## HELP FILE

The data collected consists of about overall of 7000 images including the vehicles, pedestrians and the background images.

These images are trained and tested on the videos for required output. The detection phase of the proposed system is that, mainly a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. The biggest advantage of this method is the calculation speed.



**Fig 3.3.1: Frame of a Sample Video.**

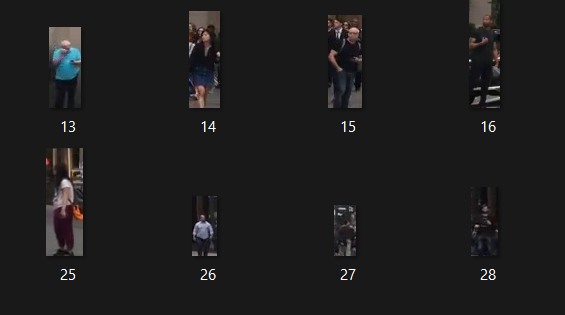
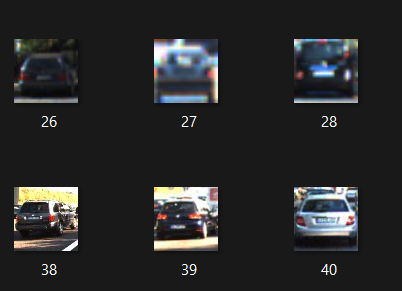
**Building includes the following steps:**

**Data Collection: Collection of Positive and Negative Image Set.**

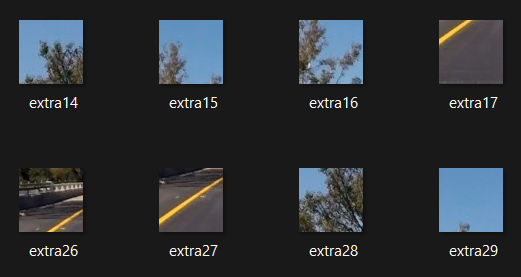
Collect the images of the pedestrians and the objects as a dataset from Kaggle.

Classify the images into positive and negative dataset.

The positive images consist of the objects and the pedestrian images (represented as p) and the Negative images consists of the background images (represented as n).

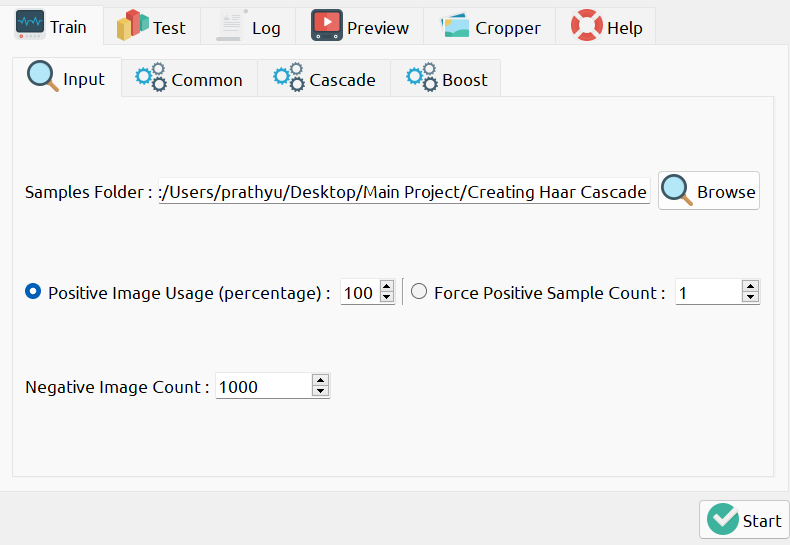
**Fig 3.3.4: Positive Images of Pedestrians Fig 3.3.5: Positive Images of Vehicles**



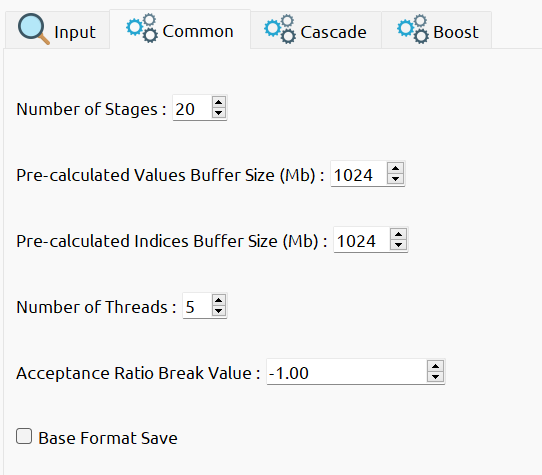
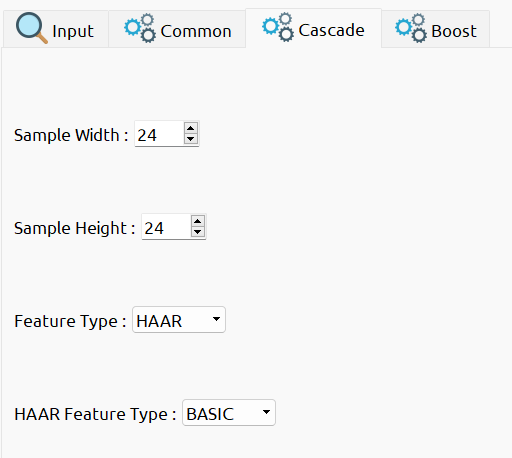
**Fig 3.3.6: Negative Images of Vehicles and Pedestrians**

**Data Pre-processing.**

Now train the Images using Cascade Classifier GUI by setting the number of stages including height and width.

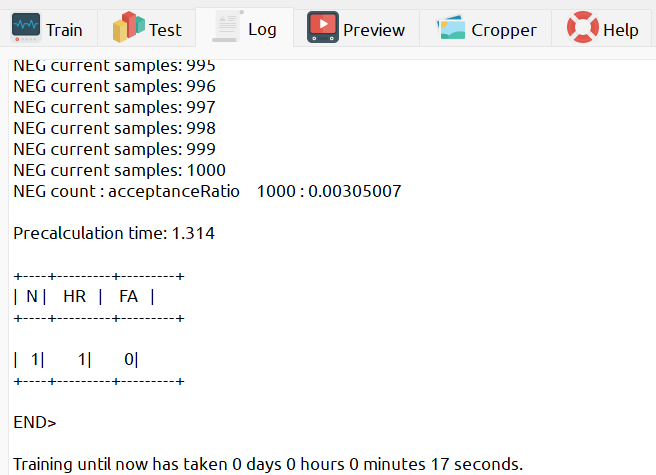


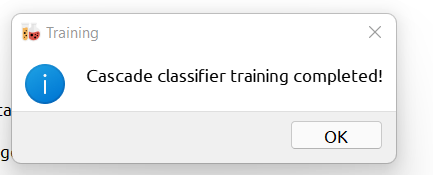
**Fig 3.3.7: Loading The images into GUI**

**Fig 3.3.8: Setting the number of Stages Fig 3.3.9: Setting the size for same amount of images.**

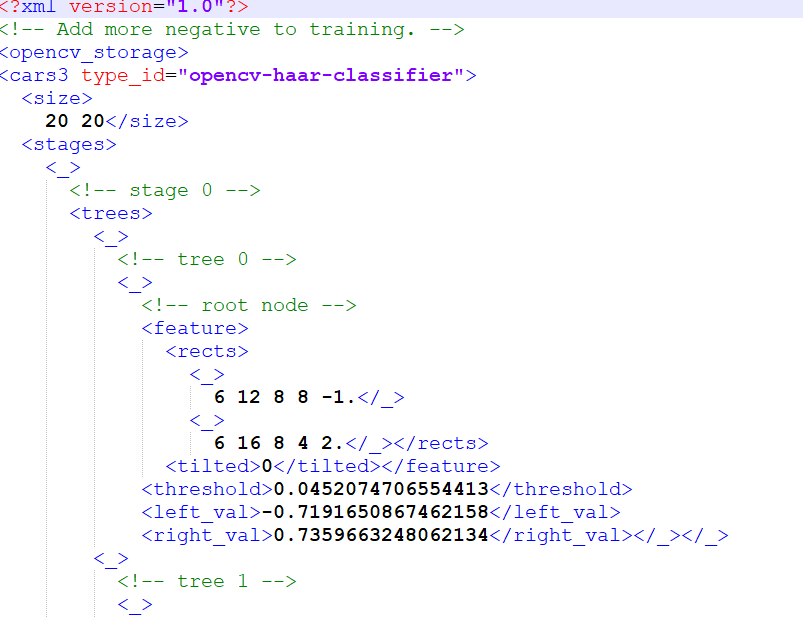
**Training the images:**

****

****

The Training images result as an XML file which is used as a classifier.

The classifier File which contains XML File

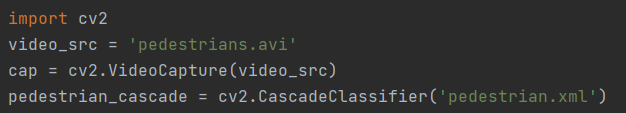


**Fig 3.3.10: XML File Visualization**

**Testing on Videos:**

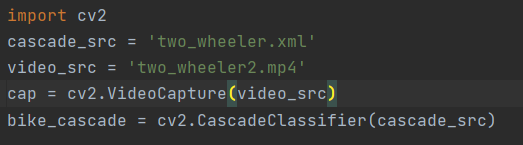
The different kinds of XML Files are imported accordingly including the testing videos.

**For Pedestrian Detection:**



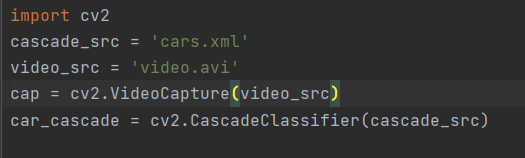
**Fig 3.3.11: Implementing Pedestrian Detection**

**For Bike Detection:**



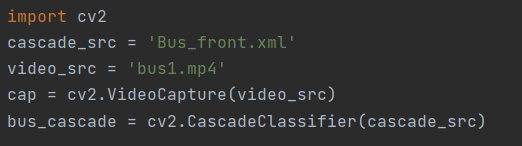
**Fig 3.3.12: Implementing Bike Detection**

**For Car Detection:**



**Fig 3.3.13: Implementing Car Detection**

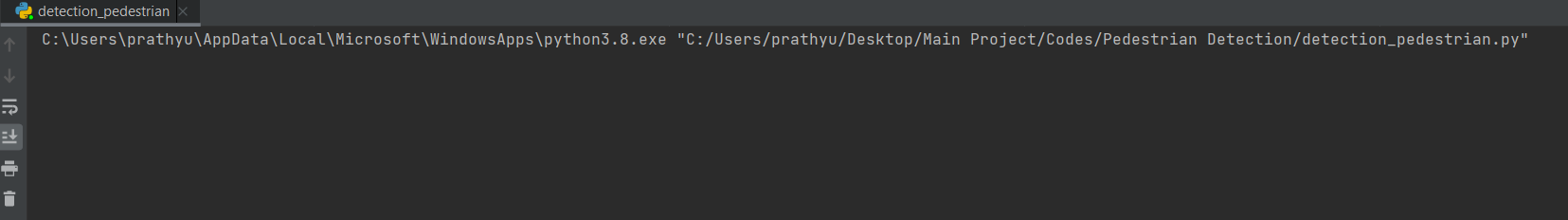
**For Bus Detection:**



**Fig 3.3.14: Implementing Bus Detection**

**Running The Application:**

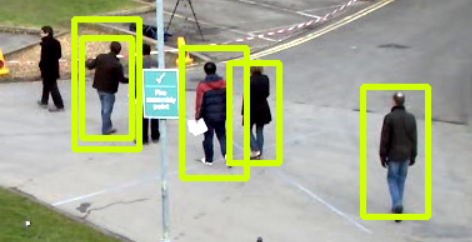
* Run the application by running the `app.py` file. By default, It displays the output.

****

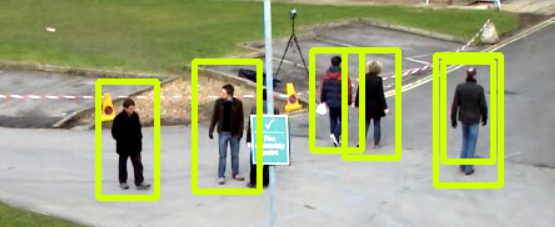
**Fig 4.1: Running the File**

**Outputs:**

**Detecting Pedestrians:**

****  ****

****

****

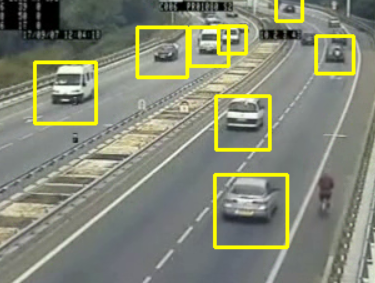
**Fig 4.2: Detecting Pedestrians**

**Detecting Bus:**

**** ****

**Fig 4.3: Detecting Bus**

**Detecting Cars:**

** **

**Fig 4.4: Detecting Cars**

**Detecting Two-Wheelers:**

****

****

**Fig 4.5: Detecting Two-Wheelers**