Real-time Face Mask Detection using TensorFlow Object Detection API

By: Robin Manchanda

# **Business Case and Objectives**

## Executive Summary

In this phase of the world where it is extremely critical in wearing masks to prevent the communicable diseases. It is necessary to set up certain measures to curb the further spread. Wearing of masks will prevent most of the chances of contracting the disease. However, people are not aware of the consequences and appear to roam without masks.

In this project, we will be implementing a real-time face mask detection system that can detect if the person captured by the camera is wearing a mask or not.

## Problem Statement

## As the disease is wide-spread and the economies are slowly opening up, it is important to employ less staffs in places where it is necessary to manage the crowds. To overcome this, we can have a real-time object detection system which can alert the system when it captures a person without any mask.

Installation

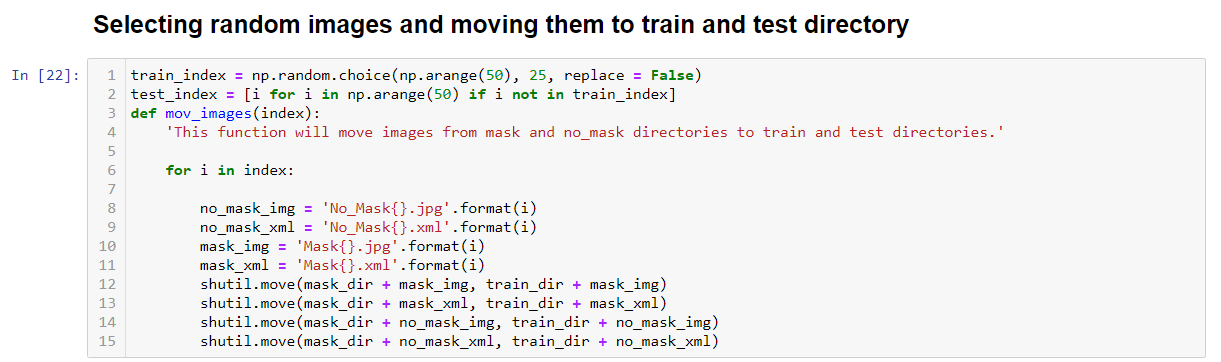
We need to install the following libraries,

* TensorFlow (See [TensorFlow Installation](https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html))
* TensorFlow Object Detection API (See [TensorFlow Object Detection API Installation](https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html))
* Protobuf
* Cudnn/Cuda
* labelImg
* PyQt
* XML

Capture Images

A script has been written using OpenCV to capture mask and no mask images with webcam. Then, these images are randomly selected using numpy.random.choice attribute, and moved to train and test directories using shutil library.





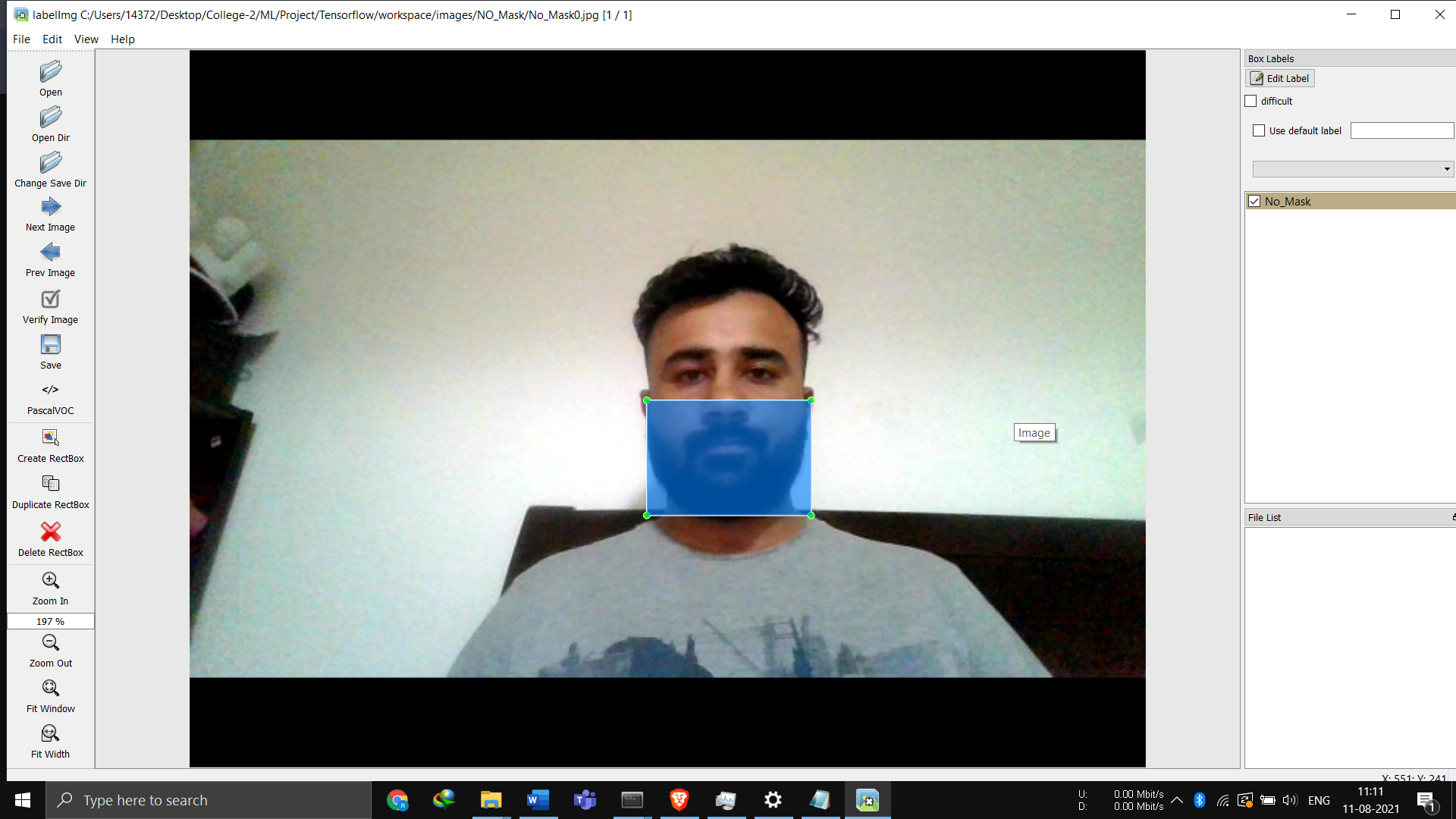
## Preparing the Dataset

LabelImg has been used to annotate the dataset

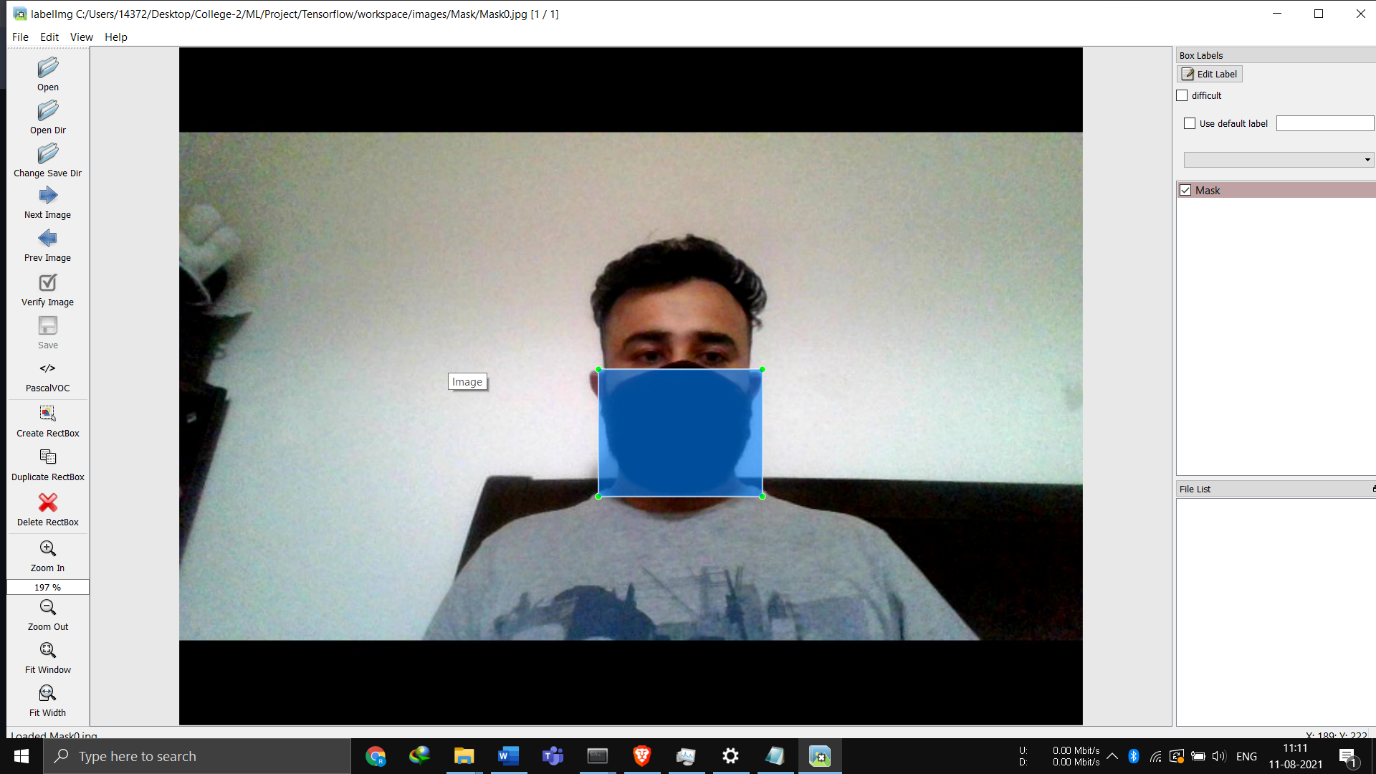
After installing LabelImg and the images of both with face mask and without facemask are collected and then annotated using the following command

*labelImg <PATH\_TO\_TF>/TensorFlow/workspace/face-mask-detection/images*

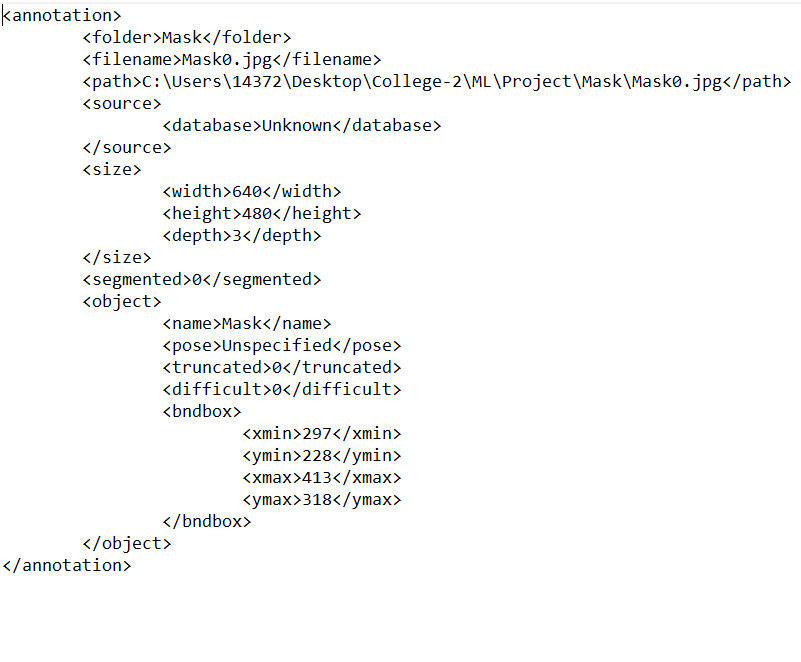
Annotating the image (No\_Mask)



Annotating the image (Mask)

**

After annotating a .xml file is created (Mask)



After annotating a .xml file is created (No\_Mask)

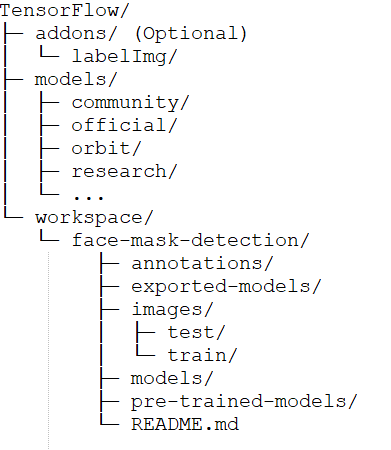


**Partition the Dataset**

Once the data is annotated, it is split into train and test using numpy.random.choice attribute, and moved to train and test directories using shutil library.

## Preparing the Workspace

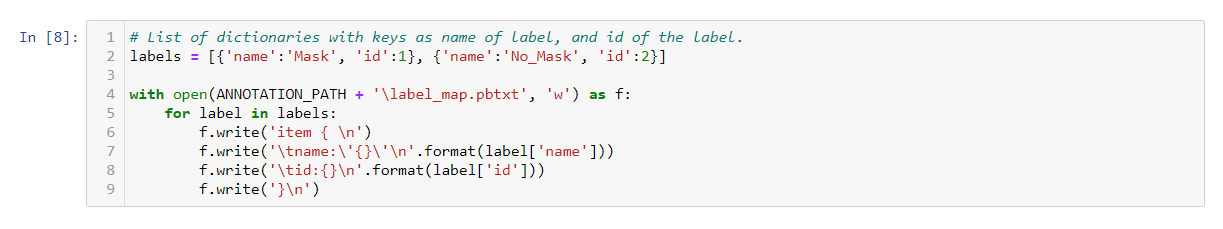
After the installation of TensorFlow, a workspace is setup with the name of the module along with the other required folders.



Create Label Map

A label map is created for both the type of classes in our dataset as shown below.

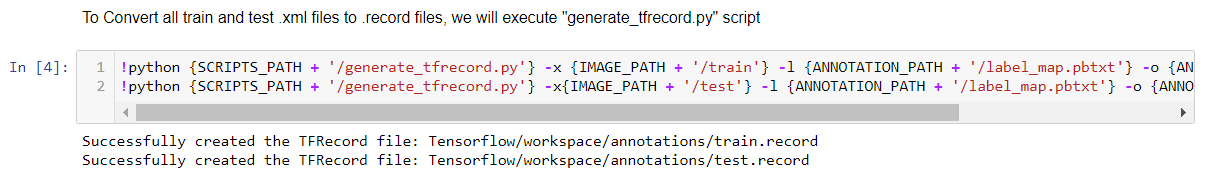
|  |
| --- |
| item {  id: 1  name: ‘mask'  }  item {  id: 2  name: ‘nomask'  } |



Create TensorFlow Records

**Convert \*.xml to \*.record**

A simple script is created which can iterate through all \*.xml files in the face-mask-detection/images/train and face-mask-detection/images/test folders, and generate a \*.record file for each of the two. Here is an example script that allows us to do just that:



Configuring a Training Job

For the purpose of this project, we are not building the detection model from scratch instead we are going to leverage the transfer learning concept.

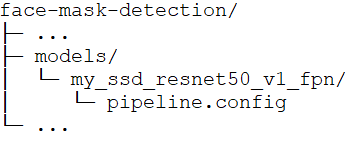
Ware using SSD MOBILENET V2 FPN 320\*320, as this model provides good detection speed for our real time detection and also provides good mean accuracy performance.

Link to the TensorFlow Object Detection Model used: [TensorFlow 2 Detection Model Zoo](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md)



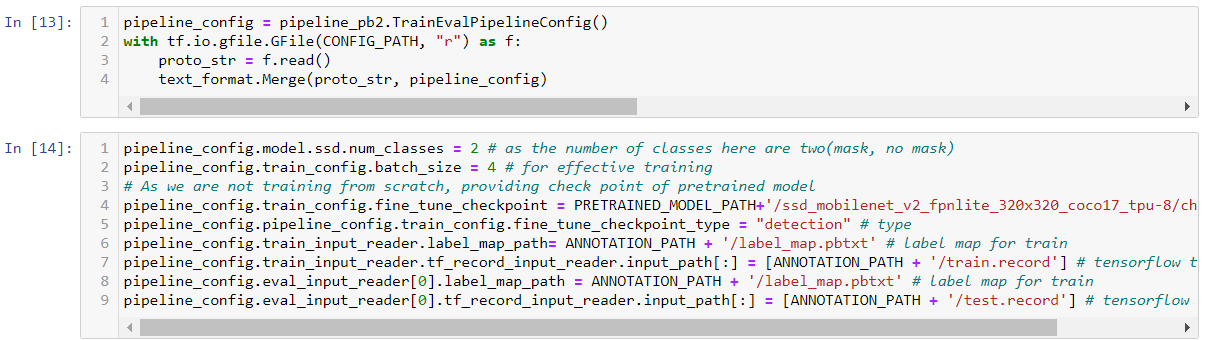
Configure the Training Pipeline

The pipeline.config file from the downloaded model is placed in the below directory structure



It is then configured by altering the below mentioned variables.

1. num\_classes
2. batch\_size
3. fine\_tune\_checkpoint
4. fine\_tune\_checkpoint\_type
5. label\_map\_path
6. Train & Test record path



Training the Model

The changes we have made in pipeline.config file are stored in the structure described in the last slide. After copying the model\_main\_tf2.py file from Tensorflow package to workspace and running the following command will train the model.

**Command to Train the model:**

|  |
| --- |
| *python model\_main\_tf2.py --model\_dir=models/my\_ssd\_resnet50\_v1\_fpn --pipeline\_config\_path=models/my\_ssd\_resnet50\_v1\_fpn/pipeline.config* |

Load the trained model:

The model that is trained is restored from the checkpoints.

Graphical user interface, text, application

Description automatically generated

Real-time detection

A script has been written that is used to initialize camera, reading frame, converting the image to array.

Then tensor. viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array is used to setup the boxes, scores and labels on screen.

Real-time detection of face mask is done custom TensorFlow object detection model.

References:

1. TensorFlow 2 Object Detection API tutorial, retrieved from <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/>
2. Pre-trained model downloaded from <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md>
3. Nicholas Renotte, <https://www.youtube.com/watch?v=yqkISICHH-U&t=17274s>