For windows:

1. clone the tensorflow model: git clone https://github.com/tensorflow/models.git

2. download protocolbuffer from: https://github.com/protocolbuffers/protobuf/releases; choose protoc-3.9.1-win64.zip depending on your operating system

3. install dependency package:

# For CPU

pip install tensorflow

# For GPU

pip install tensorflow-gpu

pip install --user Cython

pip install --user contextlib2

pip install --user pillow

pip install --user lxml

pip install --user jupyter

pip install --user matplotlib

4. install cocoapi: <https://github.com/philferriere/cocoapi>  
5. Extract python file from proto file  
official guide: ./bin/protoc object\_detection/protos/\*.proto --python\_out=.  
use python code: create py file: use\_protobuf.py. Inside the file:

import os

import sys

args = sys.argv

directory = args[1]

protoc\_path = args[2]

for file in os.listdir(directory):

if file.endswith(".proto"):

os.system(protoc\_path+" "+directory+"/"+file+" --python\_out=.")

enter the research folder, log into the console:

python use\_protobuf.py Example: python use\_protobuf.py object\_detection/protos C:/Users/Gilbert/Downloads/bin/protoc  
6. add the paths to enviroment variables:

set PYTHONPATH=%PYTHONPATH%;<PATH\_TO\_TF>/TensorFlow/models/research

set PYTHONPATH=%PYTHONPATH%;<PATH\_TO\_TF>/TensorFlow/models/research/slim

7. # From within TensorFlow/models/research/

python setup.py build

python setup.py install

8. test the install: python object\_detection/builders/model\_builder\_test.py

9. resize the training image, create a new py file named transform\_image\_resolution.py:

from PIL import Image

import os

import argparse

def rescale\_images(directory, size):

for img in os.listdir(directory):

im = Image.open(directory+img)

im\_resized = im.resize(size, Image.ANTIALIAS)

im\_resized.save(directory+img)

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser(description="Rescale images")

parser.add\_argument('-d', '--directory', type=str, required=True, help='Directory containing the images')

parser.add\_argument('-s', '--size', type=int, nargs=2, required=True, metavar=('width', 'height'), help='Image size')

args = parser.parse\_args()

rescale\_images(args.directory, args.size)

create a console in the folder which contains the image folder: python transform\_image\_resolution.py -d images/ -s 800 600

10. label the train image, add bounding box:  
<https://github.com/tzutalin/labelImg>; <https://tzutalin.github.io/labelImg/>  
  
11. Create images folder in objection\_detection folder, create train, test folder inside images folder, put train image and test image seperately

12. download xml\_to\_csv.py and generate\_tf\_record.py from <https://github.com/datitran/raccoon_dataset> into objection\_detection folder:

Adjust xml\_to\_csv.py:

# Old:

def main():

image\_path = os.path.join(os.getcwd(), 'annotations')

xml\_df = xml\_to\_csv(image\_path)

xml\_df.to\_csv('raccoon\_labels.csv', index=None)

print('Successfully converted xml to csv.')

# New:

def main():

for folder in ['train', 'test']:

image\_path = os.path.join(os.getcwd(), ('images/' + folder))

xml\_df = xml\_to\_csv(image\_path)

xml\_df.to\_csv(('images/'+folder+'\_labels.csv'), index=None)

print('Successfully converted xml to csv.')

open a console in objection\_detection: python xml\_to\_csv.py

Adjust generate\_tf\_record.py:

# TO-DO replace this with label map

def class\_text\_to\_int(row\_label):

if row\_label == 'signature':

return 1

elif row\_label == 'others':

return 2

else:

return None

open a console in objection\_detection:   
python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record

python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record

13. create a folder called training inside objectiong\_detection folder:  
create a file called labelmap.pbtxt inside training folder:

item {

id: 1

name: 'signature'

}

item {

id: 2

name: 'others'

}

Download faster\_rcnn\_inception\_v2\_coco into objection\_detection folder from:

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

extract the zipped file into current directory

14. Copy faster\_rcnn\_inception\_v2\_pets.config in objection\_detection/samples/config to training folder:

Adjust the config file:

Line 9: change the number of classes to number of objects you want to detect (2 in my case)

Line 106: change fine\_tune\_checkpoint to the path of the model.ckpt file:  
fine\_tune\_checkpoint: "C:/Users/Gilbert/Downloads/Other/models/research/object\_detection/faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt"  
  
Line 123: change input\_path to the path of the train.records file:

input\_path: "C:/Users/Gilbert/Downloads/Other/models/research/object\_detection/train.record"

Line 135: change input\_path to the path of the test.records file:

input\_path: "C:/Users/Gilbert/Downloads/Other/models/research/object\_detection/test.record"

Line 125–137: change label\_map\_path to the path of the label map:  
label\_map\_path: "C:/Users/Gilbert/Downloads/Other/models/research/object\_detection/training/labelmap.pbtxt"

Line 130: change num\_example to the number of images in your test folder.

15. actual train the model:

To train the model we will use the train.py file, which is located in the object\_detection/legacy folder. We will copy it into the object\_detection folder and then we will open a command line and type:

python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_pets.config  
  
16. Monitor the result:

About every 5 minutes the current loss gets logged to Tensorboard. We can open Tensorboard by opening a second command line, navigating to the object\_detection folder and typing:

tensorboard --logdir=training

17. Now that we have a trained model we need to generate an inference graph, which can be used to run the model. For doing so we need to first of find out the highest saved step number. For this, we need to navigate to the training directory and look for the model.ckpt file with the biggest index.

Then we can create the inference graph by typing the following command in the command line.

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/faster\_rcnn\_inception\_v2\_pets.config --trained\_checkpoint\_prefix training/model.ckpt-XXXX --output\_directory inference\_graph

XXXX represents the highest number.

18 test the model open object\_detection\_tutorial.ipynb inside objection\_detection folder:

We only need to replace the fourth code cell.

From:

# What model to download.

MODEL\_NAME = 'ssd\_mobilenet\_v1\_coco\_2017\_11\_17'

MODEL\_FILE = MODEL\_NAME + '.tar.gz'

DOWNLOAD\_BASE = 'http://download.tensorflow.org/models/object\_detection/'

# Path to frozen detection graph. This is the actual model that is used for the object detection.

PATH\_TO\_FROZEN\_GRAPH = MODEL\_NAME + '/frozen\_inference\_graph.pb'

# List of the strings that are used to add a correct label for each box.

PATH\_TO\_LABELS = os.path.join('data', 'mscoco\_label\_map.pbtxt')

To:

MODEL\_NAME = 'inference\_graph'

PATH\_TO\_FROZEN\_GRAPH = MODEL\_NAME + '/frozen\_inference\_graph.pb'

PATH\_TO\_LABELS = 'training/labelmap.pbtxt'

References:

<https://towardsdatascience.com/creating-your-own-object-detector-ad69dda69c85>

<https://gilberttanner.com/blog/live-object-detection>

<https://gilberttanner.com/blog/installing-the-tensorflow-object-detection-api>

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md>

<https://www.microsoft.com/developerblog/2018/05/07/handwriting-detection-and-recognition-in-scanned-documents-using-azure-ml-package-computer-vision-azure-cognitive-services-ocr/>

source of image: <https://www.gsa.gov/real-estate/real-estate-services/leasing-policy-procedures/lease-documents>