**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ДЕТЕКЦИЯ ОБЪЕКТОВ НА ФОТОГРАФИЯХ С ПОМОЩЬЮ

ГЛУБОКИХ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

**ТЕКСТ ПРОГРАММЫ**

КР.ПО4.190344-01 12 00

Листов 9

|  |  |
| --- | --- |
| Руководитель | А. А. Крощенко |
| Выполнил | А. С. Луд |
| Консультант |  |
| по ЕСПД | А. А. Крощенко |

2021

**СОДЕРЖАНИЕ**

Модуль yolo.py – для распознавания объектов на изображении.

Модуль yolo-video.py - для распознавания объектов на кадрах из видео.

**yolo.py**

# импорт необходимых пакетов

import numpy as np

import argparse

import time

import cv2

import os

# построить аргумент синтаксический анализ и разбор аргументов

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--image", required=True,

help="path to input image")

ap.add\_argument("-y", "--yolo", required=True,

help="base path to YOLO directory")

ap.add\_argument("-c", "--confidence", type=float, default=0.5,

help="minimum probability to filter weak detections")

ap.add\_argument("-t", "--threshold", type=float, default=0.3,

help="threshold when applying non-maxima suppression")

args = vars(ap.parse\_args())

# загрузить метки класса COCO, на которые была обучена наша модель YOLO

labelsPath = os.path.sep.join([args["yolo"], "coco.names"])

LABELS = open(labelsPath).read().strip().split("\n")

# инициализировать список цветов для представления каждой возможной метки класса

np.random.seed(42)

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),

dtype="uint8")

# производные пути к весам YOLO и конфигурации модели

weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])

configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])

# загрузить наш детектор объектов YOLO, обученный на наборе данных COCO (80 классов)

print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# загрузите наше входное изображение и захватите его пространственные размеры

image = cv2.imread(args["image"])

(H, W) = image.shape[:2]

# определяют только имена \*выходных\* слоев, которые нам нужны из YOLO

ln = net.getLayerNames()

ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

# создайте большой двоичный объект из входного изображения, а затем выполните прямое

# проход детектора объектов YOLO, дающий нам наши ограничители и

# связанные вероятности

blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),

swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()

# показать информацию о времени на YOLO

print("[INFO] YOLO took {:.6f} seconds".format(end - start))

# инициализируйте наши списки обнаруженных ограничителений, доверительных связей и

# идентификаторы классов соответственно

boxes = []

confidences = []

classIDs = []

# цикл над каждым из выходов слоя

for output in layerOutputs:

# цикл над каждым из обнаружений

for detection in output:

# извлеките идентификатор класса и достоверность (т.е. вероятность)

# обнаружение текущего объекта

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

# отфильтровывайте слабые прогнозы, обеспечивая обнаружение

# вероятность больше минимальной вероятности

if confidence > args["confidence"]:

# масштабируйте координаты ограничительной рамки относительно

# размер изображения, имея в виду, что YOLO на самом деле

# возвращает центральную (x, y)-координаты границы

# box, за которым следуют ширина и высота коробок

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype("int")

# использовать центральные (x, y)-координаты для получения верхней и

# и левый угол ограничителя

x = int(centerX - (width / 2))

y = int(centerY - (height / 2))

# обновить наш список координат ограничителю, доверительных ложей,

# и идентификаторы классов

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

# применять подавление без максимумов для подавления слабых, перекрывающихся границ

# коробки

idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],

args["threshold"])

# убедитесь, что существует хотя бы одно обнаружение

if len(idxs) > 0:

# цикл над индексами, которые мы сохраняем

for i in idxs.flatten():

# извлечь координаты ограничителя

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

# нарисуйте прямоугольник ограничителя и метку на изображении

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])

cv2.putText(image, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX,

0.5, color, 2)

# показать выходное изображение

cv2.imshow("Image", image)

cv2.waitKey(0)

**yolo-video.py**

# импорт необходимых пакетов

import numpy as np

import argparse

import imutils

import time

import cv2

import os

# построить аргумент синтаксический анализ и разбор аргументов

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--input", required=True,

help="path to input video")

ap.add\_argument("-o", "--output", required=True,

help="path to output video")

ap.add\_argument("-y", "--yolo", required=True,

help="base path to YOLO directory")

ap.add\_argument("-c", "--confidence", type=float, default=0.5,

help="minimum probability to filter weak detections")

ap.add\_argument("-t", "--threshold", type=float, default=0.3,

help="threshold when applyong non-maxima suppression")

args = vars(ap.parse\_args())

# загрузить метки класса COCO, на которые была обучена наша модель YOLO

labelsPath = os.path.sep.join([args["yolo"], "coco.names"])

LABELS = open(labelsPath).read().strip().split("\n")

# инициализировать список цветов для представления каждой возможной метки класса

np.random.seed(42)

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),

dtype="uint8")

# производные пути к весам YOLO и конфигурации модели

weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])

configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])

# загрузить наш детектор объектов YOLO, обученный на наборе данных COCO (80 классов)

# определяют только имена \*выходных\* слоев, которые нам нужны из YOLO

print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

ln = net.getLayerNames()

ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

# инициализируйте видеопоток, указатель на вывод видеофайл и

# размеры рамы

vs = cv2.VideoCapture(args["input"])

writer = None

(W, H) = (None, None)

# попробуйте определить общее количество кадров в видеофайле

try:

prop = cv2.cv.CV\_CAP\_PROP\_FRAME\_COUNT if imutils.is\_cv2() \

else cv2.CAP\_PROP\_FRAME\_COUNT

total = int(vs.get(prop))

print("[INFO] {} total frames in video".format(total))

# произошла ошибка при попытке определить общее количество

# количество кадров в видеофайле

except:

print("[INFO] could not determine # of frames in video")

print("[INFO] no approx. completion time can be provided")

total = -1

# цикл над кадрами из потока видеофайла

while True:

# чтение следующего кадра из файла

(grabbed, frame) = vs.read()

# если кадр не был схвачен, то мы дошли до конца

# потока

if not grabbed:

break

# если размеры рамы пусты, захватите их

if W is None or H is None:

(H, W) = frame.shape[:2]

# создать большой двоичный объект из входного кадра, а затем выполнить прямой

# проход детектора объектов YOLO, дающий нам наши ограничители

# и связанные с ним вероятности

blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),

swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()

# инициализируйте наши списки обнаруженных ограничителений, доверительных связей,

# и идентификаторы классов соответственно

boxes = []

confidences = []

classIDs = []

# цикл над каждым из выходов слоя

for output in layerOutputs:

# цикл над каждым из обнаружений

for detection in output:

# извлеките идентификатор класса и уверенность (т.е. вероятность)

# обнаружения текущего объекта

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

# отфильтровывайте слабые прогнозы, обеспечивая обнаружение

# вероятность больше минимальной вероятности

if confidence > args["confidence"]:

# масштаб координат ограничительной рамки относительно

# размер изображения, имея в виду, что YOLO

# фактически возвращает центр (x, y)-координаты

# ограничителя, за которым следует ширина полей и

# высота

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype("int")

# использовать центральные (x, y)-координаты для получения верхней части

# и левый угол ограничителя

x = int(centerX - (width / 2))

y = int(centerY - (height / 2))

# обновить наш список координат ограничителю,

# доверительные степени и идентификаторы классов

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

# применять подавление без максимумов для подавления слабых, перекрывающихся

# ограничители

idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],

args["threshold"])

# убедитесь, что существует хотя бы одно обнаружение

if len(idxs) > 0:

# цикл над индексами, которые мы сохраняем

for i in idxs.flatten():

# извлечь координаты ограничителя

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

# нарисуйте прямоугольник ограничителя и метку на рамке

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

text = "{}: {:.4f}".format(LABELS[classIDs[i]],

confidences[i])

cv2.putText(frame, text, (x, y - 5),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

# проверьте, является ли видео Нет

if writer is None:

# инициализировать наш видеописец

fourcc = cv2.VideoWriter\_fourcc(\*"MJPG")

writer = cv2.VideoWriter(args["output"], fourcc, 30,

(frame.shape[1], frame.shape[0]), True)

# некоторая информация по обработке одного кадра

if total > 0:

elap = (end - start)

print("[INFO] single frame took {:.4f} seconds".format(elap))

print("[INFO] estimated total time to finish: {:.4f}".format(

elap \* total))

# записать выходной кадр на диск

writer.write(frame)

# отпустите указатели файлов

print("[INFO] cleaning up...")

writer.release()

vs.release()