**Mask RCNN Final Training & Prediction Document**

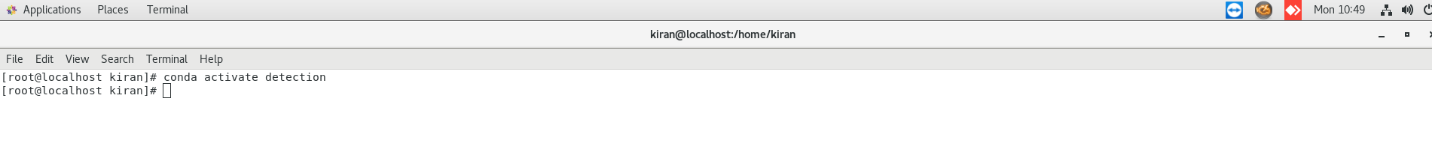
Mask RCNN custom object detection Model for signature verification Implementation.

1. Create your own environment

**conda create -n detection pip python=3.6**

1. Activate your environment

**conda activate detection**



1. Install compatible tensorflow version

**pip install –ignore-installed –upgrade tensorflow==1.12**

1. Install required packages for Mask\_RCNN object model detection

**pip install pillow**

**pip install lxml**

**pip install jupyter**

**pip install matplotlib**

**pip install cython**

**pip install opncv-python**

1. **Folder structure**

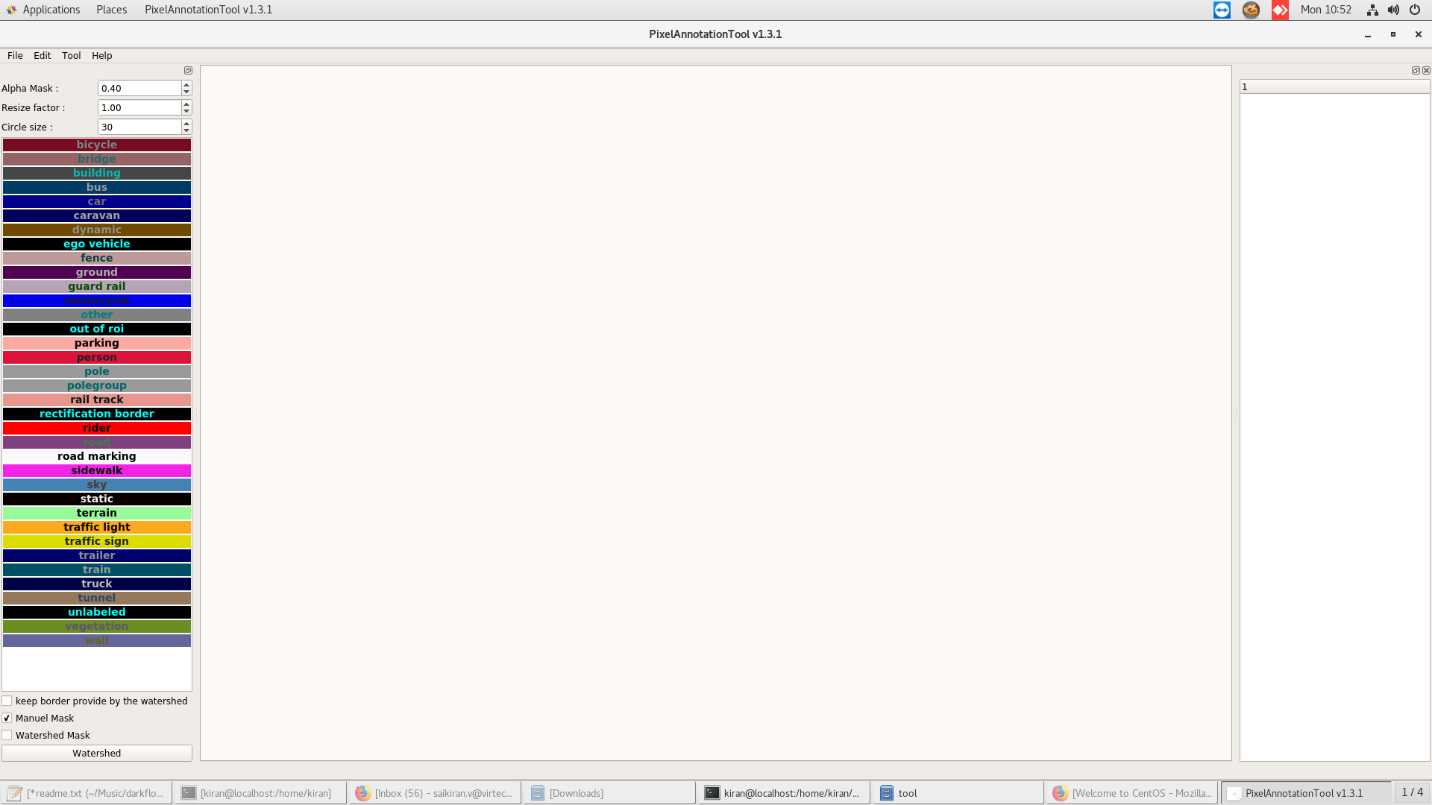
Tensorflow\_API-Custom\_Mask\_RCNN

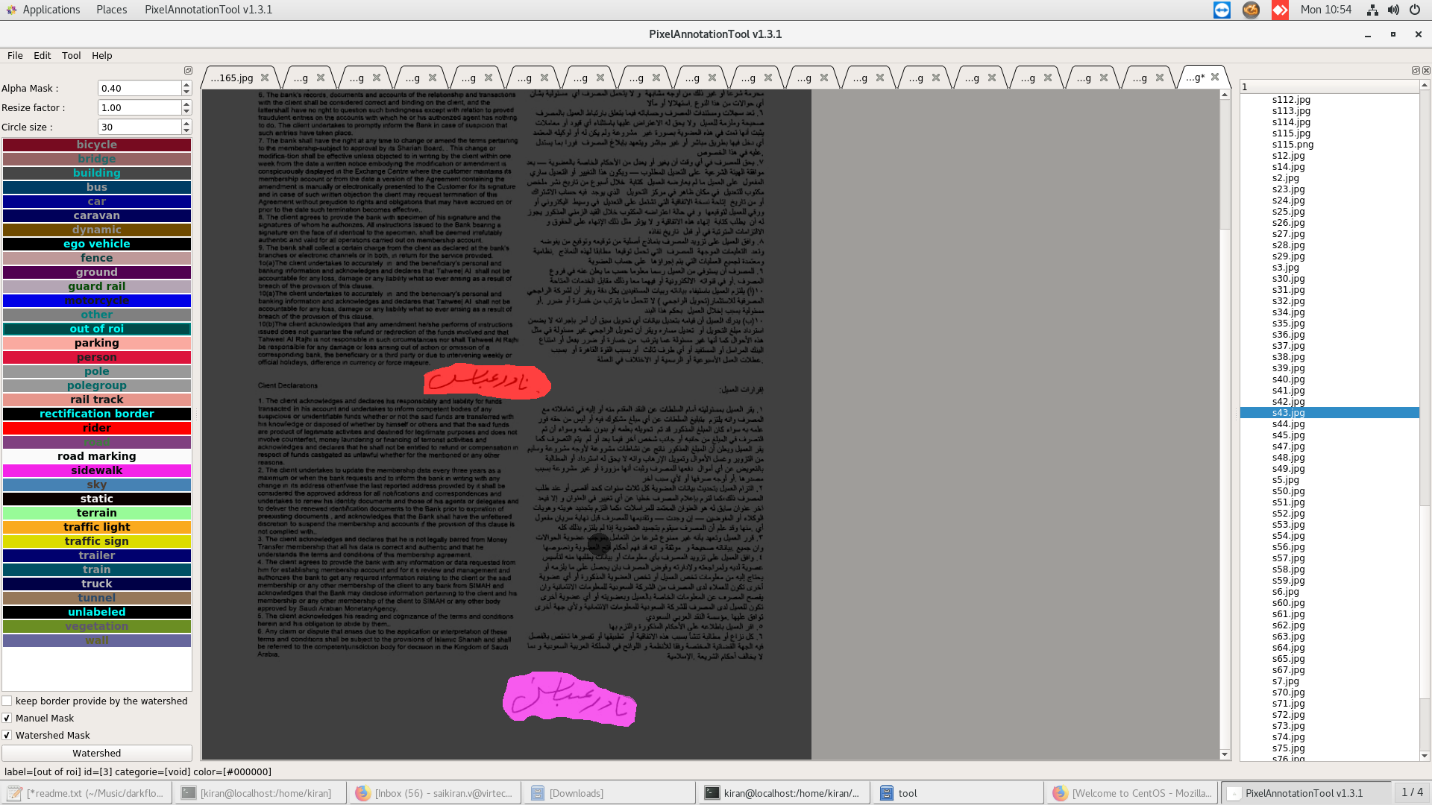
* pre\_trained\_models
  + downloaded files for the choosen pre-trained model will come here
* dataset
  + Annotations
    - xmls
      * files with bounding box annotations will come here (needed only when handling case where images have multiple object of the same class in the scene)
    - maskss for training images will come here
  + JPEGImages
    - all of images for training will come here
  + testImages
    - all 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

1. **Label the data**

Label the training data using pixel annotation tool. This tool will generate three files in the image folder

* IMAGENAME\_color\_mask.png
* IMAGENAME\_mask.png
* IMAGENAME\_watershed\_mask.png.





1. **Annotating images**

To annotate images we will be using the [labelImg](https://github.com/tzutalin/labelImg) package

**conda create -n labelImg pyqt=4**

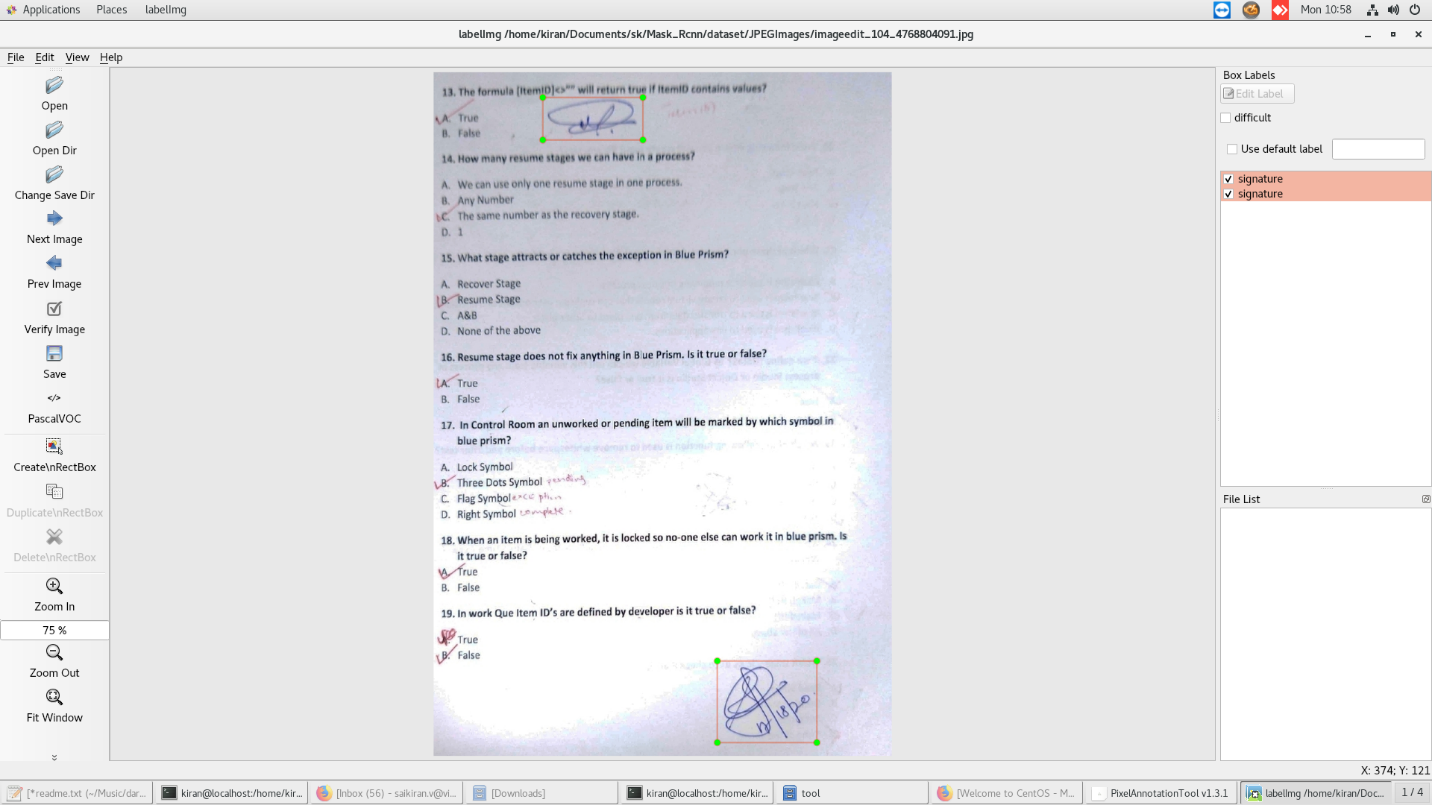
**activate labelImg**

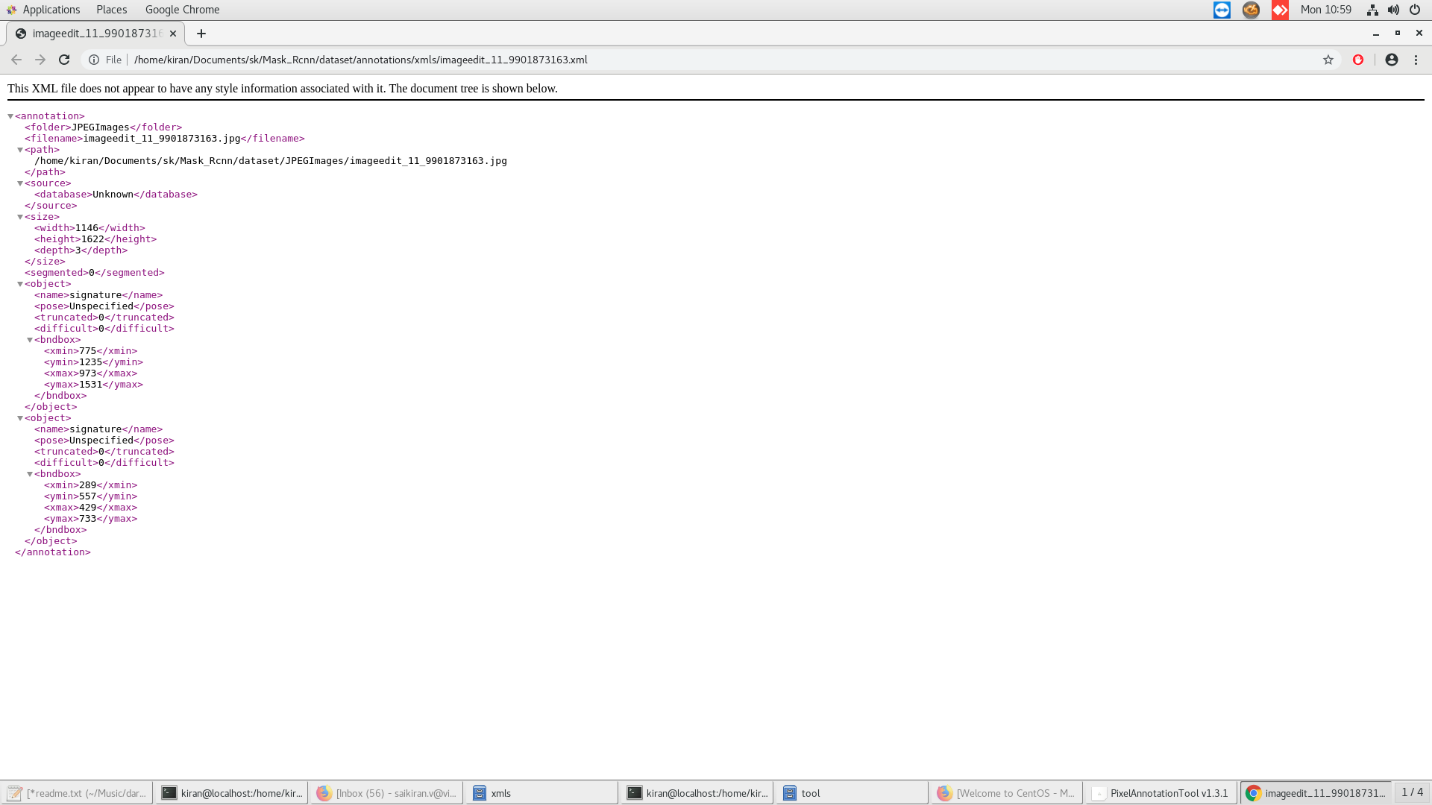
**conda install pyqt=4**

**conda install lxml**

**pyrcc4 -py3 -o resources.py resources.qrc**

**python labelImg.py**





**6.Creating Label Map**

item {

id: 1

name: 'stamp' }

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

## 7. Creating TensorFlow Records

## 7.1 Converting \*.xml to \*.csv= *python xml\_to\_csv.py -i [PATH\_TO\_IMAGES\_FOLDER]/train -o [PATH\_TO\_ANNOTATIONS\_FOLDER]/train\_labels.cs*

7.2 Converting from \*.csv to \*.record= # Create train data: python generate\_tfrecord.py --label=<LABEL> --csv\_input=<PATH\_TO\_ANNOTATIONS\_FOLDER>/train\_labels.csv --output\_path=<PATH\_TO\_ANNOTATIONS\_FOLDER>/train.record

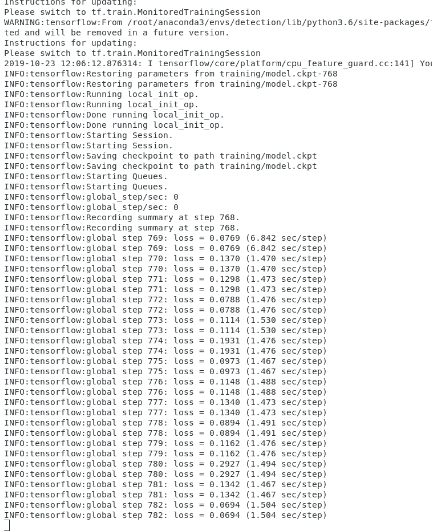
## There should be 2 new files under the training\_demo\annotations folder, named test.record and train.record, respectively.

## 8. Configuring a 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. Since we shall be using the faster\_rcnn\_inception\_v2\_coco model, we shall be downloading the corresponding mask\_rcnn\_inception\_v2.cfg file.

## 9. Training the Model

**python train.py --logtostderr --train\_dir=training/ pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_coco.config**



**Observations:**

