How to run the application:

Step 1: Data Collection.

1. The data that is being collected here is for the purpose of object detection. So, first find the number of classes of objects that is to be detected.
2. Images are captured using a mobile camera. So, make sure the quality of images is good enough to distinguish the objects in the image.
3. Object detection requires a good number of images for training for each class of the object. Therefore collect at least 50 images per class.
4. Also, collect images in such a way that they contain substantial amounts of noise, objects of other classes, objects that don’t belong to any of the defined classes and images covering the objects in different angles of vision. This helps in making the model more robust in detection.
5. Some examples from our dataset,

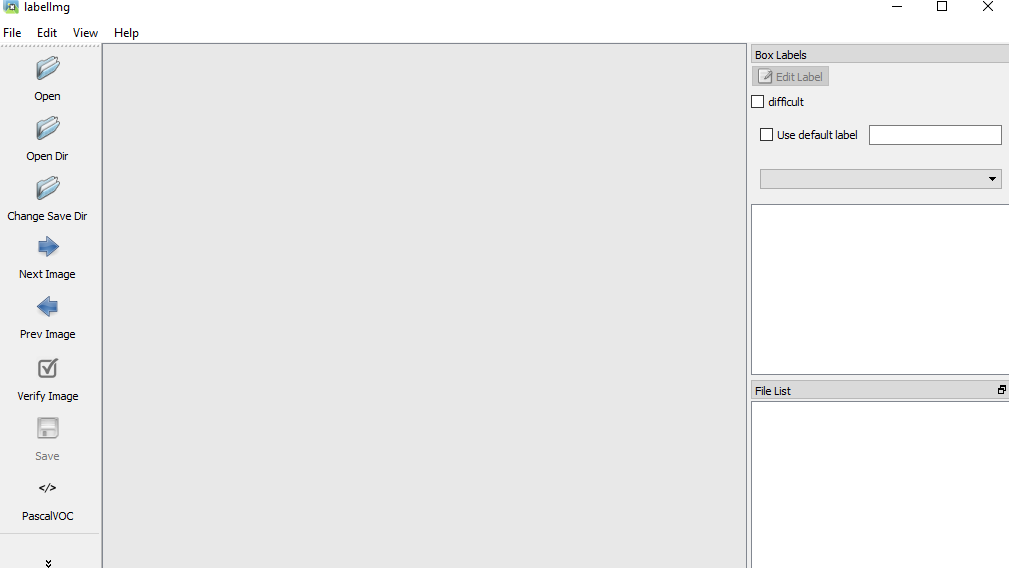






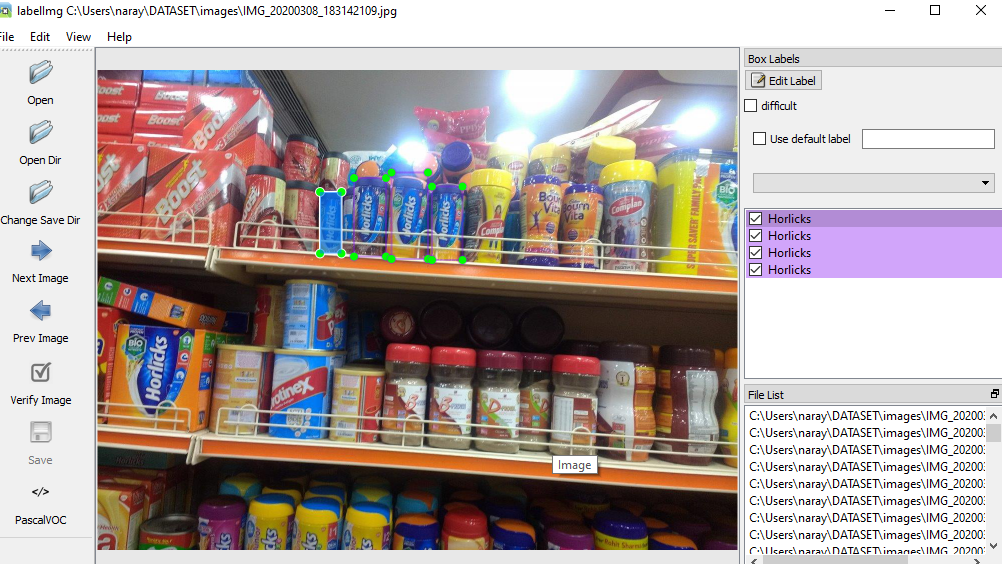
Step 2: Annotation of Data.

1. This is a very important step because this where the labelling of objects to be detected is done. This classification paves way for the supervised learning of the detection process.
2. We annotated our images using the open source annotation application LabelImg.



1. Using the “Open” or “Open Dir” option choose to open the image or directory of images to be annotated using the application.

An example of annotation of the product of class “Horlicks”,



1. The results of annotation are stored as an XML file. The XML is file is the converted to the CSV format for a more structured approach.

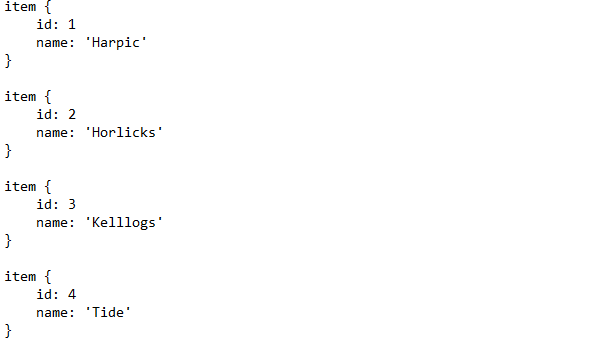
Step 3: Downloading the most suitable model for the application.

There are various object detection models available, but not all of them provide the best results for all kinds of object detection. Object detection models like SSD, YOLO are more suitable for real-time processing of images and detecting the objects in them. Our goal is to detect objects on stored images, therefore Faster-RCNN is the most suitable as it provides the maximum accuracy of all the object detection techniques.

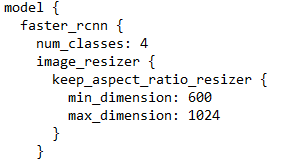
The model can be downloaded from <https://github.com/tensorflow/models>.

Step 4: Training of the model on your dataset.

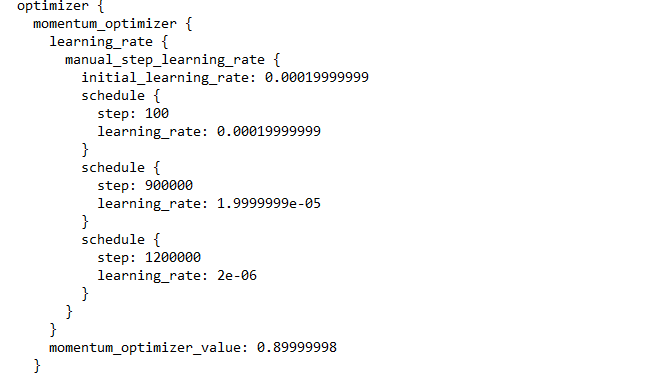
1. Create a “.pbtxt” file called “label\_map” that gives and “id” for the different labels given to the objects in the training images that looks like this,



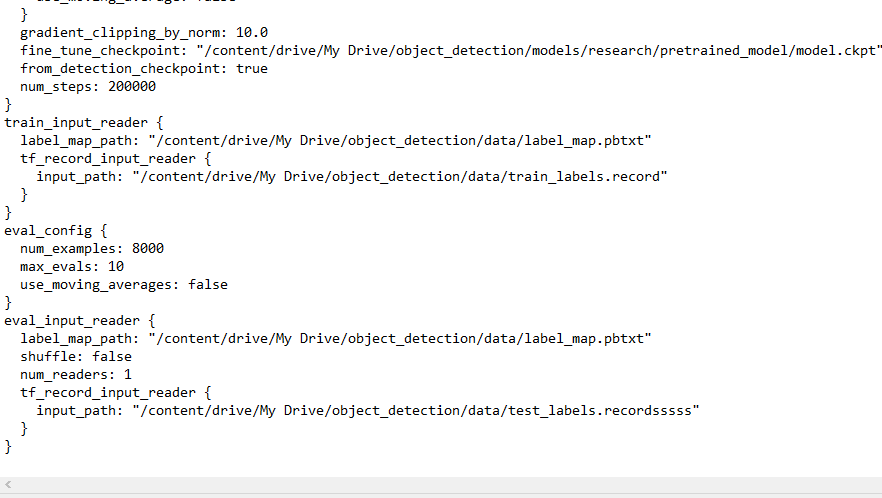
1. Make the desired configurations to the “pipeline.config”.



Change the “num\_classes” to the original number of classes of objects that are to be detected. Here we are detecting 4 classes.



Change the “learning\_rate” in the optimizer based on the outputs that you are arriving at.



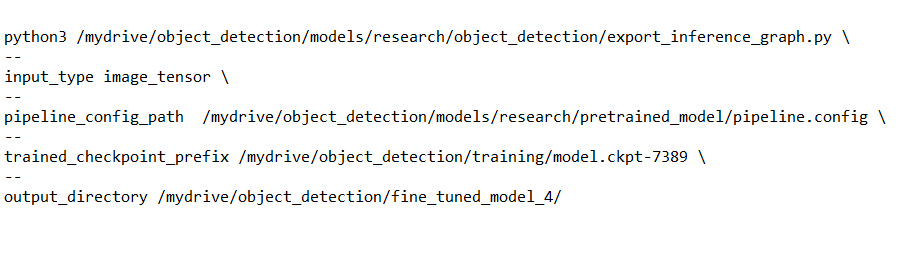
1. Set the “fine\_tune\_checkpoint” to the location that contains the model checkpoint saved while training as “model.ckpt”.
2. Set the “label\_map\_path” to the location that contains the file “label\_map.pbtxt”.
3. The “input\_path” in “train\_input\_reader” ‘s “tf\_record\_input\_reader” must be set to the location that contains the tensorflow record “train\_labels.record” that is obtained by processing the training images and training labels.
4. The “input\_path” in “eval\_input\_reader” ‘s “tf\_record\_input\_reader” must be set to the location that contains the tensorflow record “test\_labels.record” that is obtained by processing the testing images and testing labels.
5. Run the following script in your command-prompt to train your model on the dataset,



1. The “model\_main.py” is available in the “object\_detection” folder of the “models/research” folder of the “models” folder downloaded from the tensorflow git.
2. The “pipeline\_config\_path” must be set to the location of the configured version of “pipeline.config”.
3. The model after training is interrupted will be saved in the path specified in “model\_dir”.

Step 4: Exporting the inference graph.

Run the following script in the command-prompt to export the inference graph of the model.



1. The “export\_inference\_graph.py” can be found in the “models/research/object\_detection” directory of the “models” directory downloaded from tensorflow git.
2. As we are using images as the major input the “input\_type” is to be specified as “image\_tensor”.
3. The “pipeline\_config\_path” must be given with the value of the location of the configured file “pipeline.config”.
4. Choose the model checkpoint that has the minimum loss and maximum accuracy. Here “model.ckpt-7389” is used. The “trained\_checkpoint\_prefix” must be given with the location of the said checkpoint.
5. The frozen inference graph will be stored in the path provided in the parameter “output\_directory”.

Step 5: Testing