1**安装tensorflow(version>=1.4.0)**

**2部署tensorflow models**

[在这里下载](https://github.com/tensorflow/models)

解压后重命名为models复制到tensorflow/目录下

[protoc-3.3.0-win32.zip](https://github.com/google/protobuf/releases?after=v3.3.2)下载

解压后将protoc.exe放入C:\Windows下

在tensorflow/models/research/打开powershell，运行protoc  object\_detection/protos/\*.proto --python\_out=.

3**训练数据准备(标记分类的图片)**

安装[labelImg](https://github.com/tzutalin/labelImg) 用来手动标注图片 ，图片需要是png或者jpg格式

标注信息会被保存为xml文件，使用 [这个脚本](https://github.com/datitran/raccoon_dataset/blob/master/xml_to_csv.py) 将所有xml文件转换为一个csv文件（xml文件路径识别在29行，根据情况自己修改）

把生成的csv文件分成训练集和测试集

**4.生成TFRecord文件**  
　　- 使用 [这个脚本](https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py) 将两个csv文件生成出两个TFRecord文件（训练自己的模型，必须使用TFRecord格式文件。图片路径识别在86行，根据情况自己修改）

**5.创建label map文件**  
　　id需要从1开始，class-N便是自己需要识别的物体类别名，文件后缀为.pbtxt  
　　　　item{  
　　　　　　id:1  
　　　　　　name: 'class-1'  
　　　　　　}  
　　　　item{  
　　　　　　id:2  
　　　　　　name: 'class-2'  
　　　　　　}

**6.下载模型并配置文件**  
　　- 下载一个[模型](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md)（文件后缀.tar.gz）  
　　- 修改对应的训练[pipline配置文件](https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs)   
　　　　- 查找文件中的PATH\_TO\_BE\_CONFIGURED字段，并做相应修改  
　　　　　　- num\_classes 改为你模型中包含类别的数量  
　　　　　　- fine\_tune\_checkpoint 解压.tar.gz文件后的路径 + /model.ckpt  
　　　　　　- from\_detection\_checkpoint:true  
　　　　　　- train\_input\_reader  
　　　　　　　　- input\_path 由train.csv生成的record格式训练数据  
　　　　　　　　- label\_map\_path 第5步创建的pbtxt文件路径  
　　　　　　- eval\_input\_reader  
　　　　　　　　- input\_path 由test.csv生成的record格式训练数据  
　　　　　　　　- label\_map\_path 第5步创建的pbtxt文件路径

**7. 训练模型**  
　　- 进入tensorflow/models/research/目录，运行  
　　python object\_detection/train.py --logtostderr  --pipeline\_config\_path=${PATH\_TO\_YOUR\_PIPELINE\_CONFIG} //第六步中修改的pipline配置文件路径// --train\_dir=${PATH\_TO\_TRAIN\_DIR} //生成的模型保存路径//

**8.导出模型**  
　　- 在第7步中，--train\_dir指向的路径中会生成一系列训练中自动保存的checkpoint，一个checkpoint由三个文件组成，后缀分别是.data-00000-of-00001 .index和.meta，任然在第7步的路径中，运行  
　　　　python object\_detection/export\_inference\_graph.py --input\_type image\_tensor  --pipeline\_config\_path ${PIPELINE\_CONFIG\_PATH} //第六步中修改的pipline配置文件路径//--trained\_checkpoint\_prefix ${TRAIN\_PATH} //上述的一个checkpoint，例如model.ckpt-112254// --output\_directory ${OUTPUT\_PATH} //输出模型文件的路径//  
　　- 目前直接使用export\_infernce\_graph.py会报错，解决方法是将object\_detection/目录下的export.py文件中第71行的  
　　rewrite\_options = rewriter\_config\_pb2.RewriterConfig(layout\_optimizer=rewriter\_config\_pb2.RewriterConfig.ON)  
　　改为  
　　rewrite\_options = rewriter\_config\_pb2.RewriterConfig()

**9.使用新模型识别图片**  
　　- 在jupyter notebook中打开object\_detection/路径下的object\_detection\_tutorial.ipynb文件  
　　- 修改第四步代码中的PATH\_TO\_CKPT，指向第8步导出模型路径中的frozen\_inference\_graph.pb文件  
　　- 修改第四步代码中的PATH\_TO\_LABELS，指向第5步创建的pbtxt文件路径  
　　- 修改第四步代码中的NUM\_CLASSES为模型实际包含的类别数  
　　- 注释掉第五步所有代码  
　　- 修改第九步代码中的PATH\_TO\_TEST\_IMAGES\_DIR， 指向待识别图片文件夹路径  
　　- 修改第九步代码中TEST\_IMAGE\_PATHS， 将文件名和扩展名修改为实际图片格式，如图片为1.bmp、2.bmp... 则对应改为'{}.bmp'，同时修改i值范围以覆盖所有待检测图片