC-%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80-%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9-%D0%B4%D0%BB%D1%8F-%D1%82%D1%80%D0%B5%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82%D0%B0" \t "_self)**

* Поскольку в некоторых классах слишком мало данных, применим ImageDataGenerator

from tensorflow.keras.preprocessing.image import ImageDataGenerator

image\_gen\_train = ImageDataGenerator(

rotation\_range=20,

width\_shift\_range=.15,

height\_shift\_range=.15,

horizontal\_flip=True,

zoom\_range=0.3

)

data\_train\_gen = image\_gen\_train.flow(X\_train, y\_train)

* 1. **Изобразим работу датагенератора**

def plotImages(images\_arr):

fig, axes = plt.subplots(1, 5, figsize=(20,20))

axes = axes.flatten()

for img, ax **in** zip( images\_arr, axes):

ax.imshow(img)

ax.axis('off')

plt.tight\_layout()

plt.show()

augmented\_images = [data\_train\_gen[5][0][20] for i **in** range(5)]

plotImages(augmented\_images)



Рисунок 5.1 – результат работы генератора изображений

1. **Обучение модели**

history = model.fit(x=data\_train\_gen, epochs=20, verbose=1, shuffle=True, validation\_data=(X\_val, y\_val))

except **KeyboardInterrupt**:

print('**\n\n**Stopped')

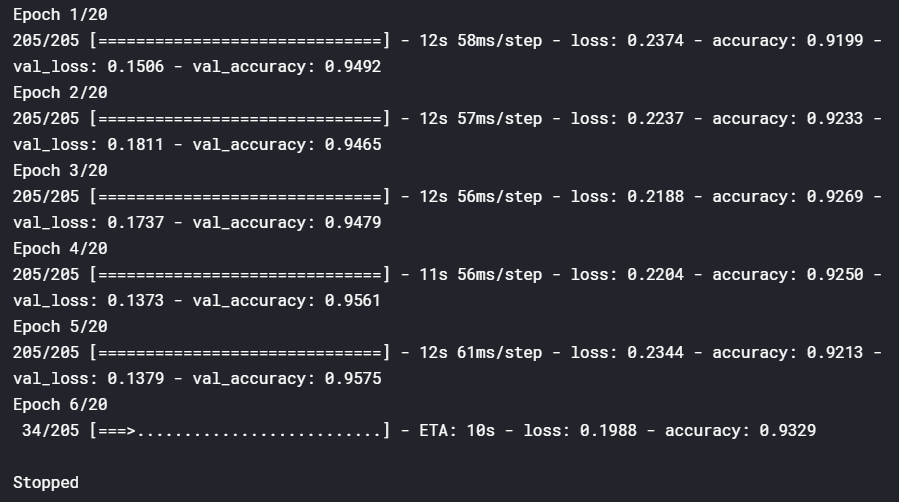


Рисунок 6.1 – Процесс обучений нейронной сети

* 1. **Проверка качества модели**

Посмотрим на основные метрики для классификации на валидационном сете

import sklearn

from sklearn.metrics import classification\_report

print('**\n**', sklearn.metrics.classification\_report(np.argmax(y\_val, axis=1,

np.argmax(model.predict(X\_val), axis=1),

target\_names=list(map\_characters.values())), sep='')

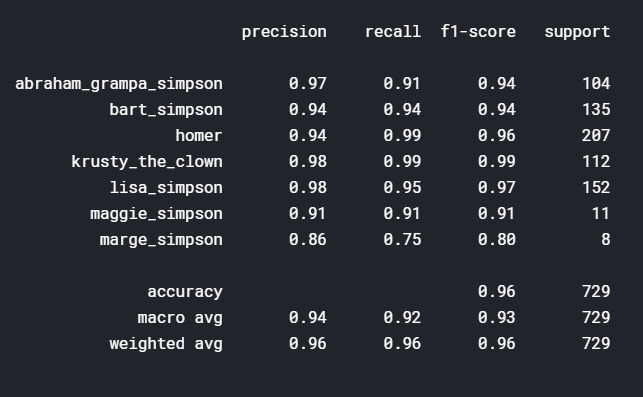


Рисунок 6.2. – Основные классификационные метрики на валидационном сете

* 1. **Сохрание весов**

*#model.save('../input/weightsbesthdf5/model\_8classes\_als.h5')*

model.save('model\_7classes\_als.h5')

1. **Проверим модель на тестовом наборе данных**

* Загрузим тестовый сет и посмотрим на метрики

X\_test, y\_test = load\_data\_from\_np('../input/simpsons-train-numpy-my/np\_images/test',

['bart\_simpson', 'marge\_simpson', 'lisa\_simpson', 'krusty\_the\_clown',

'homer', 'abraham\_grampa\_simpson', 'maggie\_simpson'])

y\_test = ohe.transform(y\_test.reshape(-1, 1))

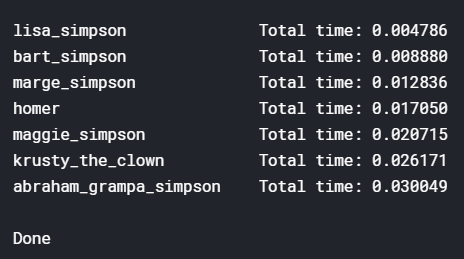


Рисунок 7.1. – Загрузка тестового сета

print('**\n**', sklearn.metrics.classification\_report(np.argmax(y\_test, axis=1),

np.argmax(model.predict(X\_test), axis=1),

target\_names=list(map\_characters.values())), sep='')

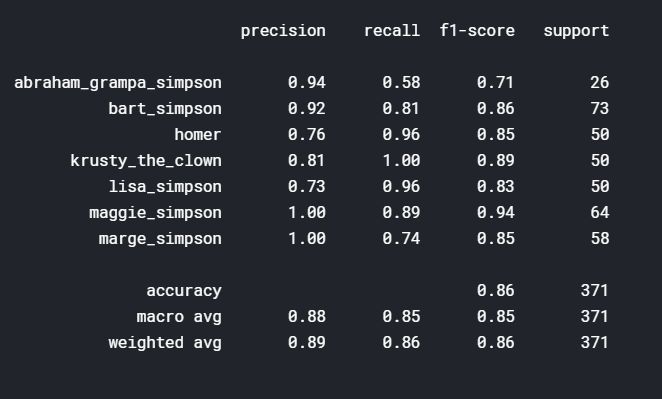


Рисунок 7.2. – Основные классификационные метрики на валидационном сете

* 1. **Изобразим предсказания на тестовом сете**

TEST\_PATH = '../input/simpsons-train-numpy-my/simpsons\_test\_set\_8\_classes'

In [42]:

F = plt.figure(1, (15,20))

grid = AxesGrid(F, 111, nrows\_ncols=(5, 5), axes\_pad=0, label\_mode="1")

for i **in** range(25):

class\_path = glob.glob(TEST\_PATH + '/\*')[i%n\_classes]

img = cv2.imread(np.random.choice(glob.glob(class\_path + '/\*')))

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

temp = cv2.resize(img,(64,64)).astype('float32') / 255.

img = cv2.resize(img, (352, 352))

a = model.predict(temp.reshape(1, 64, 64,3))[0]

actual = class\_path.split('/')[-1].split('\_')[0]

text = sorted(['**{:s}** : **{:.1f}**%'.format(map\_characters[k].split('\_')[0].title(), 100\*v) for k,v **in** enumerate(a)],

key=lambda x:float(x.split(':')[1].split('%')[0]), reverse=True)[:3]

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.rectangle(img, (0,260),(215,352),(255,255,255), -1)

cv2.putText(img, 'Actual : **%s**' % actual, (10, 280), font, 0.7,(0,0,0),2,cv2.LINE\_AA)

for k, t **in** enumerate(text):

cv2.putText(img, t,(10, 300+k\*18), font, 0.65,(0,0,0),2,cv2.LINE\_AA)

grid[i].imshow(img)



Рисунок 7.3 – Результат работы обученной неросети на тестовом датасете

**Заключение**

Таким образом в ходе выполнения домашнего задания 2 были изучены приемы transfer learning. Была обучена предобученная модель на собственном наборе данных и проведена оценка качества модели.

**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)