**Жуковский Павел, 3 курс, 12 группа, кафедра КТС**

**Задание 1.**

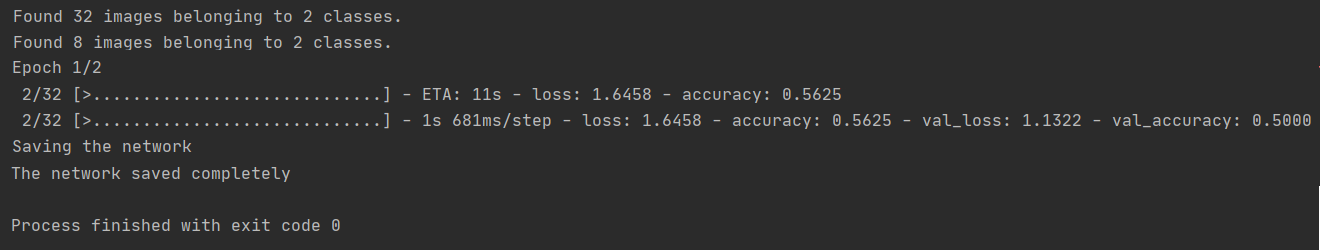
Дообучите нейронную сеть VGG16 для распознавания изображений только двух классов. Используйте дообученную сеть для классификации своих изображений.

Для начала я установил все нужные библиотеки в среду разработки, после чего для начала запустил исходный код для обучения нейронной сети.

Код для обучения сети (task\_1\_network\_training.py):

from keras.models import Model  
from keras.layers import GlobalAveragePooling2D, Dense  
from keras.preprocessing.image import ImageDataGenerator  
from keras import applications  
  
# Размер изображений  
img\_width, img\_height = 150, 150  
# Путь к каталогу с изображениями для обучения  
train\_data\_dir = 'data/train'  
# Путь к каталогу с изображениями для валидации  
validation\_data\_dir = 'data/validation'  
# Количество изображений для обучения  
nb\_train\_samples = 32  
# Количество изображений для валидации  
nb\_validation\_samples = 8  
# Количество эпох  
epochs = 2  
# Размер выборки  
batch\_size = 16  
  
# Загружаем сеть VGG16 без части, которая отвечает за классификацию  
base\_model = applications.VGG16(weights='imagenet', include\_top=False)  
  
# Добавляем слои классификации  
x = base\_model.output  
x = GlobalAveragePooling2D()(x)  
x = Dense(1024, activation='relu')(x)  
# Выходной слой с двумя нейронами для классов "кот" и "собака"  
predictions = Dense(2, activation='softmax')(x)  
  
# Составляем сеть из двух частей  
model = Model(inputs=base\_model.input, outputs=predictions)  
  
# "Замораживаем" сверточные уровни сети VGG16  
# Обучаем только вновь добавленные слои  
for layer in base\_model.layers:  
 layer.trainable = False  
  
# Компилируем модель  
model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
# Создаем генератор данных для обучения  
datagen = ImageDataGenerator(rescale=1. / 255)  
train\_generator = datagen.flow\_from\_directory(  
 train\_data\_dir,  
 target\_size=(img\_width, img\_height),  
 batch\_size=batch\_size,  
 class\_mode= 'categorical')  
  
# Создаем генератор данных для валидации  
validation\_generator = datagen.flow\_from\_directory(  
 validation\_data\_dir,  
 target\_size=(img\_width, img\_height),  
 batch\_size=batch\_size,  
 class\_mode= 'categorical')  
  
# Обучаем модель с помощью генератора  
model.fit\_generator(  
 train\_generator,  
 steps\_per\_epoch=nb\_train\_samples,  
 epochs=epochs,  
 validation\_data=validation\_generator,  
 validation\_steps = nb\_validation\_samples)  
  
print("Saving the network")  
# Сохраняем сеть для последующего использования  
# Генерируем описание модели в формате json  
model\_json = model.to\_json()  
json\_file = open("vgg16\_cat\_dogs.json", "w")  
# Записываем архитектуру сети в файл  
json\_file.write(model\_json)  
json\_file.close()  
# Записываем данные о весах в файл  
model.save\_weights("vgg16\_cat\_dogs.h5")  
print("The network saved completely")

Вывод:



Таким образом, мы обучили нашу сеть и сохранили её в json-файле.

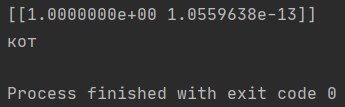
Теперь я использовал данную сеть для распознавания изображений. Для теста я взял следующее изображение кота, хранящееся в файле cat\_for\_test.jpg:



Код для использования обученной сети (task\_1\_recognition.py):

from tensorflow.keras.models import model\_from\_json  
from keras.applications.vgg16 import preprocess\_input  
from keras.preprocessing import image  
import numpy as np  
  
  
# Список классов  
classes = ['кот', 'собака']  
  
json\_file = open("vgg16\_cat\_dogs.json", "r")  
loaded\_model\_json = json\_file.read()  
json\_file.close()  
loaded\_model = model\_from\_json(loaded\_model\_json)  
loaded\_model.load\_weights("vgg16\_cat\_dogs.h5")  
  
loaded\_model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
img = image.load\_img('cat\_for\_test.jpg', target\_size=(224, 224))  
  
x = image.img\_to\_array(img)  
x = np.expand\_dims(x, axis=0)  
x = preprocess\_input(x)  
  
prediction = loaded\_model.predict(x)  
  
print(prediction)  
print(classes[np.argmax(prediction)])

Вывод:



Мы видим, что почти с полной уверенностью наша сеть определила, что это действительно кот.

**Задание 2.**

Дообучите нейронную сеть ResNet34 распознаванию Вашего лица на фотографии. Используйте дообученную сеть для распознавания Вашего лица на других фотографиях (другой возраст; другой ракурс; часть лица закрыта, например, очками).

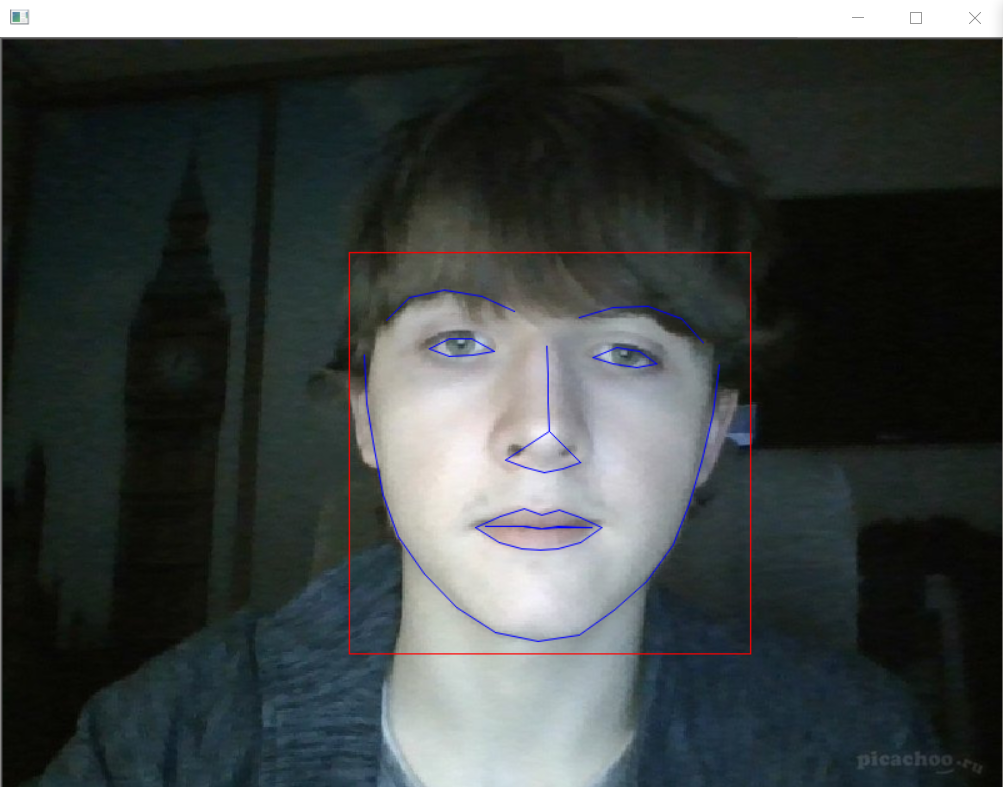
Для теста я взял парочку своих фотографий:

Код для распознавания парочки моих фотографией с помощью данной нейронной сети (task\_2.py):

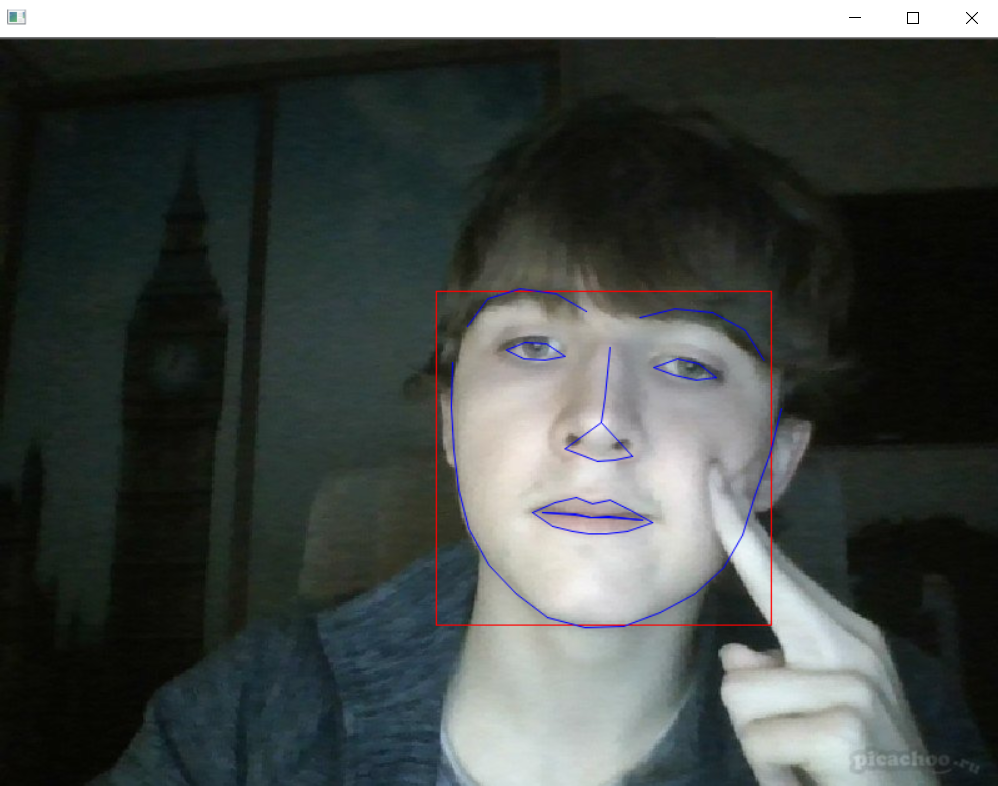
import dlib  
from scipy.spatial import distance  
from skimage import io  
  
sp = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')  
facerec = dlib.face\_recognition\_model\_v1('dlib\_face\_recognition\_resnet\_model\_v1.dat')  
detector = dlib.get\_frontal\_face\_detector()  
img = io.imread('Pavel\_1.jpg')  
win1 = dlib.image\_window()  
win1.clear\_overlay()  
win1.set\_image(img)  
dets = detector(img, 1)  
for k, d in enumerate(dets):  
 print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(  
 k, d.left(), d.top(), d.right(), d.bottom()))  
 shape = sp(img, d)  
 win1.clear\_overlay()  
 win1.add\_overlay(d)  
 win1.add\_overlay(shape)  
face\_descriptor1 = facerec.compute\_face\_descriptor(img, shape)  
img = io.imread('Pavel\_2.jpg')  
win2 = dlib.image\_window()  
win2.clear\_overlay()  
win2.set\_image(img)  
dets\_webcam = detector(img, 1)  
for k, d in enumerate(dets\_webcam):  
 print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(  
 k, d.left(), d.top(), d.right(), d.bottom()))  
 shape = sp(img, d)  
 win2.clear\_overlay()  
 win2.add\_overlay(d)  
 win2.add\_overlay(shape)  
face\_descriptor2 = facerec.compute\_face\_descriptor(img, shape)  
wait = input("PRESS ENTER TO CONTINUE.")  
a = distance.euclidean(face\_descriptor1, face\_descriptor2)  
print(a)

Вот так программа распознала моё лицо на фотографиях:

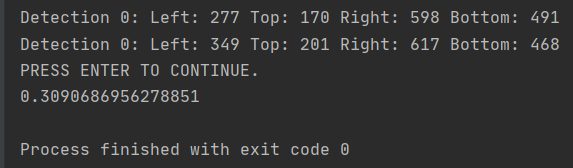
1) Pavel\_1.jpg



2) Pavel\_2.jpg



Вывод:



Из-за того, что я слегка наклонил голову, параметры 2-ой фотографии несколько изменились относительно 1-ой фотографии, потому можем наблюдать коэффициент примерно 0.3.