Custom Faster RCNN using Tensorflow Object Detection API

Vijendra Singh Follow
Oct 2, 2018 · 6 min read

A sample project for building Faster RCNN model to detect the custom object using Tensorflow object detection API. I have tried to make this post as explanatory as possible. In case you are stuck at any step, please comment for support. This post will be pointing you to the project Github repository at every step. You could found the project Github repository HERE

Folder Structure

- Tensorflow_API-Custom_object_detection
  - pre_trained_models
    - *downloaded files for the choosen pre-trained model will come here*
  - dataset
    - Annotations
      - *Annotations for your training images will come here*
    - JPEGImages
      - *all of your images for training will come here*
    - testImages
      - *all your images for testing will come here*
    - lable.pbtxt
    - train.record
  - IG
    - *inference graph of the trained model will be saved here*
  - CP
    - *checkpoints of the trained model will be saved here*
  - eval.ipynb
  - train.ipynb
  - *config file for the choosen model*

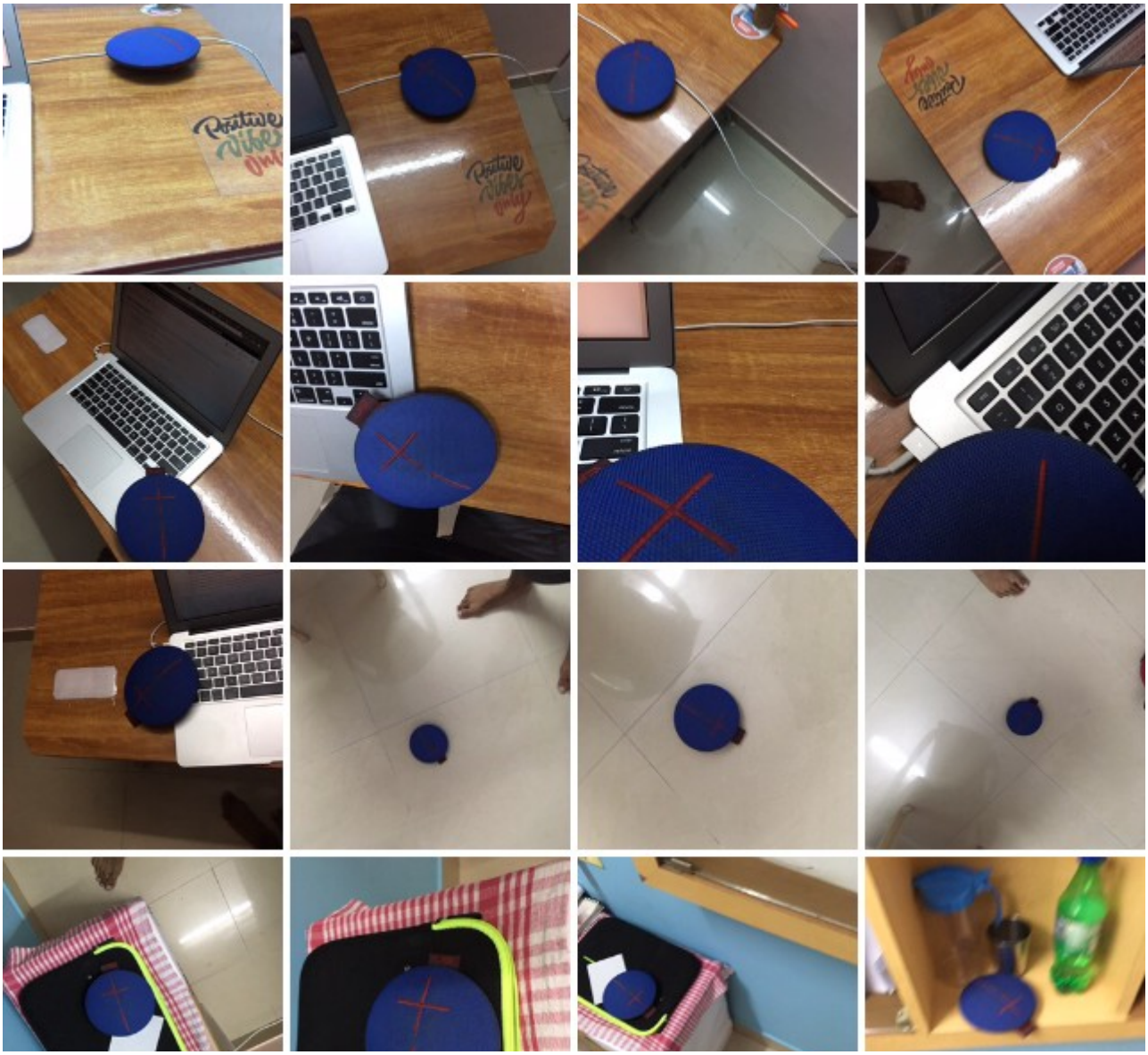
Steps

Create folders

Create the folders following the structure given above (If you want, you could use a different name for any of the folders or files)

Prepare train and test images

The Github repository contains train and test images for detection of “UE Roll” blue bluetooth speaker but I will highly recommend you to create your own dataset. Pick up an object you want to detect and take some pics of it with varying backgrounds, angles and distances. Some of the sample images used in this project are given below:

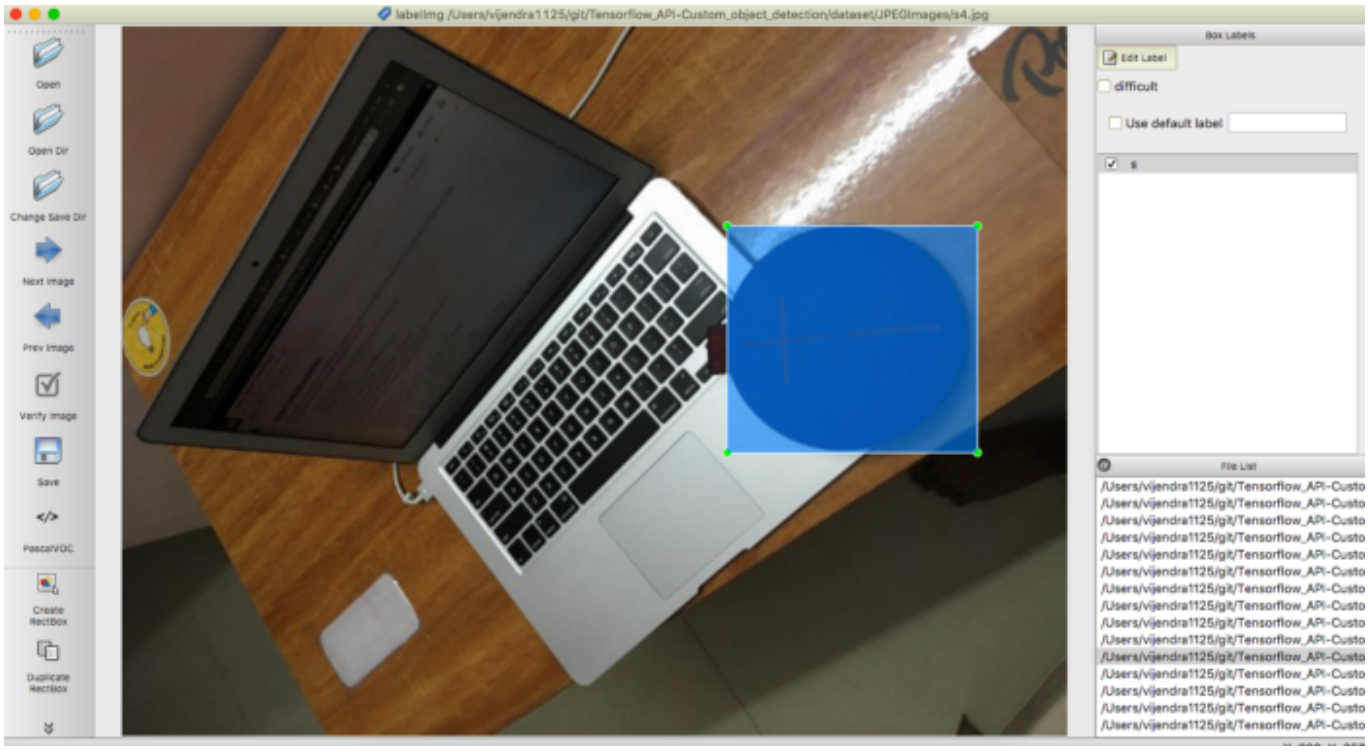


Training Images

Once you have captured images, transfer it to your PC and resize it to a smaller size (given images have the size of 605 x 454) so that your training will go smoothly without running out of memory. Now rename (for better referencing later) and divide the captured images into two chunks, one chunk for training(80%) and other for testing(20%). Finally, move training images into *JPEGImages* folder and testing images into *testImages* folder.

Label the data

Now its time to label the training data. We will be doing it using the labelImg library. To download this library along with its dependencies go to THIS LINK. Once you have the labelImg library downloaded on your PC, run lableImg.py. Select *JPEGImages* directory by clicking on *Open Dir* and change the save directory to *Annotations* by clicking on *Change Save Dir*. Now all you need to do is to draw rectangles around the object you are planning to detect. You will need to click on *Create RectBox* and then you will get the cursor to label the objects. After drawing rectangles around objects, give the name for the label and save it so that Annotations will get saved as a .xml file in *Annotations* folder.



Labeling in labeleImg

Setup Tensorflow models repository

Now it's time when we will start using Tensorflow object detection API so go ahead and clone it using the following command

```
git clone https://github.com/tensorflow/models.git
```

Once you have cloned the repository, change your present working directory to models/reserarch/ and add it to your python path. If you want to add it permanently then you will have to make the change in your .bashrc file or you could add it temporarily for current session using the following command:

```
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

You also need to run following command in order to get rid of the *string_int_label_map_pb2* issue (more details HERE)

```
protoc object_detection/protos/*.proto --python_out=.
```

Now your Environment is all set to use Tensorlow object detection API

Convert the data to Tensorflow record format

In order to use Tensorflow API, you need to feed data in Tensorflow record format. Thankfully Tensorflow gives python script to convert Pascal VOC format dataset to Tensorflow record format. Path for the file I mentioned in last line is *models/research/object_detection/dataset_tools/create_pascal_tf_record.py* Now you have two options, either follow Pascal VOC dataset format or modify the Tesorflow script as per your need. I modified the script and I have placed the same in the project repository inside the folder named as *extra*. All you need to do is to take this script and replace the original script with this. If you do so, you don't need to follow Pascal VOC format. Now,

one last remaining task is to create a label map file, in the project repository, it is named as label.pbtxt. Check *label.pbtxt* given in the repository to understand the format, its pretty simple (Note: name of the label should be same as what you had given while labeling object using the labelImg). Now it time to create record file. From models/research as present working directory run the following command to create Tensorflow record:

```
python object_detection/dataset_tools/create_pascal_tf_record.py --
data_dir=<path_to_your_dataset_directory> --annotations_dir=
<name_of_annotations_directory> --output_path=
<path_where_you_want_record_file_to_be_saved> --label_map_path=
<path_of_label_map_file>
```

For more help run the following command:

```
python object_detection/dataset_tools/create_pascal_tf_record.py -h
```

An example will be:

```
Python object_detection/dataset_tools/create_pascal_tf_record.py --
data_dir=/Users/vijendra1125/Documents/tensorflow/object_detection/sp
eaker_detection/dataset --annotations_dir=Annotations --
output_path=/Users/vijendra1125/Documents/tensorflow/object_detection
/speaker_detection/dataset/train.record --
label_map_path=/Users/vijendra1125/Documents/tensorflow/object_detect
ion/speaker_detection/dataset/label.pbtxt
```

Training

Now that we have data in the right format to feed, we could go ahead with training our model. The first thing you need to do is to select the pre-trained model you would like to use. You could check and download a pre-trained model from Tensorflow detection model zoo Github page. Once downloaded, extract all files to the folder you had created for saving the pre-trained model files. Next you need to copy *models/research/object_detection/sample/configs/<your_model_name.config>* and paste it in the project repo. You need to configure 5 paths in this file. Just open this file and

search for PATH_TO_BE_CONFIGURED and replace it with the required path. I used the pre-trained *faster_rcnn_resnet101_coco model* and I have added the modified config file (along with PATH_TO_BE_CONFIGURED as comment above lines which has been modified) for same in this repo. You could also play with other hyperparameters if you want. Now you are all set to train your model, just run the following command with models/research as present working directory

```
python object_detection/legacy/train.py --train_dir=<path_to_the
folder_for_saving_checkpoints> --pipeline_config_path=
<path_to_config_file>
```

An example will be

```
python object_detection/legacy/train.py --
train_dir=/Users/vijendra1125/Documents/tensorflow/object_detection/s
peaker_detection/CP --
pipeline_config_path=/Users/vijendra1125/Documents/tensorflow/object_
detection/speaker_detection/faster_rcnn_resnet101_coco.config
```

Let it train till loss will be below 0.1 or even lesser. once you see that loss is as low as you want then give keyboard interrupt. Checkpoints will be saved in CP folder. Now its time to generate inference graph from saved checkpoints

```
python object_detection/export_inference_graph.py --
input_type=image_tensor --pipeline_config_path=<path_to_config_file>
--trained_checkpoint_prefix=<path to saved checkpoint> --
output_directory=<path_to_the_folder_for_saving_inference_graph>
```

An example will be

```
python object_detection/export_inference_graph.py --
input_type=image_tensor --
pipeline_config_path=/Users/vijendra1125/Documents/tensorflow/object_
detection/speaker_detection/faster_rcnn_resnet101_coco.config --
trained_checkpoint_prefix=/Users/vijendra1125/Documents/tensorflow/ob
ject_detection/speaker_detection/CP/model.ckpt-1691 --
```

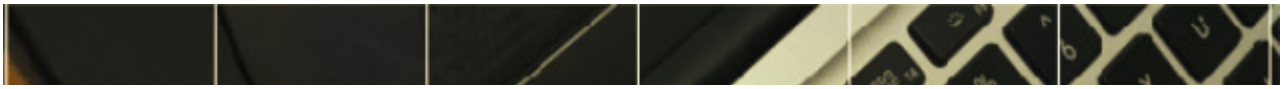
```
output_directory=/Users/vijendra1125/Documents/tensorflow/object_dete
ction/speaker_detection/IG
```

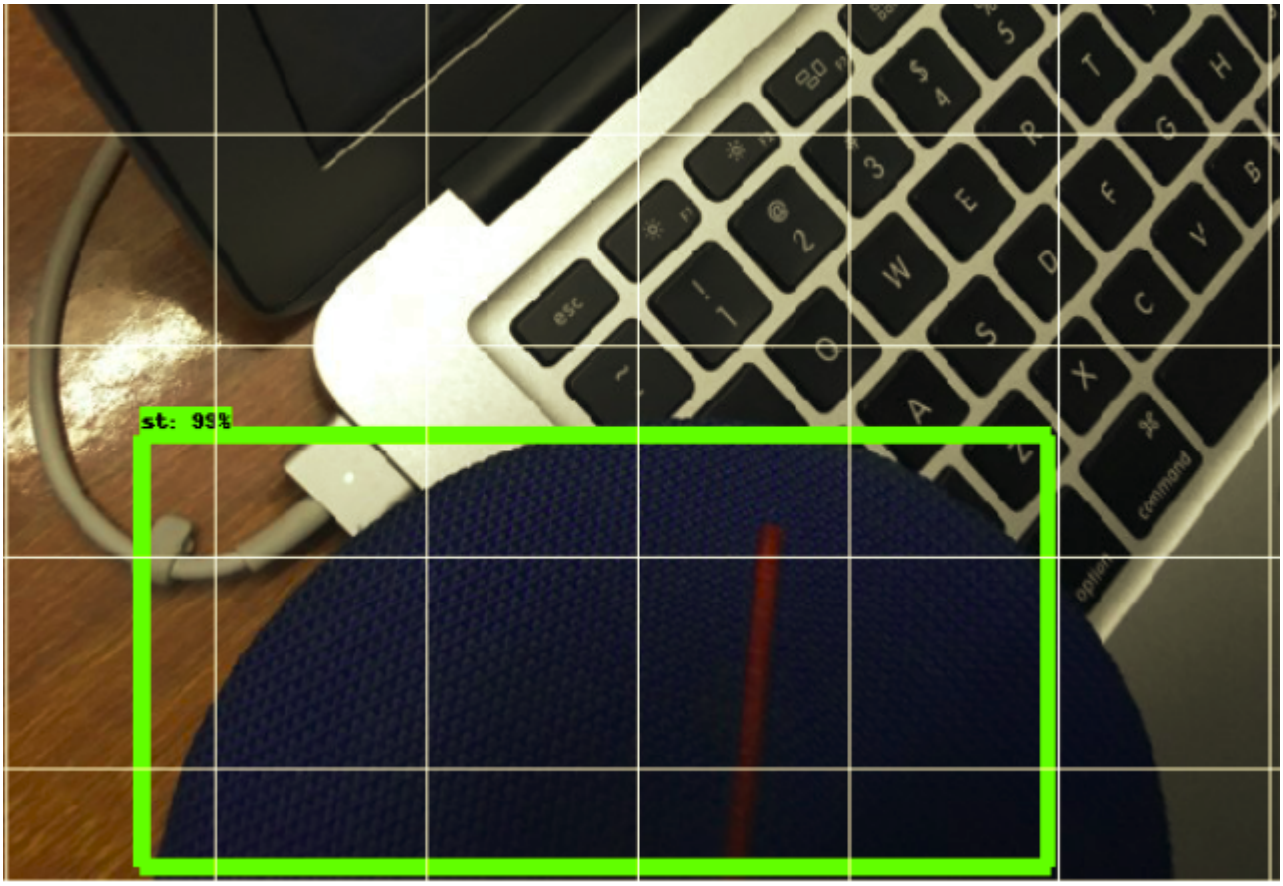
Bonus: If you want to train your model using Google Colab then check out the *train.ipynb* file given in Github repository

Test the trained model

Finally, it's time to check the result of all the hard work you did. All you need to do is to copy model/research/object_detection/object_detection_tutorial.ipynb and modify it to work with you inference graph. A modified file is already given as *eval.ipynb* in the Github repository, you just need to change the path to frozen detection graph, the number of classes and the number of images you have given as test image. Below is the result of the model trained for detection of "UE Roll" blue Bluetooth speaker.

Detection in the test image 1





Detection in the test image 2

Detection in the test image 3

Model is trained with very less number of image, performance could be improved if more training images will be used.

TensorFlow Object Detection Deep Learning