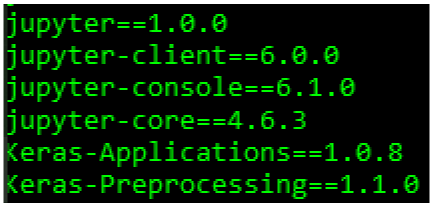
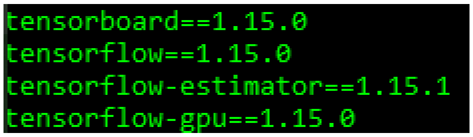
System Specifics:

- Environment variables
  - Python 3.5
  - Models\research
  - Models\research\object_detection
  - Models\research\slim
  - Opencv-4.1.1
- Environment used
  - Anaconda command prompt



```
jupyter==1.0.0
jupyter-client==6.0.0
jupyter-console==6.1.0
jupyter-core==4.6.3
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
```



```
tensorboard==1.15.0
tensorflow==1.15.0
tensorflow-estimator==1.15.1
tensorflow-gpu==1.15.0
```

Using my data and trained model:

- Should just have to change a bunch of paths and maybe download some packages
- The following paths needs to be adjusted:
  - In faster_rcnn_inception_v2_pets.config (raccoon-dataset_master/training)
    - Line 106, 123, 135, 125, 137
  - Object_detection_tutorial.ipynb (model/research/object_detection)
    - Ln [4]
      - PATH_TO_LABELS
    - Ln[8]
      - PATH_TO_TEST_IMAGES_DIR
- Now skip to the last step and run object_detection_tutorial.ipynb

Using your own data:

1. **Take the images**

   -Images need to be jpegs in order to be used by the program
   -Size is also important when it comes to training, the included ImageResize.py file will resize any image using the following command:

   ```
   python ImageResize.py -d *path to directory of images* -s 800 600
   ```

   -Divide the images into two folders, test and train. The number of images in test should be less than or equal to the number of images in train.

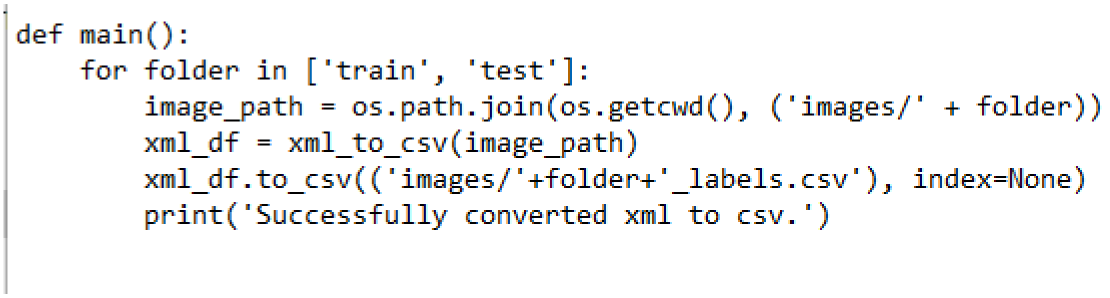
2. **Label the images using Labelimg**

-GitHub: https://github.com/tzutalin/labelImg
-Download Link: https://tzutalin.github.io/labelImg/
-Using Labelimg:

- -Select “Open Dir” and open train/test folder
- -Select “Create RectBox” and draw bounding box
- -Click “Save” and save the .xml (save it right there in the train/test folder)
- -Use “Next Image” arrow to go through whole directory

3. Generate TFRecords

-Navigate to the raccoon_dataset-master folder (original GitHub: https://github.com/datitran/raccoon_dataset)
-Within the xml_to_csv.py file change the path here in the main function to the folder that holds the test and train folders

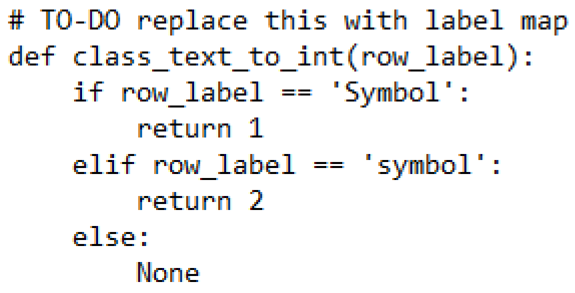


```
def main():
    for folder in ['train', 'test']:
        image_path = os.path.join(os.getcwd(), ('images/' + folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/'+folder+'_labels.csv'), index=None)
        print('Successfully converted xml to csv.')
```

-Run the xml_to_csv.py file using the following command:

```
python xml_to_csv.py
```

-After running that command, there will be two files created called test_labels.csv and train_labels.csv. If those two files are empty, there is an issue with the path you input.
-Now open the generate_tfrecords.py file and change the row_label section based on what you want to detect



```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'Symbol':
        return 1
    elif row_label == 'symbol':
        return 2
    else:
        None
```

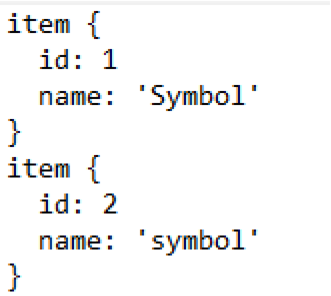
-Now the command to run generate_tfrecords.py is:

```
python generate_tfrecord.py --csv_input=*path to train_labels.csv* --image_dir=*path to train images folder* --output_path=train.record
```

```
python generate_tfrecord.py --csv_input=*path to test_labels.csv* --image_dir=*path to test images folder* --output_path=test.record
```

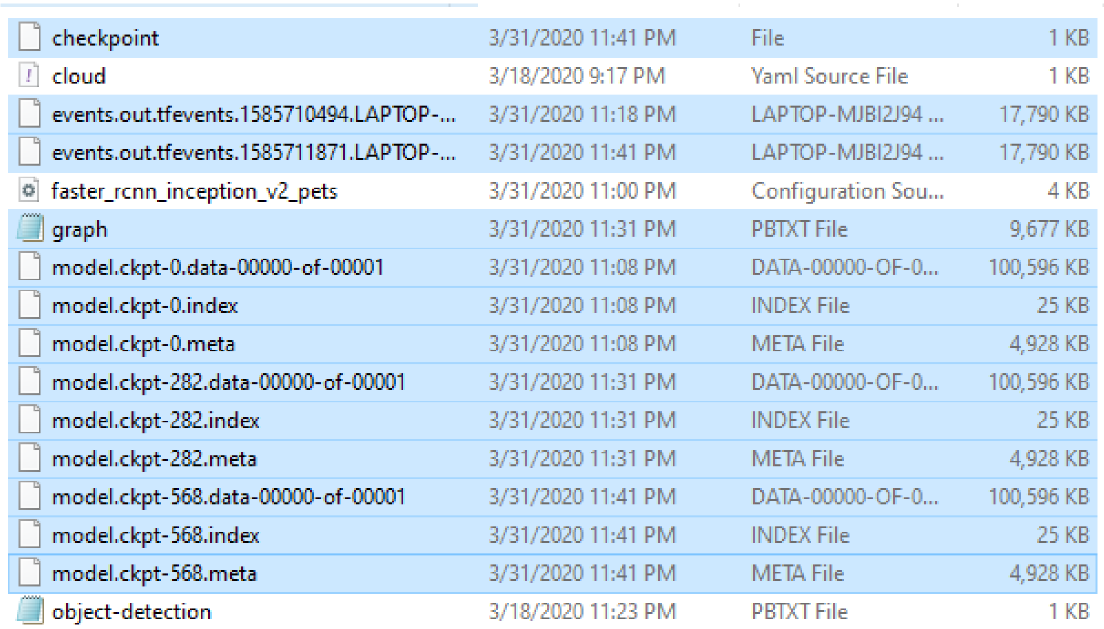
-The tfrecords are now in train.record and test.record

4. **Create Your Labelmap**
   -In the training folder (raccoon_dataset-master/training) there is a PBTXT file called object_detection which is the labelmap. You can add or remove items, just make sure the id number matches the number returned in the tf_record program.



```
item {
  id: 1
  name: 'Symbol'
}
item {
  id: 2
  name: 'symbol'
}
```

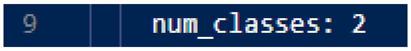
5. **Create Training Config**
   -The model (faster_rcnn_inception_v2_pets.config) is already placed in the training folder. If you want to train the model on your own images, make sure to delete the files I added. The files to delete would be all those highlighted below:



checkpoint	3/31/2020 11:41 PM	File	1 KB
cloud	3/18/2020 9:17 PM	Yaml Source File	1 KB
events.out.tfevents.1585710494.LAPTOP-...	3/31/2020 11:18 PM	LAPTOP-MJBI2J94 ...	17,790 KB
events.out.tfevents.1585711871.LAPTOP-...	3/31/2020 11:41 PM	LAPTOP-MJBI2J94 ...	17,790 KB
faster_rcnn_inception_v2_pets	3/31/2020 11:00 PM	Configuration Sou...	4 KB
graph	3/31/2020 11:31 PM	PBTXT File	9,677 KB
model.ckpt-0.data-00000-of-00001	3/31/2020 11:08 PM	DATA-00000-OF-0...	100,596 KB
model.ckpt-0.index	3/31/2020 11:08 PM	INDEX File	25 KB
model.ckpt-0.meta	3/31/2020 11:08 PM	META File	4,928 KB
model.ckpt-282.data-00000-of-00001	3/31/2020 11:31 PM	DATA-00000-OF-0...	100,596 KB
model.ckpt-282.index	3/31/2020 11:31 PM	INDEX File	25 KB
model.ckpt-282.meta	3/31/2020 11:31 PM	META File	4,928 KB
model.ckpt-568.data-00000-of-00001	3/31/2020 11:41 PM	DATA-00000-OF-0...	100,596 KB
model.ckpt-568.index	3/31/2020 11:41 PM	INDEX File	25 KB
model.ckpt-568.meta	3/31/2020 11:41 PM	META File	4,928 KB
object-detection	3/18/2020 11:23 PM	PBTXT File	1 KB

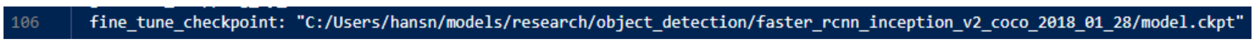
-Open the faster_rcnn_inception_v2_pets.config file

-On Line 9: Change the number of classes to the number of objects you want to detect



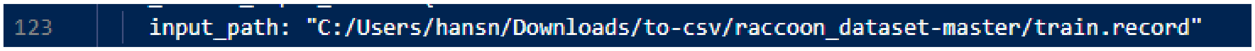
```
9    num_classes: 2
```

-On Line 106: Change fine_tune_checkpoint to the path of the model.ckpt file:



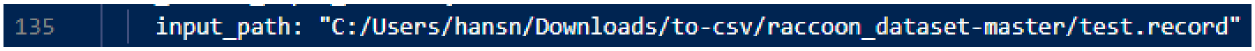
```
106  fine_tune_checkpoint: "C:/Users/hansn/models/research/object_detection/faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt"
```

-On Line 123: Change input_path to the path of train.records



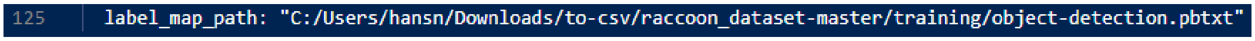
```
123    input_path: "C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/train.record"
```

-On Line 135: Change input_path to the path of test.records

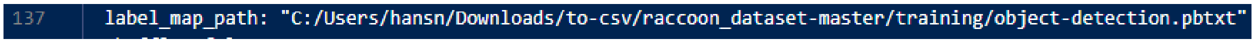


```
135    input_path: "C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/test.record"
```

-On Line 125 and 137: Change label_map_path to the path of the label map

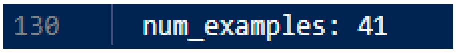


```
125  label_map_path: "C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/training/object-detection.pbtxt"
```



```
137  label_map_path: "C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/training/object-detection.pbtxt"
```

-On Line 130: Change num_example to the number of images in your test folder



```
130  num_examples: 41
```

6. Train

-Run the model_main.py file in the model/research/object_detection folder:

python model_main.py --logtostderr --model_dir=*path to training folder in raccoon-dataset_master* --pipline_config_path=*path to training folder*/faster_rcnn_inception_v2_pets.config

7. Get Inference Graph

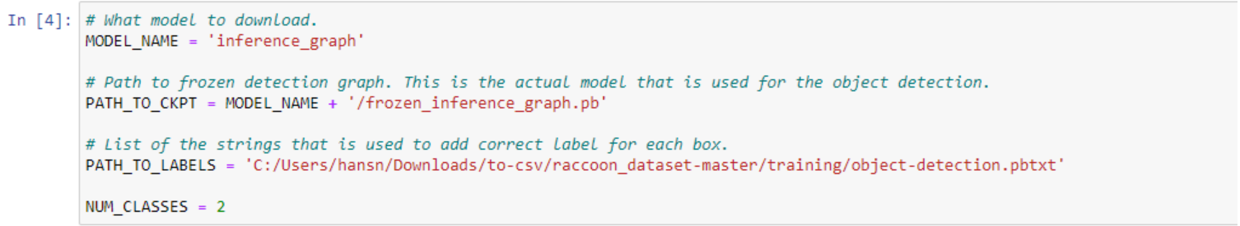
-Export the inference graph using export_inference_graph.py and the following command:

python export_inference_graphy.py --input_type image_tensor --pipeline_config_path *path to training folder*/faster_rcnn_inception_v2_pets.config --trained_checkpoint_prefix *path to training folder*/model.ckpt--XXXX --output_directory inference_graph

-The XXXX represents the highest number. In the training folder after the model is trained, so just use the highest indexed checkpoint in that folder.

8. **Test**

-In the model/research/object_detection folder there is an object_detection_tutorial IPYNB file. In that file, you have to change a couple things



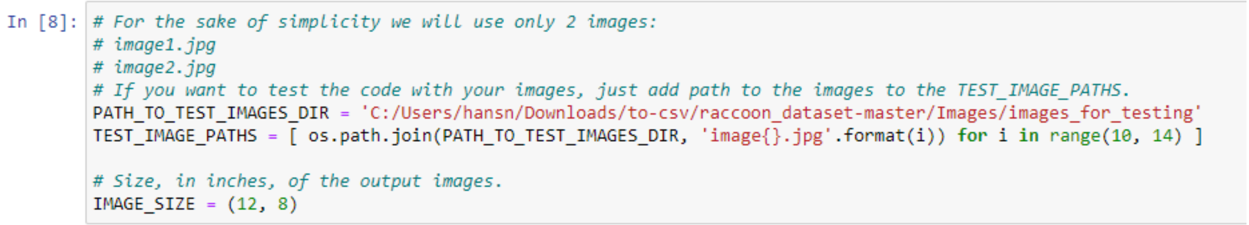
```
In [4]: # What model to download.
        MODEL_NAME = 'inference_graph'

        # Path to frozen detection graph. This is the actual model that is used for the object detection.
        PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

        # List of the strings that is used to add correct label for each box.
        PATH_TO_LABELS = 'C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/training/object-detection.pbtxt'

        NUM_CLASSES = 2
```

MODEL_NAME = the folder that you stored the inference graph in
PATH_TO_CKPT should stay the same
PATH_to_LABELS is the path to the labelmap
NUM_CLASSES is the number of objects that we are trying to detect



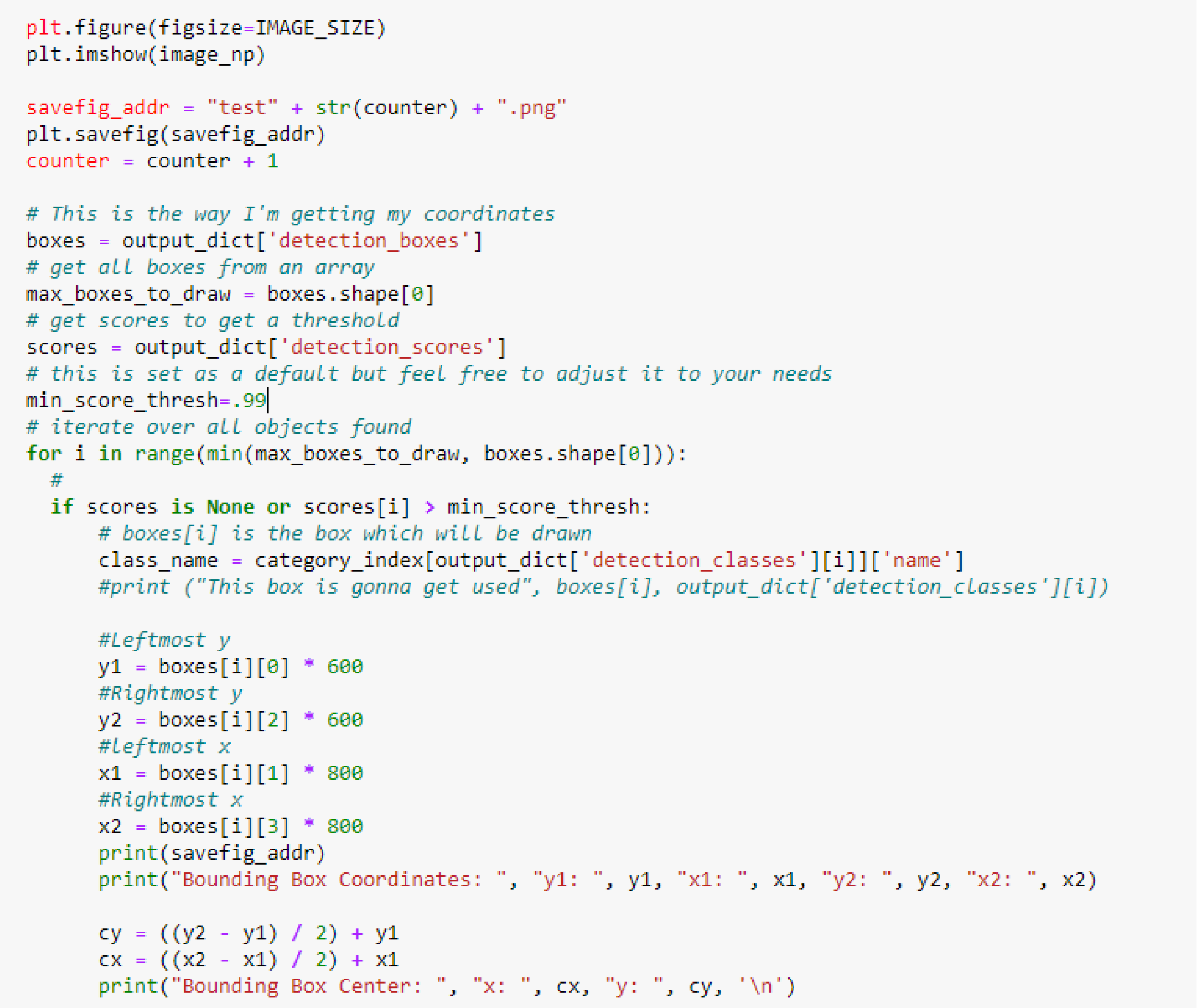
```
In [8]: # For the sake of simplicity we will use only 2 images:
        # image1.jpg
        # image2.jpg
        # If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
        PATH_TO_TEST_IMAGES_DIR = 'C:/Users/hansn/Downloads/to-csv/raccoon_dataset-master/Images/images_for_testing'
        TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(10, 14) ]

        # Size, in inches, of the output images.
        IMAGE_SIZE = (12, 8)
```

Change PATH_TO_TEST_IMAGES to the path of wherever images used for testing are (do not use the test folder you used during training)

Change the range(10-14) to the range of whatever your images are labeled

In In[10]:



```
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)

savefig_addr = "test" + str(counter) + ".png"
plt.savefig(savefig_addr)
counter = counter + 1

# This is the way I'm getting my coordinates
boxes = output_dict['detection_boxes']
# get all boxes from an array
max_boxes_to_draw = boxes.shape[0]
# get scores to get a threshold
scores = output_dict['detection_scores']
# this is set as a default but feel free to adjust it to your needs
min_score_thresh=.99
# iterate over all objects found
for i in range(min(max_boxes_to_draw, boxes.shape[0])):
  #
  if scores is None or scores[i] > min_score_thresh:
      # boxes[i] is the box which will be drawn
      class_name = category_index[output_dict['detection_classes'][i]]['name']
      #print ("This box is gonna get used", boxes[i], output_dict['detection_classes'][i])

      #Leftmost y
      y1 = boxes[i][0] * 600
      #Rightmost y
      y2 = boxes[i][2] * 600
      #leftmost x
      x1 = boxes[i][1] * 800
      #Rightmost x
      x2 = boxes[i][3] * 800
      print(savefig_addr)
      print("Bounding Box Coordinates: ", "y1: ", y1, "x1: ", x1, "y2: ", y2, "x2: ", x2)

      cy = ((y2 - y1) / 2) + y1
      cx = ((x2 - x1) / 2) + x1
      print("Bounding Box Center: ", "x: ", cx, "y: ", cy, '\n')
```

Change the numbers being multiplied for y1, y2, x1, x2 to the dimensions of your image
Change min_score_thresh to a higher or lower number to get only the best fitting boxes in output

If you have any further questions, please reach out to me at ethan.hansen@uconn.edu or refer to the tutorial that I used at the following link: https://towardsdatascience.com/creating-your-own-object-detector-ad69dda69c85