Aakarsh Yadav

[yaakarsh1011@gmail.com](mailto:yaakarsh1011@gmail.com)

8826194573

<https://www.linkedin.com/in/aakarsh1011>

<https://www.github.com/aakarsh1011>

**My approach to the face mask detection and tracking:**

1. Initially I started with the looking for dataset which is already in *tfrecord,* to save time and get good data. Luckily I found a data set from Roboflow’s data archives which was already in tfrecord format and can cloned using the following command, “!curl -L "https://public.roboflow.com/ds/TxNNu00ZCH?key=DPAiT4TZLy" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip”
2. Next I thought of using transfer learning instead of creating a custom neural network from scratch and hence looked for pretrained model on tensorflow. Meanwhile I cloned template created by *Jayy Bhatt* at <https://github.com/jakkcoder/widows-object-detection-setup> for easy organization of files required for getting the dataset and config files ready for the training.
3. I was using Tensorflow’s Object detection API on the pretrained model of ssd\_efficient\_det\_d0\_512x512. I used this as it had both optimal time delay and has better average precision score.
4. I placed the tfrecords of both train and test data in the research folder as mentioned in the tensorflow’s object detection api and the labelmap.pbtxt in the newly created images folder. This folder could contain images, if have to create tfrecords from them using the generate\_tfrecords.py by *Jayy Bhatt’s* repo.
5. Then I copied the ssd\_efficientnet’s config file from the model’s folder of object detection Api and edited the ‘num\_class’, ‘lowered the learning rate’, ‘changed the type from classification to detection’ and provided the path to out tfrecords and labelmap.
6. then I compressed the whole project folder and moved it to the google drive, so that I can use it for training in google colab.
7. In google colab I mounted the drive and unzipped all the data. Created helper functions to load and save inference files which is also accessible in the api’s documentation.
8. After setting up the environment, downloading dependencies and setting up the path. With the help of train python command from the documentation I started the training. I trained it for almost 2 hours, loss reduced to almost ~0.098. I stopped the training and transferred the config and inference\_graph to the object detection folder in my local machine to run it in Jupyter notebook.
9. In my local machine I loaded the saved\_model.pb along with few dependencies, I used OpenCV to live stream and make predictions using the model I had and also taking input from the file.

**Result:**

1. The accuracy of the model seems lower than expected. It can be clearly seen that it needs more training plus a bigger dataset.
2. It is confusing a person with beard close to having mask.
3. Not able to detect in hazy video input.

**Problems I faced and can be improved:**

1. Time constraint: Time needed to get a big labelled dataset needed annotation of images which could be really time consuming but could have improved the dataset.
2. Latency issue: reading the 1920\*1080 video file was quite difficult for OpenCV without latency. Module like ffmpeg could be tried.
3. I wasn’t able to save the output video as the output video was always coming blank which I couldn’t understand why.
4. More training should have been done without overfitting to improve the model.

**Could have done:**

1. If time permitted, I tried to understand the working of DeepSORT tracking algorithm and would have tried to implement it.