

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

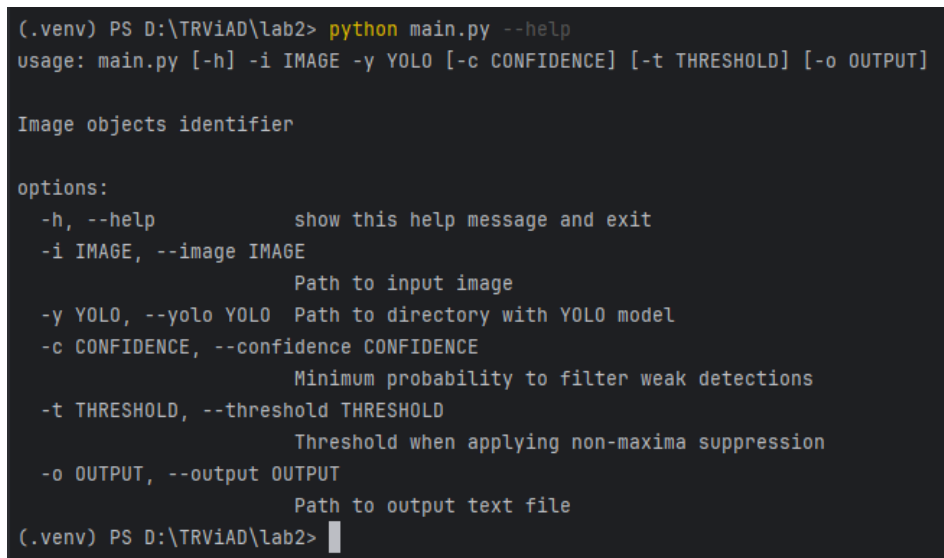
1 ЗАДАНИЕ

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.



```
(.venv) PS D:\TRViAD\lab2> python main.py --help
usage: main.py [-h] -i IMAGE -y YOLO [-c CONFIDENCE] [-t THRESHOLD] [-o OUTPUT]

Image objects identifier

options:
  -h, --help            show this help message and exit
  -i IMAGE, --image IMAGE
                        Path to input image
  -y YOLO, --yolo YOLO  Path to directory with YOLO model
  -c CONFIDENCE, --confidence CONFIDENCE
                        Minimum probability to filter weak detections
  -t THRESHOLD, --threshold THRESHOLD
                        Threshold when applying non-maxima suppression
  -o OUTPUT, --output OUTPUT
                        Path to output text file
(.venv) PS D:\TRViAD\lab2>
```

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 обнаружения 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).

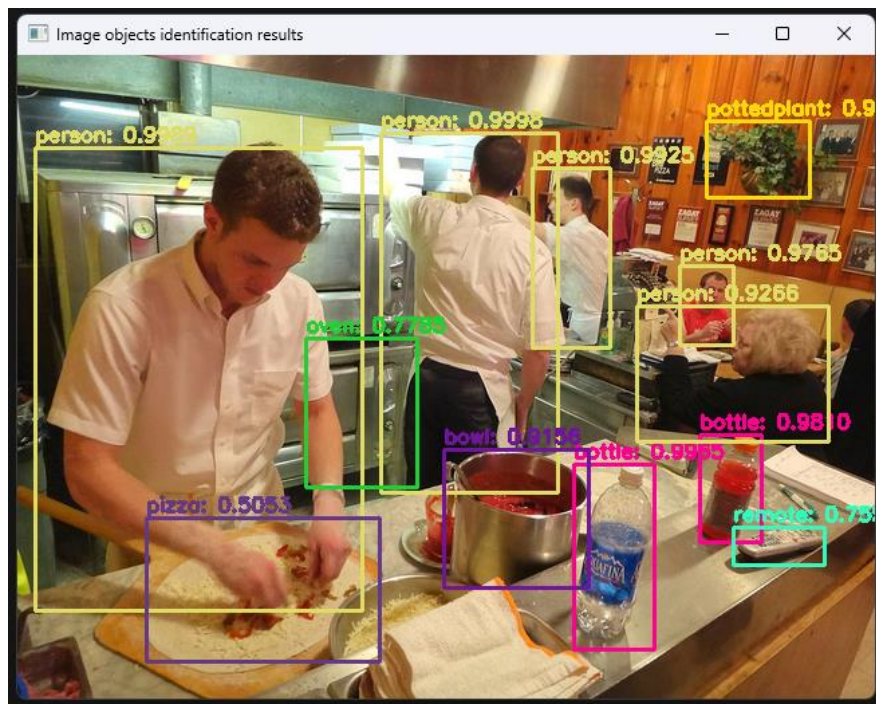


Рисунок 2.2 – Результирующее изображение идентификации объектов

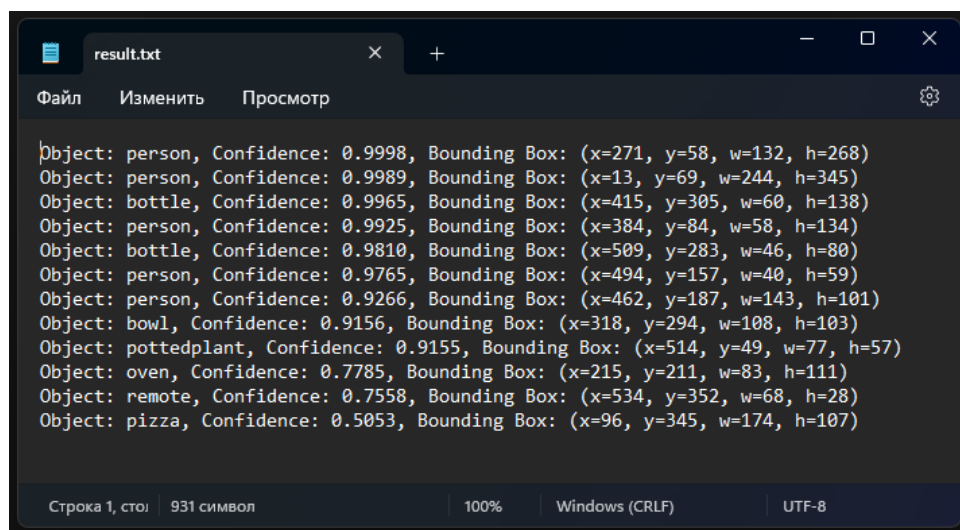


Рисунок 2.3 – Результирующий файл идентификации объектов

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 идентификации 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 с.