

第十八届智能车竞赛智能视觉 组目标检测教程——逐飞科技



目录

目录	1
1. 软件安装	3
1.1. Python 安装	3
1.2. OpenMV IDE 安装	5
1.3. 移动文件夹	5
2. 标记图片	7
2.1. 打开工具	7
2.2. 创建工程	8
2.3. 导入图片	10
2.4. 标记图片	11
2.5. 保存标记图片	12
3. 训练模型	15
3.1. 安装 python 依赖库	15
3.2. 使用测试已标记图片进行训练	16
3.2.1. 将测试图片放入文件夹	16
3.2.2. 将标记图片生成训练集和测试集	16
3.2.3. 使用聚类算法整合训练集	18
3.2.4. 开始训练	19
3.2.5. 验证模型	19
3.3. 使用自己标记图片进行训练	21
3.3.1. 将标记图片生成训练集和测试集	21

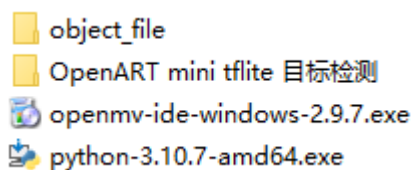
3.3.2. 使用聚类算法整合训练集.....	22
3.3.3. 开始训练.....	23
3.3.4. 验证模型.....	24
4. 模型使用.....	25
5. 文档版本.....	29

1. 软件安装

1.1. Python 安装

1. 解压“03 所需文件.7z”。

可以看到在文件中有四个内容：



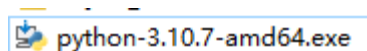
object_file: 包含目标检测所需要的文件。

OpenART mini tflite 目标检测: 使用 OpenART mini 测试目标检测模型。

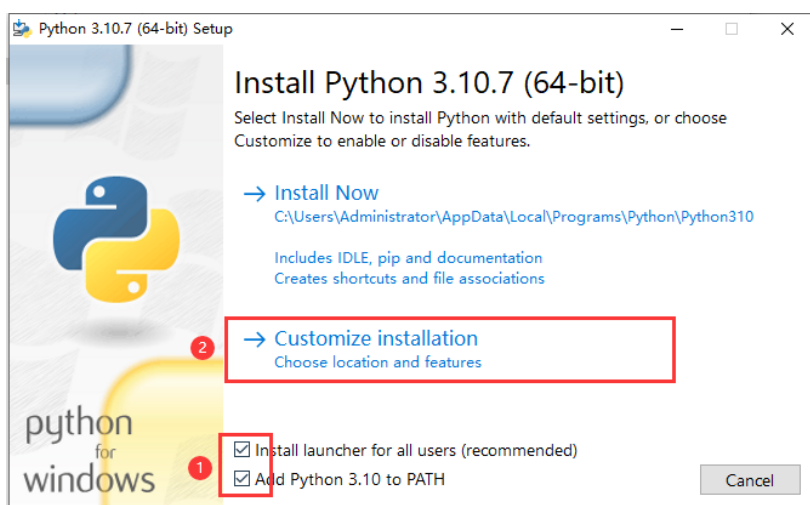
openmv-ide-windows-2.9.7.exe 为 OpenMV IDE 安装包。

python-3.10.7-amd64.exe 为 python 安装包（所安装的 python 版本必须为 3.10.x）。

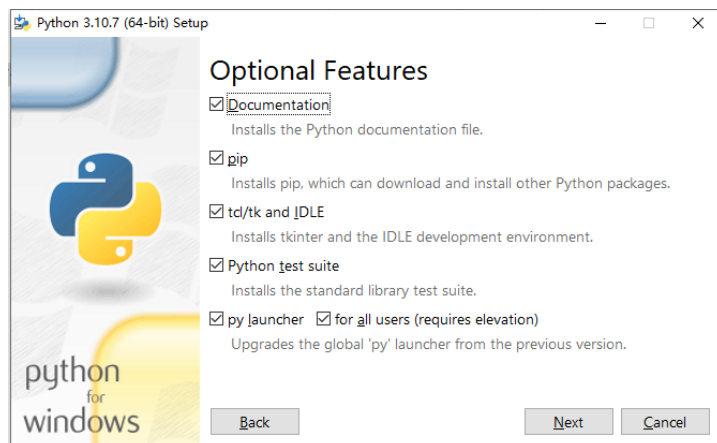
2. 双击安装包进行 python 安装。



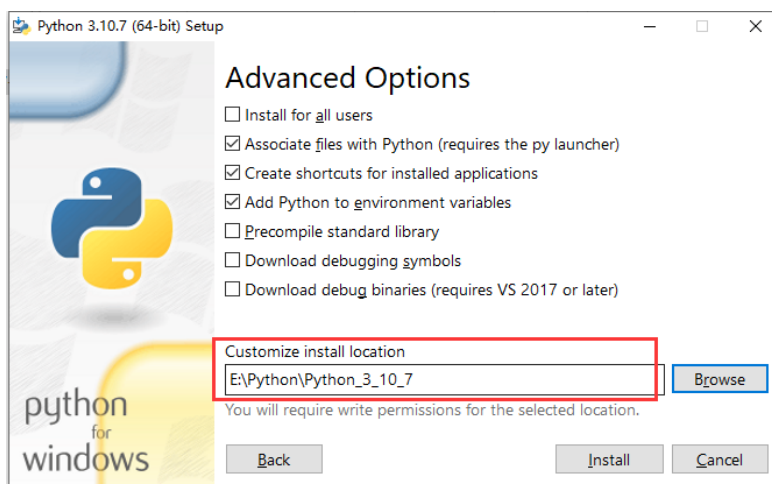
3. 勾选最下方两个框，然后点击自定义安装。



4. Next。

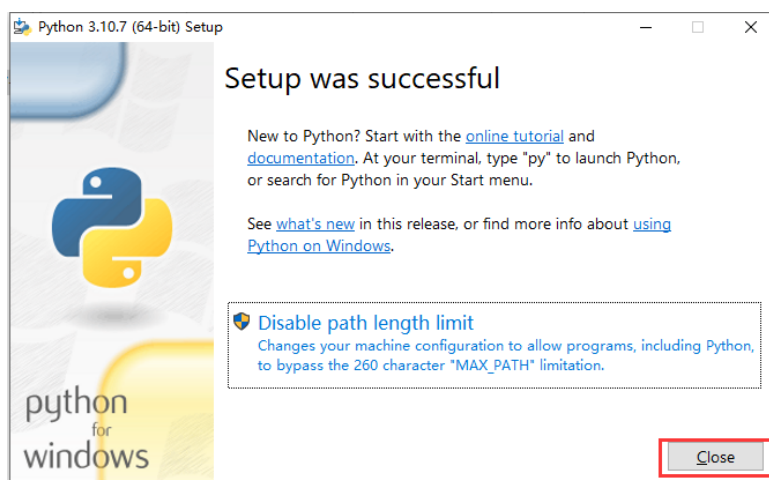


5. 修改路径，点击 Install。

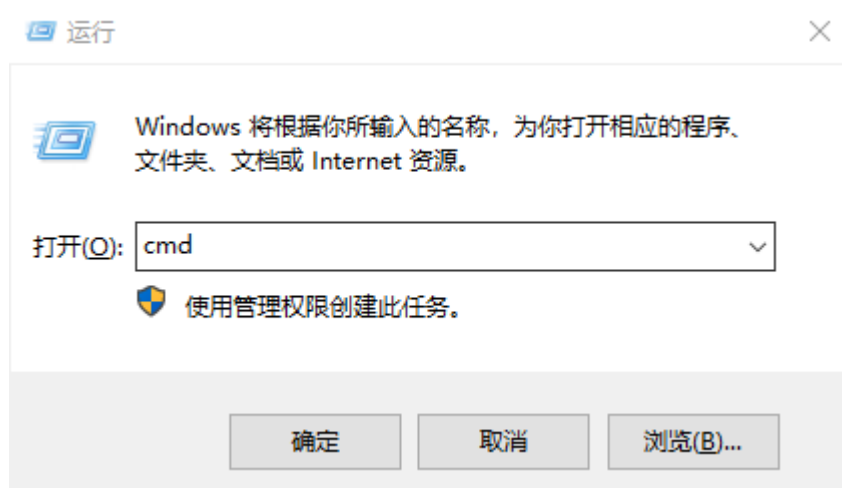


6. 等待安装。

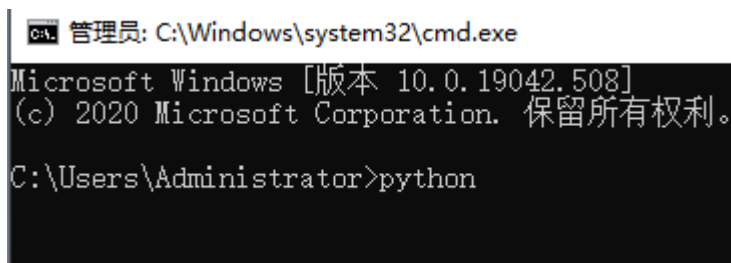
7. 安装完成后点击 Close。



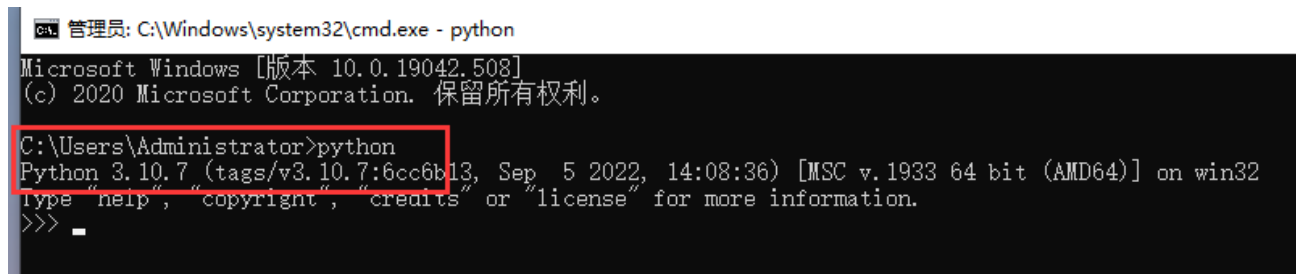
8. 判断是否安装完成，win+r 打开运行窗口，输入 cmd 回车。




9. 输入 python, 回车。



10. 如果看到显示 Python 3.10.7 则安装完成。

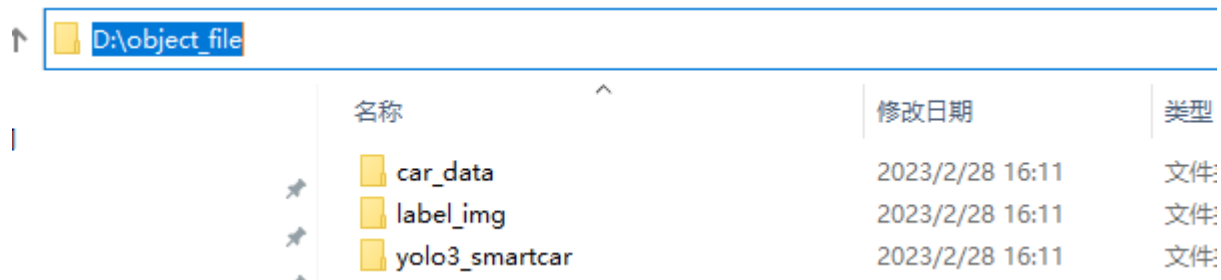


1.2. OpenMV IDE 安装

双击  openmv-ide-windows-2.9.7.exe 开始安装。请务必使用资料包中提供的安装包进行安装, 如果已经安装过 Openmv IDE 可以卸载后重新安装。

1.3. 移动文件夹

将上一章压缩包中的 object_file 文件夹整体移动到一个全英文路径下, 我这里移动到 D 盘的根目录下。



在文件夹中包含三个文件夹：

car_data 为已标记好的数据集（用于测试，实际情况还需要自行拍摄和标记）。

label_img 为标记图片的工具。

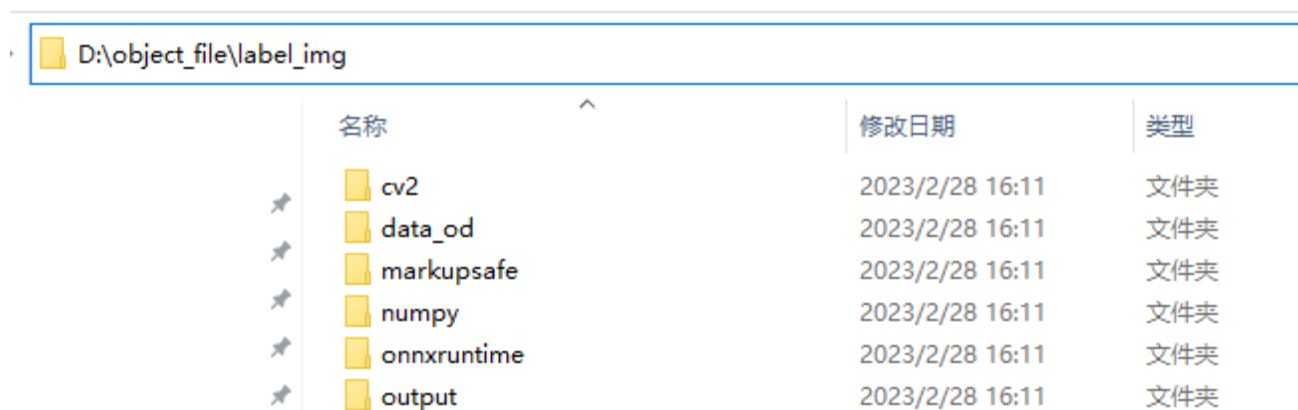
yolo3_smartcar 为模型训练和导出的脚本。

2. 标记图片

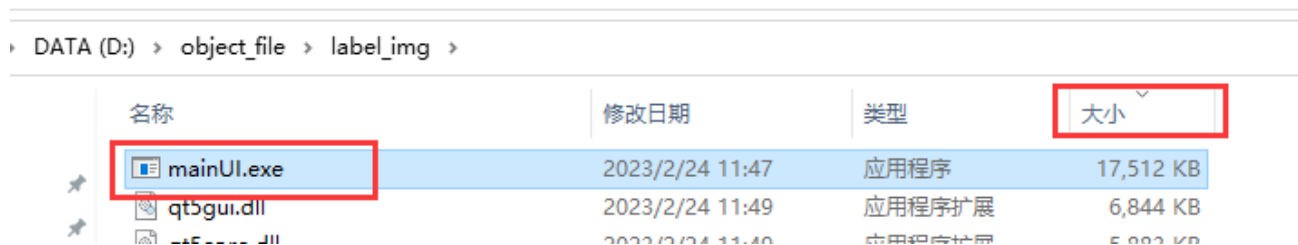
注意：需要完成 1.3 的[移动文件夹](#)步骤！

2.1. 打开工具

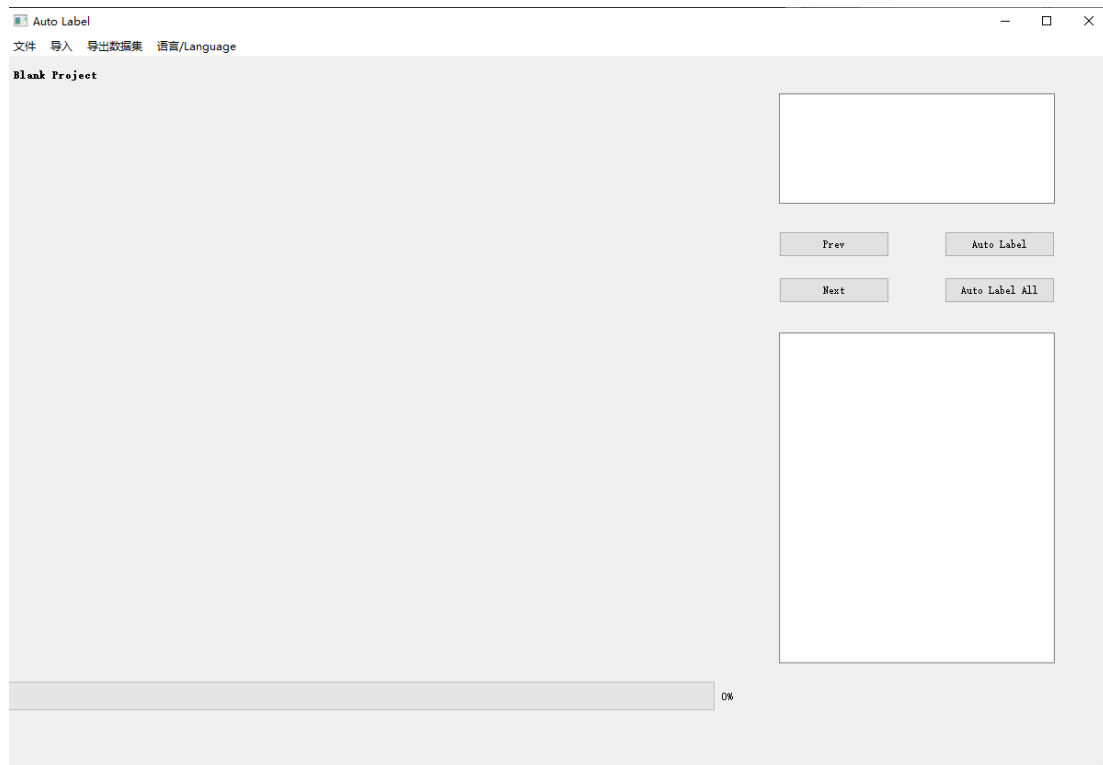
1. 进入 object_file 中的 label_img 文件夹。



2. 按照大小排序，找到 mainUI.exe 应用，双击运行。



可以看到软件主界面。

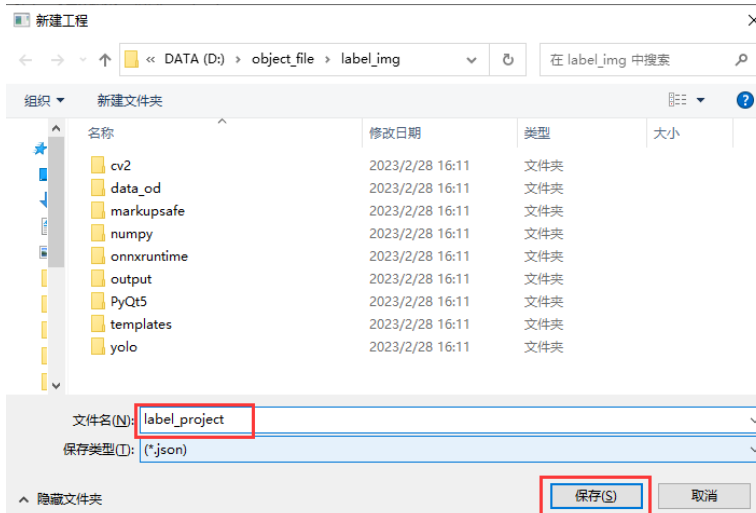


2.2. 创建工程

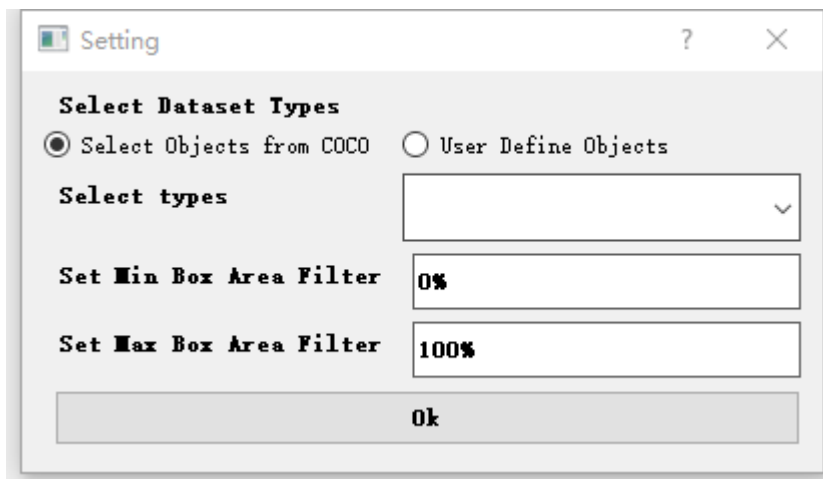
1. 在软件中点击左上角文件，选择新建工程。



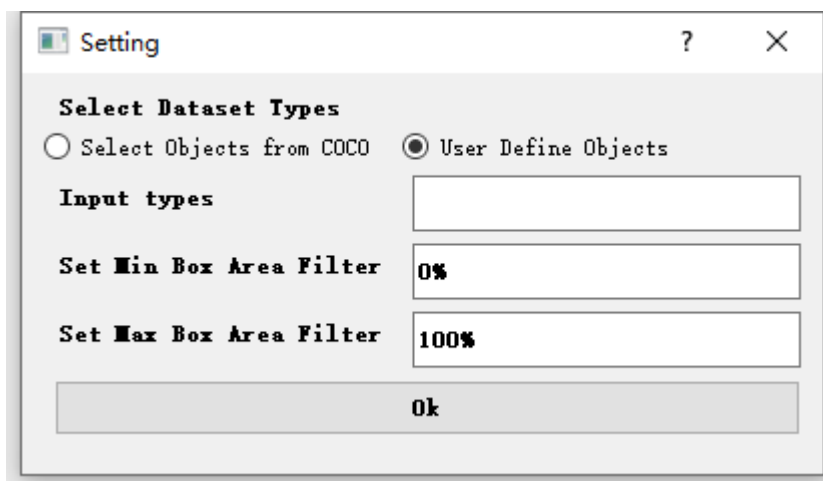
2. 输入工程名，然后保存。



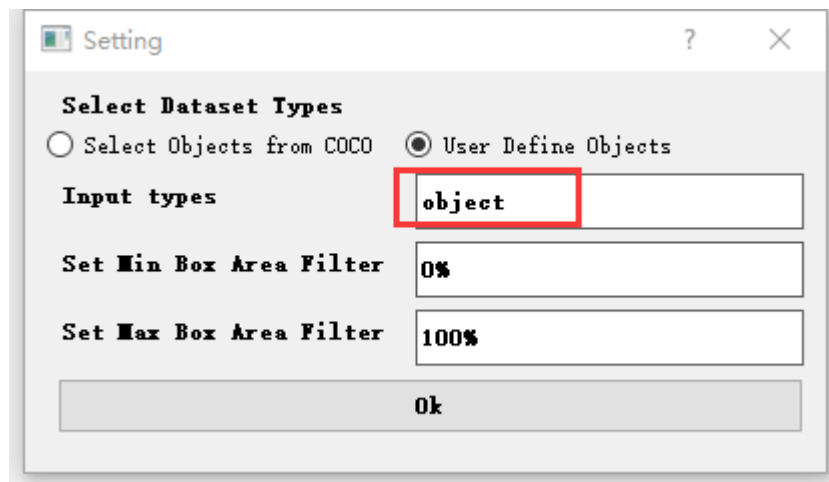
3. 保存后，会弹出一个设置窗口。



4. 这里选择右侧选项，User Define Objects。

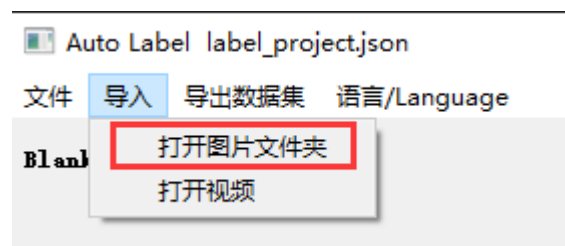


5. 输入目标类名，这里只能输入 object，否则后面训练会报错!!! 其他默认，然后点击 Ok。

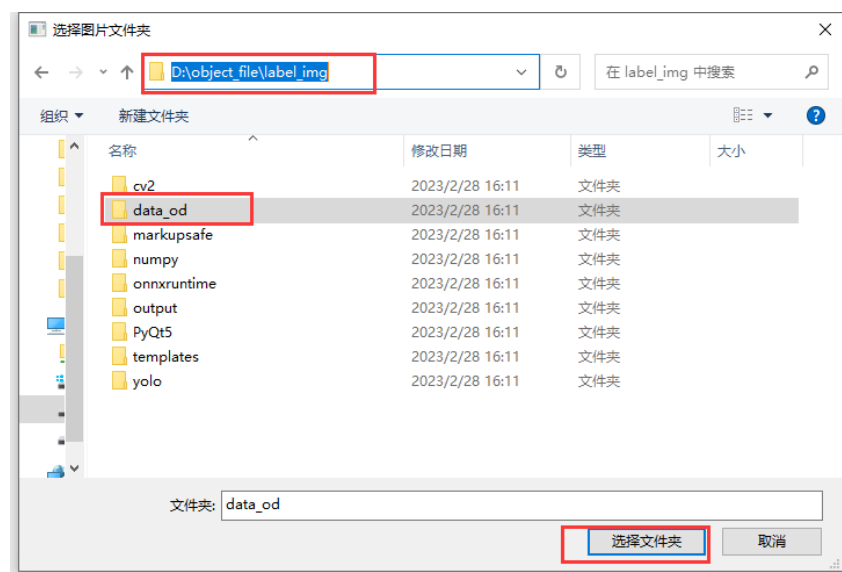


2.3. 导入图片

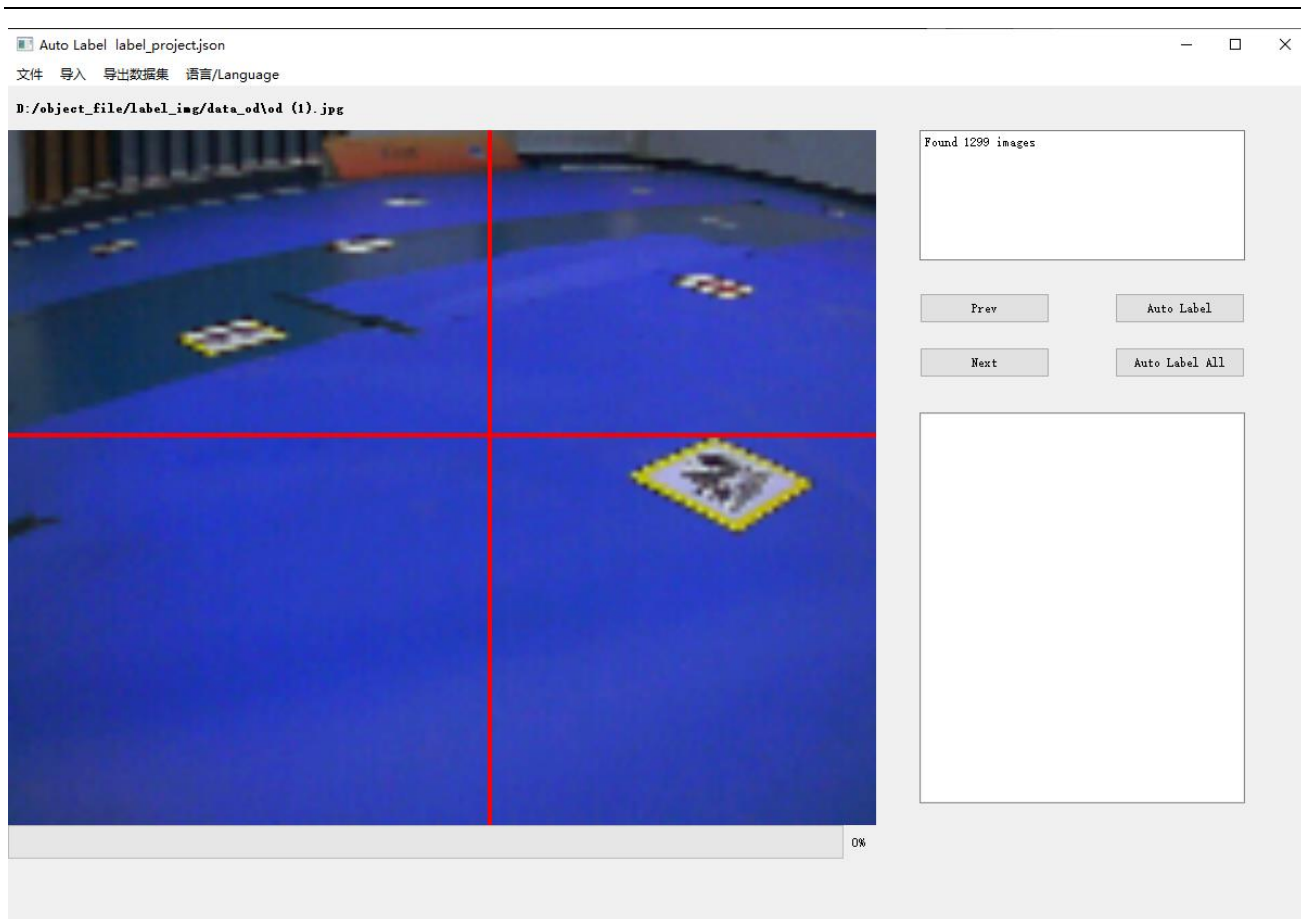
1. 在软件中点击左上角导入，选择打开图片文件夹。



2. 选择软件目录下的 data_od 文件夹，这个文件夹是我们使用 OpenART mini 拍摄的一些实际场地图片，用于本次测试，实际还需要同学们自己拍摄。

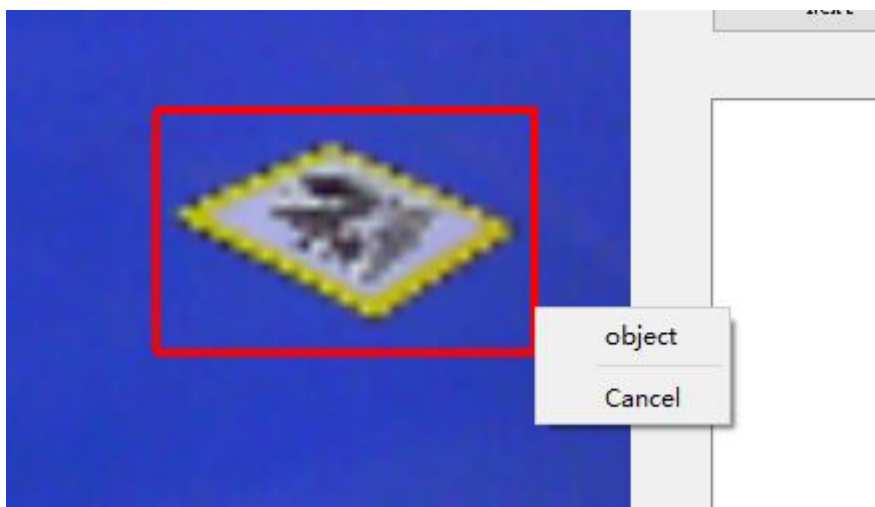


3. 然后可以看到图片已经导入软件中。

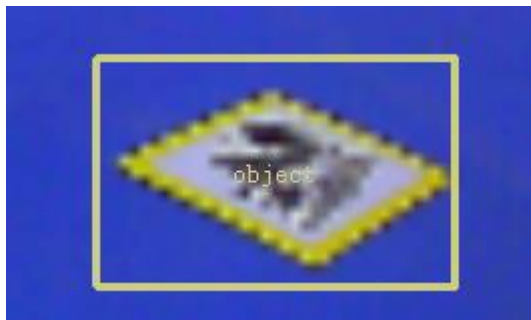


2.4. 标记图片

1. 使用鼠标左键来框选图片位置，先鼠标左键点击图片左上角位置，然后不松开左键拖动到图片右下角，松开鼠标左键，然后弹出类别提示。

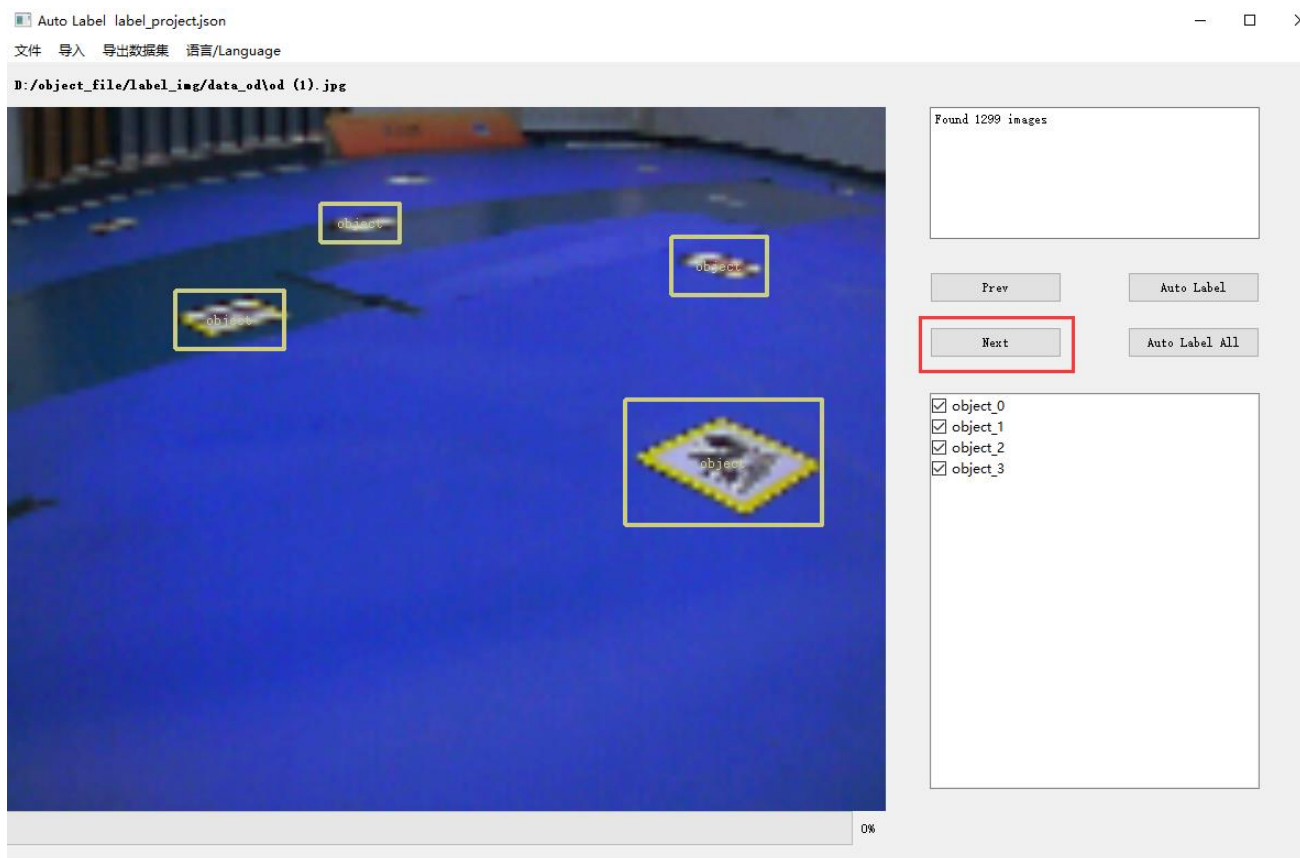


2. 选择 object，标记完成图片中第一个目标。



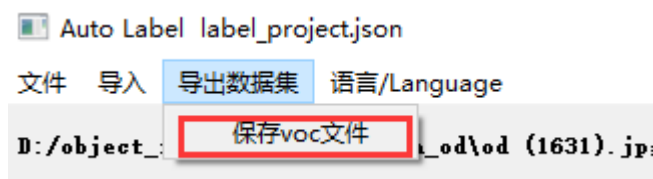
3. 然后将图片中的较为突出的目标都标记上，随后点击下一张图片进行标记。

注意：需要至少标记 30 张，否则后面训练会报错！

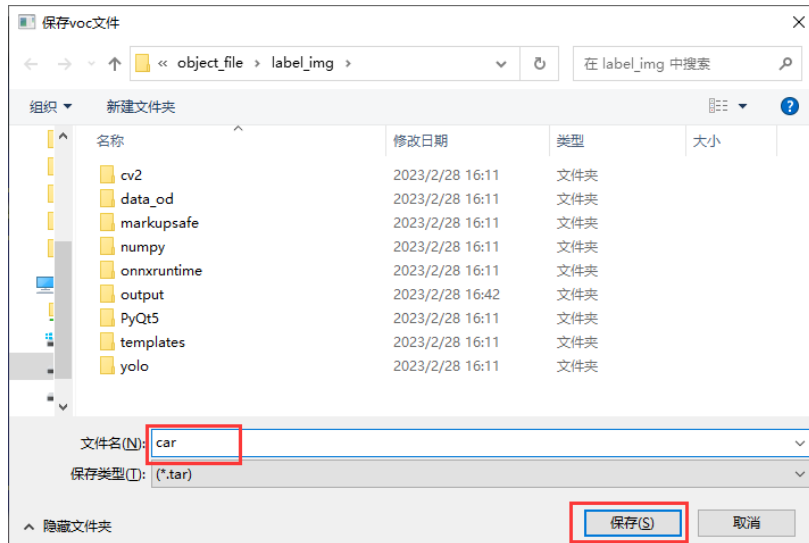


2.5. 保存图片和对应的标记文件

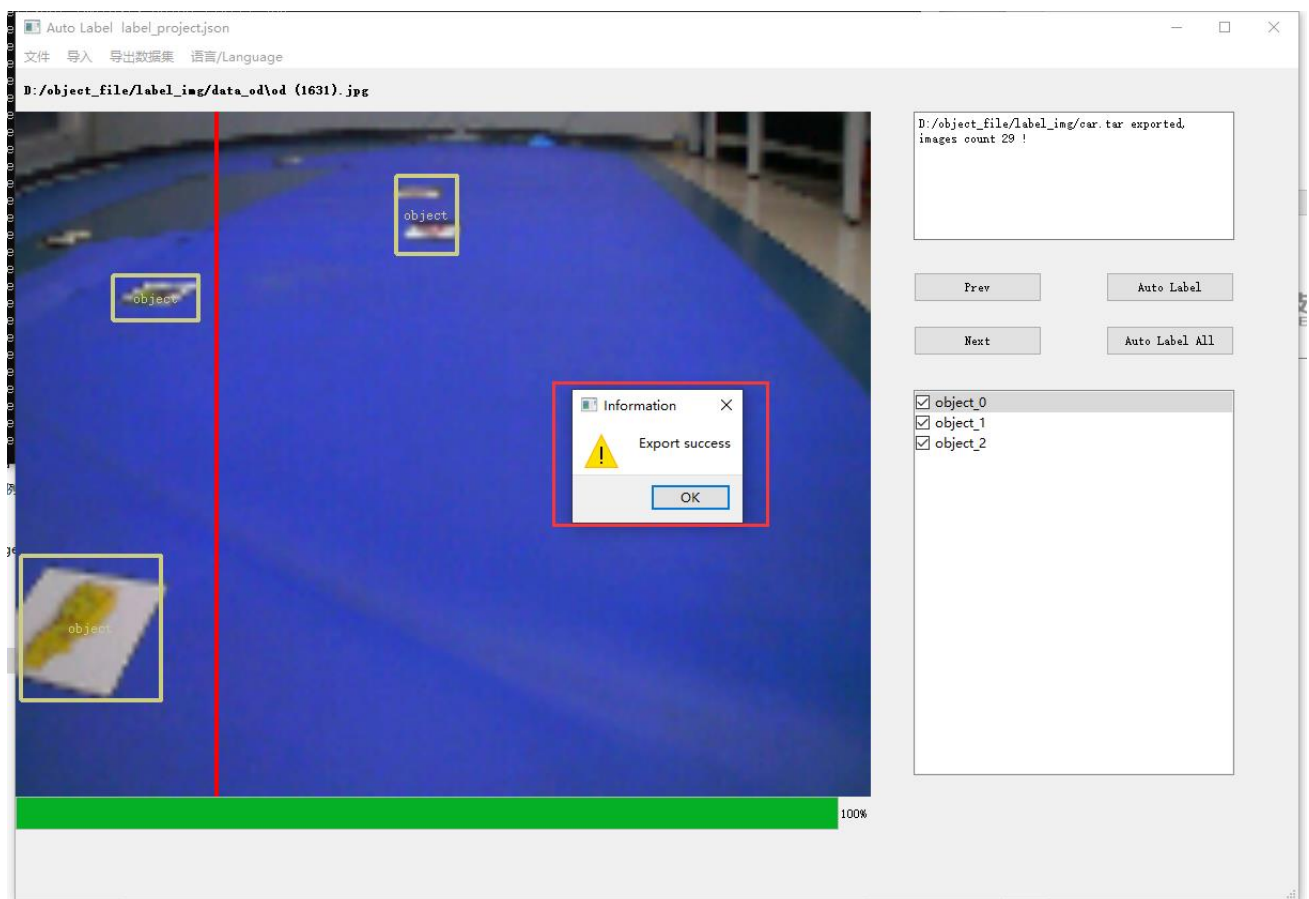
1. 图片标记完成后，点击右上角导出数据集，选择保存 voc 文件。



2. 输入文件名，然后保存，保存的是一个压缩包。



3. 等待保存完成，点击 OK 即可。



4. 我们可以在目录下看到这个压缩包。

DATA (D:) > object_file > label_img >

名称	修改日期	类型	大小
mainUI.exe	2023/2/24 11:47	应用程序	17,512 KB
qt5gui.dll	2023/2/24 11:49	应用程序扩展	6,844 KB
qt5core.dll	2023/2/24 11:49	应用程序扩展	5,883 KB
qt5widgets.dll	2023/2/24 11:49	应用程序扩展	5,370 KB
python310.dll	2023/2/24 11:49	应用程序扩展	4,389 KB
qt5quick.dll	2023/2/24 11:49	应用程序扩展	4,052 KB
qt5qml.dll	2023/2/24 11:49	应用程序扩展	3,508 KB
libcrypto-1_1.dll	2023/2/24 11:49	应用程序扩展	3,361 KB
libcrypto-1_1-x64.dll	2023/2/24 11:49	应用程序扩展	3,131 KB
libeay32.dll	2023/2/24 11:49	应用程序扩展	1,942 KB
qt5network.dll	2023/2/24 11:49	应用程序扩展	1,309 KB
unicodedata.pyd	2023/2/24 11:49	Python Extensio...	1,095 KB
ucrtbase.dll	2023/2/24 11:49	应用程序扩展	993 KB
car.tar	2023/2/28 16:49	TAR 压缩文件	920 KB
qt5multimedia.dll	2023/2/24 11:49	应用程序扩展	729 KB
libssl-1_1.dll	2023/2/24 11:49	应用程序扩展	687 KB

5. 将这个压缩包解压到 object_file\yolo3_smartcar 文件夹中。

DATA (D:) > object_file > yolo3_smartcar >

名称	修改日期	类型
.vscode	2023/2/28 16:11	文件夹
pycache	2023/2/28 16:11	文件夹
car	2023/2/28 16:53	文件夹
image	2023/2/28 16:11	文件夹

6. 解压后的文件夹中就包含原图像和标记文件。

DATA (D:) > object_file > yolo3_smartcar > car >

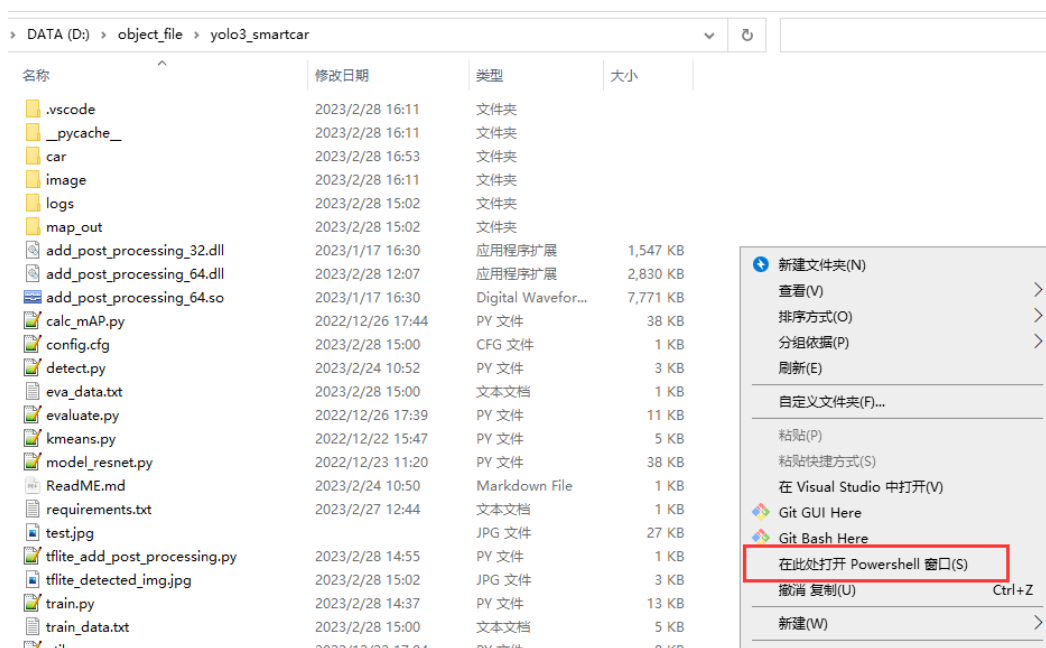
名称	修改日期
Annotations	2023/2/28 16:53
JPEGImages	2023/2/28 16:53

3. 训练模型

注意：需要完成 1.3 的[移动文件夹](#)步骤！

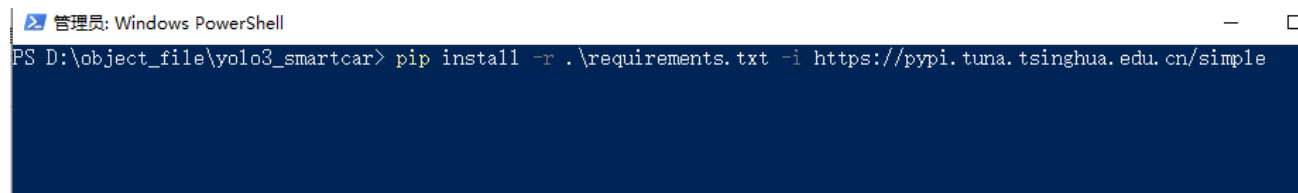
3.1. 安装 python 依赖库

1. 进入 yolo3_smartcar 文件夹。
2. 按住 shift，鼠标右键点击文件夹空白位置，选择打开。Powershell 窗口。



3. 在窗口中输入下面命令，并回车进行安装依赖库。

```
pip install -r .\requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
```



4. 等待安装完成。
5. 看到以下信息表示安装完成。


```
Successfully installed MarkupSafe-2.1.2 absl-py-1.4.0 argparse-1.4.0 astunparse-1.6.3 cachetools-5.3.0 certifi-2022.12.7
charset-normalizer-3.0.1 configparser-5.3.0 contourpy-1.0.7 cycler-0.11.0 flatbuffers-23.1.21 fonttools-4.38.0 gast-0.4
google-auth-2.16.1 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.51.3 h5py-3.8.0 idna-3.4 keras-2.11.0 kiwis
olver-1.4.4 libclang-15.0.6.1 markdown-3.4.1 matplotlib-3.7.0 numpy-1.23.4 oauthlib-3.2.2 opencv-python-4.7.0.72 opt-ein
sum-3.3.0 packaging-23.0 pillow-9.4.0 protobuf-3.19.6 pyasn1-0.4.8 pyasn1-modules-0.2.8 pyparsing-3.0.9 python-dateutil-
2.8.2 requests-2.28.2 requests-oauthlib-1.3.1 rsa-4.9 six-1.16.0 tensorboard-2.11.2 tensorboard-data-server-0.6.1 tensor
board-plugin-wit-1.8.1 tensorflow-2.11.0 tensorflow-estimator-2.11.0 tensorflow-intel-2.11.0 tensorflow-io-gcs-filesystem
m-0.31.0 termcolor-2.2.0 typing-extensions-4.5.0 urllib3-1.26.14 werkzeug-2.2.3 wheel-0.38.4 wrapt-1.15.0

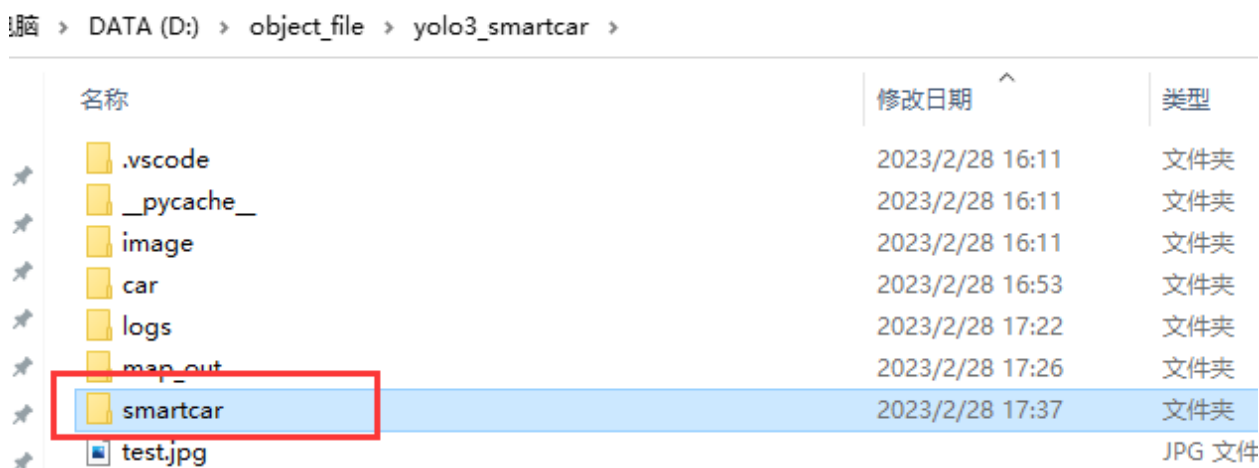
[notice] A new release of pip available: 22.2.2 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS D:\object_file\yolo3_smartcar>
```

3.2. 使用测试的图片 and 对应标记文件进行训练

我们提供了用于测试的图片源文件以及对应的标记文件，可以用于训练模型，并运行测试效果。关于图片标记的步骤查看第 2 章。

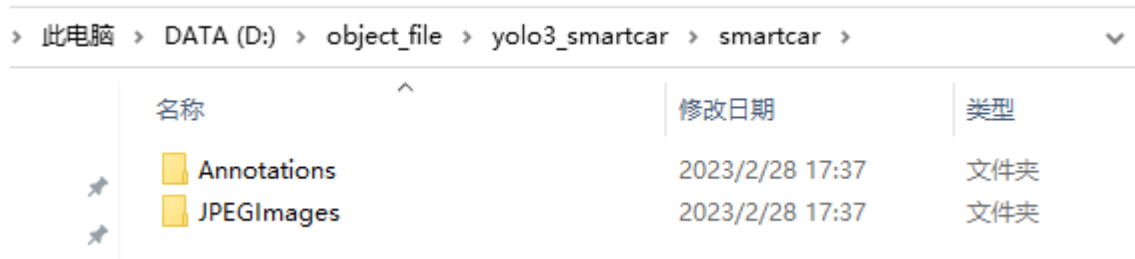
3.2.1. 将所需文件放入文件夹

1. 在提供的 object_file 文件夹中有 car_data 文件夹。
2. 将文件夹内的 smartcar 文件夹复制到 yolo3_smartcar 文件夹中。

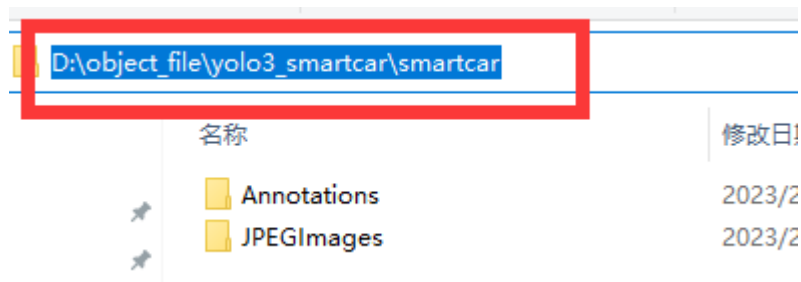


3.2.2. 将图片和对应标记文件生成统一的训练集和测试集

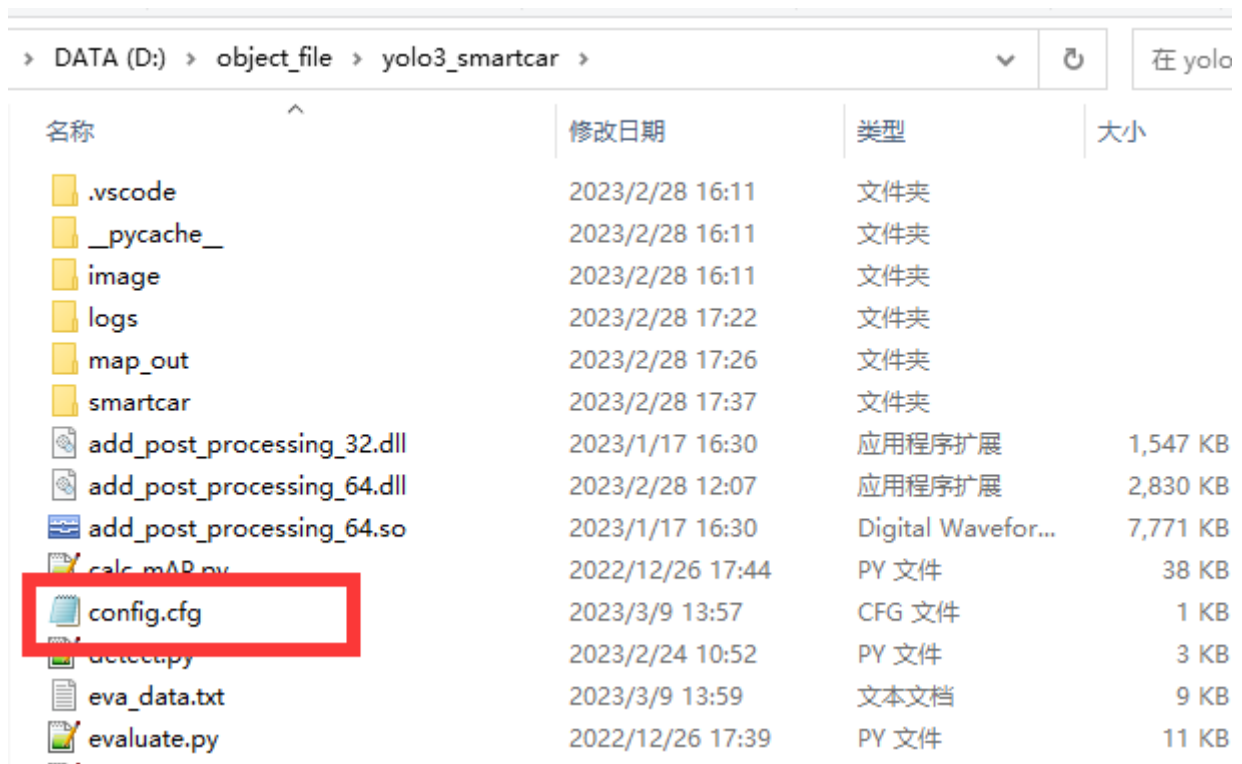
1. 打开 smartcar 文件夹。



2. 复制如下图所示路径。



3. 进入 yolo3_smartcar 文件夹，找到 config.cfg 文件并打开。



4. 在文件中修改文件夹路径，将刚刚复制的粘贴到如下图所示位置，并保存。

config.cfg - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[train]
voc_folder=D:\object_file\yolo3_smartcar\smartcar
train_data=train_data.txt
eva_data=eva_data.txt
cluster_anchor=yolo3_anchors.txt
batch_size=32
total_epochs=150
```

5. 按照 3.1 的步骤再次进入 Powershell 窗口。

6. 输入下方命令，并回车。

```
python .\voc_convertor.py
```

7. 看到下方提示即可。

```
PS D:\object_file\yolo3_smartcar> python .\voc_convertor.py
----voc_convertor_start----
----voc_convertor_finish----
```

如果有问题会提示报错。

3.2.3. 使用聚类算法整合训练集

1. 输入下方命令，并回车。

```
python .\kmeans.py
```

2. 会输出训练集的聚类程度。

```
PS D:\object_file\yolo3_smartcar> python .\kmeans.py
D:\object_file\yolo3_smartcar\smartcar\JPEGImages\od (2502). jpg$71, 38, 98, 49, 0$68, 17, 83, 25, 0$10, 35, 10, 45, 0
D:\object_file\yolo3_smartcar\smartcar\JPEGImages\od (1930). jpg$10, 59, 36, 73, 0$83, 23, 103, 31, 0$10, 28, 10, 37, 0
box error count:2
K anchors:
[[ 9 4]
 [12 6]
 [23 14]]
(3, 2)
Avg Accuracy: 74.91%
Accuracy: 74.91%
```

3.2.4. 开始训练

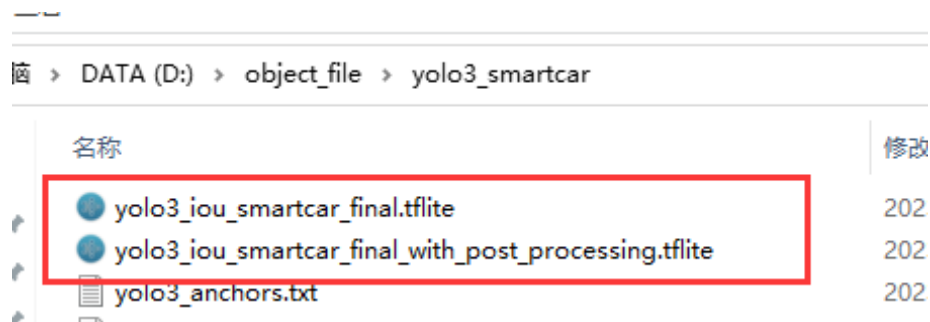
1. 输入下方命令，并回车。

```
python .\train.py
```

2. 等待训练完成。
3. 训练过程中如果判断多轮的 loss 趋于稳定会提前退出。

```
Epoch 68: LearningRateScheduler setting learning rate to 0.00564608834148813.
Epoch 68/150
20/20 [=====] - 5s 230ms/step - loss: 8.4795 - val_loss: 9.8457 - lr: 0.0056
Epoch 68: early stopping
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 26). These functions will not be directly callable after loading.
E:\Python\Python_3.10.7\lib\site-packages\tensorflow\lite\python\convert.py:765: UserWarning: Statistics for quantized inputs were expected, but not specified, continuing anyway.
  warnings.warn("Statistics for quantized inputs were expected, but not ")
2023-02-28 17:49:04.414794: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:362] Ignored output_for
at.
2023-02-28 17:49:04.414882: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:365] Ignored drop_contr
ol dependency.
2023-02-28 17:49:04.416418: I tensorflow/cc/saved_model/loader.cc:45] Reading SavedModel from: C:\Users\ADMINI~1\AppData
\Local\Temp\tmpoixsy6ej
2023-02-28 17:49:04.429638: I tensorflow/cc/saved_model/loader.cc:89] Reading meta graph with tags { serve }
2023-02-28 17:49:04.429704: I tensorflow/cc/saved_model/loader.cc:130] Reading SavedModel debug info (if present) from:
C:\Users\ADMINI~1\AppData\Local\Temp\tmpoixsy6ej
2023-02-28 17:49:04.471893: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass i
s not enabled
2023-02-28 17:49:04.480390: I tensorflow/cc/saved_model/loader.cc:229] Restoring SavedModel bundle.
2023-02-28 17:49:04.674241: I tensorflow/cc/saved_model/loader.cc:213] Running initialization op on SavedModel bundle a
t path: C:\Users\ADMINI~1\AppData\Local\Temp\tmpoixsy6ej
2023-02-28 17:49:04.735402: I tensorflow/cc/saved_model/loader.cc:305] SavedModel load for tags { serve }; Status: succ
ess: OK. Took 318979 microseconds.
2023-02-28 17:49:04.904050: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash rep
roducer, set env var 'MLIR_CRASH_REPRODUCER_DIRECTORY' to enable.
fully_quantize: 0, inference_type: 6, input_inference_type: INT8, output_inference_type: INT8
config file: config.cfg, model_file: yolo3_iou_smartcar_final.tflite
Training Complete
PS D:\object_file\yolo3_smartcar>
```

4. 在文件中也会生成两个文件。



3.2.5. 验证模型

1. 输入下方命令，并回车。

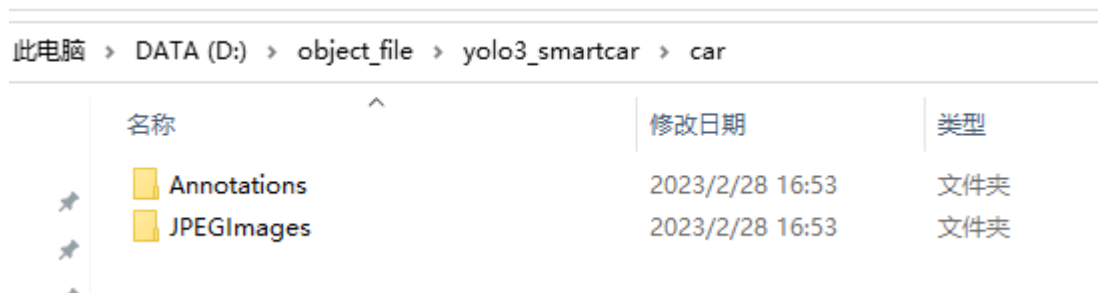
```
python .\evaluate.py
```

2. 会生成一个准确度的图

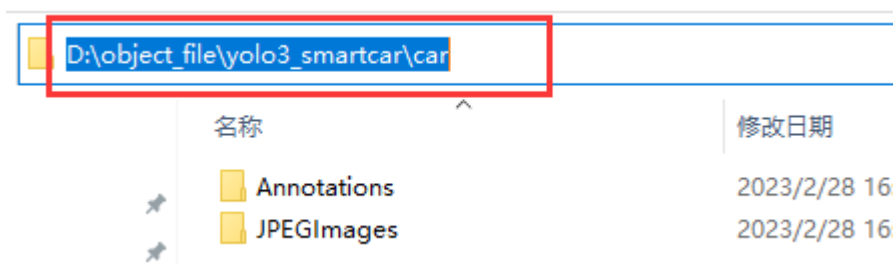
3.3. 使用自己的图片和对应标记文件进行训练

3.3.1. 将图片和对应标记文件生成训练集和测试集

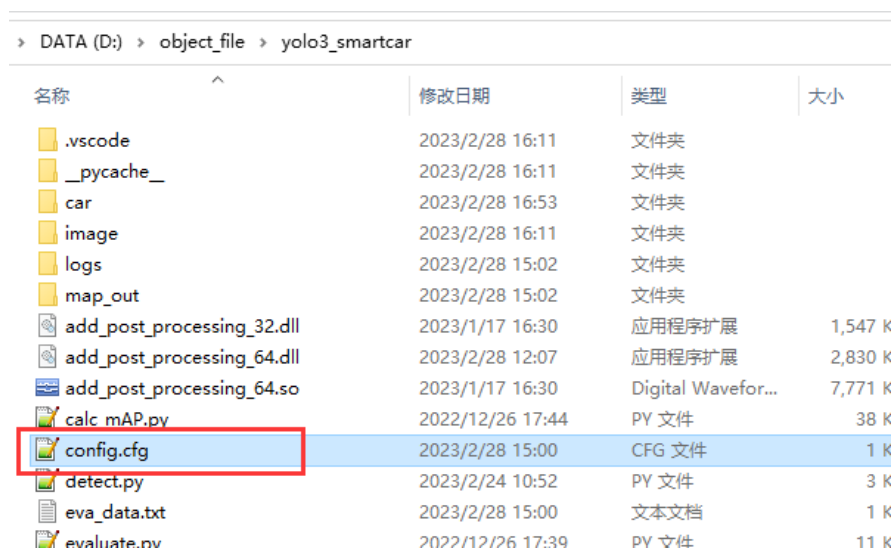
1. 进入 yolo3_smartcar 文件夹，找到第 2 章生成的 car 文件夹。



2. 复制如下图所示路径。



3. 进入 yolo3_smartcar 文件夹，找到 config.cfg 文件并打开。



4. 在文件中修改文件夹路径，将刚刚复制的粘贴到如下图所示位置，并保存。

```
*config.cfg - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

[train]
voc_folder=D:\object_file\yolo3_smartcar\car
train_data=train_data.txt
eva_data=eva_data.txt
cluster_anchor=yolo3_anchors.txt
batch_size=32
total_epochs=150
```

5. 按照 3.1 的步骤再次进入 Powershell 窗口。

6. 输入下方命令，并回车。

```
python .\voc_convertor.py
```

7. 看到下方提示即可。

```
PS D:\object_file\yolo3_smartcar> python .\voc_convertor.py
---voc_convertor_start---
---voc_convertor_finish---
```

如果有问题会提示报错。

3.3.2. 使用聚类算法整合训练集

1. 输入下方命令，并回车。

```
python .\kmeans.py
```

2. 会输出训练集的聚类程度。

```
PS D:\object_file\yolo3_smartcar> python .\kmeans.py
box error count:0
K anchors:
[[ 9  6]
 [22 16]
 [23 12]]
(3, 2)
Avg Accuracy: 75.34%
Accuracy: 75.34%
```


3.3.3. 开始训练

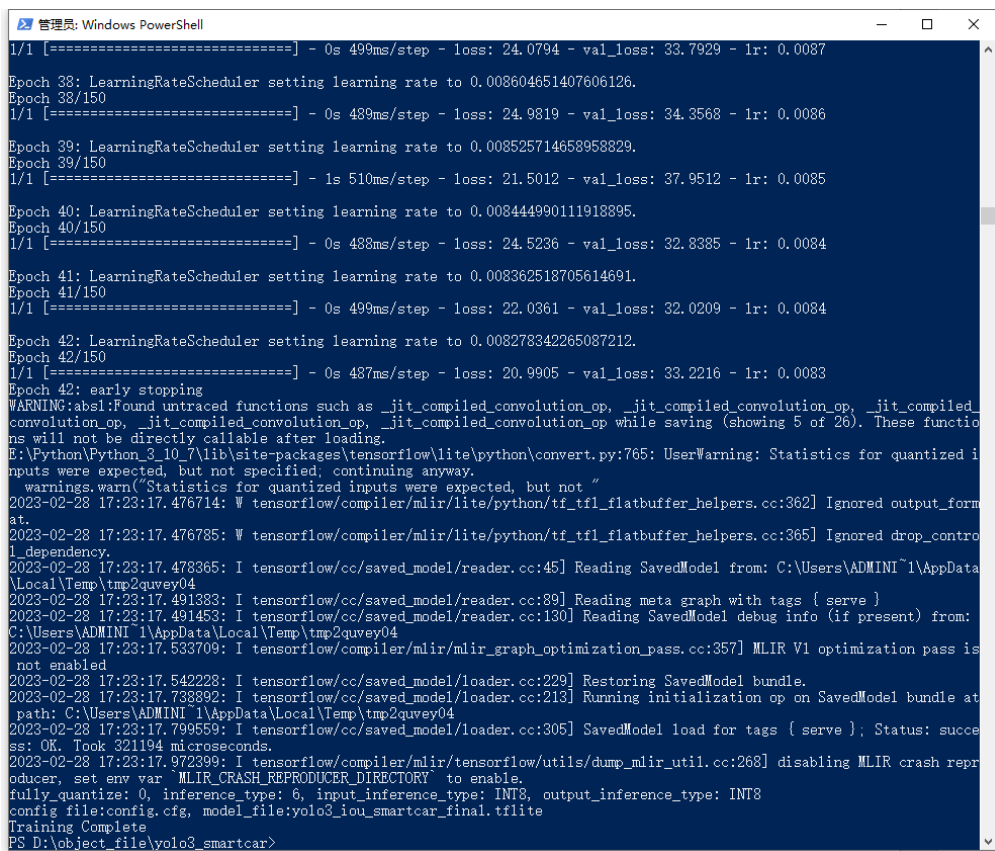
注意：需要之前至少标记 30 张，否则训练会报错！

1. 输入下方命令，并回车。

```
python .\train.py
```

2. 等待训练完成。

3. 训练过程中如果判断多轮的 loss 趋于稳定会提前退出。

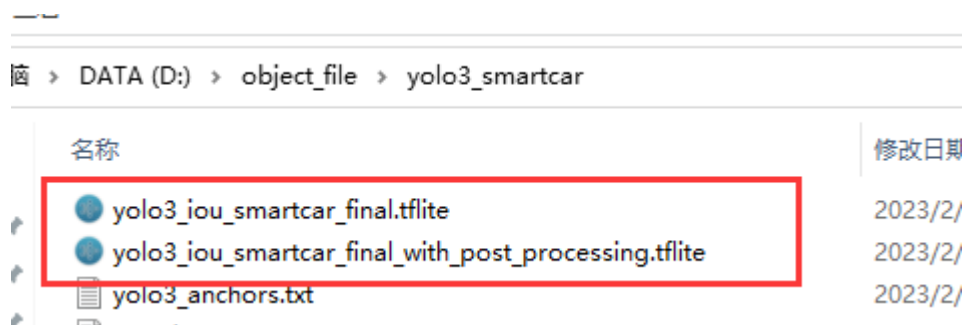


```

I/1 [=====] - 0s 499ms/step - loss: 24.0794 - val_loss: 33.7929 - lr: 0.0037
Epoch 38: LearningRateScheduler setting learning rate to 0.008604651407606126.
Epoch 38/150
I/1 [=====] - 0s 439ms/step - loss: 24.9819 - val_loss: 34.3568 - lr: 0.0086
Epoch 39: LearningRateScheduler setting learning rate to 0.008525714658958829.
Epoch 39/150
I/1 [=====] - 1s 510ms/step - loss: 21.5012 - val_loss: 37.9512 - lr: 0.0085
Epoch 40: LearningRateScheduler setting learning rate to 0.008444990111918895.
Epoch 40/150
I/1 [=====] - 0s 438ms/step - loss: 24.5236 - val_loss: 32.8385 - lr: 0.0084
Epoch 41: LearningRateScheduler setting learning rate to 0.008362518705614691.
Epoch 41/150
I/1 [=====] - 0s 499ms/step - loss: 22.0361 - val_loss: 32.0209 - lr: 0.0084
Epoch 42: LearningRateScheduler setting learning rate to 0.008278342265087212.
Epoch 42/150
I/1 [=====] - 0s 437ms/step - loss: 20.9905 - val_loss: 33.2216 - lr: 0.0083
Epoch 42: early stopping
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 26). These functions will not be directly callable after loading.
E:\Python\Python310_7\lib\site-packages\tensorflow\lite\python\convert.py:765: UserWarning: Statistics for quantized inputs were expected, but not specified; continuing anyway.
  warnings.warn("Statistics for quantized inputs were expected, but not ")
2023-02-28 17:23:17.476714: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:362] Ignored output format.
2023-02-28 17:23:17.476785: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:365] Ignored drop control dependency.
2023-02-28 17:23:17.478365: I tensorflow/cc/saved_model/loader.cc:45] Reading SavedModel from: C:\Users\ADMINI~1\AppData\Local\Temp\tmp2quvey04
2023-02-28 17:23:17.491383: I tensorflow/cc/saved_model/loader.cc:89] Reading meta graph with tags { serve }
2023-02-28 17:23:17.491453: I tensorflow/cc/saved_model/loader.cc:130] Reading SavedModel debug info (if present) from: C:\Users\ADMINI~1\AppData\Local\Temp\tmp2quvey04
2023-02-28 17:23:17.533709: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass is not enabled
2023-02-28 17:23:17.542228: I tensorflow/cc/saved_model/loader.cc:229] Restoring SavedModel bundle.
2023-02-28 17:23:17.738892: I tensorflow/cc/saved_model/loader.cc:213] Running initialization op on SavedModel bundle at path: C:\Users\ADMINI~1\AppData\Local\Temp\tmp2quvey04
2023-02-28 17:23:17.799559: I tensorflow/cc/saved_model/loader.cc:305] SavedModel load for tags { serve }; Status: success: OK. Took 321194 microseconds.
2023-02-28 17:23:17.972399: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash reproducer, set env var 'MLIR_CRASH_REPRODUCER_DIRECTORY' to enable.
fully_quantize: 0, inference_type: 6, input_inference_type: INT8, output_inference_type: INT8
config_file:config.cfg, model_file:yolo3_iou_smartcar_final.tflite
Training Complete
PS D:\object_file\yolo3_smartcar>

```

4. 在文件中也会生成两个文件。

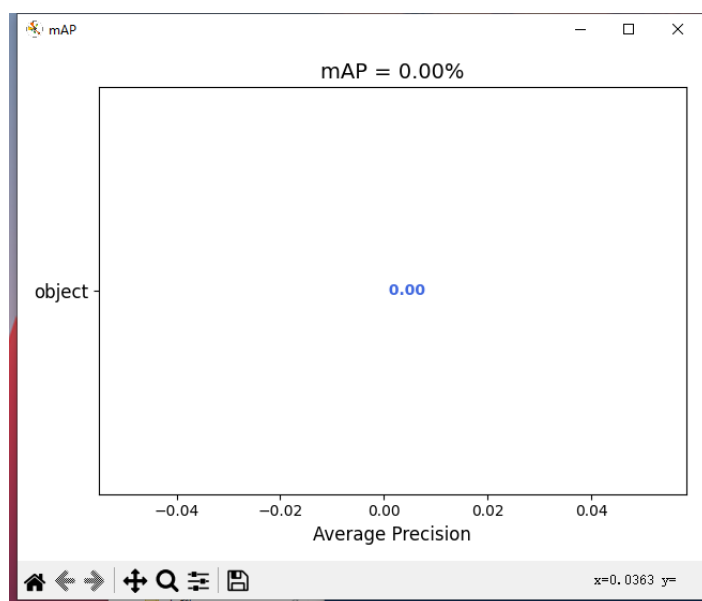


3.3.4. 验证模型

1. 输入下方命令，并回车。

```
python .\evaluate.py
```

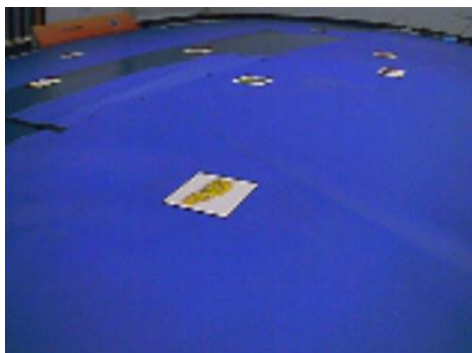
2. 会生成一个准确度的图，如果标记的图片太少模型会出现错误，如下图所示，显示为 0。正确的效果查看 3.2.5 的示意图。测试需要 600 张以上才有结果输出。



3. 输入下方命令，并回车。

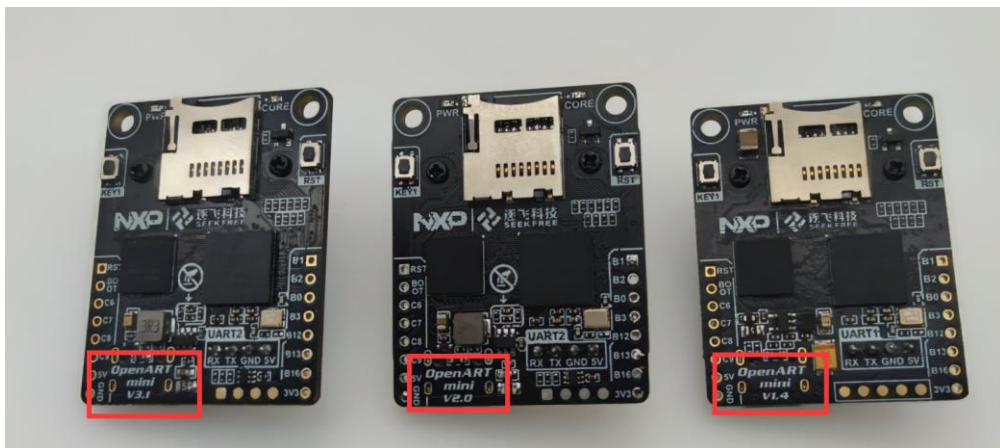
```
python detect.py -model yolo3_iou_smartcar_final.tflite -image test.jpg
```

4. 会生成一个目标检测后的图，如果标记图片过少模型会出现错误，如下图所示，显示为没有效果。正确的效果查看 3.2.5 的示意图。测试需要 600 张以上才有效果。

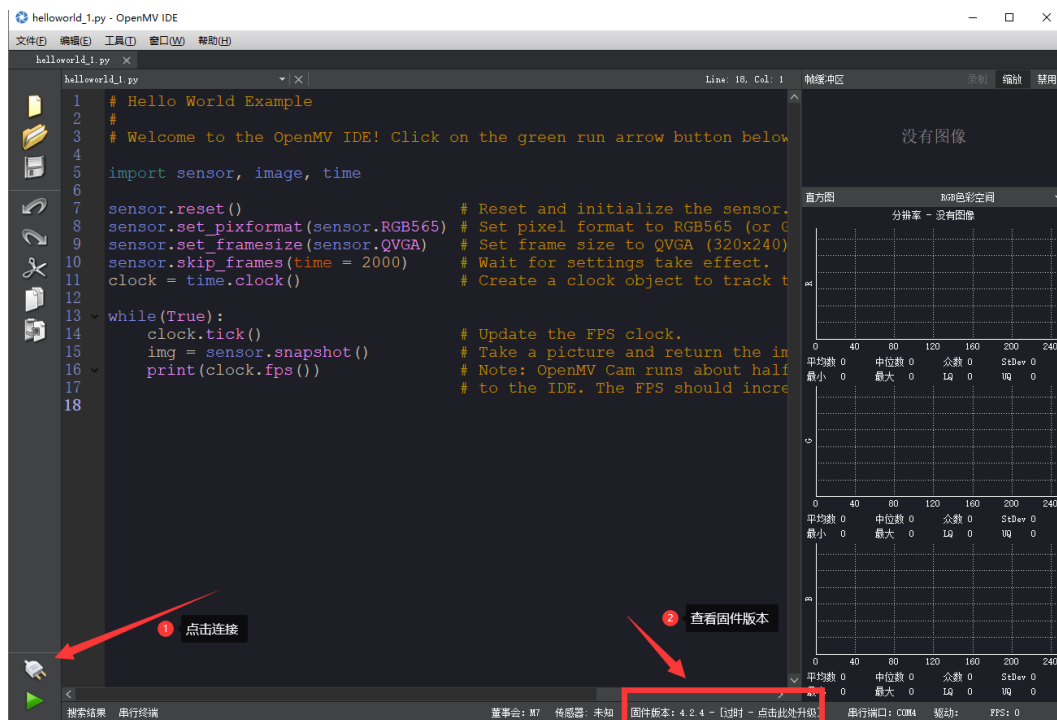


4.模型使用

首先查看 OpenART 背面的硬件版本，主要有 1.X，2.X，3.X 版本，如下图所示：



查看 OpenART 的固件版本，将 Type-C 数据线连接 OpenART mini 到电脑上，打开 OpenMV IDE，连接到 IDE 后，在软件的右下角查看 OpenART mini 的固件版本，如下图所示：



1. 如果硬件版本是 1.X 和 2.X 版本，固件版本需要大于等于 4.1.2。如果硬件版本为 3.X 版本，固件版本需要大于等于 4.2.2，没达到的找在智能组交流群中找技术客服进行升级。可以找如下几位技术客服。



逐飞软件--尚羽




逐飞软件--jay



逐飞软件-布丁

2. 打开“03 所需文件”中的“OpenART mini tflite 目标检测”文件夹。
3. 将 openart_mul_od.py 文件在 OpenMV IDE 中打开。

本地磁盘 (C:) > 用户 > Administrator > 桌面 > 03 所需文件 > OpenART mini tflite 目标检测

名称	修改日期	类型	大小
 openart_mul_od.py	2022/12/29 9:57	PY 文件	2 KB

```

1 import seekfree, pyb
2 import sensor, image, time, tf, gc
3
4 sensor.reset() # Reset and initialize the sensor.
5 sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
6 sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
7 sensor.skip_frames(time = 2000) # Wait for settings take effect.
8 clock = time.clock() # Create a clock object to track the FPS.
9
10 #设置模型路径
11 face_detect = '/sd/yolo3_iou_smartcar_final_with_post_processing.tflite'
12 #载入模型
13 net = tf.load(face_detect)
14
15 while(True):
16     clock.tick()
17     img = sensor.snapshot()
18
19     #使用模型进行识别
20     for obj in tf.detect(net,img):
21         x1,y1,x2,y2,label,scores = obj
22
23         if(scores>0.70):
24             print(obj)
25             w = x2- x1
26             h = y2- y1
27             x1 = int((x1-0.1)*img.width())
28             y1 = int(y1*img.height())
29             w = int(w*img.width())
30             h = int(h*img.height())
31             img.draw_rectangle((x1,y1,w,h),thickness=2)
32
33     print(clock.fps())
34

```

4. 查看将 3.2 章节训练后输出的两个文件。



yolo3_iou_smartcar_final.tflite



yolo3_iou_smartcar_final_with_post_processing.tflite

5. 我们只需要 yolo3_iou_smartcar_final_with_post_processing.tflite 文件。

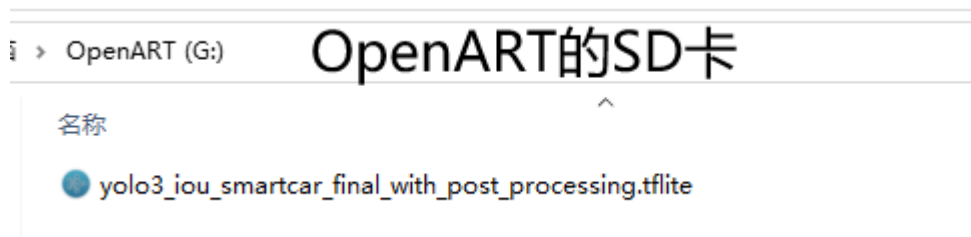


yolo3_iou_smartcar_final.tflite



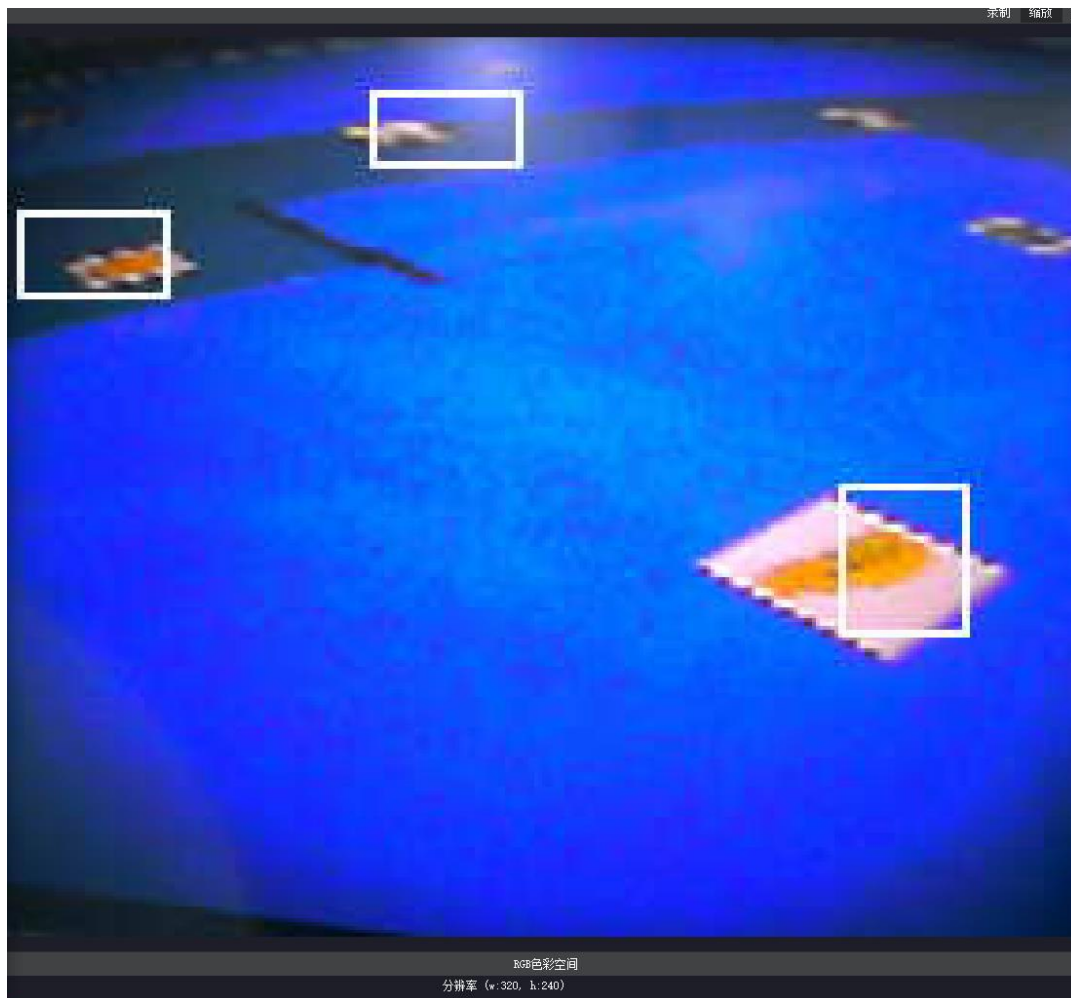
yolo3_iou_smartcar_final_with_post_processing.tflite

6. 把带 SD 卡的 OpenART mini 插到电脑上，然后将这个文件放到 OpenART mini 的 SD 卡中，再按一下 OpenART mini 的 RST 或者重新上电，确保文件保存完成。



7. 将重新上电后的 OpenART mini 连接到 OpenMV IDE 中，点击运行，测试结果如下：

如果想要更好的效果就需要采集和标记更多的图像。



8. 我们可以在窗口看到打印的目标信息，如下图所示

```
[0.3802053, 0.07091305, 0.5151919, 0.2316537, 0.0, 0.9915678]
[0.9146731, 0.1238717, 0.9963049, 0.3305721, 0.0, 0.8525074]
[0.8802933, 0.5134045, 0.9558858, 0.6463051, 0.0, 0.8184607]
6.931702
```

其中有三个数组和一个数字，数组代表识别到了三个目标，数字代表当前帧率。

```
[0.3802053, 0.07091305, 0.5151919, 0.2316537, 0.0, 0.9915678]
```

这是长度为 6 的数组

第一个值为目标左上角的 X 坐标在图像中和图像宽度的比例

第二个值为目标左上角的 Y 坐标在图像中和图像高度的比例

第三个值为目标右下角的 X 坐标在图像中和图像宽度的比例

第四个值为目标右下角的 Y 坐标在图像中和图像高度的比例

第五个值为图片的类别值，因为我们只有一个类别，所以为 0

第六个值为当前类别的确信度

如果检测到多个目标，会将所有目标的信息依次打印出来

OpenART mini 更多的外设使用方法详见 [OpenART mini 的使用教程](#)

OpenART mini 关于图片识别的教程详见[逐飞科技公众号中上一届的 eIQ 教程](#)

5.文档版本

版本号	日期	作者	内容变更
V1.0	2023/3/1	ZSY	初始版本。