



移动机器人系列课程

机器人视觉技术与深度学习课程



深圳市史河机器人科技有限公司

模型训练与目标检测

1. 什么是目标检测

目标检测的最终目的就是对于图像中的目标物体位置和大小进行判定，但是不同物体有着不同的特征，再加上一些外界因素的影响，所以目标检测具有很大的难度，这也是机器视觉领域比较头疼的一个问题。

我们拿到一张图片后需要将其分为三个步骤进行理解（目标识别过程）：

第一步：分类。用事前确定好的类别或实例 ID 对化为信息的图像结构进行描述。

第二步：检测。上一步是对整张图片内容的描述，这一步则需要选定一个物体目标进行检测，获取物体所处位置以及类别信息。

第三步：分割。这一步需要对语义和实例进行分割，并得出像素属于哪个目标物体或哪个场景的结论。

2. 基于深度学习的目标检测算法

基于深度学习的主流目标检测算法根据有无候选框生成阶段分为双阶段目标检测算法和单阶段目标检测算法两类。双阶段目标检测算法先对图像提取候选框，然后基于候选区域做二次修正得到检测结果，检测精度较高，但检测速度较慢；单阶段目标检测算法直接对图像进行计算生成检测结果，检测速度快，但检测精度低。

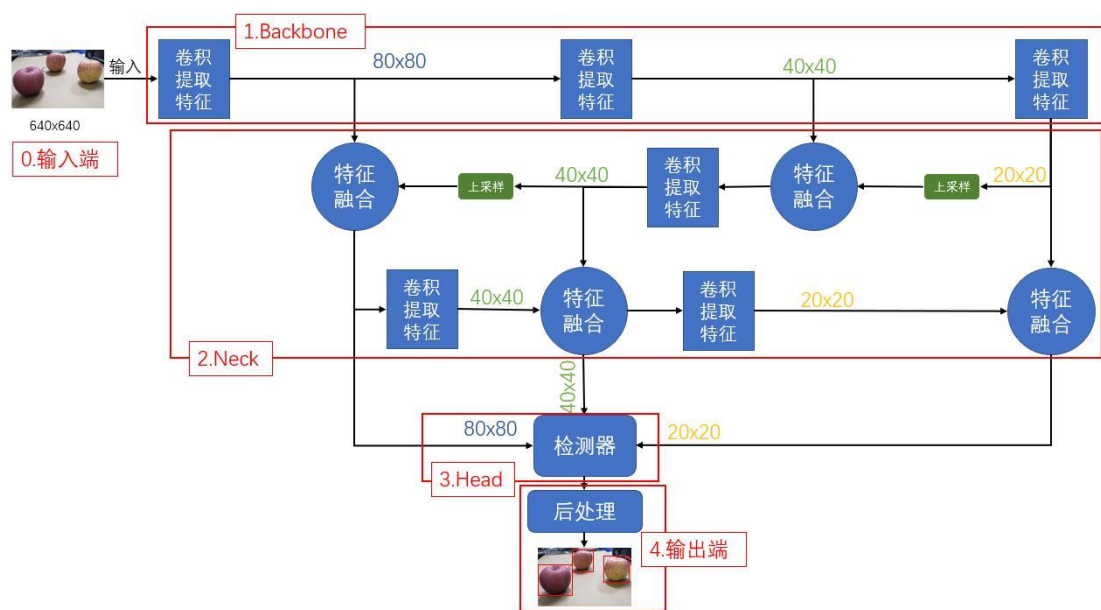
2.1 双阶段目标检测算法

该算法需要先借助 SelectiveSearch 选出图像中的候选区域, 之后还需要对候选区域进行再次检测, 从而得出最后检测结果, 比较常用的算法主要有 OverFeat、R-CNN、MaskRCNN 等。

2.2 单阶段目标检测算法

该算法依据的是回归分析思想, 所以也被称作回归分析目标检测算法。该算法之所以被称作单阶段目标检测算法是因为该算法不需要生成候选区域, 而是直接对整个图像进行检测, 从而获得目标位置类别和位置信息, 比较常用的检测算法主要有 YOLO 和 SSD。

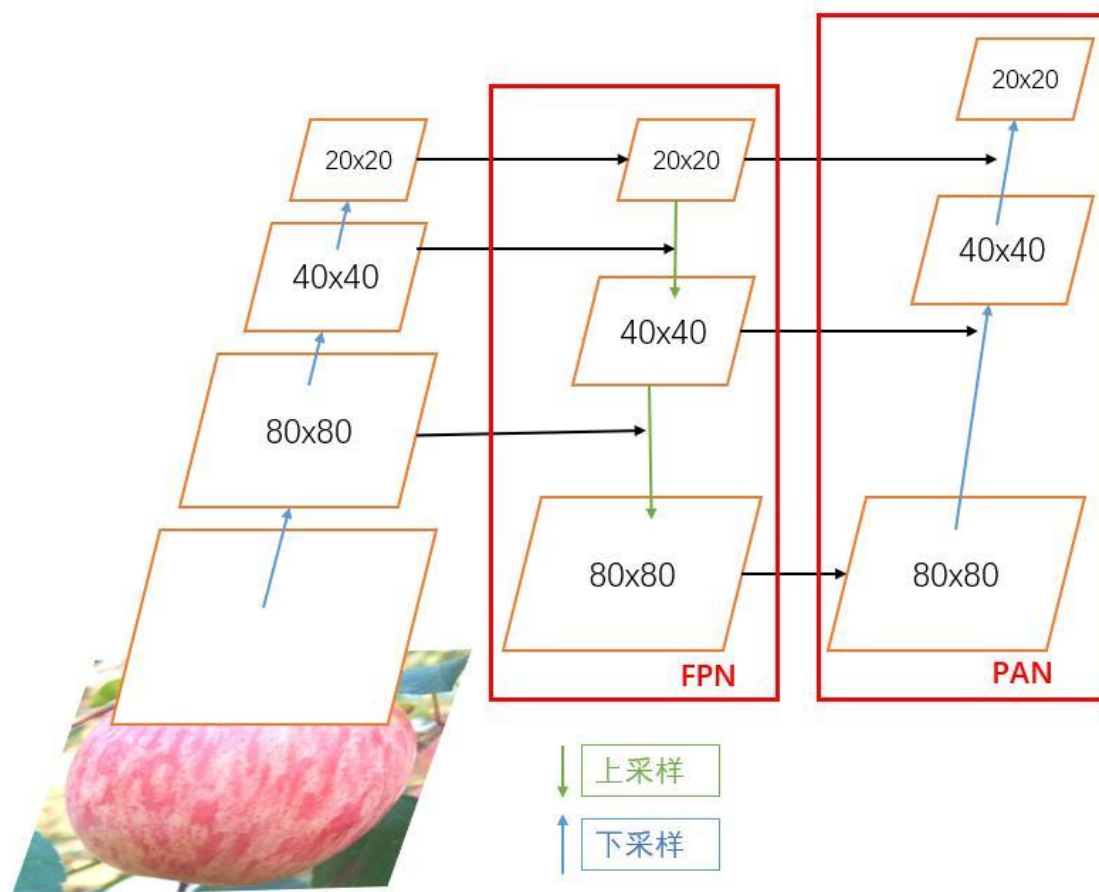
Beetle 使用的就是 YOLO 算法。JosephRedmon 等人在 2016 年的时候提出了由卷积层和 FC 层构成的 YOLO 目标检测算法, YOLO 目前已经发布到 YOLO5。YOLOV5 是一种单阶段(one-stage)目标检测算法, 具有推理速度快、精度高、易于部署的特点, 能够满足实时高精度多目标推理的应用场景。



YOLOV5 网络主要分为 5 个部分:输入端、Backbone、Neck、Head 和输出端。

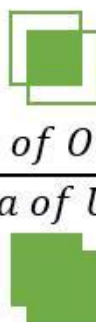
Truth)进行比对， 计算两者差距，再反向更新，迭代网络参数。YOLOV5 在每次训练时，都能自适应的计算不同训练集中的最佳锚框值。

2. Backbone 主干网络主要使用了 Focus 结构和带残差单元的 CSP 结构，负责获取不同尺度的图像特征。Focus 结构最关键的部分是采用了切片(slice)操作把高分辨率的特征图拆分成多个低分辨率的特征图，即隔列采样+拼接。然后对切片的结果进行卷积(CBL)操作。这样能减少下采样时的信息丢失。
3. Neck 部分采用了 FPN+PAN 结构,主要是将 Backbone 主干网络中获取的不同尺度的图像特征信息进行融合然后输入到 Head 部分进行预测。深层的特征图携带有更强的语义特征，较弱的定位信息。而浅层的特征图携带有较强的位置信息，和较弱的语义特征。FPN 就是把深层的语义特征传到浅层，从而增强多个尺度上的语义表达。而 PAN 则相反，把浅层的定位信息传导到深层，增强多个尺度上的定位能力。即 FPN 特征塔自顶向下传达强语义特征，而 PAN 特征塔自底向上传达强定位特征。可以实现从不同的主干层对不同的检测层进行参数聚合。



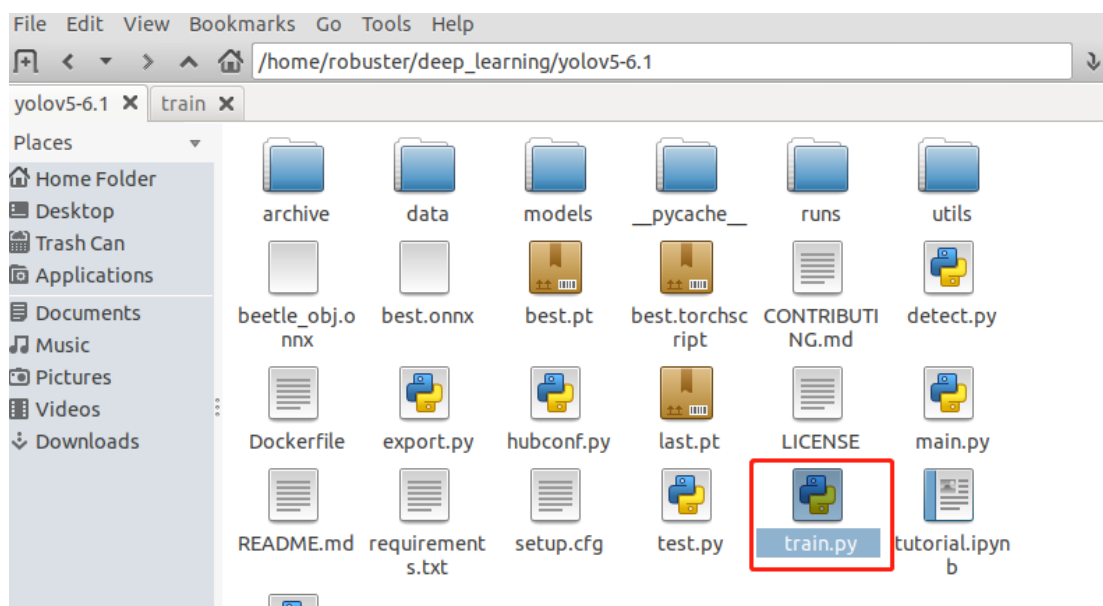
4. Head 部分的检测器则是对融合后的图像特征进行预测,生成大量边界预测框并预测类别。需要对结果使用 NMS 非极大值抑制(Non-max suppression)的后处理方法去过滤预测到的边界框,使一个目标只具有一个预测框。其原理为把置信度(预测这个网格里是否有目标的置信度)最高的那个网格的边界箱作为极大边界箱,计算极大边界箱和其他几个网格的边界箱的 IOU,如果超过一个阈值,例如 0.5,就认为这两个网格实际上预测的是同一个物体,就把其中置信度比较小的删除。最终得到一个高置信度的物体边界预测框,然后输出。




$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

3. 模型训练

训练模型的程序是在/deep_learning/yolov5-6.1 文件夹下的 train.py 文件。



在使用该文件前我们先了解下它的那些参数，以及关键参数如何配置。我们可以点击下面链接进行参看。


train.py参数解析.
txt

更多关于 train.py 函数的解析，可以参考查阅下面链接。

<https://blog.csdn.net/CharmsLUO/article/details/123542598>

1. 打开终端，执行以下指令开始模型训练工作。

```
cd ~/deep_learning/yolov5-6.1
```

```
conda activate yolov5
```

```
python train.py --device cpu --epochs 100
```

参数解释：--device cpu 是选择训练的计算硬件，BR280 中采用的是 cpu 计算。

--epochs 100 是表示训练 100 次，默认是 300。

```
robuster@robuster:~$ cd ~/deep_learning/yolov5-6.1/
robuster@robuster:~/deep_learning/yolov5-6.1$ conda activate yolov5
(yolov5) robuster@robuster:~/deep_learning/yolov5-6.1$
(yolov5) robuster@robuster:~/deep_learning/yolov5-6.1$ python train.py --device cpu --epochs 100
```

在完成第一次训练之后，我们可以看到 P（准确率）为 0，R（召回率：检测图片里是否有物体）为 0，这表示还不能检测到物体。

```
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_
warnings.warn('User provided device type of \'cuda\'', but CUDA is not available
0/99      0G      0.1096  0.02925  0.04999      6      640: 97%|
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_
warnings.warn('User provided device type of \'cuda\'', but CUDA is not available
0/99      0G      0.1099  0.02915  0.04997      3      640: 100%|
      Class      Images      Labels      P      R      mAP@.5  mAP@.5
      all         15         0      0      0      0
Epoch  gpu_mem      box      obj      cls      labels  img_size
0%|
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_
warnings.warn('User provided device_type of \'cuda\'', but CUDA is not available
1/99      0G      0.117  0.02558  0.05312      3      640: 3%|
```

通过 13 次训练之后，P 和 R 的值都有了变化，训练次数越多，他们的值越大，他们的最大值都是 1。

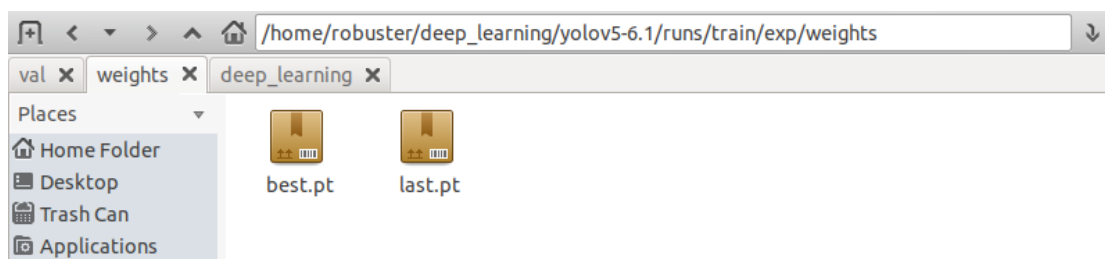

```

/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_mode.py:141: Us
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
12/99 0G 0.0811 0.03321 0.04771 5 640: 90%|
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_mode.py:141: Us
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
12/99 0G 0.08084 0.03328 0.04772 4 640: 93%|
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_mode.py:141: Us
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
12/99 0G 0.08097 0.033 0.04774 3 640: 97%|
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_mode.py:141: Us
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
12/99 0G 0.0811 0.03285 0.04775 3 640: 100%|
Class Images Labels P R mAP@.5 mAP@.5:.95: 100%|
att 15 15 0.000174 0.0667 0.000119 3.57e-05

Epoch gpu mem box obj cls labels img size
0%|12表示这是第13次训练, 99表示总共要训练100次
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/autocast_mode.py:141: Us
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
13/99 0G 0.06087 0.02503 0.04647 2 640: 3%|

```

2. 训练完成之后，会生成一个 `exp` 文件夹，这里存放了一些训练生成的文件，你可以在 `/home/robuster/deep_learning/yolov5-6.1/runs/train` 路径下找到它。在 `weights` 文件夹里面有两个 `.pt` 文件，其中 `best.pt` 表示训练效果最好，`last.pt` 存放的是最后一次的训练结果。



3. 如果在训练还未完成时，你中途退出了训练，我们也可以在上一次训练的基础上继续开始训练。打开 `/home/robuster/deep_learning/yolov5-6.1` 里的 `train.py`。找到 464 行 `--resume` 参数，将 `False` 变成 `True`。然后点击保存。

```

463 parser.add_argument('--rect', action='store_true', help='rectangular training')
464 parser.add_argument('--resume', nargs='?', const=True, default=True, help='resume most recent training')
465 parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
466 parser.add_argument('--noval', action='store_true', help='only validate final epoch')
467 parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
468 parser.add_argument('--evolve', type=int, nargs='?', const=400, help='evolve hyperparameters for x generations')
469 parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')

```

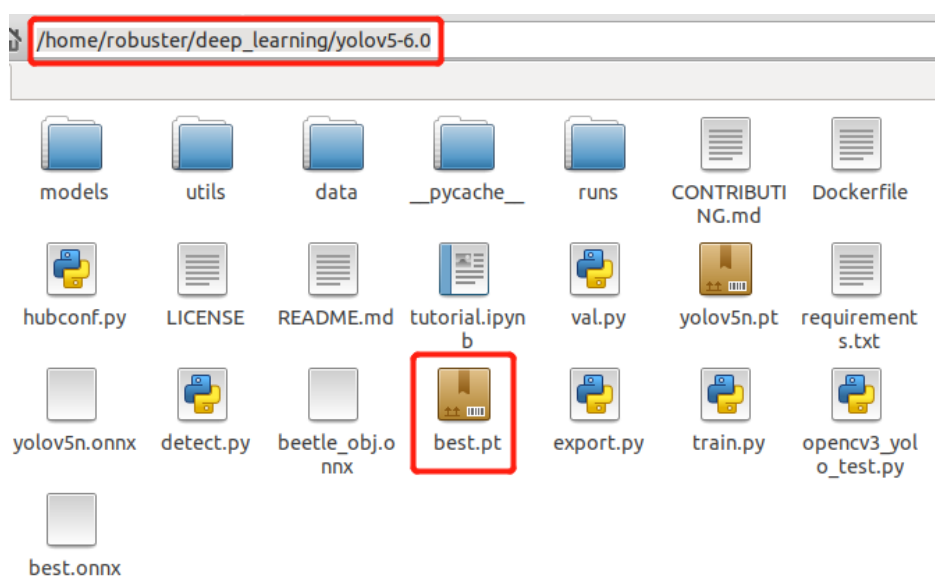
执行 `python train.py --device cpu --epochs 100`，即可恢复训练。训练完成记得将值改回 `False`，不然如果你想训练新的模型，则需要将 `exp` 文件进行删除才行。

```
train: Scanning '/home/robuster/deep_learning/yolov5-6.1/./cube_data/labels/train.cache' images and labels... 48 found
val: Scanning '/home/robuster/deep_learning/yolov5-6.1/./cube_data/labels/val.cache' images and labels... 13 found, 2
Image sizes 128 train, 128 val
Using 0 dataloader workers
Logging results to runs/train/exp
Starting training for 100 epochs...

Epoch 11/99
  gpu_mem  box    obj    cls  labels  img_size
    0G    0.02929 0.005535 0.01585    15    128: 100%| 7/7 [00:06<00:00, 1.01it/s]
  Class    Images  Labels    P    R    mAP@.5  mAP@.5:.95: 100%| 1/1 [00:00<00:
  all      15      13    0.703    0.791    0.804    0.425
```

4. 训练完成将 weights 里面的 best.pt 文件拷贝到

/home/robuster/deep_learning/yolov5-6.1 路径下。



5. 将 best.pt 文件导出为 best.onnx 文件

ONNX 全称 Open Neural Network Exchange（开放神经网络交换），ONNX 格式是一个用于表示深度学习模型的标准，可使模型在不同框架之间进行转移（一般用于中间部署阶段）。ONNX 文件不仅仅存储了神经网络模型的权重，同时也存储了模型的结构信息以及网络中每一层的输入输出和一些其它的辅助信息。

执行以下指令转换文件，前面两行是进入目录与激活 yolov5，如果你发现你的终端命令行前有 (yolov5) 的标志，说已经激活了 yolov5，就不需要输入 conda activate yolov5 命令。

```
cd ~/deep_learning/yolov5-6.1
```

```
conda activate yolov5
```

```
python export.py
```

```
(yolov5) robuster@robuster:~/deep_learning/yolov5-6.1$ python export.py
export: data=data/comp_data.yaml, weights=best.pt, imgsz=[128, 128], batch_size=1, device=cpu, half=False, inplace=False, train=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=10, verbose=False, workspace=4, nms=False, agnostic_nms=False, topk_per_class=100, topk_all=100, iou_thres=0.5, conf_thres=0.05, include=['torchscript', 'onnx']
YOLOv5 2022-2-22 torch 1.10.0+cpu CPU

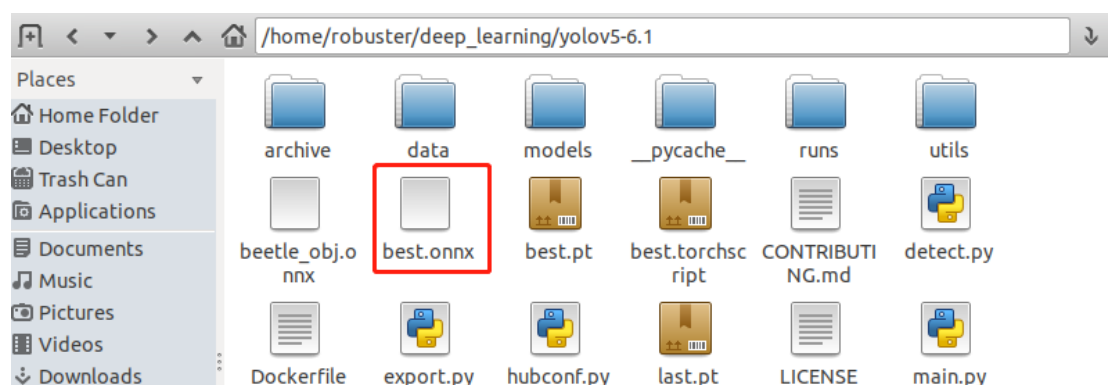
Fusing layers...
[W NNPack.cpp:79] Could not initialize NNPack! Reason: Unsupported hardware.
Model Summary: 213 layers, 1765930 parameters, 0 gradients, 4.2 GFLOPs

PyTorch: starting from best.pt with output shape (1, 1008, 10) (3.7 MB)

TorchScript: starting export with torch 1.10.0+cpu...
TorchScript: export success, saved as best.torchscript (7.3 MB)

ONNX: starting export with onnx 1.10.2...
/home/robuster/miniconda3/envs/yolov5/lib/python3.6/site-packages/torch/onnx/symbolic_helper.py:382: UserWarning: You are trying to export the model with onnx:Resize for ONNX opset version 10. This operator might cause results to not match the expected results by PyTorch.
ONNX's Upsample/Resize operator did not match PyTorch's Interpolation until opset 11. Attributes to determine how to transform the input were added in onnx:Resize in opset 11 to support PyTorch's behavior (like coordinate_transformation_mode and nearest_mode).
We recommend using opset 11 and above for models using this operator.
"" + str(_export_onnx_opset_version) + ". "
ONNX: export success, saved as best.onnx (7.1 MB)

Export complete (12.87s)
```



4. 目标检测

执行以下指令开始进行目标检测测试：

```
cd ~/deep_learning/yolov5-6.0
```

```
conda activate yolov5
```

```
python detect.py --weights best.pt --source 2
```

参数解释，--source 2 是选择相机端口，一般 2 是机械臂上的 1080p 相机，0 是深度相机。

根据训练好的模型，单目相机可以检测出物体



5. 总结

在应用 yolov5 的时候，了解其各个部分的参数解析函数是一个比较快捷上手的方式，也是以后将 yolov5 应用到其他设备或项目中一个关键的步骤，如果你想将 yolov5 应用到其他设备上可以查看相对应的官方教程。

<https://github.com/ultralytics/yolov5>

或者你想安装到 windows 设备上，可以参阅下面的链接。

https://blog.csdn.net/qq_45701791/article/details/113992622

在学习完数据集采集与标注、目标检测与模型训练后，我们可以尝试训练一个模型来识

别生活中常见的办公设备。



— 深圳史河机器人科技有限公司 —



深圳史河机器人科技有限公司

www.robotplusplus.com.cn

深圳市龙华区龙华街道清华社区建设东路青年创业园C栋4层412-415房