

# 无人机上仅使用CPU实时运行Yolov5? (OpenVINO帮你实现)

Intel CPU在运行视觉导航等算法时实时性要优于Nvidia等平台，如Jetson Tx2, NX。而Nvidia平台在运行深度学习算法方面具有很大优势，两种平台各有利弊。但是，Intel OpenVINO的推出允许NUC平台实时运行深度学习模型，如目前最流行的目标检测程序Yolov5，这样就太好了，仅使用Intel无人机平台就可以完成各种任务。本教程将教你用Prometheus在Intel无人机平台部署Yolov5目标检测。

先来个速度测试，仅使用Intel CPU，没有模型压缩与剪枝等算法，也不依赖其他任何加速硬件。

Intel CPU	Yolov5模型	输入分辨率	检测帧率
Intel i7-10710U (10代NUC)	yolov5s	256	44.5Hz
Intel i7-10710U (10代NUC)	yolov5s	320	27.7Hz
Intel i7-10710U (10代NUC)	yolov5s	384	19.8Hz
Intel i7-10710U (10代NUC)	yolov5s	512	10.1Hz
Intel i7-6700K (台式机)	yolov5s	256	43.2Hz
Intel i7-6700K (台式机)	yolov5s	320	27.1Hz
Intel i7-6700K (台式机)	yolov5s	384	19.2Hz
Intel i7-6700K (台式机)	yolov5s	512	10.0Hz

## 一、安装OpenVINO

官网教程：

[https://docs.openvinotoolkit.org/latest/openvino\\_docs\\_install\\_guides\\_installing\\_openvino\\_linux.html](https://docs.openvinotoolkit.org/latest/openvino_docs_install_guides_installing_openvino_linux.html)

### 1. 下载安装包

地址：

[https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/download.html?operatingsystem=linux&distributions=webdownload&version=2021%203%20\(latest\)&options=offline](https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/download.html?operatingsystem=linux&distributions=webdownload&version=2021%203%20(latest)&options=offline)

Select options below to download

Operating System:

Select operating system  
Linux

Distribution:

Select distribution  
Web & Local (recommended)

Version Type:

Select version  
2021.3 (latest)

Installer Type:

Select installer  
Local

1.

2.

### Offline Installer

- Includes all tools in the toolkit
- Recommended for host machines with poor or no internet connection

Initial download: 257 MB

Maximum download: 521 MB (based on operating system)

### Select Your Release Type

#### Standard Release

Recommended for new users and users that are currently prototyping. Offers new features, tools, and support to stay current with deep learning technology advancements.

#### Long-Term Support (LTS) Release

Recommended for experienced users that are ready to take their application into production and who do not require new features and capabilities for their application.

Download

## 2. 如果之前安装过openvino，重命名或删除以下文件：

```
1 ~/inference_engine_samples_build
2 ~/openvino_models
```

## 3. 打开终端，解压安装包并进入解压路径

```
1 cd ~/Downloads/
2 tar -xvzf l_openvino_toolkit_p_<version>.tgz
3 cd l_openvino_toolkit_p_<version>
```

## 4. 安装OpenVINO

```
1 sudo ./install_GUI.sh
```

## 5. 安装软件依赖

```
1 cd /opt/intel/openvino_2021/install_dependencies
2 sudo -E ./install_openvino_dependencies.sh
```

## 6. 配置模型优化器

```
1 cd
/opt/intel/openvino_2021/deployment_tools/model_optimizer/install_prerequisites
2 sudo ./install_prerequisites.sh
```

如果下载很慢，可以修改`~/.pip/pip.conf`，转到国内源

```
1 [global]
2 index-url = http://mirrors.aliyun.com/pypi/simple/
3 [install]
4 trusted-host = mirrors.aliyun.com
```

## 二、配置Yolov5，运行演示程序

## 1. 下载Prometheus子模块Yolov5，并配置

```
1 sudo apt install python3-pip
2 cd <path-to-Prometheus>/
3 ./Scripts/install_detection_yolov5openvino.sh
```

## 2. 下载模型权重或训练自己的模型

以官方权重为例

```
1 cd <path-to-Prometheus>/Modules/object_detection_yolov5openvino/weights
2 wget https://github.com/ultraalytics/yolov5/releases/download/v3.0/yolov5s.pt
```

## 3. 将.pt权重文件转换为.onnx文件

运行命令：

```
1 cd <path-to-Prometheus>/Modules/object_detection_yolov5openvino
2 python3 models/export.py --weights weights/yolov5s.pt --img 384 --batch 1
```

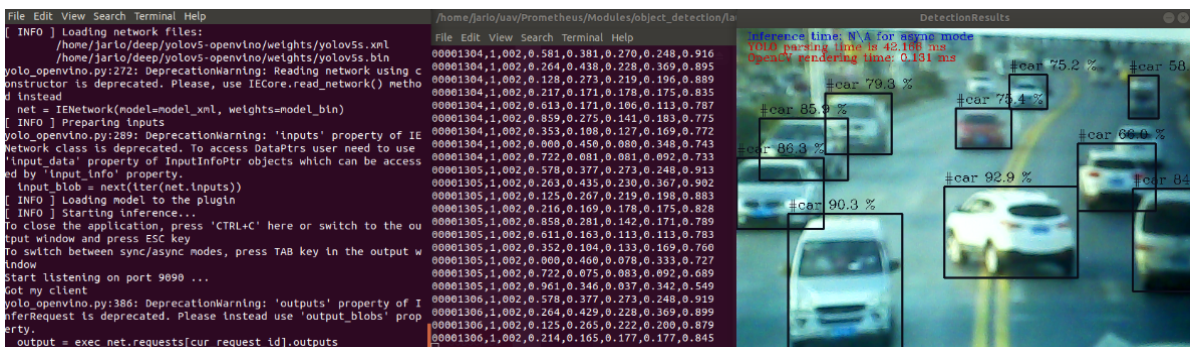
## 4. 将.onnx文件转换为IR文件

```
1 cd <path-to-Prometheus>/Modules/object_detection_yolov5openvino
2 python3 /opt/intel/opencv_2021/deployment_tools/model_optimizer/mo.py --
  input_model weights/yolov5s.onnx --model_name weights/yolov5s -s 255 --
  reverse_input_channels --output Conv_487,Conv_471,Conv_455
```

## 5. 运行演示程序

```
1 cd <path-to-Prometheus>/
2 ./Scripts/start_yolov5openvino_server.sh
3 # Ctrl+t 打开一个新的命令行页面，并运行：
4 roslaunch prometheus_detection yolov5_intel_openvino.launch
```

运行结果如下：



## 6. 输入ROS-Launch参数

以<path-to-Prometheus>/Modules/object\_detection/launch/yolov5\_intel\_openvino.launch为例：

```

1 <launch>
2   <node pkg="prometheus_detection" type="yolov5_openvino_client.py"
   name="yolov5_openvino_client" output="screen">
3     <param name="output_topic"
   value="/prometheus/object_detection/yolov5_openvino_det"/>
4     <param name="camera_parameters" type="string" value="$(find
   prometheus_detection)/shell/calib_webcam_640x480.yaml" />
5     <param name="object_names_txt" value="coco"/>
6   </node>
7 </launch>

```

其中：

- **output\_topic**: 检测结果输出话题（消息类型：`MultiDetectionInfo.msg`）
- **camera\_parameters**: 相机参数文件（为了估计视线角误差、目标位置）
- **object\_names\_txt**: 目标类别描述 `txt` 问题（具体见：`<path-to-Prometheus>/Modules/object_detection/py_nodes/yolov5_openvion_client/class_desc/coco.txt`）

## 7. 输出ROS话题解析

默认输出话题，消息类型：

```

1  ##`MultiDetectionInfo.msg`
2  Header header
3  ## 检测到的目标数量
4  int32 num_objs
5  ## 每个目标的检测结果
6  DetectionInfo[] detection_infos

```

```

1  ##`DetectionInfo.msg`
2  ## 是否检测到目标
3  bool detected
4  ## 目标类别名称
5  string object_name
6  ## 类别ID
7  int32 category
8  ## 0表示相机坐标系，1表示机体坐标系，2表示惯性坐标系
9  int32 frame
10 ## 目标位置[相机系下：右方x为正，下方y为正，前方z为正]
11 float32[3] position
12 ## 目标姿态-欧拉角-(z,y,x)
13 float32[3] attitude
14 ## 目标姿态-四元数-(qx,qy,qz,qw)
15 float32[4] attitude_q
16 ## 视线角度[相机系下：右方x角度为正，下方y角度为正]
17 float32[2] sight_angle
18 ## 偏航角误差
19 float32 yaw_error

```

注意：默认情况下**每个目标**仅有 `detected`，`object_name`，`category`，`frame`，`sight_angle` 的输出。

如果想输出 `position`，需要在类别描述文件（如 `<path-to-Prometheus>/Modules/object_detection/py_nodes/yolov5_openvion_client/class_desc/coco.txt`）中填写目标的高度与宽度（单位：`m`）。

```
Open ▾  coco.txt
~/uav/Prometheus/Modules/object_detection/py_nodes/yolov5_openvino_client/class_desc
person,0.5,1.8
bicycle
car
motorcycle
airplane
bus
train
truck
boat
traffic_light
fire_hydrant
stop_sign
parking_meter
bench
```

如上图，以人（person）为例，设置宽度 `0.5m`，高度 `1.8m`

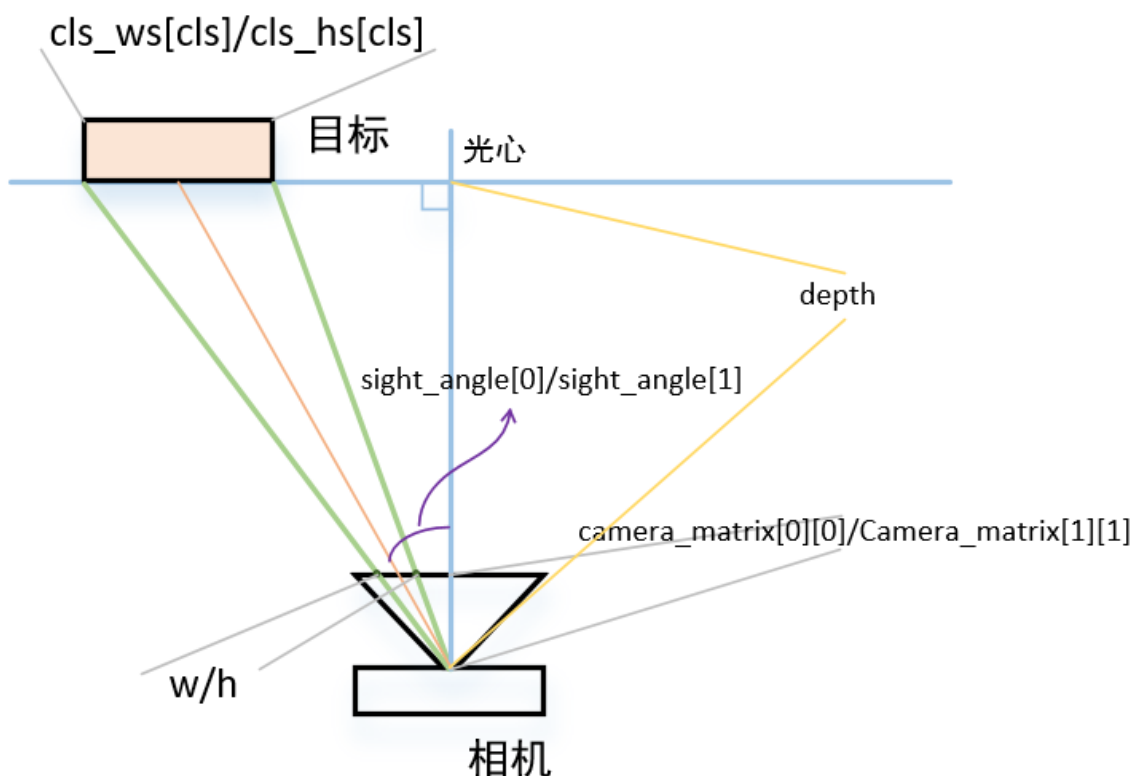
然后，需要修改源代码，这里以高度估计人距离摄像机的距离，并以此估计 `position`。

源代码位置：`<path-to-Prometheus>/Modules/object_detection/py_nodes/yolov5_openvion_client/yolov5_openvion_client.py`，（112-114 行）

```
111
112         if cls == 0 and cls_hs[cls] > 0:
113             depth = (cls_hs[cls]*camera_matrix[1][1]) / (h*image_height)
114             d_info.position = [math.tan(d_info.sight_angle[0])*depth, math.tan(d_info.sight_angle[1])*depth, depth]
115
```

`cls==0` 用来判断是否为类别—person，`cls_hs[cls]` 用来读取目标高度（读到的数就是我们写在 `coco.txt` 里的 `1.8m`），`camera_matrix[1][1]` 为垂直方向像素焦距（由相机标定参数文件决定），`h` 物体的像素高度（为实时检测结果，并被归一化到0-1）

## 附：目标位置测量原理



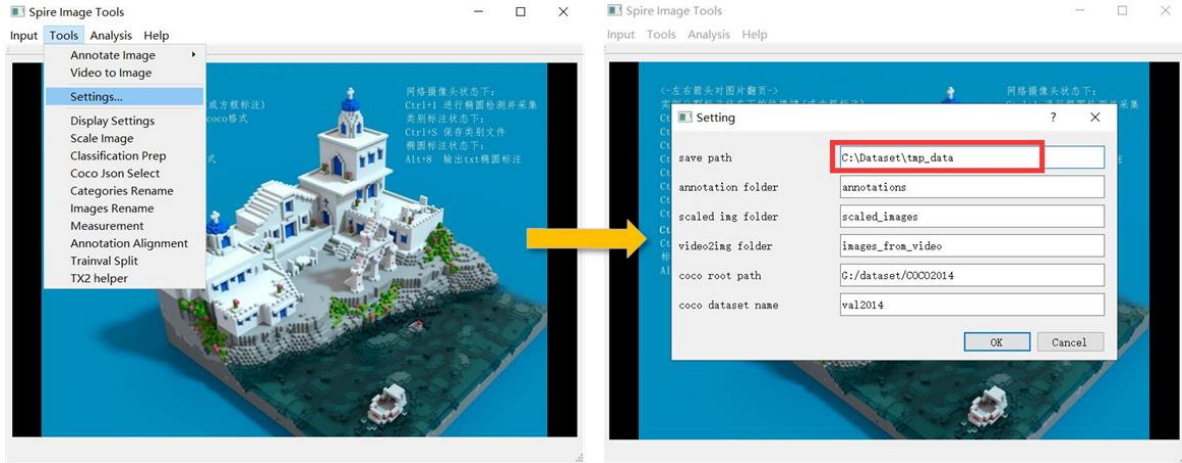
## 三、训练自己的yolov5模型并部署

# 1. 数据标注

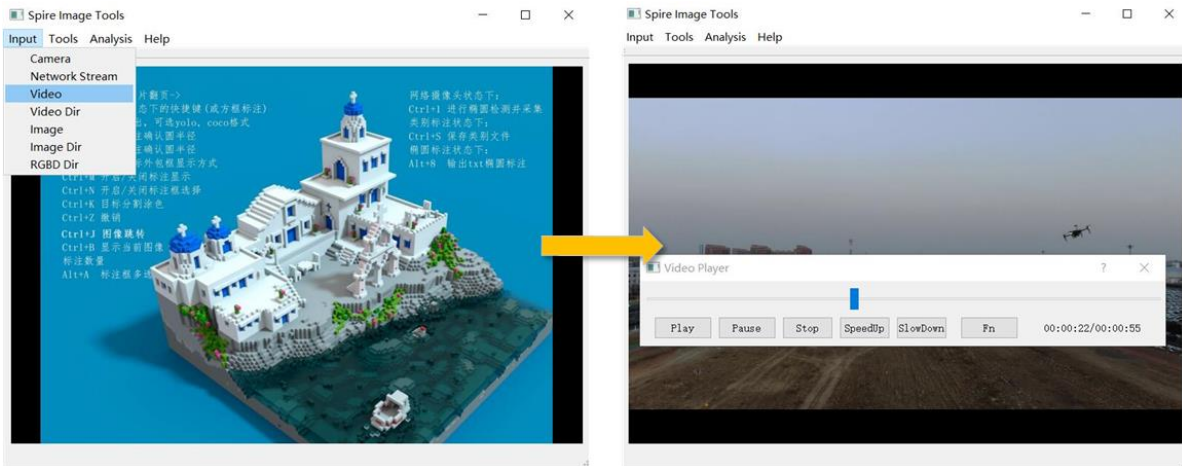
下载数据集标注工具，下载地址：[Spire Web](#)或者[百度网盘](#) (密码: l9e7)，数据集管理软件 SpireImageTools: [gitee地址](#)或者[github地址](#)。

- 解压，打开标注软件 SpireImageTools\_x.x.x.exe

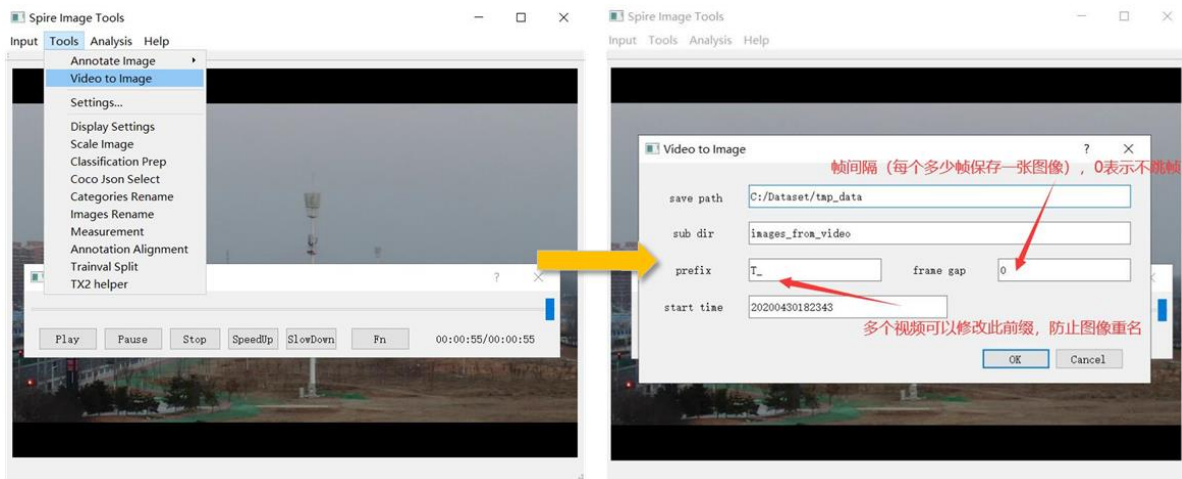
首先点击 Tools->Setting...，填写一个 save path (所有的标注文件都会存储在这个文件夹中)



- 将拍摄的视频转为图像 (如果采集的是图像，则跳过这一步骤)，点击 Input->Video，选择要标注的视频。

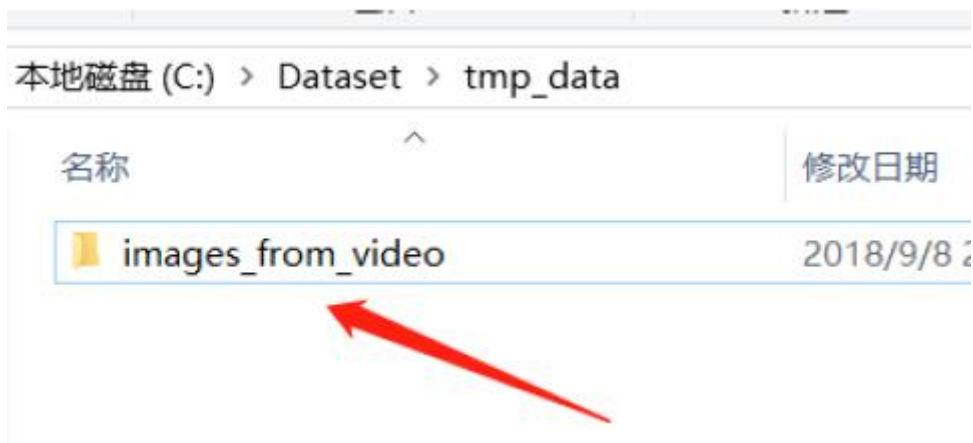


然后，点击 Tools->Video to Image

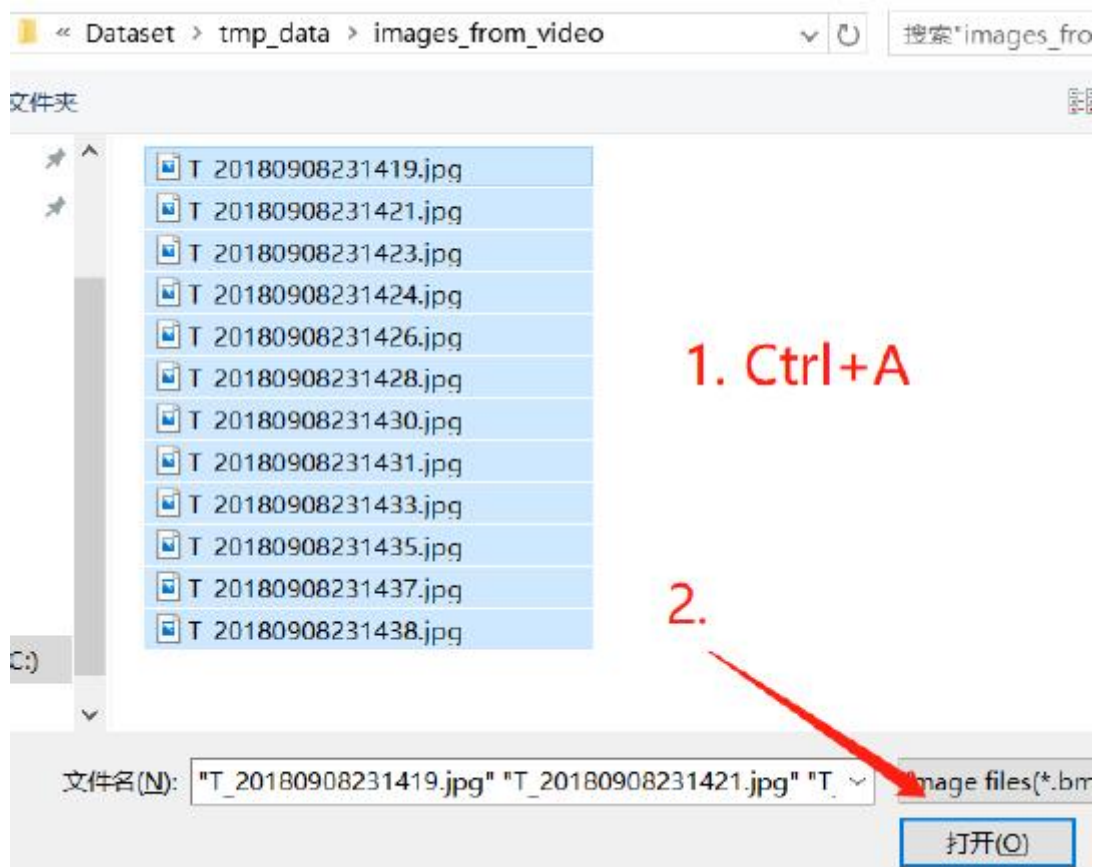


点击OK 后，等待完成，结果会存储在：

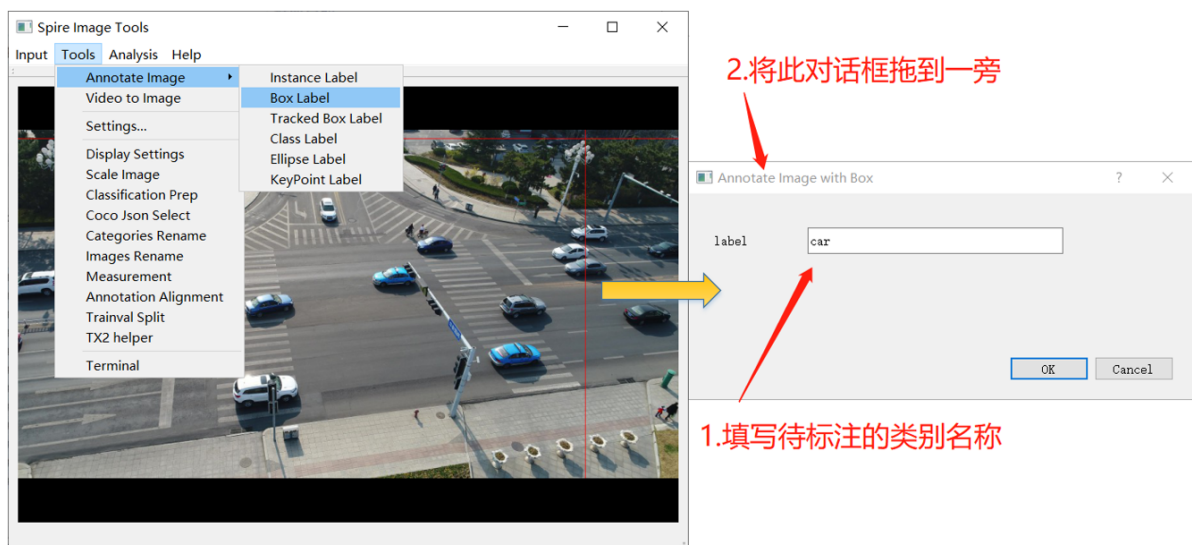




- 打开需要标注的图像，点击菜单 Input->Image Dir，找到需要标注的图像所在文件夹，按 Ctrl+A，全选，打开所有图像：

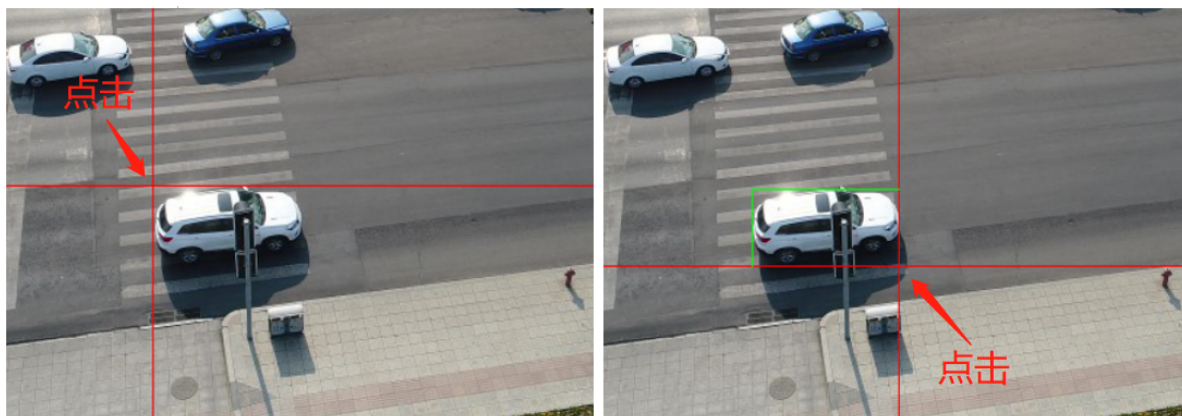


- 点击菜单：Tools->Annotate Image->Box Label，开始标注图像



在 label 中填写待标注目标名称，然后将对话框拖到一边。

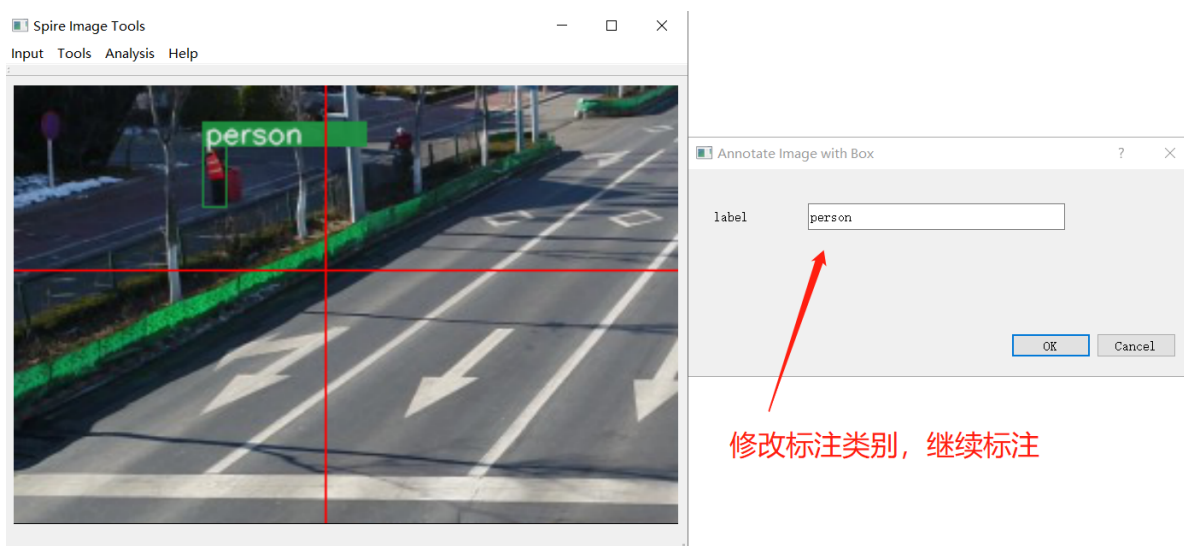
- **开始标注**，在主窗口中开始标注，鼠标滚轮放大缩小图像，按住左键移动可视图像区域不断点击左键将目标框包围，使用 Yolo 训练时，**点击2个点即可**：



**标注时**，如果点错，按鼠标右键可以取消。**标注完成后**，如果不满意，可以点击绿色边框(边框会变红，如下图所示)，按 Delete 删除

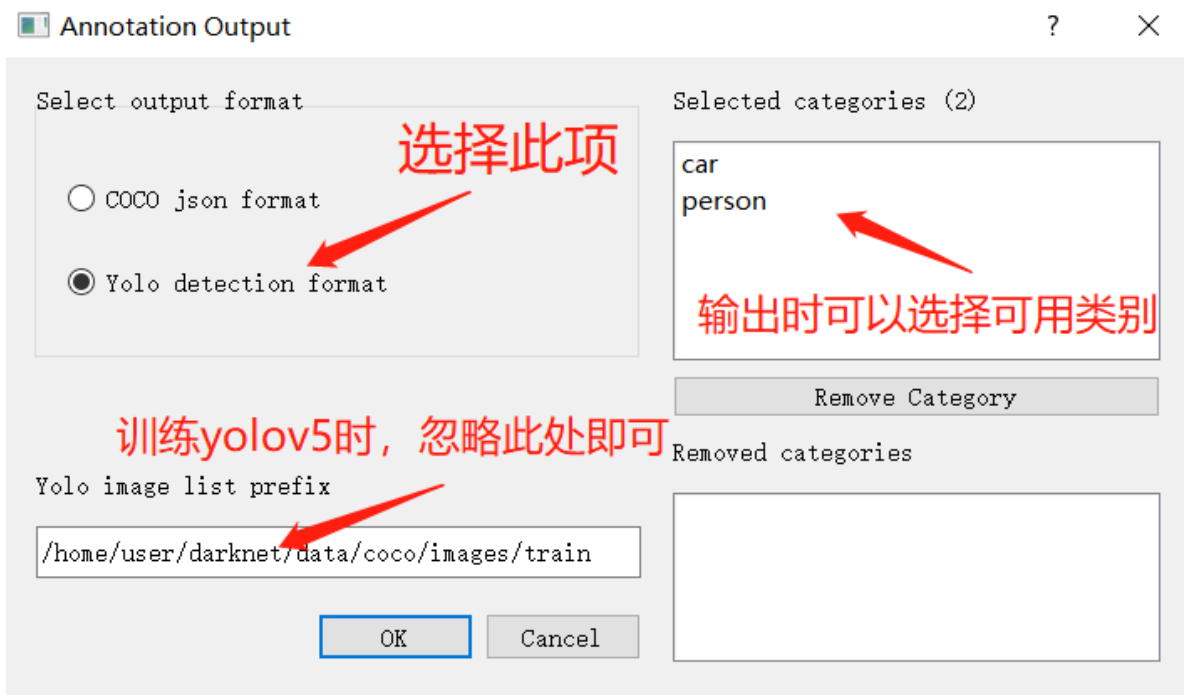


- **继续标注行人类别**：



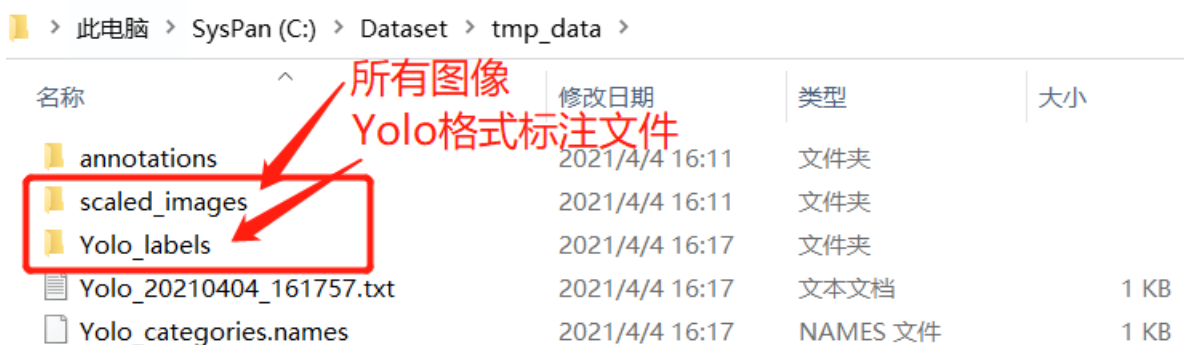
- **全部标注完成后**，将标注输出为 Yolo 格式，准备训练——在标注完成之后，按下 Ctrl+o





点击 OK 即可，需要等待转换。

- 注意，如下两个文件夹是我们训练 Yolov5 需要的



## 2. 开始训练Yolov5

在准备好 `scaled_images` 和 `Yolo_labels` 两个文件夹之后，我们就可以训练 Yolov5 了。首先，创建一个 `car_person.yaml`，将其放到 `<path-to-Prometheus>/Modules/object_detection_yolov5openvino/data/` 文件夹下。 `car_person.yaml` 的具体内容如下：

```
1 # train and val data as 1) directory: path/images/, 2) file: path/images.txt,
  or 3) list: [path1/images/, path2/images/]
2 train: data/car_person/images/train/
3 val: data/car_person/images/train/
4
5 # number of classes
6 nc: 2
7
8 # class names
9 names: ['car', 'person']
```

**注意1：** `car_person` 是自定义名称，我们这次标注的数据集仅有这2个类别。

**注意2：** `names: ['car', 'person']` 这里的类别顺序需要跟 `Yolo_categories.names` 里的类别顺序一致。

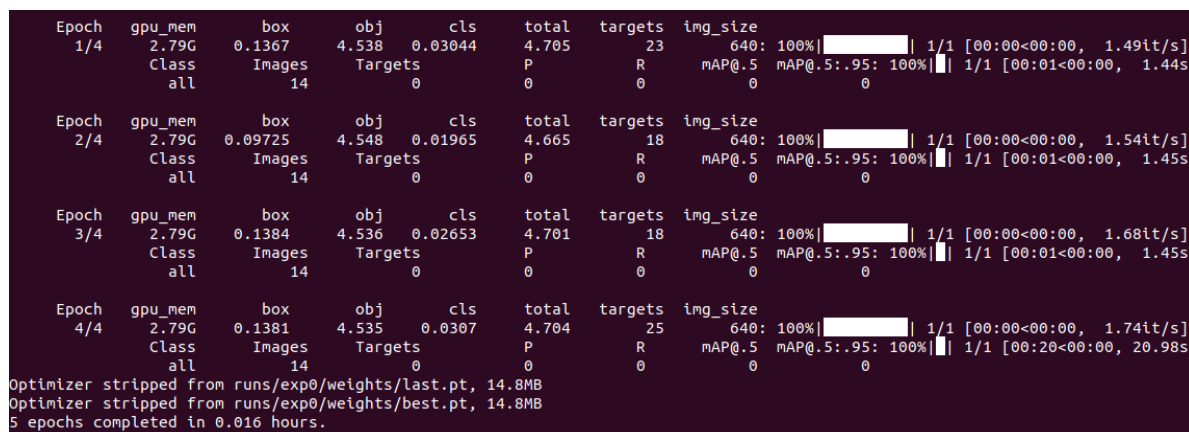
- 将训练图像与标注拷贝到对应位置

首先，在 `<path-to-Prometheus>/Modules/object_detection_yolov5openvino/data/` 下新建一个文件夹 `car_person`。然后，在 `car_person` 下再新建2个文件夹 `images` 和 `labels`。最后，将准备好的 `scaled_images` 拷贝到 `images` 下，并重命名为 `train`；将准备好的 `yolo_labels` 拷贝到 `labels` 下，并重命名为 `train`。

结合 `car_person.yaml` 里的内容，我想你应该明白上面目录结构的含义啦。

- 开始训练

```
1 cd <path-to-Prometheus>/Modules/object_detection_yolov5openvino/  
2 python3 train.py --img 640 --batch 16 --epochs 5 --data data/car_person.yaml  
   --weights weights/yolov5s.pt
```



Epoch	gpu_mem	box	obj	cls	total	targets	img_size
1/4	2.79G	0.1367	4.538	0.03044	4.705	23	640: 100% ██████████  1/1 [00:00<00:00, 1.49it/s]
Class		Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% ██████████  1/1 [00:01<00:00, 1.44s]
all		14	0		0	0	0
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
2/4	2.79G	0.09725	4.548	0.01965	4.665	18	640: 100% ██████████  1/1 [00:00<00:00, 1.54it/s]
Class		Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% ██████████  1/1 [00:01<00:00, 1.45s]
all		14	0		0	0	0
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
3/4	2.79G	0.1384	4.536	0.02653	4.701	18	640: 100% ██████████  1/1 [00:00<00:00, 1.68it/s]
Class		Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% ██████████  1/1 [00:01<00:00, 1.45s]
all		14	0		0	0	0
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
4/4	2.79G	0.1381	4.535	0.0307	4.704	25	640: 100% ██████████  1/1 [00:00<00:00, 1.74it/s]
Class		Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% ██████████  1/1 [00:20<00:00, 20.98s]
all		14	0		0	0	0

Optimizer stripped from runs/exp0/weights/last.pt, 14.8MB  
Optimizer stripped from runs/exp0/weights/best.pt, 14.8MB  
5 epochs completed in 0.016 hours.

显示以上内容说明训练成功！可以增加训练期数（`--epochs 5`）提升效果。

- 部署训练好的模型

刚刚训练好的模型会保存在 `<path-to-Prometheus>/Modules/object_detection_yolov5openvino/runs/exp?/weights/best.pt`，?需根据自己的情况而定（最新训练的模型?为最大的数字），将 `best.pt` 重命名为 `yolov5s.pt`，拷贝到 `<path-to-Prometheus>/Modules/object_detection_yolov5openvino/weights/` 下，然后执行第二部分 3-5 的操作进行 OpenVINO 部署。