Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №2

по курсу

«Технологии распределенных вычислений и анализа данных»

|  |  |
| --- | --- |
| Выполнила: | магистрант группы 355841  А.В. Деркач |
| Проверил: | к.т.н., доцент каф. ЭВМ  Д.Ю. Перцев |

Минск 2024

**1 ЗАДАНИЕ**

На вход подается один из универсальных dataset’ов. На выходе создается множество файлов (или файл) с описанием того, что в данном dataset’е было обнаружено.

Допускается использовать: любую операционную систему, любой язык программирования, любые технологии и библиотеки алгоритмов.

**2 ВЫПОЛНЕНИЕ РАБОТЫ**

Реализована пользовательская утилита, которая запускается с помощью командной строки и поддерживает параметры, представленные на рисунке 2.1.

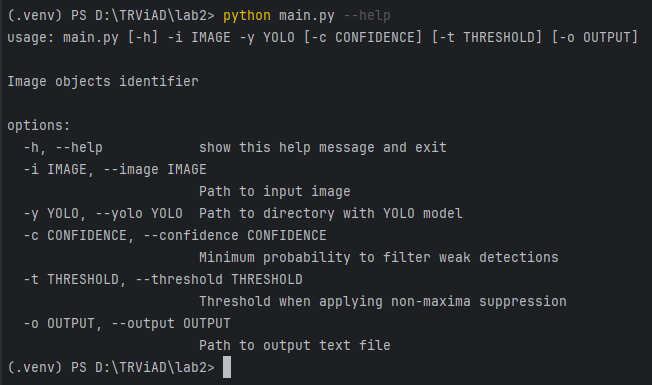


Рисунок 2.1 – Параметры для настройки индентификации объектов

Для запуска утилиты с указанием входного изображения, директории с моделью обнаружения объектов YOLO, выходного файла с результатами, а также минимальной точности обнуражения равной 0.5 и максимального порога подавления «избыточных» результатов равного 0.3, выполняется команда:

python main.py -i '../img/img7.jpg' -y '../yolo' -c 0.5 -t 0.3 -o '../result.txt'

Утилита анализирует изображение, после чего выводит его на экран с указанием найденных объектов (см. рисунок 2.2), а также сохраняет результат в указанный выходной файл (см. рисунок 2.3).

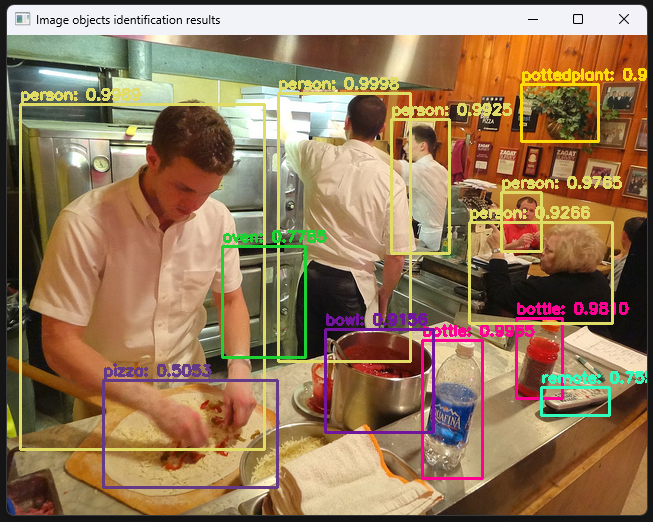


Рисунок 2.2 – Результирующее изображение идентификации объектов

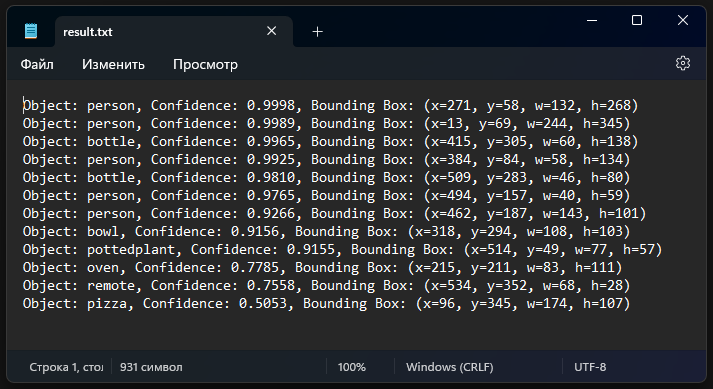


Рисунок 2.3 – Результирующий файл идентификации объектов

Код реализации пользовательской утилиты для идентификации объектов изображения на языке Python:

001. import argparse

002. import os

003. import time

004.

005. import cv2

006. import numpy as np

007.

008.

009. def load\_yolo\_model(yolo\_model\_path):

010. yolo\_labels\_path = os.path.sep.join([yolo\_model\_path, "coco.names"])

011. yolo\_labels = open(yolo\_labels\_path).read().strip().split("\n")

012.

013. np.random.seed(42)

014. colors = np.random.randint(0, 255, size=(len(yolo\_labels), 3), dtype="uint8")

015.

016. yolo\_weights\_path = os.path.sep.join([yolo\_model\_path, "yolov3.weights"])

017. yolo\_config\_path = os.path.sep.join([yolo\_model\_path, "yolov3.cfg"])

018.

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

020. net = cv2.dnn.readNetFromDarknet(yolo\_config\_path, yolo\_weights\_path)

021.

022. return net, yolo\_labels, colors

023.

024.

025. def detect\_objects(image, net, min\_confidence, nms\_threshold):

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

027.

028. ln = net.getLayerNames()

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

030.

031. blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)

032. net.setInput(blob)

033. start = time.time()

034. layer\_outputs = net.forward(ln)

035. end = time.time()

036.

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

038.

039. boxes = []

040. confidences = []

041. class\_ids = []

042.

043. for output in layer\_outputs:

044. for detection in output:

045. scores = detection[5:]

046. class\_id = np.argmax(scores)

047. confidence = scores[class\_id]

048. if confidence > min\_confidence:

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

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

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

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

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

054. confidences.append(float(confidence))

055. class\_ids.append(class\_id)

056.

057. idxs = cv2.dnn.NMSBoxes(boxes, confidences, min\_confidence, nms\_threshold)

058.

059. return idxs, boxes, confidences, class\_ids

060.

061.

062. def display\_result\_image(image, idxs, boxes, confidences, class\_ids, labels, colors):

063. if len(idxs) > 0:

064. for i in idxs.flatten():

065. (x, y, w, h) = boxes[i]

066. color = [int(c) for c in colors[class\_ids[i]]]

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

068. text = "{}: {:.4f}".format(labels[class\_ids[i]], confidences[i])

069. cv2.putText(image, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX,0.5, color, 2)

070. cv2.imshow("Image objects identification results", image)

071. cv2.waitKey(0)

072.

073.

074. def create\_result\_txt(idxs, boxes, confidences, class\_ids, labels, output\_file):

075. with open(output\_file, 'w') as f:

076. if len(idxs) > 0:

077. for i in idxs.flatten():

078. (x, y, w, h) = boxes[i]

079. object\_label = labels[class\_ids[i]]

080. object\_confidence = confidences[i]

081. text = f"Object: {object\_label}, Confidence: {object\_confidence:.4f}, Bounding Box: (x={x}, y={y}, w={w}, h={h})\n"

082. f.write(text)

083.

084.

085. def main():

086. parser = argparse.ArgumentParser(description='Image objects identifier')

087. parser.add\_argument('-i', '--image', type=str, required=True, help='Path to input image')

088. parser.add\_argument('-y', '--yolo', type=str, required=True, help='Path to directory with YOLO model')

089. parser.add\_argument('-c', '--confidence', type=float, default=0.5, help='Minimum probability to filter weak detections')

090. parser.add\_argument('-t', '--threshold', type=float, default=0.3, help='Threshold when applying non-maxima suppression')

091. parser.add\_argument('-o', '--output', type=str, default='result.txt', help='Path to output text file')

092. args = parser.parse\_args()

093.

094. net, LABELS, COLORS = load\_yolo\_model(args.yolo)

095. image = cv2.imread(args.image)

096.

097. idxs, boxes, confidences, classIDs = detect\_objects(image, net, args.confidence, args.threshold)

098.

099. create\_result\_txt(idxs, boxes, confidences, classIDs, LABELS, args.output)

100. display\_result\_image(image, idxs, boxes, confidences, classIDs, LABELS, COLORS)

101.

102.

103. if \_\_name\_\_ == '\_\_main\_\_':

104. main()

**3 ВЫВОДЫ**

В ходе выполнения лабораторной работы была подготовлена пользовательская утилита для индентификации объектов на изображении. Утилита реализована на языке Python с применением новейшей сети обнаружения объектов YOLO. Тестирование производилось на ОС Microsoft Windows 10 и 11, среднее время обнаружения ~1.1 c.