计算机视觉 (实验三)
物体分类识别和电子围栏

智科三班　严中圣　222020335220177

2022 年 12 月 12 日

1　实验目的

图像处理 (image processing)，用计算机对图像进行分析，以达到所需结果的技术。图像处理将会是物联网产业发展的重要支柱之一。本实验旨在加深本课程中图像预处理、人脸检测、后处理等重要的知识点的理解和实践，并实现对视频中物体的分类识别，最终生成标注后的视频/流媒体。

2　实验环境

- PyCharm 2022.1.3 (Professional Edition)
- OS: Windows 11 22H2
- CPU: 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
- Packages: python 3.8.15 torch>=1.7.0 torchvision>=0.8.1 numpy>=1.18.5 opencv-python>=4.1.1

3　实验内容

(1) 对视频（固定拍摄位置）进行逐帧检测，对其中的物体进行识别；

(2) 实现电子围栏，即对特定类别的物体进入制定区域后进行报警；

(3) 对检测结果进行后处理，将结果在图像中进行标注，并实时播放或者保存视频；

(4) 效率思考，若逐帧处理效率较低，考虑提高优化方法。

4　实验步骤

4.1　物体检测分类识别

4.1.1　目标检测算法概述

物体检测（object detection）是计算机视觉中一个重要的分支，其大致功能是在一张图片中，用最小矩形框框出目标物体位置，并进行分类。

物体检测的两个步骤可以概括为：

图 1: 目标检测任务

1. 步骤 1：检测目标位置（生成矩形框）

2. 步骤 2：对目标物体进行分类

物体检测主流的算法框架大致分为 one-stage 与 two-stage。two-stage 算法代表有 R-CNN 系列，onestage 算法代表有 Yolo 系列。two-stage 算法将步骤一与步骤二分开执行，输入图像先经过候选框生成网络（例如 faster rcnn 中的 RPN 网络），再经过分类网络；one-stage 算法将步骤一与步骤二同时执行，输入图像只经过一个网络，生成的结果中同时包含位置与类别信息。two-stage 与 one-stage 相比，精度高，但是计算量更大，所以运算较慢。

为了实现物体分类识别任务，我们调用了现今最为流行的 Yolov5 框架进行检测，下面对 Yolo 算法进行介绍，完整项目见https://github.com/ZS-Yan/yolov5。

4.1.2 Yolo 算法介绍

(1) 算法概述

现在的大多数目标检测方式都是将物体检测问题，最后会转变成一个分类问题。在检测中，detection systems 采用一个 classifier 去评估一张图像中，各个位置一定区域的 window 或 bounding box 内，是否包含一个物体以及包含了哪种物体。而 R-CNN、Fast R-CNN 则采用的是 region proposals 的方法，先生成一些可能包含待检测物体的 potential bounding box，再通过一个 classifier 去判断每个 bounding box 里是否包含有物体，以及物体所属类别的 probability 或者 confidence。这种方法的 pipeline 需要经过好几个独立的部分，所以检测速度很慢，也难以去优化，因为每个独立的部分都需要单独训练。Yolo 算法将 object detection 的框架设计为一个 regression problem。直接从图像像素到 bounding box 以及 probabilities。这个 YOLO 系统如图看了一眼图像就能 predict 是否存在物体，他们在哪个位置，所以也才叫 You Only Look Once。

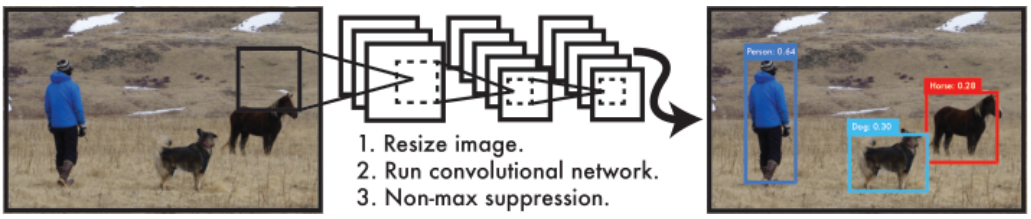

图 2: Yolo 模型框架

这样的统一的架构，对比之前如 R-CNN、Fast R-CNN 的 pipeline，有以下几点好处：

(1) YOLO 检测系统非常非常的快。受益于将 detection 架构设计成一个 regression problem，以及简单的 pipeline。在 Titan X 上，不需要经过批处理，标准版本的 YOLO 系统可以每秒处理 45 张图像；YOLO 的极速版本可以处理 150 帧图像。这就意味着 YOLO 可以以小于 25 毫秒延迟的处理速度，实时地处理视频。同时，YOLO 实时检测的 mean Average Precision（mAP）是其他实时检测系统的两倍。

(2) YOLO 在做 predict 的时候，YOLO 使用的是全局图像。与 sliding window 和 region proposals 这类方法不同，YOLO 一次“看”一整张图像，所以它可以将物体的整体（contextual）的 class information 以及 appearance information 进行 encoding。目前最快最好的 Fast R-CNN ，较容易误将图像中的 background patches 看成是物体，因为它看的范围比较小。YOLO 的 background errors 比 Fast R-CNN 少一半多。

(3) YOLO 学到物体更泛化的特征表示。当在自然场景图像上训练 YOLO，再在 artwork 图像上去测试 YOLO 时，YOLO 的表现甩 DPM、R-CNN 好几条街。YOLO 模型更能适应新的 domain。

(2) 检测方式

Yolo 算法采用了 Unified Detection 的方式，YOLO 检测系统，先将输入图像分成 $S \times S$ 的网格，如果一个物体的中心掉落在一个 grid cell 内，那么这个 grid cell 就负责检测这个物体。

每个网格单元预测这些盒子的 B 个边界框和置信度分数。这些置信度分数反映了该模型对盒子是否包含目标的信息，以及它预测盒子的准确程度。在形式上，模型将置信度定义为 $P_r(Object) \times IOU_{pred}^{truth}$。如果该单元格中不存在目标，则置信度分数应为零。否则，模型希望置信度分数等于预测框与真实值之间联合部分的交集（IOU）。

每个边界框包含 5 个预测：x, y,w, h 和置信度。(x, y) 坐标表示边界框相对于网格单元边界框的中心。宽度和高度是相对于整张图像预测的。最后，置信度预测表示预测框与实际边界框之间的 IOU。

每个网格单元还预测 C 个条件类别概率 $P_r(Class_i|Object)$。这些概率以包含目标的网格单元为条件。每个网格单元模型只预测的一组类别概率，而不管边界框的的数量 B 是多少。在测试时，我们乘以条件类概率和单个盒子的置信度预测，

$$P_r\left(Class_i \mid Object\right) \times P_r(Object) \times IOU_{pred}^{truth} = P_r\left(Class_i\right) \times IOU_{pred}^{truth} \tag{1}$$

它为我们提供了每个框特定类别的置信度分数。这些分数编码了该类出现在框中的概率以及预测框拟合目标的程度。

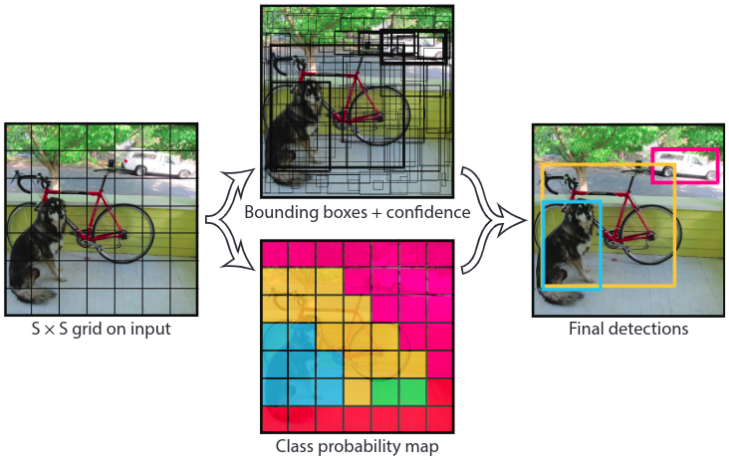

图 3: Yolo 目标检测模型

(3) 网络结构设计

Yolo 将此模型作为卷积神经网络来实现，网络的初始卷积层从图像中提取特征，而全连接层预测输出概率和坐标。网络有 24 个卷积层，后面是 2 个全连接层。模型只使用 1×1 降维层，后面是 3×3 卷积层，完整的网络如图所示。

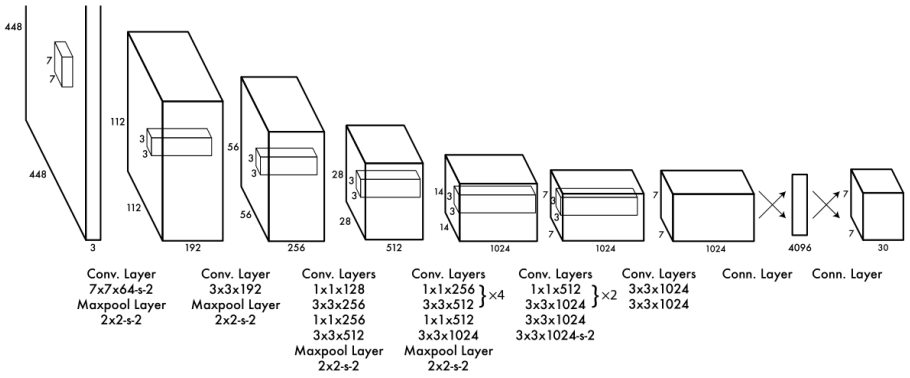

图 4: Yolo 算法网络结构设计

4.1.3 实时物体检测分类识别

我们调用 Yolov5 的预训练模型对采集视频进行物体检测与分类识别，效果如下：

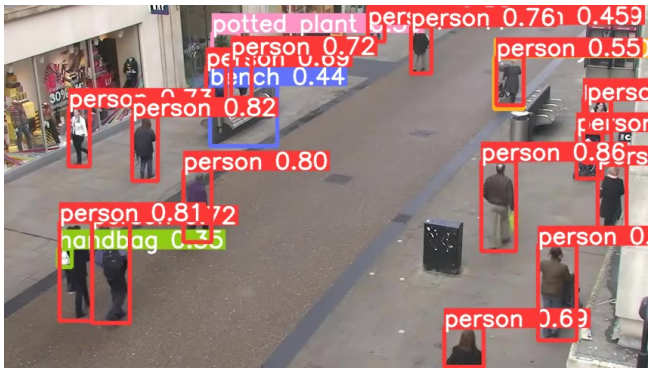

图 5: 物体检测与分类识别效果展示

4.2 电子围栏检测报警

为了实现电子围栏的报警检测任务，我们划定一片固定区域，实时判断该区域是否有行人经过，如检测框重叠即发出报警，实现代码如下：

```
limit_box = [600, 200, 800, 800]
            p1, p2 = (int(box[0]), int(box[1])), (int(box[2]), int(box[3]))
            cv2.rectangle(self.im, (limit_box[0], limit_box[1]), (limit_box[2],
                limit_box[3]), color=(255, 255, 0), thickness=3)
            cv2.rectangle(self.im, p1, p2, color, thickness=self.lw, lineType=
                cv2.LINE_AA)
            if label:
                if label[:6] == 'person':
                    if p1[0] > limit_box[0] and p1[1] > limit_box[1] and p2[0] <
                        limit_box[2] and p2[1] < limit_box[3]:
                        cv2.putText(self.im, "Warning!", (50, 50), fontFace=0,
                            fontScale=self.lw / 3, color=(255,0,0),thickness=2)
```

检测效果如下：

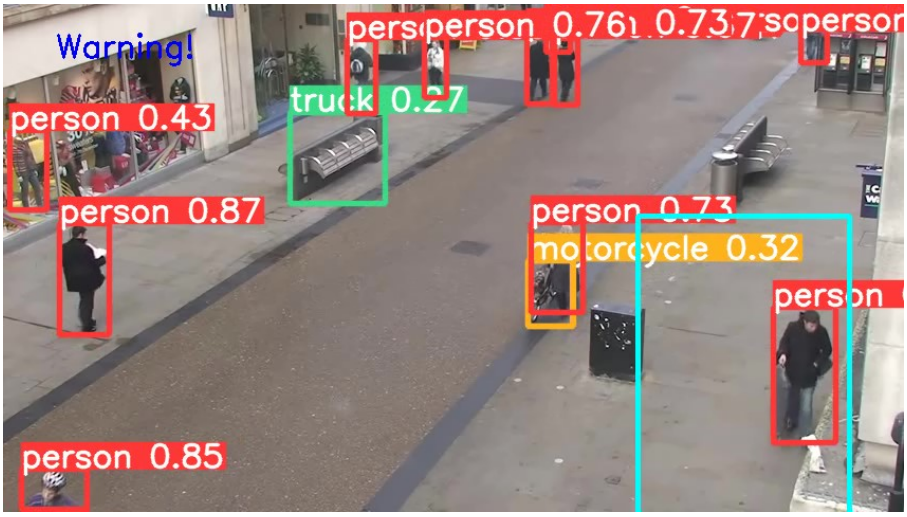

图 6: 电子围栏检测效果展示

A 附录

A.1 物体检测代码

```python
import argparse
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
    LoadScreenshots, LoadStreams
from utils.general import (LOGGER, Profile, check_file, check_img_size,
    check_imshow, check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args,
                              scale_boxes, strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
        weights=ROOT / 'yolov5s.pt',  # model path or triton URL
        source=ROOT / 'data/images',  # file/dir/URL/glob/screen/0(webcam)
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
```

```
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.streams') or (is_url and not is_file)
    screenshot = source.lower().startswith('screen')
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
```

```
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt,
            vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True)
                if visualize else False
            pred = model(im, augment=augment, visualize=visualize)

        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes,
                agnostic_nms, max_det=max_det)

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s
            )

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode
                == 'image' else f'_{frame}')  # im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain
                whwh
```

```
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(
                names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).
                    round()

                # Print results
                for c in det[:, 5].unique():
                    n = (det[:, 5] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to
                        string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).
                            view(-1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)
                             # label format
                        with open(f'{txt_path}.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or save_crop or view_img:  # Add bbox to image
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if hide_conf
                            else f'{names[c]} {conf:.2f}')
                        annotator.box_label(xyxy, label, color=colors(c, True))
                    if save_crop:
                        save_one_box(xyxy, imc, file=save_dir / 'crops' / names[
                            c] / f'{p.stem}.jpg', BGR=True)

            # Stream results
            im0 = annotator.result()
            if view_img:
                if platform.system() == 'Linux' and p not in windows:
                    windows.append(p)
                    cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.
                        WINDOW_KEEPRATIO)  # allow window resize (Linux)
                    cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
```

```
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on results videos
                        vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer[i].write(im0)

            # Print time (inference-only)
            LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt', help='model path or triton URL')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images', help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
```

```
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU
        threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum
        detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or
        0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to
         *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save
        confidences in --save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped
        prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save
        images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class:
         --classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-
        agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented
        inference')
    parser.add_argument('--visualize', action='store_true', help='visualize
        features')
    parser.add_argument('--update', action='store_true', help='update all models
        ')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save
        results to project/name')
    parser.add_argument('--name', default='exp', help='save results to project/
        name')
    parser.add_argument('--exist-ok', action='store_true', help='existing
        project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding
        box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true',
        help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help=
        'hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-
        precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for
        ONNX inference')
    parser.add_argument('--vid-stride', type=int, default=1, help='video frame-
        rate stride')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt
```

```
222
223
224 def main(opt):
225     check_requirements(exclude=('tensorboard', 'thop'))
226     run(**vars(opt))
227
228
229 if __name__ == "__main__":
230     opt = parse_opt()
231     main(opt)
```