Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики
Кафедра Математического обеспечения и суперкомпьютерных технологий

Летняя школа Intel – ННГУ по компьютерному зрению

Практическая работа №3
Детектирование объектов с использованием методов глубокого обучения

Васильев Е.П.

Нижний Новгород
2021

1 Цели и задачи работы

Цель работы состоит в изучении и применении глубоких моделей для решения задачи детектирования объектов на изображениях с применением инструмента Intel Distribution of OpenVINO Toolkit.

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить лекцию «Детектирование объектов разных классов на изображениях с использованием глубоких нейронных сетей».
- Изучить глубокие модели для детектирования объектов, входящие в состав Open Model Zoo, и выбрать модель для дальнейшего решения задачи. В данной работе демонстрируется пример использования модели SSD300 [4].
- Разработать программный код для детектирования объектов с использованием OpenVINO. Результат детектирования необходимо отобразить на исходном изображении.

Отметим, что процедура настройки рабочего окружения подробно описана в предыдущей практике, поэтому данный шаг в настоящем описании опущен.

2 Глубокие модели для детектирования объектов, входящие в состав Open Model Zoo

В настоящее время распространение получило несколько групп глубоких моделей, которые применяются для решения задачи детектирования объектов.

- Region-based Convolutional Networks (RCNN) [2].
- Region-Based Fully Convolutional Networks (RFCN) [3].
- Single Shot MultiBox Detector (SSD) [4].
- You Only Look Once (YOLO) [5].

Более полную информацию можно найти в лекции «Детектирование объектов разных классов на изображениях с использованием глубоких нейронных сетей». Здесь же остановимся на некоторых моделях для детектирования объектов, входящих в Open Model Zoo.

В Open Model Zoo присутствует ***семейство моделей SSD***: ssd300, ssd512, mobilenet-ssd и другие. Указанная группа моделей имеет идентичный вход и выход. Вход SSD моделей, обученных в Caffe, представляет собой тензор размера $[B \times C \times H \times W]$, где B – количество обрабатываемых изображений в пачке, подаваемой на вход сети, C – количество каналов изображений, H, W – высота и ширина входного изображения соответственно. Выход SSD-моделей, сконвертированных в промежуточное представление OpenVINO, представляет собой тензор размера $[1 \times 1 \times N \times 7]$, в котором каждая строка (последняя размерность тензора) содержит следующие параметры: [image_number, classid, score, left, bottom, right, top], где image_number – номер изображения; classid – идентификатор класса; score – достоверность присутствия объекта в выделенной области; left, bottom, right, top – нормированные координаты окаймляющих прямоугольников в диапазоне от 0 до 1.

Наряду с этим, в состав Open Model Zoo входит семейство ***моделей, построенных на базе RCNN***. К таковым относятся faster_rcnn_resnet50_coco, faster_rcnn_inception_v2_coco, mask_rcnn_resnet50_atrous_coco, mask_rcnn_inception_v2_coco и некоторые другие. Вход RCNN-моделей, обученных с использованием библиотеки TensorFlow, отличается от SSD-моделей. Сконвертированые RCNN модели содержат два входных тензора: первый тензор размера $[B \times C \times H \times W]$, отвечающий набору обрабатываемых изображений, второй тензор размерности $[B \times C]$, где C – вектор, состоящий из трех значений в формате $[H, W, S]$, где S – коэффициент масштабирования исходного изображения. Выход сконвертированных моделей по аналогии с SSD-моделями представляет собой тензор размера $[1 \times 1 \times N \times 7]$ или $[N \times 7]$, в котором каждая строка (последняя размерность тензора) содержит следующие параметры: [image_id, label, conf,

(x_min, y_min), (x_max, y_max)], где image_id – номер изображения; label – номер класса; conf – достоверность присутствия объекта в выделенной области; (x_min, y_min), (x_max, y_max) – нормированные координаты окаймляющих прямоугольников в диапазоне от 0 до 1.

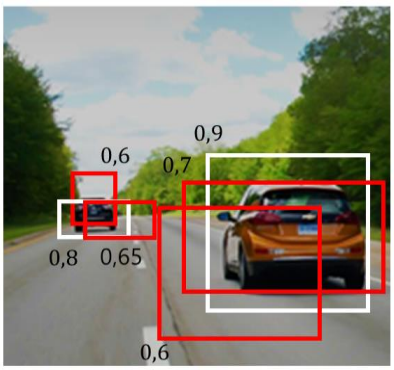
В Open Model Zoo помимо публичных моделей, приведенных выше, также имеются собственные модели Intel, которые решают задачи детектирования специфичных объектов (транспортных средств, автомобильных номеров, людей, лиц, текста). К таким моделям, в частности, относятся face-detection-retail, person-detection-retail, vehicle-detection-adas, vehicle-license-plate-detection-barrier-0106 и другие. Данные модели имеют вход и выход, аналогичный SSD-моделям. Полное описание детектируемых классов объектов, входа и выхода для каждой модели доступно в документации или директории OpenVINO, указанной ниже.

```
<openvino_dir>/deployment_tools/open_model_zoo/models/intel
```

2.1 Model API для использования моделей глубокого обучения в OpenVINO

Рассмотрим процедуру запуска

Как было показано выше, результатом работы модели у некоторых моделей в Open Model Zoo является тензор размера $[1 \times 1 \times N \times 7]$ или $[N \times 7]$, однако это не «настоящий» выход нейросети. Выходом работы нейросети как правило является большой список ограничивающих прямоугольников, с разной степенью уверенности в локализации объекта.



19×19

Далее для списка полученных прямоугольников проводится процедура подавления немаксимумов – Non-Maximum Suppression (NMS). Для части моделей в Open Model Zoo данная процедура встроена отдельным слоем в нейросеть, но для моделей без встроенного NMS приходится выполнять его самостоятельно, причем код для NMS зависит от конкретного типа модели. Чтобы пользователям OpenVINO было удобней, в пакете OpenVINO содержится несколько готовых реализаций, в которых достаточно только загрузить веса моделей и не программировать NMS.

Для C++ и Python разработаны обертки для 5 моделей: centernet, faceboxes, retinaface, ssd, yolo (https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos/common/cpp/models/include/models), (https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos/common/python/models).

Познакомиться с использованием Model API можно в Object Detection Demo в Open Model Zoo (https://github.com/openvinotoolkit/open_model_zoo/blob/master/demos/object_detection_demo/python/object_detection_demo.py).

В качестве наиболее полезных вещей для широкого круга пользователей из Model API можно выделить:

- возможность использовать единую функцию для открытия изображения 1 изображения, папки изображений, видео с возможностью зацикливания получения данных;
- возможность легкого создания асинхронного конвейера обработки данных;

- возможность использования готового препроцессинга и постпроцессинда данных для моделей из OpenModelZoo.

В данной практике мы попробуем использовать готовое Model API для object detection моделей.

2.2 Запуск примера OpenVINO для детектирования объектов на изображениях

Рассмотрим процедуру запуска

примера для детектирования объектов, входящего в состав OpenVINO. В пакете OpenVINO содержится файл **detection_sample.py**, который позволяет детектировать объекты на изображениях при помощи глубоких нейронных сетей. На официальном сайте присутствует полное описание данного примера и инструкций по его запуску [9]. Здесь приведем краткую информацию, необходимую для быстрого старта.

Для запуска примера скачайте, сконвертируйте и запустите модель SSD300, последовательность команд приведена ниже. В приведенном перечне команд необходимо заменить пути в угловых скобках на реальные пути на вашем компьютере.

```
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\download
er.py" --name ssd300 --output_dir <destination_folder>
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\converte
r.py" --name ssd300 --download_dir <destination_folder>
python "C:\Program Files
(x86)\Intel\openvino_2021\inference_engine\samples\python\object_detection
_sample_ssd\object_detection_sample_ssd.py" -i <path_to_image> -m
<path_to_model>\ssd300.xml
```

После запуска данного приложения в консоли должен появиться выход работы данного примера, а в текущей директории появится изображение **out.bmp** с обведенными объектами (если директория будет открыта для записи файлов).

```
[ INFO ] Loading Inference Engine
[ INFO ] Loading network files:
ssd300.xml
ssd300.bin
[ INFO ] Device info:
CPU
MKLDNNPlugin version ......... 2.1
Build ........... 2021.1.0-1237-bece22ac675-releases/2021/1
inputs number: 1
input shape: [1, 3, 300, 300]
input key: data
[ INFO ] File was added:
[ INFO ] dog.jpg
[ WARNING ] Image dog.jpg is resized from (1200, 2274) to (300, 300)
[ INFO ] Preparing input blobs
[ INFO ] Batch size is 1
[ INFO ] Preparing output blobs
[ INFO ] Loading model to the device
[ INFO ] Creating infer request and starting inference
[ INFO ] Processing output blobs
[0,3] element, prob = 0.015166 (1551,32)-(1695,129) batch id : 0
```

```
[1,3] element, prob = 0.00843134 (1507,22)-(1727,174) batch id : 0
[2,3] element, prob = 0.00616033 (1589,58)-(1711,146) batch id : 0
...
[197,18] element, prob = 0.00396589 (-132,-63)-(532,257) batch id : 0
[198,18] element, prob = 0.00327471 (-256,571)-(692,828) batch id : 0
[199,18] element, prob = 0.00327383 (-265,698)-(696,968) batch id : 0
[ INFO ] Image out.bmp created!
[ INFO ] Execution successful
[ INFO ] This sample is an API example, for any performance measurements
please use the dedicated benchmark_app tool
```

Код данного и остальных примеров можно использовать для изучения программного интерфейса OpenVINO. Далее приводится пошаговая инструкция по разработке приложения для детектирования объектов.

3 Разработка приложения для детектирования объектов на изображениях с использованием OpenVINO

3.1 Скачивание модели YOLOv3

Модель YOLOv3 является широко используемой моделью детектирования объектов, которую легко обучить под собственный набор данных, поэтому в данной работе будем использовать ее. Не забудьте заранее установить зависимости для конвертации TensorFlow моделей, поскольку в репозитории модель именно в этом формате.

```
pip install –r " C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\model_optimizer\requirements_tf
.txt"
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\download
er.py" --name yolo-v3-tiny-tf --output_dir <destination_folder>
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\converte
r.py" --name yolo-v3-tiny-tf --download_dir <destination_folder>
```

3.2 Рабочие скрипты

Шаблон для выполнения практической работы расположен в файле **`object_detector.py`**, содержащий функции **`get_plugin_configs`**, **`build_argparser`**, **`draw_detections`**, **`main`**.

Далее поэтапно реализуем приведенные методы.

3.3 Функция отрисовки обнаруженных объектов

Выходом Model API для Object Detection моделей является список объект Detection (https://github.com/openvinotoolkit/open_model_zoo/blob/master/demos/common/python/models/utils.py), который содержит следующие параметры: [left, bottom, right, top, score, id], где id – номер класса; score – достоверность присутствия объекта в данном месте; left, bottom, right, top – нормированные координаты окаймляющих прямоугольников.

Для обработки выхода необходимо реализовать метод **`draw_detection`**, который отрисует на изображении обнаруженные объекты. Последовательность работы указанного метода следующая.

1. Получить текущую строку матрицы.
2. Получить достоверность детектирования (score).
3. Если достоверность больше некоторого порогового значения (для определенности можно выбрать 0.5), то получить идентификатор класса и координаты окаймляющего прямоугольника.

4. Отрисовать прямоугольник на изображении средствами OpenCV. Для отрисовки прямоугольника воспользуйтесь функцией **cv2.rectangle**. Прототип и описание параметров данной функции приведены ниже.

```
cv2.rectangle(img, point1, point2, color, line_width)
```

- **img** – изображение, на котором необходимо выполнить отрисовку.
- **point1 = (x,y)** – кортеж из двух целых чисел, соответствующий координатам левого верхнего угла окаймляющего прямоугольника.
- **point2 = (x,y)** – кортеж из двух целых чисел, соответствующий координатам правого нижнего угла окаймляющего прямоугольника.
- **color = (B,G,R)** – кортеж из трех целых чисел от 0 до 255, определяющий цвет линий.
- **line_width = 1** – вещественное положительное число, определяющее толщину линий.

Для отображения текста, содержащего класс объекта, воспользуйтесь функцией **cv2.puttext**. Прототип и описание параметров данной функции приведены ниже.

```
cv2.putText(img, text, point, cv2.FONT_HERSHEY_COMPLEX, text_size, color,
1)
```

- **img** – изображение, на котором необходимо нарисовать.
- **text** – строка надписи.
- **point** – точка старта надписи на изображении.
- **color** – кортеж из трех целых чисел от 0 до 255, определяющий цвет текста.
- **text_size = 0.45** – вещественное положительное число, определяющее размер текста.

3.4 Создание основной функции

Создайте функцию **main** для примера, которая выполняет следующие действия:

1. Разбор аргументов командной строки.
2. Открытие входных данных.
3. Инициализацию OpenVINO
4. Загрузку модели YOLOv3
5. Инициализацию асинхронного конвейера
6. Отправку данных на обработку
7. Получение обработанных данных
8. Отрисовка детектированных объектов на изображении.
9. Отображение полученного изображения на экране.

С помощью **open_images_capture** функции откройте источник данных.

```
cap = open_images_capture(args.input, True)
```

Инициализируйте OpenVINO (как в прошлой практике)

```
ie = IECore()
```

Инициализируйте параметры плагинов

```
plugin_configs = get_plugin_configs('CPU', 0, 0)
```

Загрузите YOLOv3

```
detector = models.YOLO(ie, pathlib.Path(args.model), labels=args.classes,
threshold=args.prob_threshold, keep_aspect_ratio=True)
```

Инициализируйте асинхронных конвейер обработки

```
detector_pipeline = AsyncPipeline(ie, detector, plugin_configs,
                                  device='CPU', max_num_requests=1)
```

В цикле получите очередной фрейм

```
img = cap.read()
```

Выполните асинхронную обработку фрейма с помощью YOLOv3

```
frame_id = 0
detector_pipeline.submit_data(img,frame_id,{'frame':img,'start_time':0})
detector_pipeline.await_any()
results, meta = detector_pipeline.get_result(frame_id)
```

Отрисуйте результат на фрейме и выведите результат на экран

```
draw_detections(img, results, None, args.prob_threshold)
cv2.imshow('Image with detections', img)
```

Для возможности прервать бесконечный цикл сделайте выход по клавише q

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

4 Запуск приложения

Запуск разработанного приложения можно выполнить из командной строки. Для этого необходимо открыть командную строку. Строка запуска имеет вид:

```
python object_detector.py -i image.jpg -m yolo-v3-tiny-tf.xml
```

где аргумент **-i** задает путь до изображения, аргумент **-m** задает путь до конфигурации модели.

Результат запуска приложения выглядит следующим образом: выводится сообщение о старте примера, затем открывается графическое окно, в котором отрисовано изображение с обнаруженными объектами (рис. 1).

```
 [ INFO ] Start IE detection sample
```

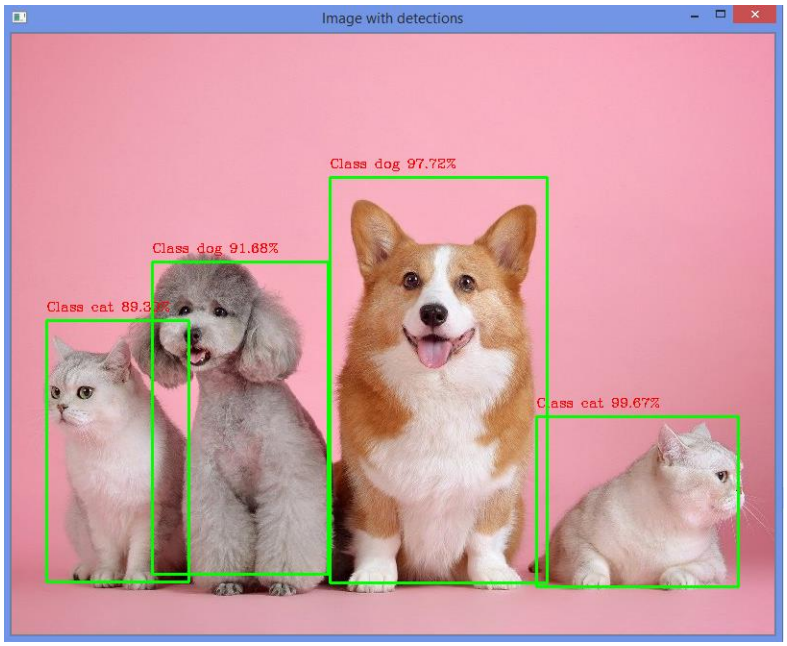


Рис. 1. Пример вывода программы детектирования

5 Дополнительные задания

Созданный пример детектирования объектов содержит минимально необходимый функционал. В качестве дополнительных заданий предлагается обеспечить поддержку следующих возможностей:

1. Добавление записи обработанных фреймов в видеофайл.
2. Вычисление времени обработки одного изображения.

Данные задания предлагается выполнить самостоятельно, опираясь на документацию и примеры, входящие состав пакета OpenVINO.

6 Литература

1. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер. – 2018. – 400с.
2. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014.
3. Dai J., Li Y., He K., Sun J. R-FCN: Object detection via region-based fully convolutional networks. – 2016.
4. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016.
5. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015.
6. Рамальо Л. Python. К вершинам мастерства / Пер. с англ. Слинкин А.А. – М.: ДМК Пресс. – 2016. – 768 с.
7. Страница репозитория Open Model Zoo [https://github.com/openvinotoolkit/open_model_zoo].
8. Документация Intel Distribution of OpenVINO Toolkit [https://docs.openvinotoolkit.org/latest/index.html].
9. OpenVINO detection sample [https://docs.openvinotoolkit.org/latest/openvino_inference_engine_ie_bridges_python_sample_object_detection_sample_ssd_README.html].