**Faster RCNN Documentation**

**Overview:**

# Data collection

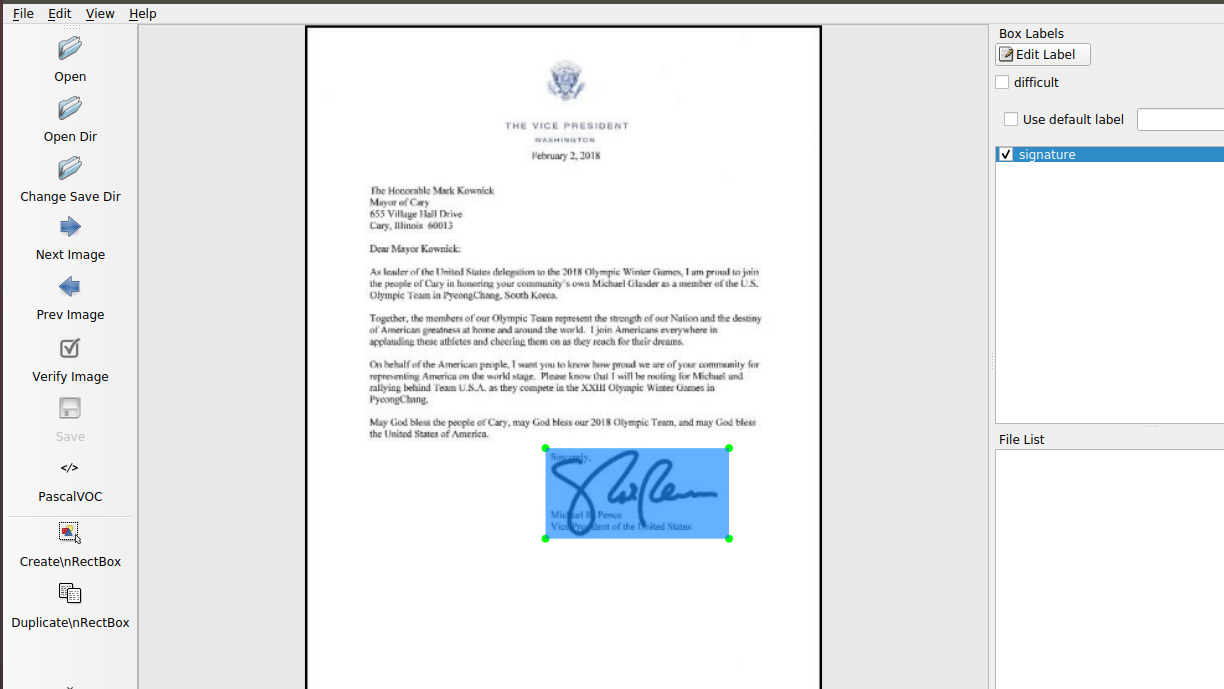
1. Annotation
2. Creating mapping files
3. Creating Tensorflow records
4. .xml to .csv conversion
5. .csv to .tfrecord conversion
6. Configuring the Training Pipeline
7. Training
8. Prediction

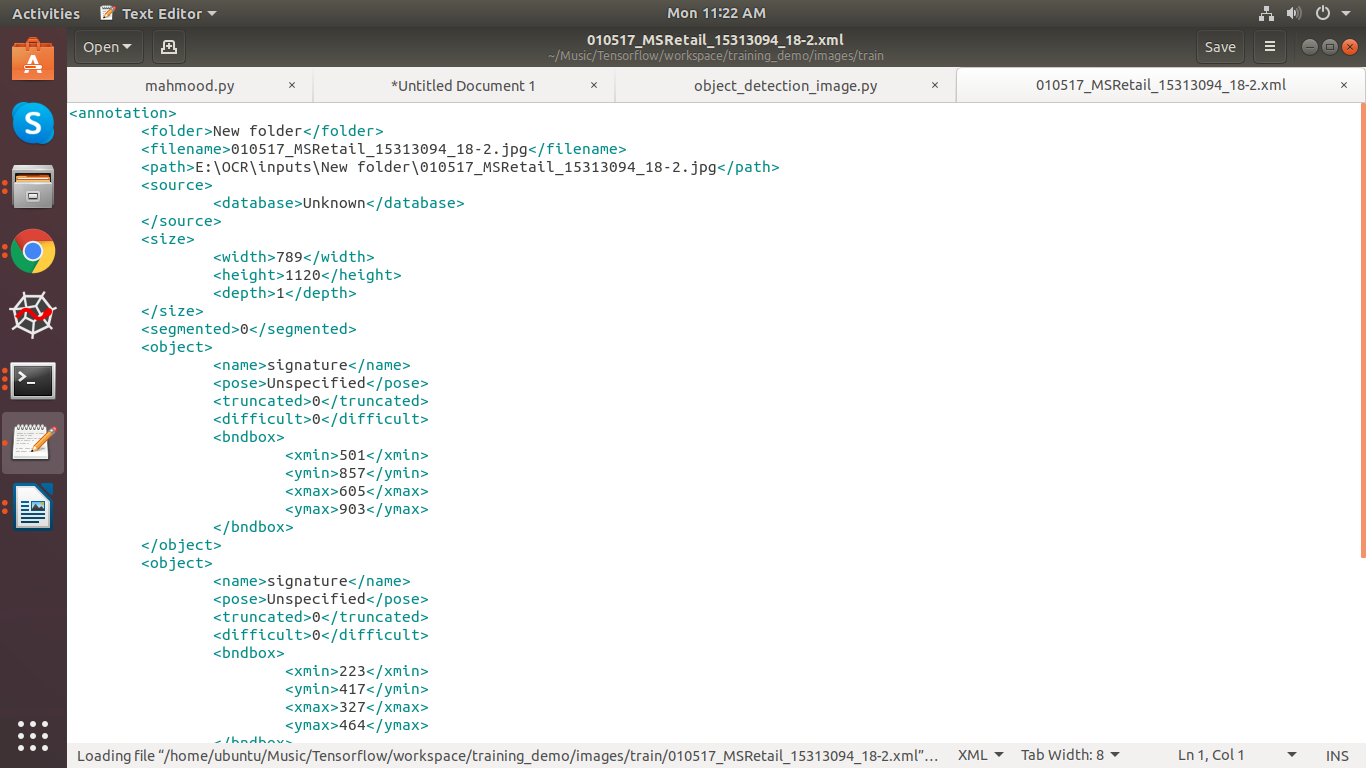
## Data collection

* 1. Split and convert .tiff, .png, .jpeg file into .jpg file
  2. Image Augmentation for increasing dataset

## Annotation

* 1. Install Annotation tool - for multi class
     1. Conda environment
        + Python 3.6
        + pip install labelImg
  2. Boundary box on image
* Launch by python labelImg.py
* Load image and start annotating with the new classes
* An xml file with will be generated when you save the annotated image.





1. Creating Label Map:

TensorFlow requires a label map, which namely maps each of the used labels to an integer values. This label map is used both by the training and detection processes.

item {

id: 1

name: 'signature'

}

Label map files have the extention .pbtxt and should be placed inside the training\_demo\annotations folder.

1. Creating Tensorflow Records:

We have generated our annotations and split our dataset into the desired training and testing subsets, it is time to convert our annotations into the so called TFRecord format.

There are two steps in doing so:

* Converting the individual \*.xml files to a unified \*.csv file for each dataset.
* Converting the \*.csv files of each dataset to \*.record files (TFRecord format).
* create a directory where we can store some scripts. Under the TensorFlow folder, create a new folder TensorFlow\scripts, which we can use to store some useful scripts. To make things even tidier, let’s create a new folder TensorFlow\scripts\preprocessing, where we shall store scripts that we can use to preprocess our training inputs. Below is out TensorFlow directory tree structure, up to now



1. Converting .xml to .csv:

* write a script that iterates through all \*.xml files in the training\_demo\images\train and training\_demo\images\test folders, and generates a \*.csv for each of the two.
* Create a new file with name xml\_to\_csv.py under TensorFlow\scripts\preprocessing, open it, paste the above code inside it and save.
* Install the pandas package.
* Finally, cd into TensorFlow\scripts\preprocessing and run:

*# Create train data:*

python xml\_to\_csv.py -i [PATH\_TO\_IMAGES\_FOLDER]/train -o [PATH\_TO\_ANNOTATIONS\_FOLDER]/train\_labels.csv

*# Create test data:*

python xml\_to\_csv.py -i [PATH\_TO\_IMAGES\_FOLDER]/test -o [PATH\_TO\_ANNOTATIONS\_FOLDER]/test\_labels.csv

1. Converting .csv to .records:

* we have obtained our \*.csv annotation files, we will need to convert them into TFRecords.

# Create train data:

python generate\_tfrecord.py --label=<LABEL> --csv\_input=<PATH\_TO\_ANNOTATIONS\_FOLDER>/train\_labels.csv

--img\_path=<PATH\_TO\_IMAGES\_FOLDER>/train --output\_path=<PATH\_TO\_ANNOTATIONS\_FOLDER>/train.record

# Create test data:

python generate\_tfrecord.py --label=<LABEL> --csv\_input=<PATH\_TO\_ANNOTATIONS\_FOLDER>/test\_labels.csv

--img\_path=<PATH\_TO\_IMAGES\_FOLDER>/test

--output\_path=<PATH\_TO\_ANNOTATIONS\_FOLDER>/test.record

1. Configuring the Training pipeline:

The model we shall be using in our examples is the Faster\_rcnn\_inception\_v2\_coco model, since it provides a relatively good trade-off between performance and speed.

Download the pre-trained model from tensorflow model detection zoo.

1. Training the Model:

To initiate a new training job, cd inside the training\_demo folder and type the following: python train.py --logtostderr --train\_dir=training/ -- pipeline\_config\_path=training/ssd\_inception\_v2\_coco.config

1. Prediction:

## E:\Screenshot from 2019-10-14 11-46-56.png

## E:\Screenshot from 2019-10-14 11-42-44.png