**TENSORFLOW OBJECT DETECTION STEPS FOR RETRAIN MODEL ON OWN DATASET**

1. Install tensorflow
2. Setup tensorflow directory e.g. c:\tensorflow1
3. Download tensorflow object detection API from github <https://github.com/tensorflow/models>
4. Download the tensorflow model like SSD or FastRCNN

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

1. Steps for Anconda environment
2. C:\> conda create -n tensorflow1 pip python=3.6
3. C:\> activate tensorflow1
4. (tensorflow1) C:\> pip install --ignore-installed --upgrade tensorflow
5. (tensorflow1) C:\> conda install -c anaconda protobuf
6. (tensorflow1) C:\> pip install pillow
7. (tensorflow1) C:\> pip install lxml
8. (tensorflow1) C:\> pip install Cython
9. (tensorflow1) C:\> pip install jupyter
10. (tensorflow1) C:\> pip install matplotlib
11. (tensorflow1) C:\> pip install pandas
12. (tensorflow1) C:\> pip install opencv-python
13. Configure PYTHONPATH Environment variable

(tensorflow1) C:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim

1. Run protobuf files in protoc folder. It will create .py file

protoc --python\_out=. .\object\_detection\protos\box\_predictor.proto .\object\_detection\protos\hyperparams.proto .\object\_detection\protos\matcher.proto .\object\_detection\protos\pipeline.proto .\object\_detection\protos\ssd.proto .\object\_detection\protos\anchor\_generator.proto .\object\_detection\protos\eval.proto .\object\_detection\protos\image\_resizer.proto .\object\_detection\protos\mean\_stddev\_box\_coder.proto .\object\_detection\protos\post\_processing.proto .\object\_detection\protos\ssd\_anchor\_generator.proto .\object\_detection\protos\argmax\_matcher.proto .\object\_detection\protos\faster\_rcnn.proto .\object\_detection\protos\input\_reader.proto .\object\_detection\protos\model.proto .\object\_detection\protos\preprocessor.proto .\object\_detection\protos\string\_int\_label\_map.proto .\object\_detection\protos\bipartite\_matcher.proto .\object\_detection\protos\faster\_rcnn\_box\_coder.proto .\object\_detection\protos\keypoint\_box\_coder.proto .\object\_detection\protos\multiscale\_anchor\_generator.proto .\object\_detection\protos\region\_similarity\_calculator.proto .\object\_detection\protos\train.proto .\object\_detection\protos\box\_coder.proto .\object\_detection\protos\grid\_anchor\_generator.proto .\object\_detection\protos\losses.proto .\object\_detection\protos\optimizer.proto .\object\_detection\protos\square\_box\_coder.proto

1. run the following commands from the C:\tensorflow1\models\research directory:

(tensorflow1) C:\tensorflow1\models\research> python setup.py build

(tensorflow1) C:\tensorflow1\models\research> python setup.py install

Retraining steps:-

1. Download LabelIMg exe for create xml file for your training and testing datasets.
2. Generate training datasets
3. Execute below script to convert xml files to CSV then convert CSV to tfrecord format

(tensorflow1) C:\tensorflow1\models\research\object\_detection> python xml\_to\_csv.py

filename,width,height,class,xmin,ymin,xmax,ymax

61326(BackSide)-OK-(1).jpg,128,128,61326OkBack,1,2,128,127

61326(BackSide)-OK-(2).jpg,128,128,61326OkBack,1,1,127,128

61326(FrontSide)-OK-(1).jpg,128,128,61326Okfront,2,3,128,124

61326(FrontSide)-OK-(2).jpg,128,128,61326Okfront,7,4,128,126

61326-SC-(1).jpg,128,128,61326ScratchMark,3,1,128,124

61326-SC-(2).jpg,128,128,61326ScratchMark,1,3,128,127

61326-SD-(1).jpg,128,128,61326SlotDamage,1,3,128,124

61326-SD-(2).jpg,128,128,61326SlotDamage,2,2,121,128

61326-TH-(1).jpg,128,128,61326Thinning,2,4,127,127

61326-TH-(2).jpg,128,128,61326Thinning,3,1,128,127

61326-WR-(1).jpg,128,128,61326Wrinkle,2,4,125,128

61326-WR-(2).jpg,128,128,61326Wrinkle,5,7,128,124

python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record

python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record

add classes condition in generate\_tfrecord file

if row\_label == '61326Okfront':

return 1

elif row\_label == '61326ScratchMark':

return 2

elif row\_label == '61326SlotDamage':

return 3

elif row\_label == '61326Thinning':

return 4

elif row\_label == '61326Wrinkle':

return 5

elif row\_label == '61326OkBack':

return 6

else:

None

1. Create labelmap.pbtxt file for define the classes:-

C:\tensorflow1\models\research\object\_detection\training

e.g.

item {

id: 1

name: '61326Okfront'

}

item {

id: 2

name: '61326scratchMark'

}

item {

id: 3

name: '61326SlotDamage'

}

item {

id: 4

name: '61326Thinning'

}

item {

id: 5

name: '61326Wrinkle'

}

item {

id: 6

name: '61326OkBack'

}

1. Take the config file from this folder C:\tensorflow1\models\research\object\_detection\samples\configs and change the paths , num of training examples, Training epoch and number of classes
2. python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/ssd\_mobilenet\_v1\_jbm.config
3. it will start training on your data sets. But for training high end computational resources are required .otherwise normal CPU will hang the system because it requires more memory for retraining.second if file size is very huge then it will also slow training . so decrease the file size dimensions when model is running on less computational resources
4. Once training is completed then validate your own test dataset on the graph which is created.
5. Create pb from .ckpt file

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/ssd\_mobilenet\_v1\_jbm.config --trained\_checkpoint\_prefix training/model.ckpt-1000 --output\_directory training/

1. Object\_detection python file present in github.run the file it will classify the data