# Installation

Install Tensorflow using the following command:

**$ pip install tensorflow**

**$ pip install pillow Cython lxml jupyter matplotlib pycocotools**

# Install protobuf using Home Brew  
**$ brew install protobu**f

For protobuf installation on other OS, follow the instructions [here](http://google.github.io/proto-lens/installing-protoc.html).

**PROTOC\_ZIP=protoc-3.3.0-linux-x86\_64.zip**  
**curl -OL https://github.com/google/protobuf/releases/download/v3.3.0/$PROTOC\_ZIP**  
**sudo unzip -o $PROTOC\_ZIP -d /usr/local bin/protoc**

## 1. Setting up the environment

$ **cd <path\_to\_your\_tensorflow\_installation>/models/research/**  
$ **protoc object\_detection/protos/\*.proto --python\_out=.**  
$ **export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim**

Run a quick test to confirm that the Object Detection API is working properly:

$ **python object\_detection/builders/model\_builder\_test.py**

If the result looks like the following, you're ready to proceed to the next steps!

...............  
----------------------------------------------------------------------  
Ran 15 tests in 0.123s  
  
OK

## Folder structure

data

|---------------- data\_id\_no   
 ├── annotations  
 └── xmls

└── trainval.txt

└── label\_map.pbtxt

├── images  
 ├── checkpoints  
 └── model.ckpt.data-00000-of-00001

└── model.ckpt.index

└── model.ckpt.meta

└── ssd\_mobilenet\_v2\_coco.config

├── tf\_record  
 └── train.record (will be created on execution of create\_tf\_record.py)

└── val.record (will be created on execution of create\_tf\_record.py)

├── train

|--------- eval

├── fine\_tuned\_model  
 ...

These folders will be used to store required components for our model as we proceed.

# 2. Collect images

Collect images of needed classes and place them in "images" forder.

## 3. Label your data set

Once you've collected all the images you need, we need to label them manually using[labelImg](https://github.com/tzutalin/labelImg), it saves label files (.xml) in the Pascal VOC format, which makes subsequent data conversion easier.

Double check that every image has a corresponding .xml file and save them in data/data\_id\_no/annotations/xmls/.

## 4. Create Label Map (.pbtxt)

Classes need to be listed in the label map. The label map should id and item like the following:

**item {**

**id: 1**

**name: 'person'**

**}**

**item {**

**id: 2**

**name: 'bicycle'**

**}**

Save this file as label\_map.pbtxt in data/data\_id\_no/annotations/label\_map.pbtxt

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

## 5. Create trainval.txt

trainval.txt is a list of image names without file extensions. Since we have sequential numbers for image names, the list should look like this:

|  |  |
| --- | --- |
|  | 1 |
|  | 2 |
|  | 3 |
|  | ... |
|  | 198 |
|  | 199 |
|  | 200 |

Save this file as trainval.txt in data/data\_id\_no/annotations/trainval.txt

## 6. Create TFRecord (.record)

This script is preconfigured to do 70-30 train-val split. Execute it by running:

# From model-master

$ **python research/object\_detection/dataset\_tools/create\_tf\_record.py**

If the script was successful, train.record and val.record should appear in your data/data\_id\_no/tf\_record/ directory.

## 7. Download pre-trained model

There are many pre-trained object detection models available in the [model zoo](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md). In order to train them using our custom data set, we need to *restore* them in Tensorflow using their checkpoints (.ckpt files), which are records of a previous model state.

eg. ssd\_mobilenet\_v2\_coco [here](http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz) and save its model checkpoint files (model.ckpt.meta, model.ckpt.index, model.ckpt.data-00000-of-00001) to our data/data\_id\_no/checkpoints/

.

## 8. Modify Config (.config) File

Each of the pretrained models has a config file that contains details about the model. To detect our custom class, the config file needs to be modified accordingly.

The config files are included in the models directory you cloned in the very beginning. You could find them in:

models\_master/research/object\_detection/samples/configs

In our case, we'll modify the config file for ssd\_mobilenet\_v2\_coco. Make a copy of it first and save it in the data/data\_id\_no/checkpoints/.

Here are the items we need to change:

1. Since we're trying to detect person and bicycle, change **num\_classes** to **2**
2. **fine\_tune\_checkpoint** tells the model which checkpoint file to use. Set this to data/data\_id\_no/checkpoints/model.ckpt
3. The model also needs to know where the TFRecord files and label maps are for both training and validation sets. Since our train.record and val.record are saved in **tf\_record** folder, our config should reflect that:

**train\_input\_reader: {**  
 **tf\_record\_input\_reader {**  
 **input\_path: "data/data\_id\_no/tf\_record/train.record"**  
 **}**  
 **label\_map\_path: "data/data\_id\_no/annotations/label\_map.pbtxt"**  
**}**  
  
**eval\_input\_reader: {**  
 **tf\_record\_input\_reader {**  
 **input\_path: "data/data\_id\_no/tf\_record/val.record"**  
 **}**  
 **label\_map\_path: "data/data\_id\_no/annotations/label\_map.pbtxt"**  
 **shuffle: false**  
 **num\_readers: 1**  
**}**

Now move the ssd\_mobilenet\_v2\_coco.config to data/data\_id\_no/checkpoints/.

**9. Training**

If you have successfully completed previous steps, you're ready to start training!

Follow the steps below:

# Change into models directory  
$ **cd models-master**

# Begin training  
$ **python research/object\_detection/legacy/train.py \**  
 **--logtostderr \**  
 **--train\_dir=data/data\_id\_no/train\**  
 **--pipeline\_config\_path=data/data\_id\_no/checkpoints/ssd\_mobilenet\_v2\_coco.config**

**eg:**

python research/object\_detection/legacy/train.py --logtostderr --train\_dir=data/1415/train --pipeline\_config\_path=data/1415/checkpoints/ssd\_mobilenet\_v2\_coco.config

Training time varies depending on the computing power of your machine.

## 10. Evaluation

Evaluation can be run in parallel with training. The eval.py script checks the train directory for progress and evaluate the model based on the most recent checkpoint.

# From the tensorflow/models/ directory  
$ **python research/object\_detection/legacy/eval.py \**  
 **--logtostderr \**  
 **--pipeline\_config\_path=data/data\_id\_no/checkpoints/ssd\_mobilenet\_v2\_coco.config \**  
 **--checkpoint\_dir=data/data\_id\_no/train\**  
 **--eval\_dir=data/data\_id\_no/eval**

**eg:**

python research/object\_detection/legacy/eval.py --logtostderr --pipeline\_config\_path=data/1415/checkpoints/ssd\_mobilenet\_v2\_coco.config --checkpoint\_dir=data/1415/train --eval\_dir=data/1415/eval

## 11. Model export

Once you finish training your model, you can export your model to be used for inference. If you've been following the folder structure, use the following command:

$ **python research/object\_detection/export\_inference\_graph.py \**  
 **--input\_type image\_tensor \**  
 **--pipeline\_config\_path data/data\_id\_no/checkpoints/ssd\_mobilenet\_v2\_coco.config\**  
 **--trained\_checkpoint\_prefix data/data\_id\_no/train/model.ckpt-<the\_highest\_checkpoint\_number> \**  
 **--output\_directory data/data\_id\_no/fine\_tuned\_mode**l

eg:

python research/object\_detection/export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path=data/1415/checkpoints/ssd\_mobilenet\_v2\_coco.config --trained\_checkpoint\_prefix data/1415/train/**model.ckpt-10** --output\_directory data/1415/fine\_tuned\_model

- As a result, the frozen inference graph is created in data/data\_id\_no/fine\_tuned\_models/.

- When you want to create another frozen inference graph, after few more training steps, delete the existing contents in data/data\_id\_no/fine\_tuned\_models/ folder.

## 12. Classify images

Now that you have a model, you can use it to detect objects in pictures! Before you proceed, pick an image you want to test the model with.

Open the jupyter notebook file, models/research/object\_detection/object\_detection\_tutorial.ipynb and follow the steps in the notebook:

1. **MODEL\_NAME = 'ssd\_mobilenet\_v2\_coco'**
2. **PATH\_TO\_CKPT = 'data/data\_id\_no/fine\_tuned\_models/frozen\_inference\_graph.pb'**
3. **PATH\_TO\_LABELS = ''data/data\_id\_no/annotations/label\_map.pbtxt'**
4. **NUM\_CLASSES = 2**
5. **Comment out cell #5 completely (just below Download Model)**
6. **Since we're only testing on one image, comment out PATH\_TO\_TEST\_IMAGES\_DIR and TEST\_IMAGE\_PATHS in cell #9 (just below Detection)**
7. **In cell #11 (the last cell), remove the for-loop, unindent its content, and add path to your test image:**

**image\_path = 'path/to/image\_you\_want\_to\_test.jpg'**

After following through the steps, run the notebook and you should see the objects in your test image highlighted by a bounding box!

API : custom\_od