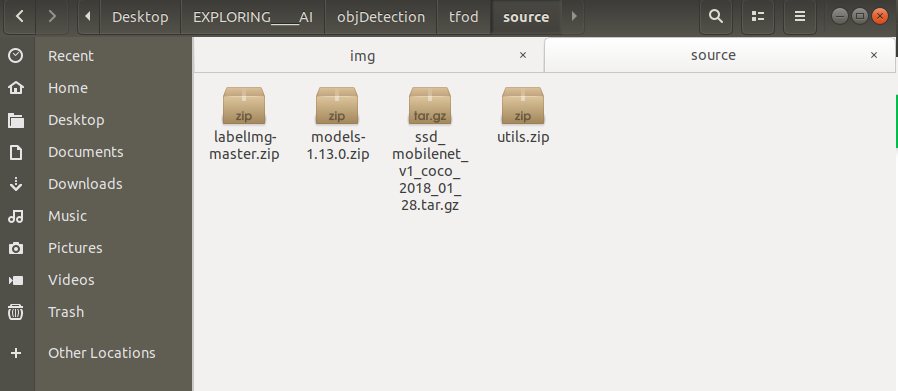
Configuration steps for TensorFlow object detection-

## STEP-1 Download the following content-

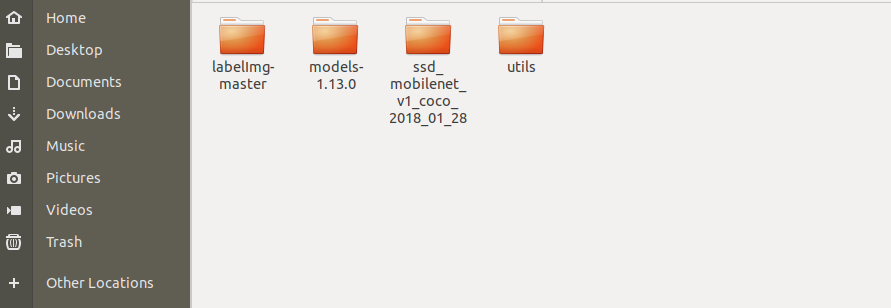
1. [Download](https://github.com/tensorflow/models/tree/v1.13.0) v1.13.0 model.
2. [Download](http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_2018_01_28.tar.gz) the ssd\_mobilenet\_v1\_coco model from the model zoo **or** any other model of your choice from [TensorFlow model zoo.](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md" \t "_blank)
3. [Download](https://drive.google.com/file/d/12F5oGAuQg7qBM_267TCMt_rlorV-M7gf/view?usp=sharing) Dataset & utils.
4. [Download](https://tzutalin.github.io/labelImg/) labelImg tool for labeling images.

before extraction, you should have the following compressed files –

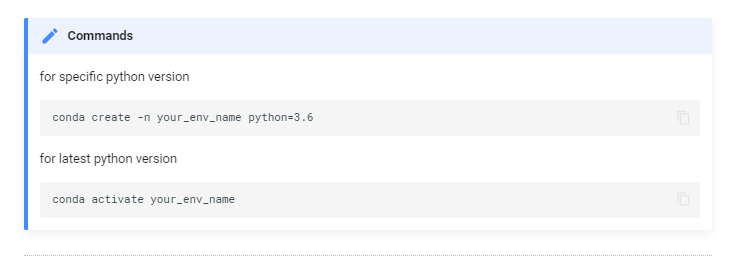


## STEP-2 Extract all the above zip files into a tfod folder and remove the compressed files-

Now you should have the following folders -



## STEP-3 Creating virtual env using conda-



## STEP-4 Install the following packages in your new environment-

### for GPU

pip install pillow lxml Cython contextlib2 jupyter matplotlib pandas opencv-python tensorflow-gpu==1.14.0

### for CPU only

pip install pillow lxml Cython contextlib2 jupyter matplotlib pandas opencv-python tensorflow==1.14.0

## STEP-5 Install protobuf using conda package manager-

conda install -c anaconda protobuf

## STEP-6 For protobuff to .py conversion download from a tool from here-

For windows -> [download](https://github.com/protocolbuffers/protobuf/releases/download/v3.11.0/protoc-3.11.0-win64.zip) source for other versions and OS - [click here](https://github.com/protocolbuffers/protobuf/releases/tag/v3.11.4)

Open command prompt and cd to research folder.

Now in the research folder run the following command-

### For Linux or Mac

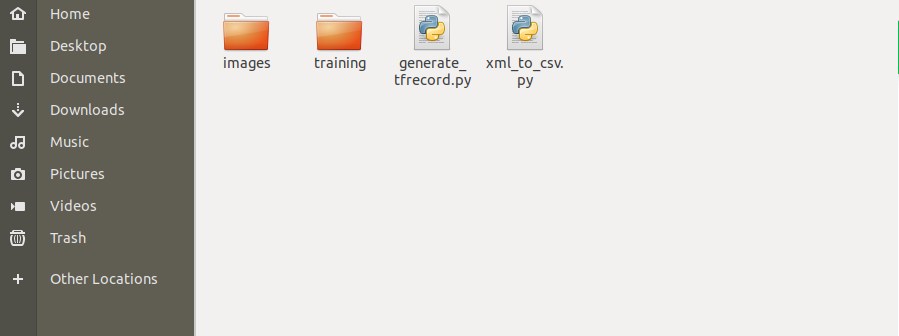
protoc object\_detection/protos/\*.proto --python\_out=.

### For Windows

protoc object\_detection/protos/\*.proto --python\_out=.

## STEP-7 Paste all content present in utils into research folder-

Following are the files and folder present in the utils folder-



## STEP-8 Paste ssd\_mobilenet\_v1\_coco or any other model downloaded from model zoo into research folder-

Now cd to the research folder and run the following python file-

python xml\_to\_csv.py

## STEP-9 Run the following to generate train and test records-

from the research folder-

for train:

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

for test:

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

## STEP-10 Copy from research/object\_detection/samples/config/ YOURMODEL.config file into research/training-

The following config file shown here is with respect to **ssd\_mobilenet\_v1\_coco**. So if you have downloaded it for any other model apart from SSD you'll see config file with YOUR\_MODEL\_NAME as shown below-

model {

YOUR\_MODEL\_NAME {

num\_classes: 6

box\_coder {

faster\_rcnn\_box\_coder {

**Hence always verify YOUR\_MODEL\_NAME before using the config file.**

## STEP-11 Update num\_classes, fine\_tune\_checkpoint ,and num\_steps plus update input\_path and label\_map\_path for both train\_input\_reader and eval\_input\_reader-

Changes to be made in the config file are highlighted in yellow color. You must update the value of those keys in the config file.

# SSDLite with Mobilenet v1 configuration for MSCOCO Dataset.

# Users should configure the fine\_tune\_checkpoint field in the train config as

# well as the label\_map\_path and input\_path fields in the train\_input\_reader and

# eval\_input\_reader. Search for "PATH\_TO\_BE\_CONFIGURED" to find the fields that

# should be configured.

model {

ssd {

num\_classes: 6

box\_coder {

faster\_rcnn\_box\_coder {

y\_scale: 10.0

x\_scale: 10.0

height\_scale: 5.0

width\_scale: 5.0

}

}

matcher {

argmax\_matcher {

matched\_threshold: 0.5

unmatched\_threshold: 0.5

ignore\_thresholds: false

negatives\_lower\_than\_unmatched: true

force\_match\_for\_each\_row: true

}

}

similarity\_calculator {

iou\_similarity {

}

}

anchor\_generator {

ssd\_anchor\_generator {

num\_layers: 6

min\_scale: 0.2

max\_scale: 0.95

aspect\_ratios: 1.0

aspect\_ratios: 2.0

aspect\_ratios: 0.5

aspect\_ratios: 3.0

aspect\_ratios: 0.3333

}

}

image\_resizer {

fixed\_shape\_resizer {

height: 300

width: 300

}

}

box\_predictor {

convolutional\_box\_predictor {

min\_depth: 0

max\_depth: 0

num\_layers\_before\_predictor: 0

use\_dropout: false

dropout\_keep\_probability: 0.8

kernel\_size: 3

use\_depthwise: true

box\_code\_size: 4

apply\_sigmoid\_to\_scores: false

conv\_hyperparams {

activation: RELU\_6,

regularizer {

l2\_regularizer {

weight: 0.00004

}

}

initializer {

truncated\_normal\_initializer {

stddev: 0.03

mean: 0.0

}

}

batch\_norm {

train: true,

scale: true,

center: true,

decay: 0.9997,

epsilon: 0.001,

}

}

}

}

feature\_extractor {

type: 'ssd\_mobilenet\_v1'

min\_depth: 16

depth\_multiplier: 1.0

use\_depthwise: true

conv\_hyperparams {

activation: RELU\_6,

regularizer {

l2\_regularizer {

weight: 0.00004

}

}

initializer {

truncated\_normal\_initializer {

stddev: 0.03

mean: 0.0

}

}

batch\_norm {

train: true,

scale: true,

center: true,

decay: 0.9997,

epsilon: 0.001,

}

}

}

loss {

classification\_loss {

weighted\_sigmoid {

}

}

localization\_loss {

weighted\_smooth\_l1 {

}

}

hard\_example\_miner {

num\_hard\_examples: 3000

iou\_threshold: 0.99

loss\_type: CLASSIFICATION

max\_negatives\_per\_positive: 3

min\_negatives\_per\_image: 0

}

classification\_weight: 1.0

localization\_weight: 1.0

}

normalize\_loss\_by\_num\_matches: true

post\_processing {

batch\_non\_max\_suppression {

score\_threshold: 1e-8

iou\_threshold: 0.6

max\_detections\_per\_class: 100

max\_total\_detections: 100

}

score\_converter: SIGMOID

}

}

}

train\_config: {

batch\_size: 24

optimizer {

rms\_prop\_optimizer: {

learning\_rate: {

exponential\_decay\_learning\_rate {

initial\_learning\_rate: 0.004

decay\_steps: 800720

decay\_factor: 0.95

}

}

momentum\_optimizer\_value: 0.9

decay: 0.9

epsilon: 1.0

}

}

fine\_tune\_checkpoint: "ssd\_mobilenet\_v1\_coco\_2018\_01\_28/model.ckpt"

from\_detection\_checkpoint: true

# Note: The below line limits the training process to 200K steps, which we

# empirically found to be sufficient enough to train the pets dataset. This

# effectively bypasses the learning rate schedule (the learning rate will

# never decay). Remove the below line to train indefinitely.

num\_steps: 20000

data\_augmentation\_options {

random\_horizontal\_flip {

}

}

data\_augmentation\_options {

ssd\_random\_crop {

}

}

}

train\_input\_reader: {

tf\_record\_input\_reader {

input\_path: "train.record"

}

label\_map\_path: "training/labelmap.pbtxt"

}

eval\_config: {

num\_examples: 8000

# Note: The below line limits the evaluation process to 10 evaluations.

# Remove the below line to evaluate indefinitely.

max\_evals: 10

}

eval\_input\_reader: {

tf\_record\_input\_reader {

input\_path: "test.record"

}

label\_map\_path: "training/labelmap.pbtxt"

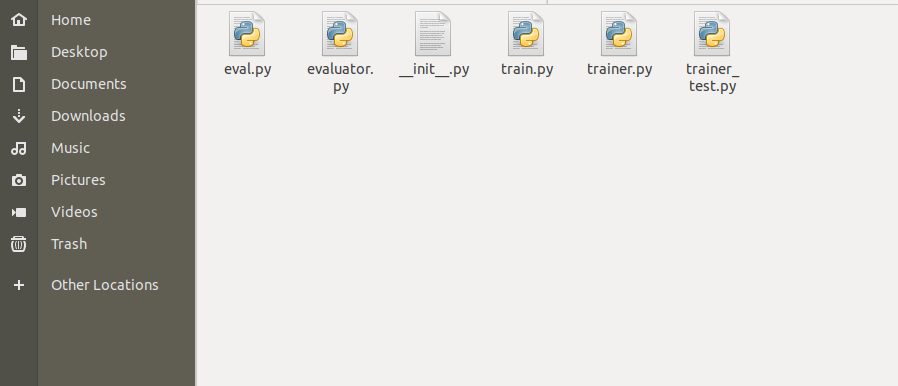
shuffle: false

num\_readers: 1

}

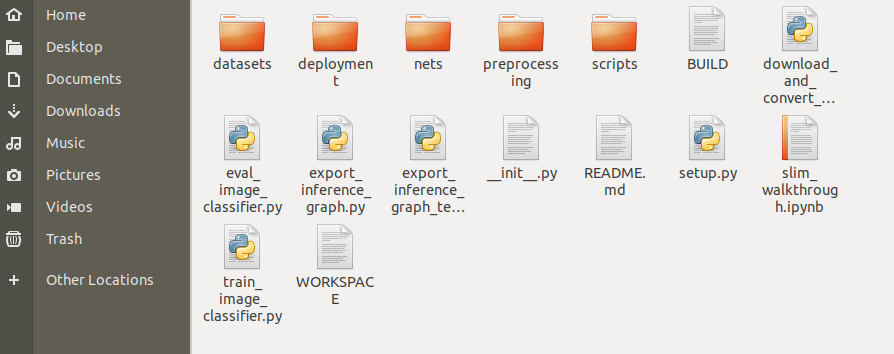
## STEP-12 From research/object\_detection/legacy/ copy train.py to research folder

legacy folder contains train.py as shown below –



## STEP-13 Copy deployment and nets folder from research/slim into the research folder-

slim folder contains the following folders -



## STEP-14 NOW Run the following command from the research folder. This will start the training in your local system-

Note : copy the command and replace **YOUR\_MODEL.config** with your own model's name for example **ssd\_mobilenet\_v1\_coco.config** and then run it in cmd prompt or terminal. And make sure you are in research folder.

python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/YOUR\_MODEL.config