ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

СОГЛАСОВАНО

Преподаватель департамента программной инженерии Факультета компьютерных наук

_____ И.М. Воронков

«____» _____ 2019 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», канд. техн. наук, профессор ДПИ ФКН

_____ В.В. Шилов

«____» _____ 2019 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

Программа для классификации объектов мебели на фотографиях

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.13-01 12 01-1-ЛУ

Исполнитель

Студент группы БПИ171

_____ / Д. А. Потапенков /

«____» _____ 2019 г.

Москва 2019

УТВЕРЖДЕН

RU.17701729. 04.13-01 12 01-1-ЛУ

Программа для классификации объектов мебели на фотографиях

Текст программы

RU.17701729.04.13-01 12 01-1

Листов 11

Инв. № подл		Подп. и дата		Взам. инв. №		Инв. № дубл.		Подп. и дата	

Москва 2019

ОГЛАВЛЕНИЕ

1. Main.py

```
import sys
import os
import tkinter as tk
from tkinter import filedialog
import tkinter.ttk as ttk
from PIL import Image, ImageTk
import numpy as np
import Models as model
class MyApp:
    def __init__(self, window):
        self.window = window
        self.configur_window()

    def loadImgClick(self):
        self.imgName =
tk.filedialog.askopenfilename(filetypes=[("Images","*.png *.jpg")])
        img = Image.open(self.imgName)
        img.thumbnail((self.canvas.winfo_width(),self.canvas.winfo_height()),
Image.ANTIALIAS)
        self.img = ImageTk.PhotoImage(img)
        self.canvas.create_image(self.canvas.winfo_width()/2,
self.canvas.winfo_height()/2, image=self.img ,anchor="c")

    def loadModelClick(self):
        self.ansCanvas.delete("all")
        self.chartCanvas.delete("all")
        modelPath = tk.filedialog.askopenfilename(filetypes=[("Models",
"*.h5")])
        _ , filename = os.path.split(modelPath)
        modelName , _ = os.path.splitext(filename)
        self.predictor.loadModel(modelName, modelPath)
        models = list(self.modelCombobox['values'])
        models.append(modelName)
        self.modelCombobox['values'] = models


    def predictClick(self):
        self.ansCanvas.delete("all")

        if self.imgName == "":
            self.ansCanvas.create_text(10, 10, anchor="nw", text="Load an
image", font="Times 11")
            return

        if self.modelCombobox.get() == "":
            self.ansCanvas.create_text(10, 10, anchor="nw", text="Select
model", font="Times 11")
            return

        predict = self.predictor.predict(self.modelCombobox.get(),
self.imgName)
        ind_max = np.argmax(predict)

        self.ansCanvas.create_text(10, 10, anchor="nw",
text=self.classes[ind_max].upper(), font="Times 12")
```

```
            self.chartCanvas.delete("all")
            for i in range(len(predict)):
                self.chartCanvas.create_text(10, 10 + i*45, anchor="nw",
text=self.classes[i], font="Times 11")
                self.chartCanvas.create_rectangle(10, 30 + i*45, 10 + predict[i]
* 150, 50 + i*45, fill="blue")


    def configur_window(self):
        self.window.title("Furniture")
        self.window.geometry("1000x620")
        self.window.resizable(False, False)

        self.canvas = tk.Canvas(self.window, heigh=600, width=800, bd=2,
relief="ridge", bg="white")
        self.canvas.pack(side="left")

        self.modelFrame = tk.LabelFrame(text="Model")
        self.modelFrame.pack(side="top", fill="x")
        self.modelCombobox = ttk.Combobox(self.modelFrame, state="readonly")
        self.modelCombobox['values'] = ["MobileNet"]
        self.modelCombobox.pack(side="top", fill="x", pady=10)
        self.loadModelButton = tk.Button(self.modelFrame, heigh=2, width=24,
text="Load Model",
                                         command=lambda:
MyApp.loadModelClick(self))
        self.loadModelButton.pack(side="top", fill="x", pady=5)

        self.modelImage = tk.LabelFrame(text="Image")
        self.modelImage.pack(side="top", fill="x")
        self.loadImgButton = tk.Button(self.modelImage, heigh=2, width=24,
text="Load Image", command= lambda :MyApp.loadImgClick(self))
        self.loadImgButton.pack(side="top", fill="x", pady=0)
        self.predictButton = tk.Button(self.modelImage, heigh=2, width=24,
text="Predict", command= lambda :MyApp.predictClick(self))
        self.predictButton.pack(side="top", fill="x", pady=2)

        self.modelAnswer = tk.LabelFrame(text="Answer")
        self.modelAnswer.pack(side="top", fill="x")
        self.ansCanvas = tk.Canvas(self.modelAnswer, heigh=30, bd=2,
relief="ridge", bg="white")
        self.ansCanvas.pack(side="top", fill="x", pady=5)
        self.chartCanvas = tk.Canvas(self.modelAnswer, heigh=200, bd=2,
relief="ridge", bg="white")
        self.chartCanvas.pack(side="top", fill="x", pady=0)

        self.provar = tk.IntVar()
        self.provar.set(0)
        self.bar = ttk.Progressbar(self.window, variable=self.provar)
        self.bar.pack(side="top", fill="x")

    def loadResurses(self):
        self.imgName = ""
        self.predictor = model.Predictor(self)
        self.predictor.loadModel("MobileNet", "models/MobileNet.h5")
        self.classes = ["chair", "couch", "plant", "table"]
```

```
window = tk.Tk()

myApp = MyApp(window)
window.after(100, myApp.loadResurses)
window.mainloop()
sys.exit()
```

2. Model.py

```
from skimage.io import imread
from skimage.transform import resize
import numpy as np
import tensorflow as tf
from keras import backend as ker_bac
from keras.models import load_model

config = tf.ConfigProto()
config.gpu_options.allow_growth = True  # Don't pre-allocate memory; allocate
as-needed
sess = tf.Session(config=config)
ker_bac.tensorflow_backend.set_session(sess)

class Predictor:
    models = dict()

    def __init__(self, App):
        self.App = App

    def loadModel(self, name, path):
        self.App.ansCanvas.create_text(10, 10, anchor="nw", text="Model is
loading...", font="Times 11")
        self.App.provar.set(50)
        self.App.window.update()
        self.models[name] = load_model(path)
        self.App.provar.set(100)
        self.App.ansCanvas.delete("all")
        self.App.ansCanvas.create_text(10, 10, anchor="nw", text="Model has
been loaded", font="Times 10")
        self.App.window.update()

    def predict(self, modelName, fileName):
        global graph
        graph = tf.get_default_graph()

        myImg = resize(imread(fileName)[:, :, :3], (224, 224),
anti_aliasing=True)
        myImg = np.reshape(myImg, [1, 224, 224, 3])
        with graph.as_default():
```

```
            ans = self.models[modelName].predict(myImg)
        return ans[0]
```

3. TrainModels.ipynb

```
#%%
from keras.applications.resnet50 import ResNet50
from keras.applications.mobilenet import MobileNet
from keras import layers
from keras import Model
#%%
base_model=ResNet50(input_shape=(224, 224, 3),
weights='imagenet',include_top=False)

#%%
base_model=MobileNet(input_shape=(224, 224, 3),
weights='imagenet',include_top=False)
#%%
#For MobileNet
x=layers.GlobalAveragePooling2D()(base_model.output)
x=layers.Dense(1024,activation='relu')(x)
x=layers.Dense(512,activation='relu')(x) #
preds=layers.Dense(4,activation='softmax')(x) #final layer with softmax
activation
#%%
#For Inception
x=layers.GlobalAveragePooling2D()(base_model.output)
preds=layers.Dense(4,activation='softmax')(x) #final layer with softmax
activation
#%%
model=Model(inputs=base_model.input,outputs=preds)
for layer in model.layers:
    layer.trainable = True
#%%
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator()
test_datagen = ImageDataGenerator()
#%%
train_dir = "path/to/train"
validation_dir = "path/to/val"
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(224,224),
                                                    color_mode='rgb',
                                                    batch_size=32,
                                                    class_mode='categorical',
                                                    shuffle=True)

validation_generator = test_datagen.flow_from_directory(
                                                   validation_dir,
                                                   target_size=(224, 224),
                                                   batch_size=32,
                                                   class_mode='categorical',
                                                   shuffle=True,
                                                   color_mode='rgb')
#%%
```

```
from keras.optimizers import SGD
from keras.optimizers import Adam

model.compile(optimizer=SGD(lr=0.001,

momentum=0.5),loss='categorical_crossentropy',metrics=['accuracy'])
#%%
step_size_train=train_generator.n//train_generator.batch_size
step_size_val=validation_generator.n//validation_generator.batch_size

history = model.fit_generator(
      train_generator,
      steps_per_epoch=step_size_train,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=step_size_val,
      verbose=1)
#%%
from skimage.io import imread
from skimage.transform import resize
import numpy as np


from keras.applications.resnet50 import preprocess_input

myImg =
resize(imread("/content/ImgClass/validation/plant/potted_plant_11029.jpg")[:,
:,:3], (224, 224),anti_aliasing=False)
myImg = np.reshape(myImg, [1, 224, 224, 3])
myImg = preprocess_input(myImg)
print(myImg.shape)
#%%
ans = model.predict(myImg)
print(ans)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729. 04.13-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

4. GetLinks.py

```
from bs4 import BeautifulSoup
import requests
import os

# You need to have directoris in imagesT like in dictionary

baseName = "images/"
myDict = {'Skandinavskiy': 'https://www.houzz.ru/photos/foto-gostinaya-
komnata-v-skandinavskom-stile-phbr1-bp~t_13807~s_22925',
          'Sovremenniy': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-
v-sovremennom-stile-phbr1-bp~t_13807~s_14086',
          'Klassicheskiy': 'https://www.houzz.ru/photos/foto-gostinaya-
komnata-v-klassicheskom-stile-phbr1-bp~t_13807~s_14089',
          'Loft': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-loft-phbr1-bp~t_13807~s_14094',
          'Fyyushn': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-fyyuzhn-phbr1-bp~t_13807~s_14087',
          'Modernizm': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-modernizm-phbr1-bp~t_13807~s_14088',
          'Vostochniy': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-
v-vostochnom-stile-phbr1-bp~t_13807~s_14090',
          'Morskoiy': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
morskom-stile-phbr1-bp~t_13807~s_14091',
          'Kantri': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-kantri-phbr1-bp~t_13807~s_14093',
          'ShebbiShink': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-
v-stile-shebbi-shik-phbr1-bp~t_13807~s_24688',
          'Sredizemnomorskiy': 'https://www.houzz.ru/photos/foto-gostinaya-
komnata-v-sredizemnomorskom-stile-phbr1-bp~t_13807~s_14095',
          'Retro': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-retro-phbr1-bp~t_13807~s_14096',
          'Rustika': 'https://www.houzz.ru/photos/foto-gostinaya-komnata-v-
stile-rustika-phbr1-bp~t_13807~s_14097',
          'SovremennayaKlassika': 'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-sovremennaya-klassika-phbr1-bp~t_13807~s_14098',
          'Viktorianskom': 'https://www.houzz.ru/photos/foto-gostinaya-
komnata-v-viktorianskom-stile-phbr1-bp~t_13807~s_22926'
          }

for (type, url) in myDict.items():
    os.makedirs(baseName + type)
    out = open(baseName + type + "/links.txt", "w")
    counter = 1000
    baseUrl = url + "?fi="
    while counter < 1200:
        curentUrl = baseUrl + str(counter)
        page = requests.get(curentUrl)
        soup = BeautifulSoup(page.text, features="html.parser")
        myimages = soup.select("picture", {"class": "hz-image-container"})
        curCount = 0
        for img in myimages:
            if "hz-image-placeholder" in img["class"]:
                continue
            curCount += 1
            imgUrl = img.contents[1]["src"]
```

```
            myImage = imgUrl[27:48]
            out.write(myImage + "\n")
            print(myImage + " " + type)
        counter += curCount
        print("---------------" + str(counter) + "----------------------")

    out.close()
```

5. DownloadLinks.py

```
import requests
import os.path

# You need to have directoris in imagesT like in dictionary and file that was
generatied by GetLinks.py

baseName = "images/"
myDict = {'Skandinavskiy':          'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-skandinavskom-stile-phbr1-bp~t_13807~s_22925',
          'Sovremenniy':            'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-sovremennom-stile-phbr1-bp~t_13807~s_14086',
          'Klassicheskiy':          'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-klassicheskom-stile-phbr1-bp~t_13807~s_14089',
          'Loft':                   'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-loft-phbr1-bp~t_13807~s_14094',
          'Fyyushn':                'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-fyyuzhn-phbr1-bp~t_13807~s_14087',
          'Modernizm':              'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-modernizm-phbr1-bp~t_13807~s_14088',
          'Vostochniy':             'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-vostochnom-stile-phbr1-bp~t_13807~s_14090',
          'Morskoiy':               'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-morskom-stile-phbr1-bp~t_13807~s_14091',
          'Kantri':                 'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-kantri-phbr1-bp~t_13807~s_14093',
          'ShebbiShink':            'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-shebbi-shik-phbr1-bp~t_13807~s_24688',
          'Sredizemnomorskiy':      'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-sredizemnomorskom-stile-phbr1-bp~t_13807~s_14095',
          'Retro':                  'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-retro-phbr1-bp~t_13807~s_14096',
          'Rustika':                'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-rustika-phbr1-bp~t_13807~s_14097',
          'SovremennayaKlassika':   'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-stile-sovremennaya-klassika-phbr1-bp~t_13807~s_14098',
          'Viktorianskom':          'https://www.houzz.ru/photos/foto-
gostinaya-komnata-v-viktorianskom-stile-phbr1-bp~t_13807~s_22926'}

counter1 = 0
counter2 = 0
w = 300
h = 300
begin = "https://st.hzcdn.com/"
dir1 = "fimgs/"
dir2 = "simgs/"
endFormat = "-w{0}-h{1}-b0-p0--.jpg"
```

```
res = [14, 9]

for (type, url) in myDict.items():
    print("--------------------------" + type + "--------------------------
")
    myLinks = open(baseName + type + "/links.txt", "r")
    for link in myLinks:
        link = link.strip()
        parts = link.split("_")
        url1 = begin + dir1 + link + endFormat.format(h, h)
        url2 = begin + dir2 + parts[0] + "_" + str(res[0]) + "-" + parts[1] +
".jpg"

        #Загрузить эту ссылку
        #print(url1)

        p = requests.get(url2)

        if (p.status_code == 404):
            counter2 += 1
            print("Уменьшили")
            url2 = begin + dir2 + parts[0] + "_" + str(res[1]) + "-" +
parts[1] + ".jpg"
            p = requests.get(url2)

        if (p.status_code == 404):
            counter1 += 1
            print("Не могу закачать")
            continue

        print(url2)
        if os.path.exists(baseName + type + "/" + url2[27:]):
            continue
        p = requests.get(url2)
        out = open(baseName + type + "/" + url2[27:], "wb")
        out.write(p.content)
        out.close()
print(counter1, counter2)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729. 04.13-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729. 04.13-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729. 04.13-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729. 04.13-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата