Tensorflow Object

Detector Training Guideline

|  |  |  |
| --- | --- | --- |
| **Version** | **Last update date** | **Name** |
| Version 1.0 | 27/08/2019 | William |

## 0.Before Training

**For the land, machine and water pound this model.**

(the **inference\_graph\_2019\_08\_16\_2736x1824** is trained model the base model is faster\_rcnn\_inception\_v2\_pets , that for the accuracy is up to 20%+.)

If you want to improve the accuracy, you can follow this guide to use the new images of the same image requirements to retrain that model.

for **inference\_graph\_2019\_08\_16\_2736x1824** model

the image requirement: (recommend):

* Dimension:   2736x1824
* Dpi:               72x72
* Color:            RGB
* File type:       JPG

## 0.1 Environment

**Get in to the environment first:**

**if you use the venv :**

|  |
| --- |
| **source** venv/bin/activate |

**if you use the ubuntu linux server**

|  |
| --- |
| cd ~/Documents/tensorflow/  conda activate tensorflow\_env |

**Check tensor-flow/python/pip version:**

|  |
| --- |
| **pip** show tensorflow  python -V  pip -V |

**In this case:**

***Tensorflow:   1.14***

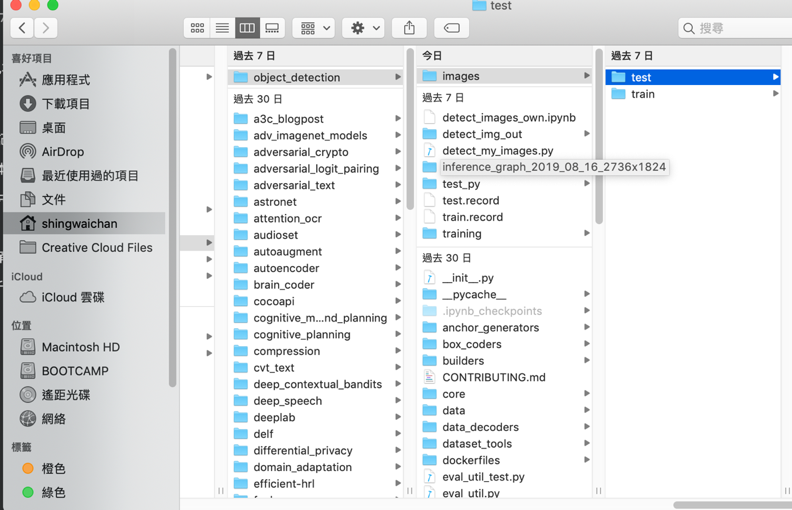
***Python:          3.6.8***

***pip:                 19.2.1***

## 1.Labeling data

**//先用labelimg將相片框起要detect的object**

Now that we have our images we need to move about **80%** of the images into the ***object\_detection/images/train*** directory and the other **20%** in the***object\_detection/images/test*** directory.



In order to label our data, we need some kind of image labeling software. LabelImg is a great tool for labeling images. It’s also freely available on Github and prebuilts can be downloaded easily.

[**LabelImg Github**](https://github.com/tzutalin/labelImg)

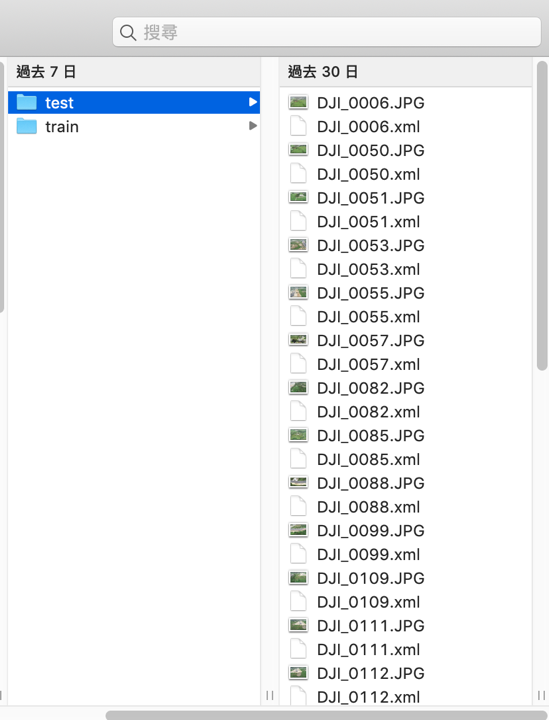
[**LabelImg download**](https://tzutalin.github.io/labelImg/)

**open labelimg application**

**cmd: labelimg**

labeled and save to current directory, the labelimg app will generate the xml file.

repeat the labelling images on test folder and train folder



## 2.Generating TFRecords for training

**//gen之前要先將labelimg生成的.xml file 轉成 .csv**

**2.1 Go to object\_detection folder**

|  |
| --- |
| **cd /Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection** |

**the xml\_to\_csv.py file are stored in object\_detection, if not exit, create it and copy the code at below**

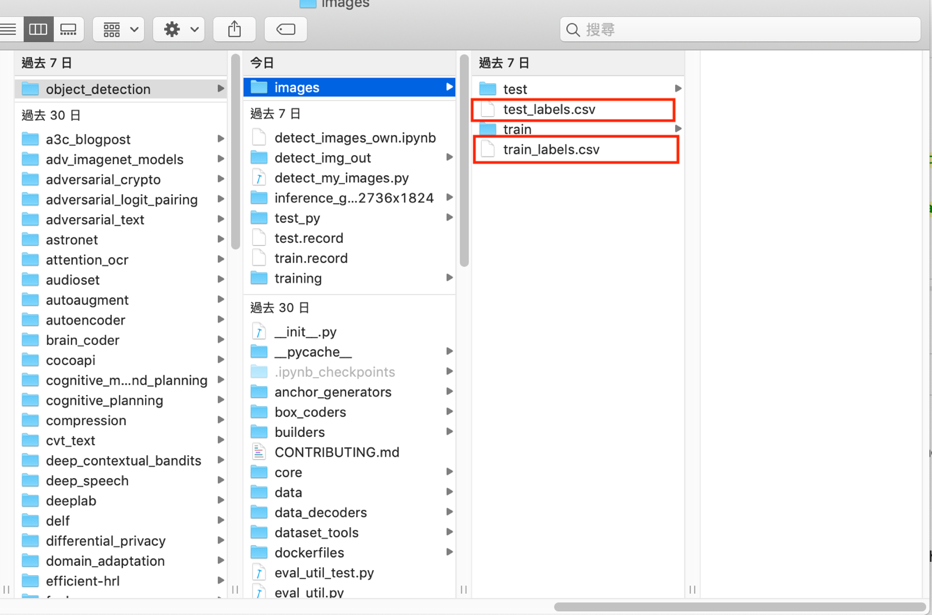
here is code for python: ***xml\_to\_csv.py***

|  |
| --- |
| # Old: this one only for your reference  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: use this one  def main():      for folder in ['train', 'test']:         image\_path = os.path.join(os.getcwd(), ('images/' + folder))#add 2 folder to path          xml\_df = xml\_to\_csv(image\_path) #changing to csv with two folder          xml\_df.to\_csv(('images/'+folder+'\_labels.csv'), index=None) #save and change the file to csv with original name          print('Successfully converted xml to csv.') |

**2.2 Then type in cmd:**

|  |
| --- |
| python xml\_to\_csv.py |

**#These creates two files in the images directory. One called *test\_labels.csv* and another one called *train\_labels.csv***

****

## 3. Modifly the TFRecords.py

3.1 Before we can transform the newly created files to TFRecords we need to change a few lines in the [*generate\_tfrecords.py*](http://generate_tfrecords.py/) file.

**the**[***generate\_tfrecords.py***](http://generate_tfrecords.py/)**file are located in:**

accounting to where you store the **tensorflow/tensorflow/models/research/object\_detection/**[***generate\_tfrecords.py***](http://generate_tfrecords.py/)

**for example in my case MacOS X: (venv environment )**

**/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/**[***generate\_tfrecords.py***](http://generate_tfrecords.py/)

3.2 Modify or add the row\_label

|  |
| --- |
| # TO-DO replace this with label map  def class\_text\_to\_int(row\_label):      if row\_label == 'Land':         return 1      elif row\_label == 'Machine':         return 2      elif row\_label == 'Water pound':         return 3      else:         None |

**Now the TFRecords can be generated by typing in cmd:**

**3.3 These two commands generate a *train.record* and a *test.record* file which can be used to train our object detector.**

|  |
| --- |
| **macOS/linux:**  python [generate\_tfrecord.py](http://generate_tfrecord.py/) --**csv\_input**=images/train\_labels.csv --**image\_dir**=images/train --**output\_path**=train.record  python [generate\_tfrecord.py](http://generate_tfrecord.py/) --**csv\_input**=images/test\_labels.csv --**image\_dir**=imagestest --**output\_path**=test.record  **Windows:**  python [generate\_tfrecord.py](http://generate_tfrecord.py/) --**csv\_input**=images\train\_labels.csv --**image\_dir**=images\train --**output\_path**=train.record  python [generate\_tfrecord.py](http://generate_tfrecord.py/) --**csv\_input**=images\test\_labels.csv --**image\_dir**=images\test --**output\_path**=test.record |

## 4.Configuring training

The last thing we need to do before training is to create a label map and a training configuration file.

**4.1 Create a folder name as training in object\_detection/**

**4.2 Creating a labelmap.pbtxt**

The label map maps an id to a name. We will put it in a folder called *training(store to training folder)*, which is located in the object\_detection directory. The labelmap for my detector can be seen below.

**4.3 Store the labelmap.pbtxt in object\_detection/training/**

The id number of each item should match the id of specified in the generate\_tfrecord.py file!!!

|  |
| --- |
| item {      id: 1      name: 'Land'  }item {      id: 2      name: 'Machine'  }item {      id: 3      name: 'Water pound'  } |

## 5.Creating a training configuration

Now we need to create a training configuration file. Because as my model of choice I will use **faster\_rcnn\_inception**, **which just like a lot of other models can be downloaded from** [**this**](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md)**page(**<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

**).**

I will start with a sample config ( **faster\_rcnn\_inception\_v2\_pets.config** ), which can be found in the **object\_detection/sample** folder.

**Depend on you model to use the config!**

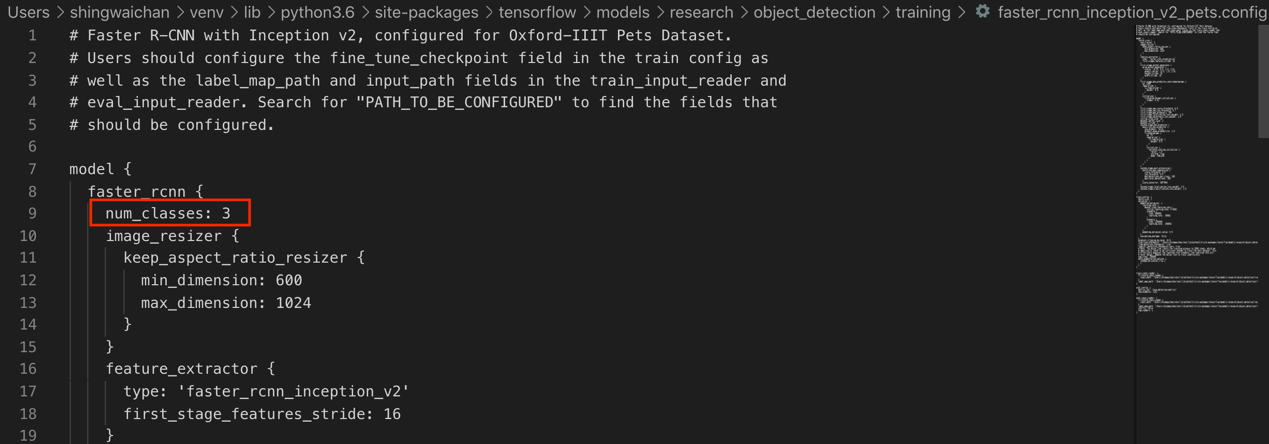
Copy the config to training folder object\_detection/training

Then modify it to change the content

In my case:

**5.1 Line 9: change the number of classes to number of objects you want to detect**

**num\_classes: 3**



**5.2 Line 106: change *fine\_tune\_checkpoint*to the path of the *model.ckpt* file:**

in my case **first time** to train a model of my own.

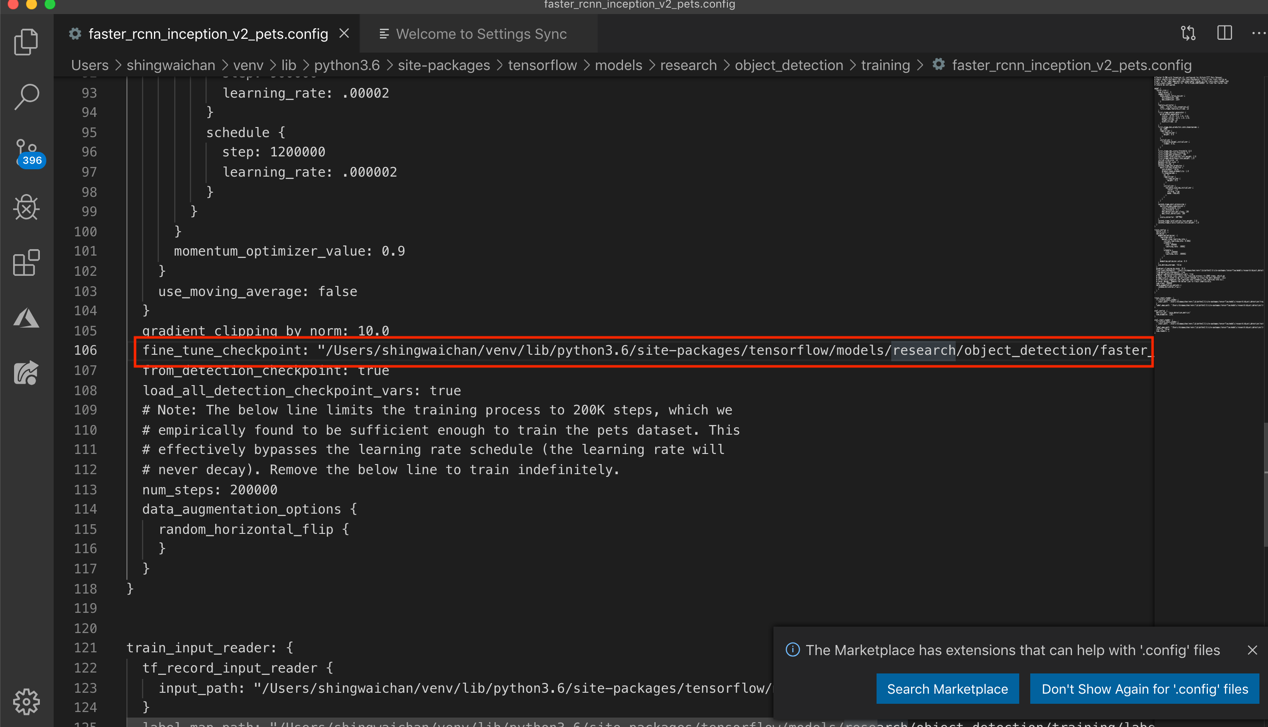
**fine\_tune\_checkpoint: "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt”**

**5.2.1 For retrain the model (first time train)**

**//<your model> type the name of model folder in here**

**//Eg:** **inference\_graph\_2019\_08\_16\_2736x1824**

**fine\_tune\_checkpoint: "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/<your model>/model.ckpt”**

****

**train\_input\_reader:**

**5.4 Line 123: change *input\_path*to the path of the *train.records* file:**

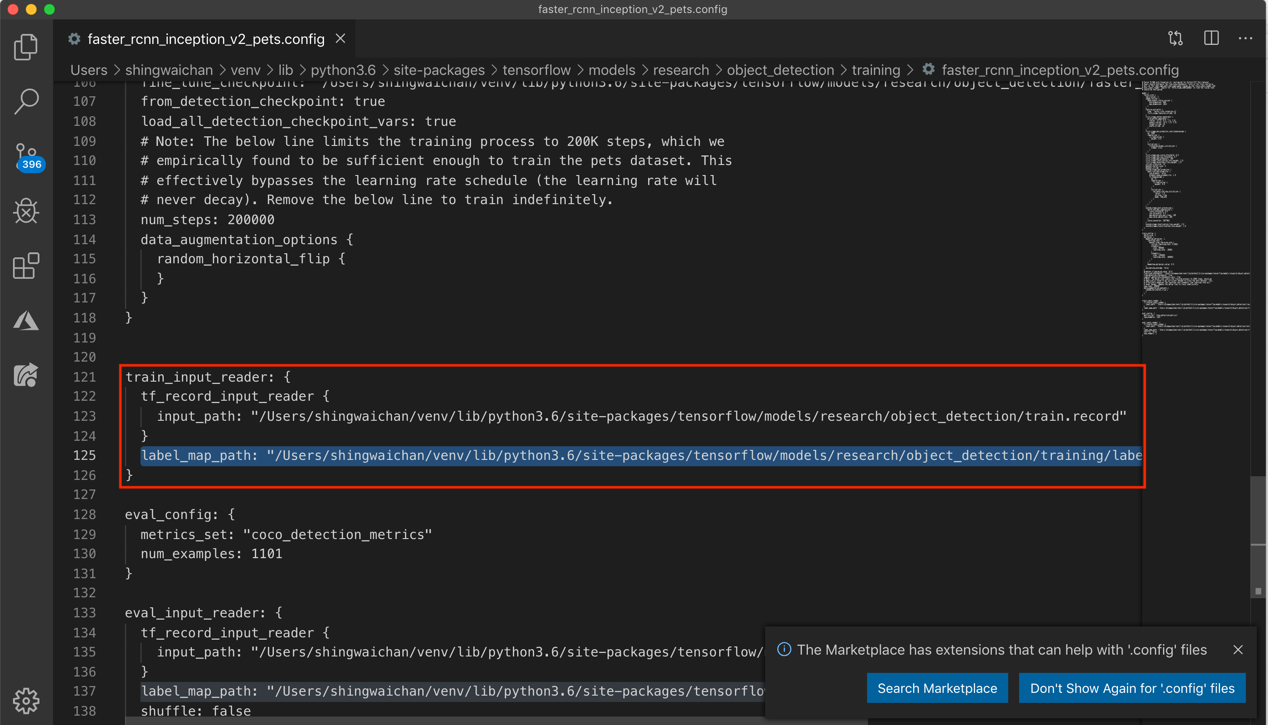
in my case

input\_path: "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/**train.record**"

**5.5 Line 125–137: change *label\_map\_path*to the path of the label map:**

in my case

label\_map\_path**:** "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/training/**labelmap.pbtxt"**

****

**eval\_input\_reader:**

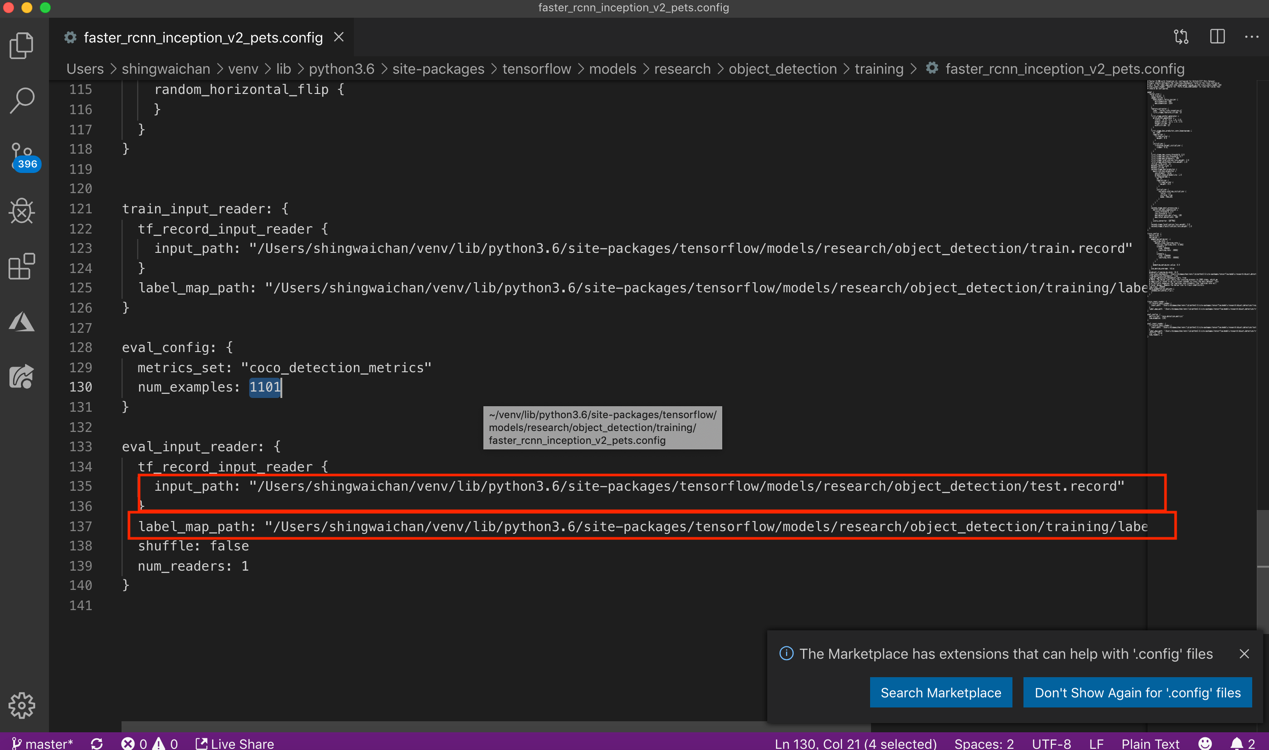
**5.6 Line 135: change*input\_path* to the path of the *test.records* file:**

in my case

input\_path: "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/**test.record**”

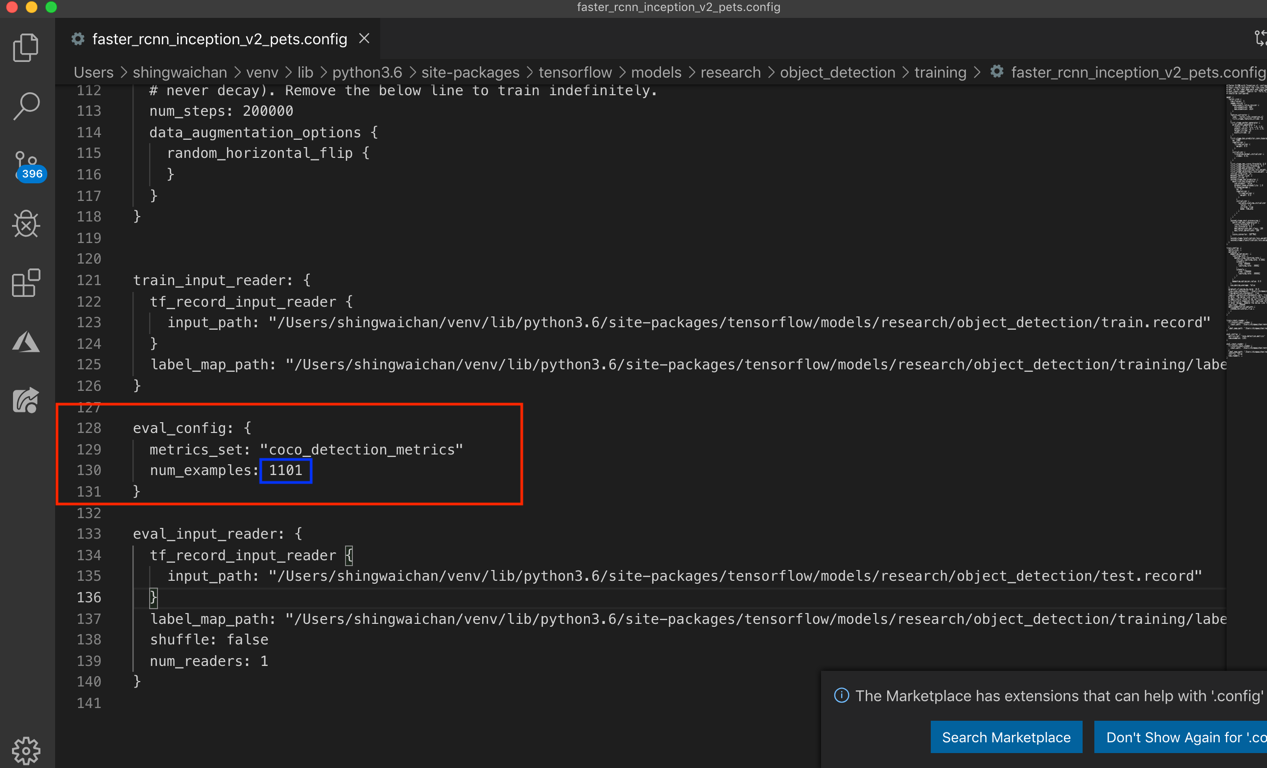
**5.7 Change the labelmap.pbtxt of label\_map\_path in eval\_input\_reader**

label\_map\_path**:** "/Users/shingwaichan/venv/lib/python3.6/site-packages/tensorflow/models/research/object\_detection/training/**labelmap.pbtxt"**



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

**in my case**

****

## 6.Training model

#In the past to train the model we will use the [*train.py*](http://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:

**6.1 Now to train the model we use the model\_main file in the object\_detection folder instead**

--**model\_dir**=**training/ (is the model save path)**

|  |
| --- |
| ***run in cmd:***  python train.py **--logtostderr** --**train\_dir**=**training/** **--pipeline\_config\_path**=**training/faster\_rcnn\_inception\_v2\_pets.config**  ***in tensorflow after v 1.14 :(recommend)***  python model\_main.py **--logtostderr** --**model\_dir**=**training/** **--pipeline\_config\_path**=**training/faster\_rcnn\_inception\_v2\_pets.config** |

***6.2 If everything was setup correctly the training should begin shortly.***

the train log same like:( depends on tensorflow versions)

**Note:**

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 in cmd:

|  |
| --- |
| tensorboard **--logdir=**training |

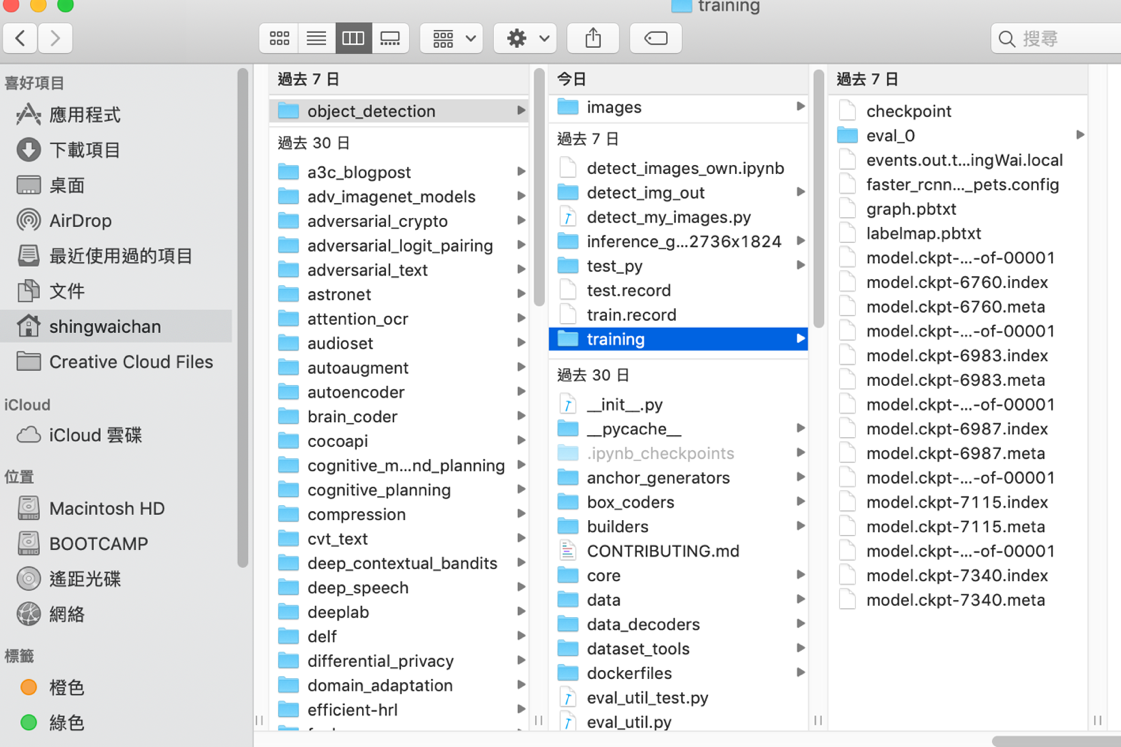
**tensorboard each mAP mean:**

|  |
| --- |
| 'DetectionBoxes\_Precision/mAP': mean average precision over classes averaged over IOU thresholds ranging from .5 to .95 with .05 increments.  'DetectionBoxes\_Precision/mAP@.50IOU': mean average precision at 50% IOU  'DetectionBoxes\_Precision/mAP@.75IOU': mean average precision at 75% IOU  'DetectionBoxes\_Precision/mAP (small)': mean average precision for small objects (area < 32^2 pixels).  'DetectionBoxes\_Precision/mAP (medium)': mean average precision for medium sized objects (32^2 pixels < area < 96^2 pixels).  'DetectionBoxes\_Precision/mAP (large)': mean average precision for large objects (96^2 pixels < area < 10000^2 pixels).  'DetectionBoxes\_Recall/AR@1': average recall with 1 detection.  'DetectionBoxes\_Recall/AR@10': average recall with 10 detections.  'DetectionBoxes\_Recall/AR@100': average recall with 100 detections.  'DetectionBoxes\_Recall/AR@100 (small)': average recall for small objects with 100.  'DetectionBoxes\_Recall/AR@100 (medium)': average recall for medium objects with 100.  'DetectionBoxes\_Recall/AR@100 (large)': average recall for large objects with 100 detections. |

You should train the model until it reaches a satisfying loss. The training process can then be terminated by pressing **Ctrl+C**

**6.3 Note: *利用Tensorboard去觀測data training 的狀態及數據是否合適，當loss值穩定時便可以在 cmd Ctrl+C 停止 training***

***The model with check point is trained in training folder:***

****

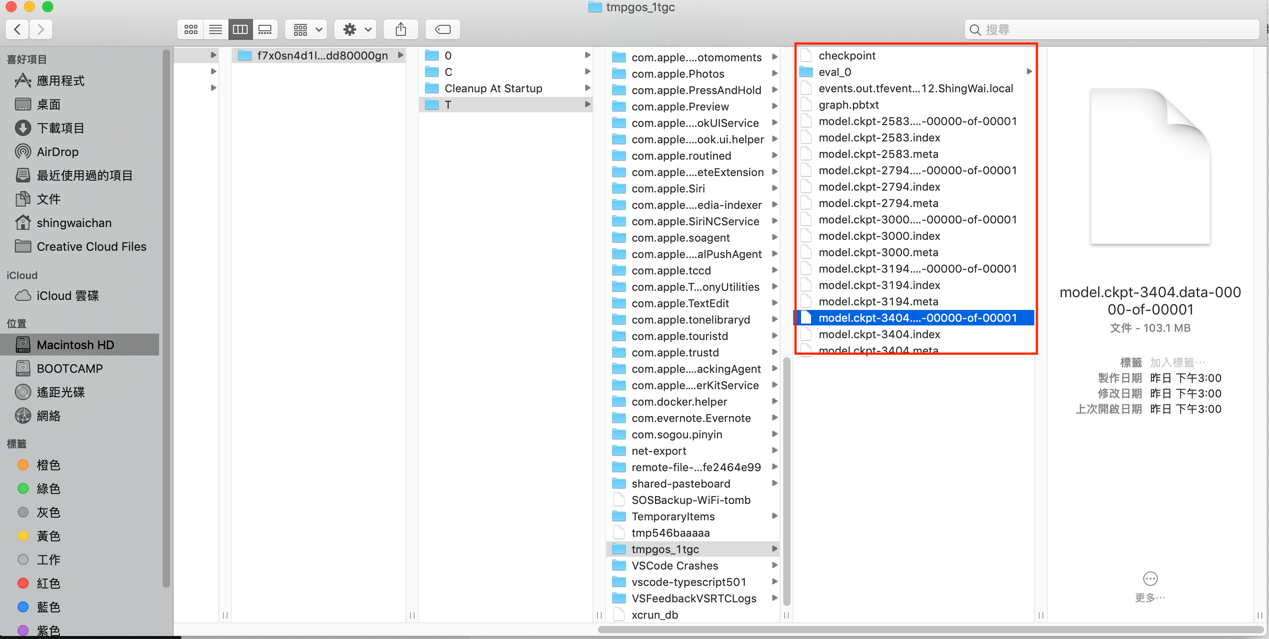
**Note: When the training finish but you can’t find the model file in object\_detection/training folder, please see the log in cmd, it will show the path in here. if the model and check point is success generating to training folder then skip the step**

**6.3.1 in the cmd log, the red text is the location of trained model check point**

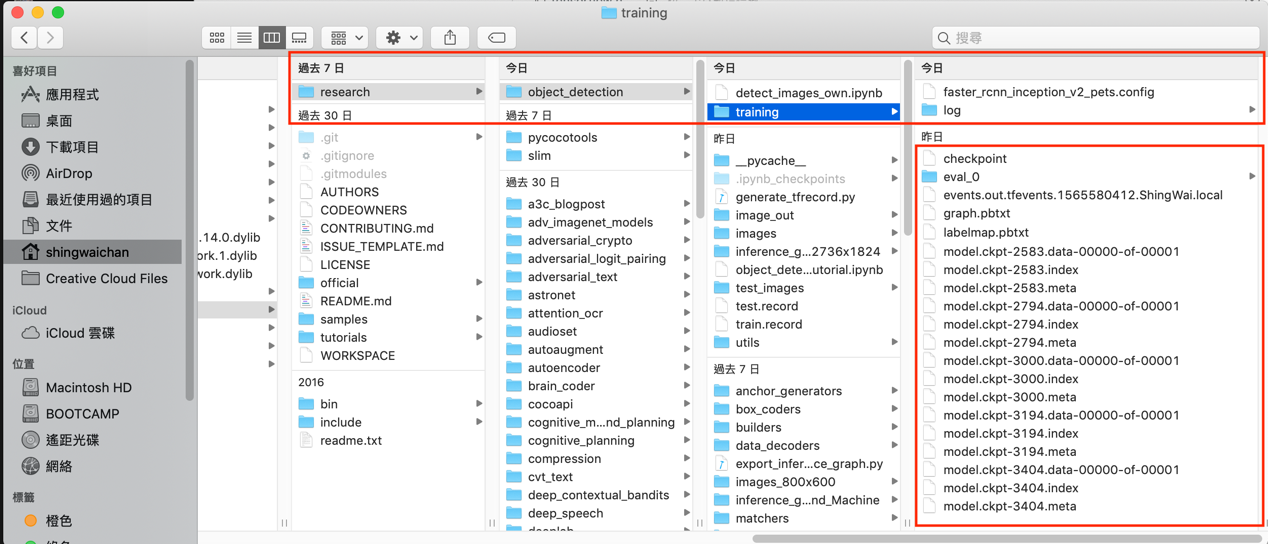
|  |
| --- |
| I0812 15:00:53.476722 4561737152 evaluation.py:275] Finished evaluation at 2019-08-12-15:00:53  I0812 15:00:53.477008 4561737152 estimator.py:2039] Saving dict for global step 3404: DetectionBoxes\_Precision/mAP = 0.09536339, DetectionBoxes\_Precision/mAP (large) = 0.10728626, DetectionBoxes\_Precision/mAP (medium) = 0.052669477, DetectionBoxes\_Precision/mAP (small) = -1.0, DetectionBoxes\_Precision/mAP@.50IOU = 0.31232026, DetectionBoxes\_Precision/mAP@.75IOU = 0.04200174, DetectionBoxes\_Recall/AR@1 = 0.06721491, DetectionBoxes\_Recall/AR@10 = 0.17719299, DetectionBoxes\_Recall/AR@100 = 0.24469298, DetectionBoxes\_Recall/AR@100 (large) = 0.2780263, DetectionBoxes\_Recall/AR@100 (medium) = 0.1125, DetectionBoxes\_Recall/AR@100 (small) = -1.0, Loss/BoxClassifierLoss/classification\_loss = 0.21557835, Loss/BoxClassifierLoss/localization\_loss = 0.18868619, Loss/RPNLoss/localization\_loss = 0.3906744, Loss/RPNLoss/objectness\_loss = 0.44417876, Loss/total\_loss = 1.239118, global\_step = 3404, learning\_rate = 0.0002, loss = 1.239118  I0812 15:00:53.576096 4561737152 estimator.py:2099] **Saving 'checkpoint\_path' summary for global step 3404: /var/folders/75/f7x0sn4d1lx7g8tp6lqmzdd80000gn/T/tmpgos\_1tgc/model.ckpt-3404** |

**if the model is not exit in your training, please open the model folder, Then take all file to your training folder**

**from:**

****

**to:**

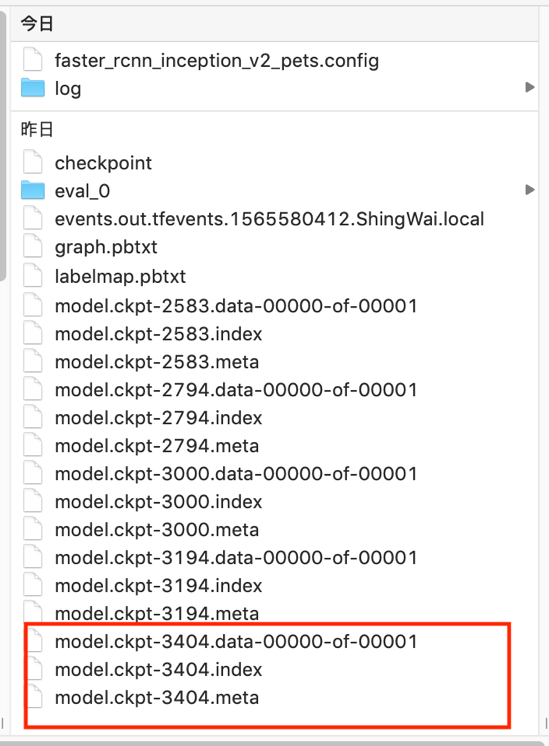
****

## 7.Exporting inference graph

***#Model is finished the training , now generate the model inference grap***

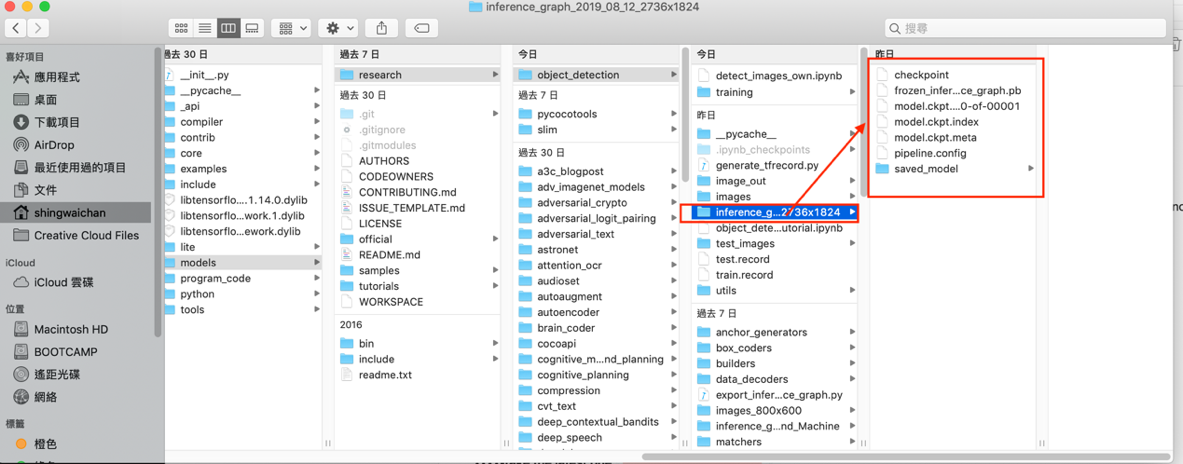
Now that we have a trained model we need to generate an inference graph, which can be used to run the model. For doing so we need to first of find out the highest saved step number. For this, we need to navigate to the **object\_detection/training** directory and look for the **model.ckpt** file with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

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

**xxxx-take the latest one **

|  |
| --- |
| **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** |

***If everything was setup correctly the inference graph model will generate in inference\_graph folder***

******

**The training is finish**

Now you can add the model to your project

**Reference:**

**tutorial:**

**Tensorflow Object Detection Tutorial #1 - Installation**

<https://www.youtube.com/watch?v=wdufj-pjE5c&t=318s>

**Tensorflow Object Detection Tutorial #2 - Live Object Detection**

<https://www.youtube.com/watch?v=NV1g0DYb_vs>

**Tensorflow Object Detection Tutorial #3 - Create your own object detector**

<https://www.youtube.com/watch?time_continue=1054&v=HjiBbChYRDw>

**Tensorflow Object Detection API Tutorial #4 - Building a Surveillance System**

<https://www.youtube.com/watch?v=8CRFChU3U-U>

**Creating your own object detector ( Im follow this tutorial )**

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

**Github tutorial from:**

<https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>

**Tensorflow installation**

<https://www.tensorflow.org/install>