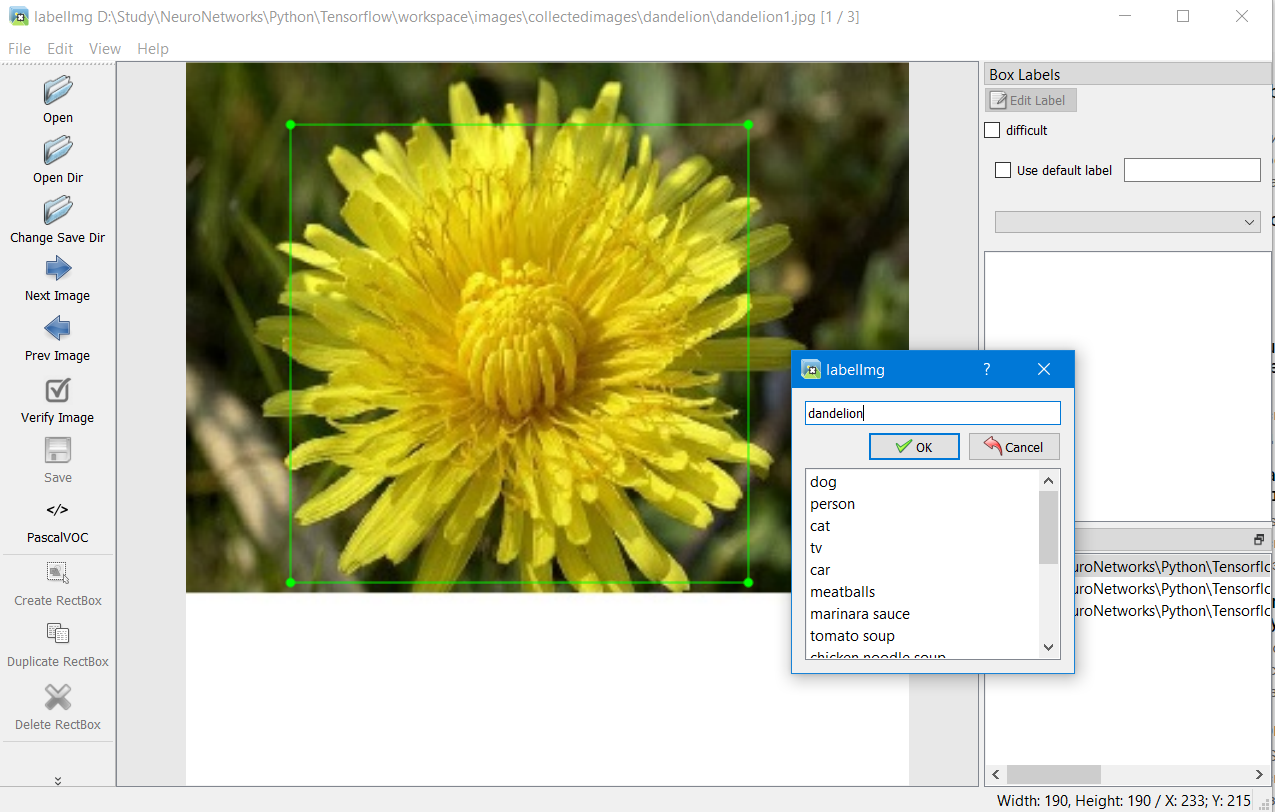
Никонов Иван, группа АиСИС

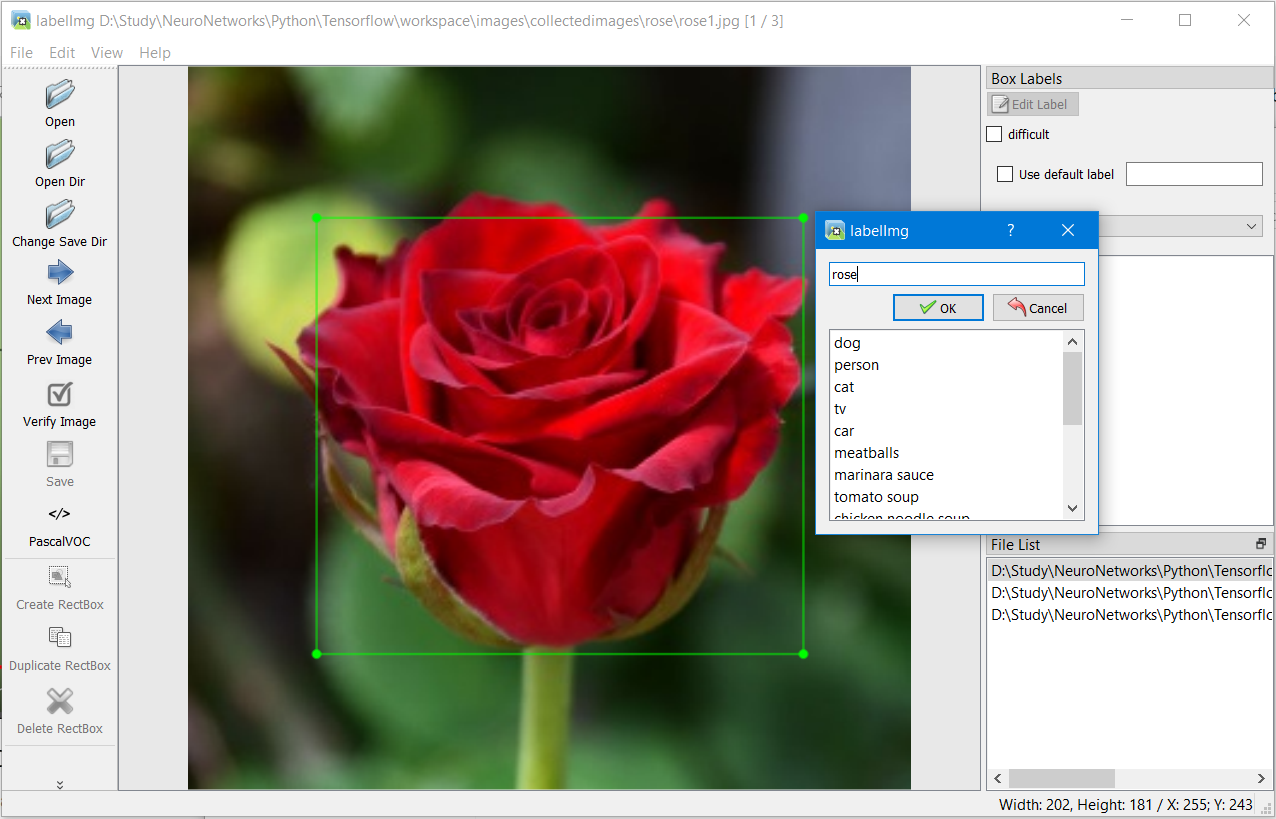
Лабораторная работа № 5

**Вариант № 8**

В данной лабораторной работе проводится обучение глубокой нейронной сети для распознавания образов цветов на изображениях.

Для начала была создана обучающая выборка по средствам разметки изображений, найденных в интернете.





В результате для каждого изображения был сформирован xml файл, описывающий положение искомого объекта на изображении. Обучающая выборка состояла всего из четырех изображений каждого типа. Это очень мало, и для хорошего обучения требуется выборка куда больших размеров.

В работе была использована заранее обученная модель, доступная для скачивания по следующей ссылке: http://download.tensorflow.org/models/object\_detection/tf2/20200711/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8.tar.gz.

Далее размеченные данные приводятся к необходимому виду, создаются дополнительные конфиг файлы:

CUSTOM\_MODEL\_NAME = 'my\_ssd\_mobnet'

PRETRAINED\_MODEL\_NAME = 'ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8'

PRETRAINED\_MODEL\_URL = 'http://download.tensorflow.org/models/object\_detection/tf2/20200711/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8.tar.gz'

TF\_RECORD\_SCRIPT\_NAME = 'generate\_tfrecord.py'

LABEL\_MAP\_NAME = 'label\_map.pbtxt'

paths = {

'WORKSPACE\_PATH': os.path.join('Tensorflow', 'workspace'),

'SCRIPTS\_PATH': os.path.join('Tensorflow','scripts'),

'APIMODEL\_PATH': os.path.join('Tensorflow','models'),

'ANNOTATION\_PATH': os.path.join('Tensorflow', 'workspace','annotations'),

'IMAGE\_PATH': os.path.join('Tensorflow', 'workspace','images'),

'MODEL\_PATH': os.path.join('Tensorflow', 'workspace','models'),

'PRETRAINED\_MODEL\_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),

'CHECKPOINT\_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM\_MODEL\_NAME),

'OUTPUT\_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM\_MODEL\_NAME, 'export'),

'TFJS\_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM\_MODEL\_NAME, 'tfjsexport'),

'TFLITE\_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM\_MODEL\_NAME, 'tfliteexport'),

'PROTOC\_PATH':os.path.join('Tensorflow','protoc')

}

files = {

'PIPELINE\_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM\_MODEL\_NAME, 'pipeline.config'),

'TF\_RECORD\_SCRIPT': os.path.join(paths['SCRIPTS\_PATH'], TF\_RECORD\_SCRIPT\_NAME),

'LABELMAP': os.path.join(paths['ANNOTATION\_PATH'], LABEL\_MAP\_NAME)

}

for path in paths.values():

if not os.path.exists(path):

if os.name == 'posix':

!mkdir -p {path}

if os.name == 'nt':

!mkdir {path}

labels = [{'name':'dandelion', 'id':1}, {'name':'rose', 'id':2}]

with open(files['LABELMAP'], 'w') as f:

for label in labels:

f.write('item { \n')

f.write('\tname:\'{}\'\n'.format(label['name']))

f.write('\tid:{}\n'.format(label['id']))

f.write('}\n')

!python {files['TF\_RECORD\_SCRIPT']} -x {os.path.join(paths['IMAGE\_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION\_PATH'], 'train.record')}

!python {files['TF\_RECORD\_SCRIPT']} -x {os.path.join(paths['IMAGE\_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION\_PATH'], 'test.record')}

if os.name =='posix':

!cp {os.path.join(paths['PRETRAINED\_MODEL\_PATH'], PRETRAINED\_MODEL\_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT\_PATH'])}

if os.name == 'nt':

!copy {os.path.join(paths['PRETRAINED\_MODEL\_PATH'], PRETRAINED\_MODEL\_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT\_PATH'])}

config = config\_util.get\_configs\_from\_pipeline\_file(files['PIPELINE\_CONFIG'])

pipeline\_config = pipeline\_pb2.TrainEvalPipelineConfig()

with tf.io.gfile.GFile(files['PIPELINE\_CONFIG'], "r") as f:

proto\_str = f.read()

text\_format.Merge(proto\_str, pipeline\_config)

pipeline\_config.model.ssd.num\_classes = len(labels)

pipeline\_config.train\_config.batch\_size = 4

pipeline\_config.train\_config.fine\_tune\_checkpoint = os.path.join(paths['PRETRAINED\_MODEL\_PATH'], PRETRAINED\_MODEL\_NAME, 'checkpoint', 'ckpt-0')

pipeline\_config.train\_config.fine\_tune\_checkpoint\_type = "detection"

pipeline\_config.train\_input\_reader.label\_map\_path= files['LABELMAP']

pipeline\_config.train\_input\_reader.tf\_record\_input\_reader.input\_path[:] = [os.path.join(paths['ANNOTATION\_PATH'], 'train.record')]

pipeline\_config.eval\_input\_reader[0].label\_map\_path = files['LABELMAP']

pipeline\_config.eval\_input\_reader[0].tf\_record\_input\_reader.input\_path[:] = [os.path.join(paths['ANNOTATION\_PATH'], 'test.record')]

config\_text = text\_format.MessageToString(pipeline\_config)

with tf.io.gfile.GFile(files['PIPELINE\_CONFIG'], "wb") as f:

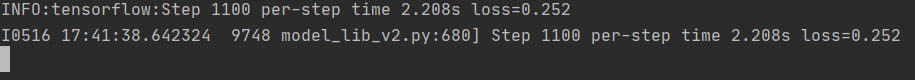
f.write(config\_text)

После чего запускается обучение сети

TRAINING\_SCRIPT = os.path.join(paths['APIMODEL\_PATH'], 'research', 'object\_detection', 'model\_main\_tf2.py')

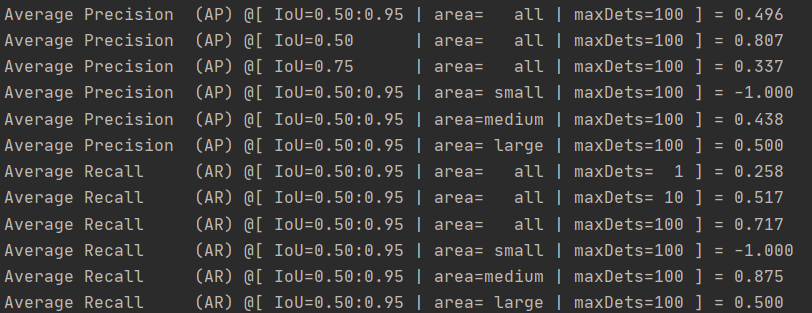
command = "python {} --model\_dir={} --pipeline\_config\_path={} --num\_train\_steps=2000".format(TRAINING\_SCRIPT, paths['CHECKPOINT\_PATH'],files['PIPELINE\_CONFIG'])

!{command}



Далее оценим ее при помощи команды:

python Tensorflow\models\research\object\_detection\model\_main\_tf2.py --model\_dir=Tensorflow\workspace\models\my\_ssd\_mobnet --pipeline\_config\_path=Tensorflow\workspace\models\my\_ssd\_mobnet\pipeline.config --checkpoint\_dir=Tensorflow\workspace\models\my\_ssd\_mobnet



Получаем неплохие результаты для такой маленькой обучающей выборки.

Запустим модель на тестовых изображениях.

import cv2

import numpy as np

from matplotlib import pyplot as plt

category\_index = label\_map\_util.create\_category\_index\_from\_labelmap(files['LABELMAP'])

IMAGE\_PATH = os.path.join(paths['IMAGE\_PATH'], 'test', 'dandelion4.jpg')

img = cv2.imread(IMAGE\_PATH)

image\_np = np.array(img)

input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)

detections = detect\_fn(input\_tensor)

num\_detections = int(detections.pop('num\_detections'))

detections = {key: value[0, :num\_detections].numpy()

for key, value in detections.items()}

detections['num\_detections'] = num\_detections

# detection\_classes should be ints.

detections['detection\_classes'] = detections['detection\_classes'].astype(np.int64)

label\_id\_offset = 1

image\_np\_with\_detections = image\_np.copy()

viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

image\_np\_with\_detections,

detections['detection\_boxes'],

detections['detection\_classes']+label\_id\_offset,

detections['detection\_scores'],

category\_index,

use\_normalized\_coordinates=True,

max\_boxes\_to\_draw=5,

min\_score\_thresh=.8,

agnostic\_mode=False)

plt.imshow(cv2.cvtColor(image\_np\_with\_detections, cv2.COLOR\_BGR2RGB))

plt.show()

Получаем следующий результат. К сожалению, определился только один цветок, однако определился весьма точно (точный процент не поместился).

