Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

Building Custom HAAR-Cascade Classifier for face Detection

Mr. Tejas R. Phase
Dept. of Electronics
K.B.P. College of Enggi.
Satara, India

Dr. Prof. Suhas S. Patil
Dept. of Electronics
K.B.P. College of Enggi.
Satara, India

***Abstract*— There are superior pre-trained HAAR-Cascade classifiers available on the Internet whose detection accuracy is quite impressive for the task of face detection in the presence of different illuminations conditions and different poses of the face. But the drawback to use such pre-trained classifies for any detection task is we never know how training of such classifiers can be done, how to prepare dataset for a particular detection task and how to use different parameters of the classifiers while training. In this paper we build our own Custom HAAR-Cascade Classifier using "Cascade Trainer GUI (a tool designed by Amin Ahmadi) to detect face/faces in any given image/images. We also create dataset which include positive and negative samples to use during training purpose. We also demonstrate how to retrain the classifier after analyzing error matrix after each detection stage and how to increase accuracy of the classifier in detection work.**

***Keywords*—HAAR-Cascade Classifier;postive samples;negative samples;GUI,Training Dataset; Test Dataset;False Positive;False Negative;face database**

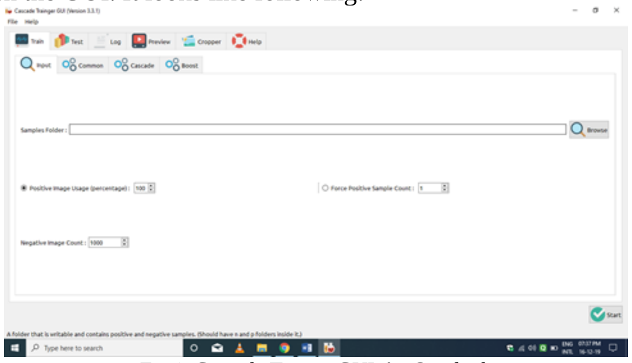
I. INTRODUCTION

Face detection is bit challenging because while detection of the face we needs to consider all possible appearance variations happened due to change in illuminations and blockage of light fall on the image etc. In addition with this, the system needs to detect faces that visible at various scale, various pose and with rotations. In spitefulness of all these problems and difficulties tremendous progress has been happened from the last few years and due to this progress many face-detection systems have shown admirable real-time performance in their operations [1]. These recent advancements in these classification models have also shown significant performance in detection of other objects include full human body/pedestrians, cars, signal pols etc. For the task of face detection most of the times there is the usage of pre-trained HAAR-Cascade Classifier whose performance is quite noticeable with presence all of the above challenges. But when we use pre-trained classifier we never know how the training of that classifier can be done, how to prepare data if we want to perform the detection task other than face etc [2]. By addressing such kind of questions we present in this paper the procedure of creating our own Classifier using very simple to use GUI tool named "Cascade Trainer GUI (a tool designed by Amin Ahmadi)". Cascade Trainer GUI is a program that can be used to train, test and improve cascade classifier models. It uses a graphical interface to set the parameters and make it easy to use OpenCV tools for training and testing classifiers. As this is my first time to create and train a Classifier I use it for face detection task. But you can create a classifier to any kind of object detection task by following the same process.

II. CASCADE TRAINER GUI

A. Installation:

Currently Cascade Trainer GUI can be used on Windows (7 or above). The installation procedure is pretty straightforward. To download the set-up, visit: https://amin-ahmadi.com/. After downloading the setup, just install it at your preferred location in your local machine. After successful installation click on the icon that is creates on your desktop to open the GUI. It looks like following:

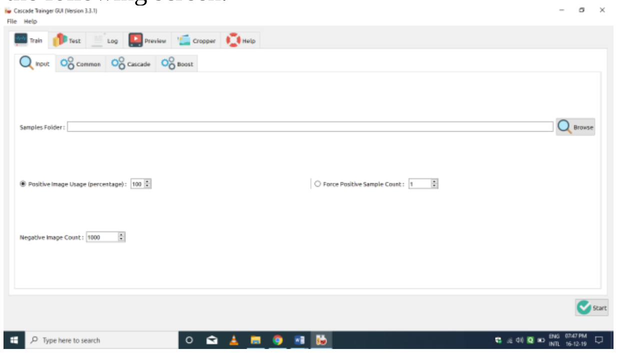


Samples Folder:
Positive Image Usage (percentage): 100
Force Positive Sample Count: 1
Negative Image Count: 1000
A folder that is writable and contains positive and negative samples. (Should have n and p folders inside it.)

Fig.1. Cascade Trainer GUI-An Overlook

B. How to use:

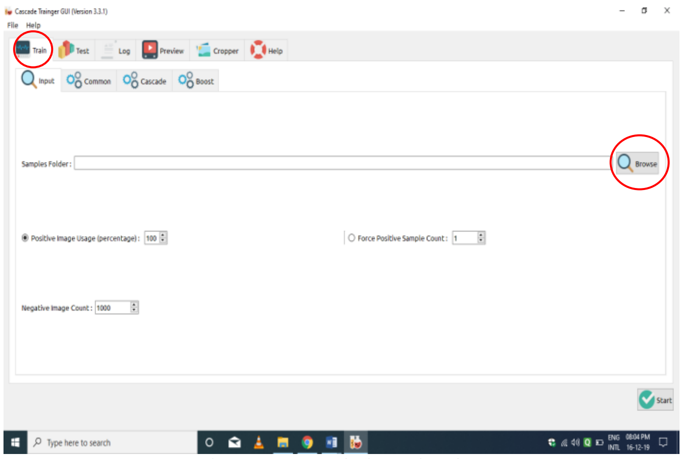
Here we see different parts and functionalities of Cascade Trainer GUI. When Cascade Trainer GUI is first started we see the following screen.



Samples Folder:
Positive Image Usage (percentage): 100
Force Positive Sample Count: 1
Negative Image Count: 1000

Fig. 2. First look when open GUI tool

This is the starting screen and it can be used for training classifiers. To train classifiers usually we need to provide thousands positive and negative image samples, but there are cases when we can achieve the same with less samples. To start the training, we need to create a folder for our classifier. Then we need to create two folders inside it. One should be named as "p" (for positive images) and the other should be named as "n" (for negative images). For this paper we created a folder in our local machine named "CUSTOM-HAAR-CASCADE-FACE-DETECTION" in which we created the above mentioned 2 folders inside it. Positive image samples are the images of the object we want to train our classifier and then detect. In our case, we want to train and detect faces in the images and thus we need lots of images that contained the face or faces. And you need also have many negative samples. Negative samples are the images that doesn't contain any face. Thing to remember: Negative images must NEVER include any positive images part and if it includes mistakenly then our classifier will generate false negative error in detection. In theory, negative samples can be any image that doesn't include any positive image part but during training, negative samples should be somehow relevant to the positive samples. For example, using dog face image as negative sample is a poor choice for training a good face detection classifier, even though it doesn't have a bad effect on the overall accuracy. Start by pressing the Browse button in Train tab. Select the folder you have created for the classifier [3] .



Samples Folder :

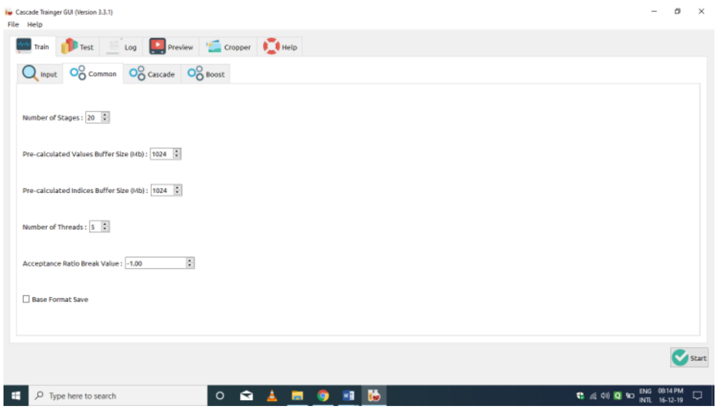
Positive Image Usage (percentage) : 100

Force Positive Sample Count : 1

Negative Image Count : 1000

Fig. 3. "Input" tab (in Train tab)

Keep the remaining parameters from that tab as it is. Then move to the "common" tab. Common, Cascade and Boost tabs can be used for setting numerous parameters for customizing the classifier training. By default, Cascade Trainer GUI sets the most optimized and recommended settings for these parameters. But still we need to tune some parameters for each iteration of the training. We have not documented detailed description of all these parameters in this paper and which is beyond the scope of this study and require deep knowledge about cascade classification techniques.



Number of Stages : 20

Pre-calculated Values Buffer Size (Mb) : 1024

Pre-calculated Indices Buffer Size (Mb) : 1024

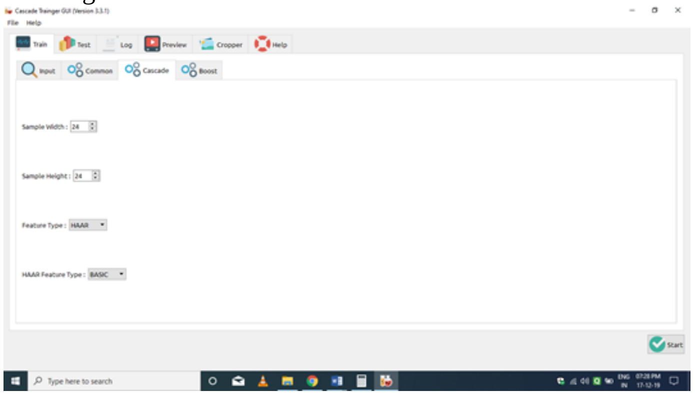
Number of Threads : 5

Acceptance Ratio Break Value : -1.00

☐ Base Format Save

Fig. 4. "Common" tab(in Train tab)

Here we set the iterations of the training. In general, more the iterations better the classifier detection accuracy. But as the number of iterations increase it takes more time to build and train the classifier. Next we set pre-calculation buffer size to help with the speed of training process. We can assign as much memory as we can for these but be careful to not assign too much or too low. For example if we have 4 GB of RAM on our computer then we can safely set both of the buffer sizes below to 2048[4]. Then in next tab which is "Cascade" we need to set the sample width and height. We need to make sure that this should not be set to a very big size because it will make our detection process very slow. Recommended settings for sample width and height is that to keep one aspect on 24 and set the other accordingly. For example, if we have sample images that are 640 × 480 then first calculate the aspect ratio which in this case is 1.33:1 then multiply the bigger number with 24. You'd get 32×24 for sample width and height.



Sample Width : 24

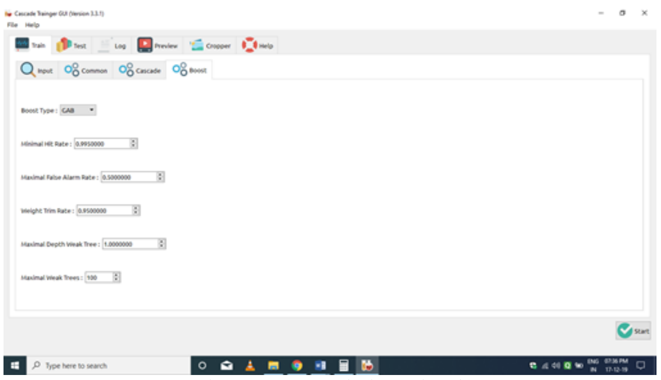
Sample Height : 24

Feature Type : HAAR

HAAR Feature Type : BASIC

Fig.5. "Cascade" tab(in Train tab)

We can also set the feature type to HAAR or LBP. We use HOG only if we have OpenCV 3.1 or later. HAAR classifiers are very accurate but require a lot more time to train so it is much wiser to use LBP if we provide our classifiers with many sample images. LBP classifiers on the other hand are less accurate but train much quicker and detect almost 3 times faster. As for the parameters in the Boost tab, keep the default values we are not much aware about those[4].



Boost Type : GAB
Minimal Hit Rate : 0.9950000
Maximal False Alarm Rate : 0.5000000
Weight Trim Rate : 0.9500000
Maximal Depth Weak Tree : 1.0000000
Maximal Weak Trees : 100

Fig. 6. "Boost" tab (in Train tab)

III. PREPARATION OF THE DATASET

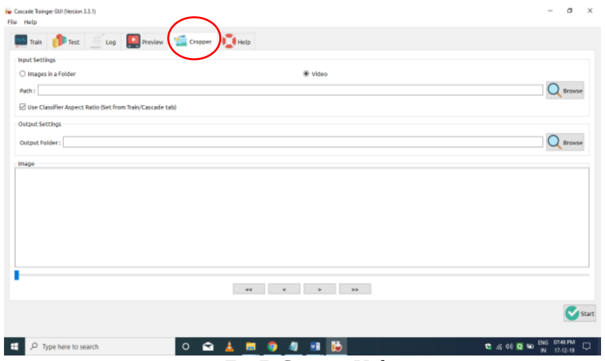
For any detection task we need a set of positive and negative images. A Cascade classifier basically tells "OpenCV" what to look in an images. How many images do we need is depends on a variety of factors which include:

--The quality of the images

--The object we want to recognize

--The method to generate samples

--The CPU Power. For our training we used 60-Positive samples and 40-Negative samples. We create positive and negative image dataset by-hand and store them in their respective folders (p & n in this case). We use "Cropper "function of Cascade Trainer GUI to make these datasets.



Input Settings
Images in a Folder
[✓ Video]
Path :
Use Classifier Aspect Ratio (Set from Train/Cascade tab)
Output Settings
Output Folder :
Image

Fig.7. Cropper Utility

To create positive image dataset we use Georgia Tech Face Database. The database include the images of 50 people and the format of them is JPEG format. For each different persons, there are 15 color images and most of the images captured in two different time slots of the day to consider different variations in illumination conditions, different facial expression, and appearance. In addition with this, the faces or face images were captured with different scales and different orientations. Here we select 6 people images with different orientation and with different illumination conditions. Let's have a look on it:

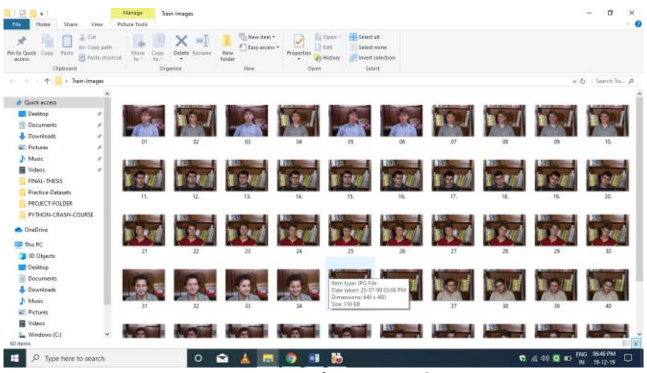
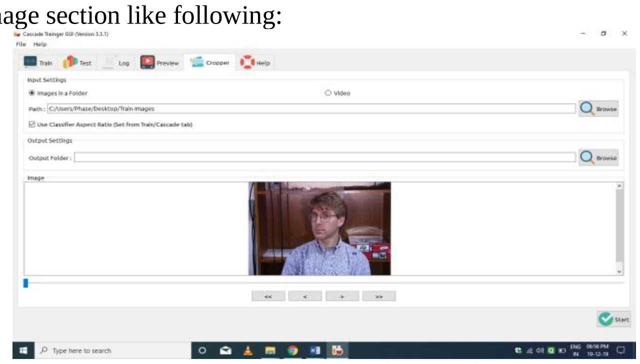


Fig.8. Face images for positive dataset

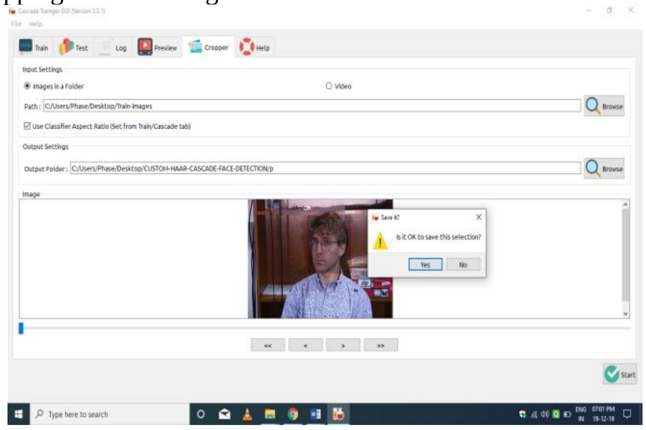
Then we store all these images (we can say raw images) in one folder and set the path of that folder in the "Input Setting" of "Cropper" function. We select "Images in folder" option which is available there. If someone wants to detect objects in video then he can choose video for input setting. Then we keep the other options as it is. Then as soon as we provide path of the image folder all of the images are popped in the image section like following:



Input Settings
[✓ Images in a Folder] Video
Path : C:/Users/Phase/Desktop/Train-Images
Use Classifier Aspect Ratio (Set from Train/Cascade tab)
Output Settings
Output Folder :
Image

Fig. 9. Images to Crop

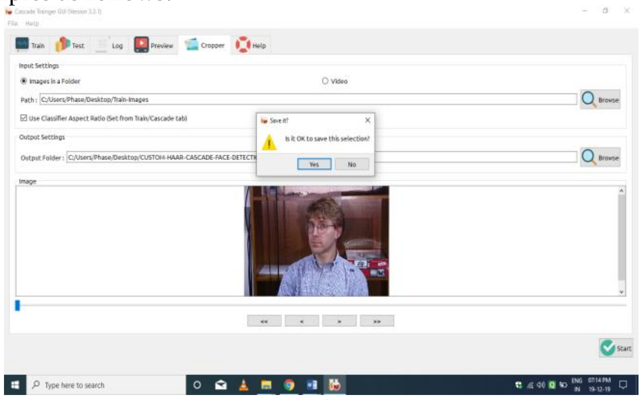
Using the "> <" buttons we can select the next or previous image for cropping. Then we provide the path of "p" folder that we have created at the beginning. Here all the cropped images are going in that "p" folder which stands "Positive" images for Cascade. Then after providing the path we start cropping like following:



Input Settings
[✓ Images in a Folder] Video
Path : C:/Users/Phase/Desktop/Train-Images
Use Classifier Aspect Ratio (Set from Train/Cascade tab)
Output Settings
Output Folder : C:/Users/Phase/Desktop/CUSTOM-HAAR-CASCADE-FACE-DETECTION/p
Image
Save it?
Is it OK to save this selection?
Yes No

Fig. 10. Cropping Face portion

Here we select only the "face portion" area in the image using mouse and as soon as you we stop the selection we get a pop-up message saying "is it OK to save this selection" and if we feel this Ok so we save this selection in the "p" folder. Kindly note: If you don't feel the selected area is not representing the exact face then you can cancel that pop-up box and reselect the desired region to save. You can do this step as many times as you can until you feel that selected area is the exact object we want to detect. Then after creating the desired positive samples (60 in this case) we move to create negative sample dataset. As we discussed earlier negative sample is anything that doesn't have to our detection object. Here anything means not entirely different but related to positive sample. For e.g. in face detection task the negative sample can part of face, background, hand etc. Like previous step, we provide path for the "n" folder where we want to store negative samples for cascade. We create the negative samples as follows:



Input Settings

(● Images in a Folder) ○ Video

Path : C:/Users/Phase/Desktop/Train-Images

☑ Use Classifier Aspect Ratio (Set from Train/Cascade tab)

Output Settings

Output Folder : C:/Users/Phase/Desktop/CUSTOM-HAAR-CASCADE-FACE-DETECTI

Image

Save it?

Is it OK to save this selection?

Yes No

Fig. 11. Creation of negative sample

Here we select the region as a negative sample from "ear" and the background. We create 40 such samples and store them in folder named "n".

A. *Positive Samples:*

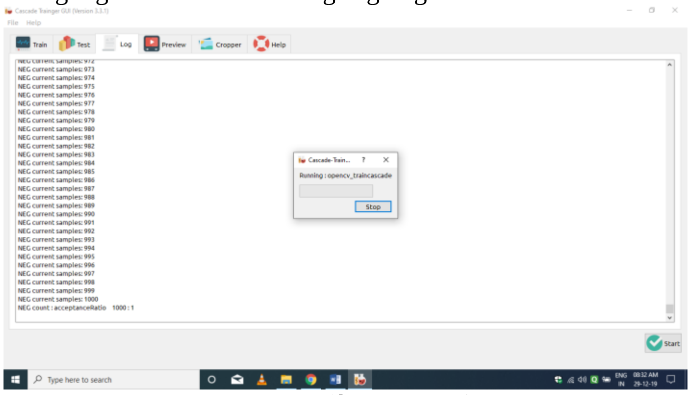
Positive samples must go in folder named 'p' (small letter 'p'). Positive samples are the images in which contain the object that we want our algorithm to detect. Here we have two methods to create such samples. First we can collect all the relevant images in which contain the object that we want to detect and can put directly into the "p" folder. The second way is to use "Cropper" tool. In our case we take the GT database of face images and select or crop only the face area from the image and save it to "p" folder. Keep in mind that the sample size should be max 32 x 32. If you put more than this then it would take more time for training.

B. *Negative Samples:*

Negative samples are the images which should not have the object that we want to detect. This does not mean that the "Negative Samples" are completely different. In general the negative sample is the sample in which contain the part of the object that we want to detect. The percentage of this part should be 20% of the original object that we want to detect. By keeping this in mind we take face parts, some background parts to create set of negative samples. Keep in mind that we need to store all these negative samples in folder named "n". And we can create this set like positive sample set which mean we can take directly negative images from Google images or we can use "Cropper" tool we crop or select the negative area in the image.

IV. MODEL CREATION & TRAINING

After all the parameters are set, press Start button at the bottom to start training your cascade classifier. You'll see the following log screen while training is going on.



```
NEG current samples: 972
NEG current samples: 973
NEG current samples: 974
NEG current samples: 975
NEG current samples: 976
NEG current samples: 977
NEG current samples: 978
NEG current samples: 979
NEG current samples: 980
NEG current samples: 981
NEG current samples: 982
NEG current samples: 983
NEG current samples: 984
NEG current samples: 985
NEG current samples: 986
NEG current samples: 987
NEG current samples: 988
NEG current samples: 989
NEG current samples: 990
NEG current samples: 991
NEG current samples: 992
NEG current samples: 993
NEG current samples: 994
NEG current samples: 995
NEG current samples: 996
NEG current samples: 997
NEG current samples: 998
NEG current samples: 999
NEG current samples: 1000
NEG count : acceptanceRatio    1000 : 1
```

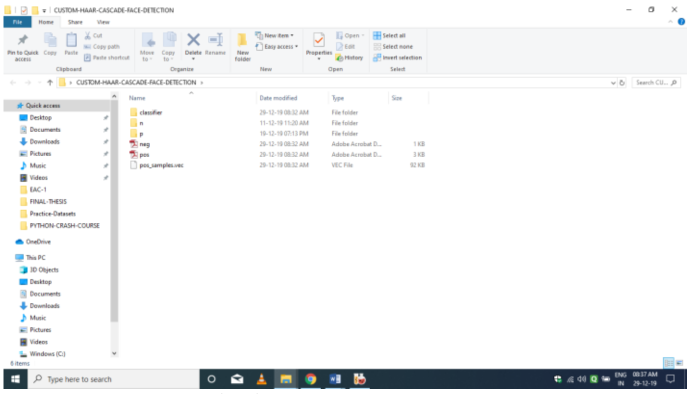
Cascade-Train...

Running : opencv_traincascade

Stop

Fig. 12. Log Screen(during Training)

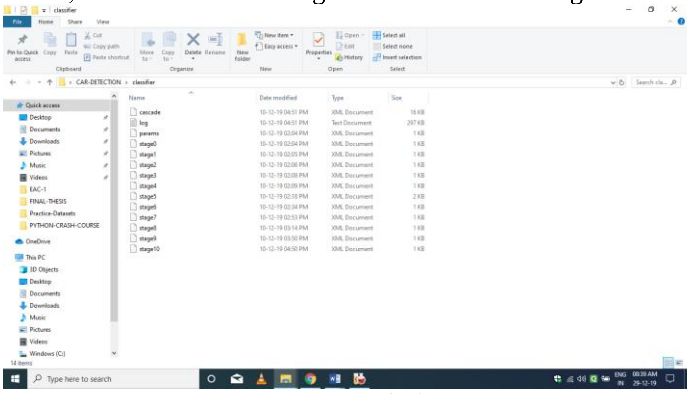
Wait for the training to complete. Now if we exit Cascade Trainer GUI and go to the classifier folder we will notice that there are new files and folders are created in this folder.



Name	Date modified	Type	Size
classifier	29-12-19 08:32 AM	File folder	
n	11-12-19 11:20 AM	File folder	
p	19-12-19 07:13 PM	File folder	
neg	29-12-19 08:32 AM	Adobe Acrobat D...	1 KB
pos	29-12-19 08:32 AM	Adobe Acrobat D...	3 KB
pos_samples.vec	29-12-19 08:32 AM	VEC File	92 KB

Fig. 13. Custom Classifier folder(created New Files and Folders inside it)

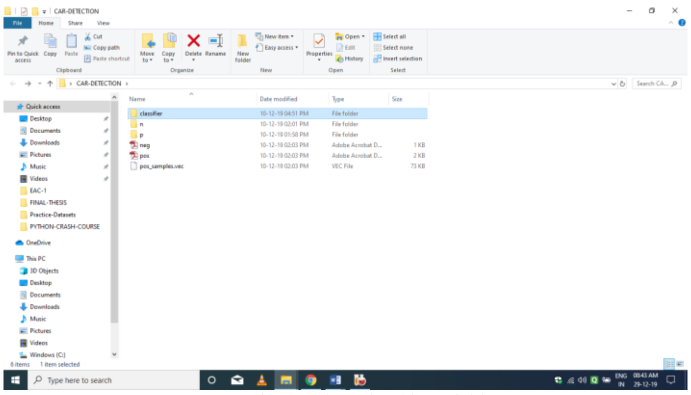
"n" and "p" are familiar to us but the rest are new. "classifier" folder contains XML files that are created during different stages of training. If we check inside "classifier" folder, we'll notice something similar to the following.



Name	Date modified	Type	Size
cascade	10-12-19 04:51 PM	XML Document	16 KB
log	10-12-19 04:51 PM	Text Document	297 KB
params	10-12-19 02:04 PM	XML Document	1 KB
stage0	10-12-19 02:04 PM	XML Document	1 KB
stage1	10-12-19 02:05 PM	XML Document	1 KB
stage2	10-12-19 02:06 PM	XML Document	1 KB
stage3	10-12-19 02:08 PM	XML Document	1 KB
stage4	10-12-19 02:09 PM	XML Document	1 KB
stage5	10-12-19 02:18 PM	XML Document	2 KB
stage6	10-12-19 02:34 PM	XML Document	1 KB
stage7	10-12-19 02:53 PM	XML Document	1 KB
stage8	10-12-19 03:14 PM	XML Document	1 KB
stage9	10-12-19 03:50 PM	XML Document	1 KB
stage10	10-12-19 04:50 PM	XML Document	1 KB

Fig. 14. Inside Classifer folder

"stage#.xml" files are temporary files that won't be needed anymore. "params.xml" contains the parameters we have used for the training. "cascade.xml" is the actual cascade classifier and if the training completed successfully then we should have this file inside classifier folder. "neg.lst", "pos.lst" and "pos_samples.vec" are temporary files created for training the classifier and they can also be removed without having any effect.



Name	Date modified	Type	Size
classifier	10-12-19 04:51 PM	File folder	
n	10-12-19 02:01 PM	File folder	
p	10-12-19 01:58 PM	File folder	
neg	10-12-19 02:03 PM	Adobe Acrobat D...	1 KB
pos	10-12-19 02:03 PM	Adobe Acrobat D...	2 KB
pos_samples.vec	10-12-19 02:03 PM	VEC File	73 KB

Fig. 15. Files inside "Classifier" folder

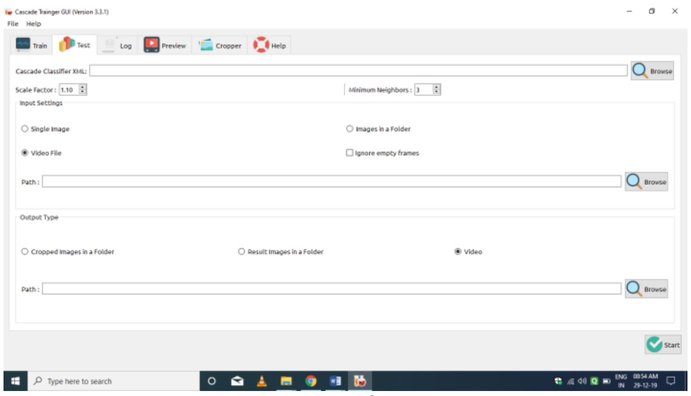
V. TESTING THE CLASSIFER

To test our classifiers, we need to go to Test tab from the tab bar at the top and set the options as described below and finally press Start button. First we select our cascade classifier using the Browse button at the top. We can also manually enter the path to cascade XML file in the Cascade Classifier XML field. This cascade classifier will be used for detection in images and/or videos. Next we can select one of the following for Input Settings and we need to set the path according to this option:

1. Single Image: A single image will be used as the scene in which detection will be done. In this case Path should point to a single image file. (Only supported images can be selected.)
2. Images in a Folder: A folder containing many images will be used for testing. In this case Path should be set to a folder that contains one or more scene images.
3. Video: A video file will be used for testing the classifier. In this case Path should point to a video file that will be used as input.

After setting the input settings it is time for output settings. Output type and path should be set according to what was chosen in input settings. See below:

1. Images in a Folder: Detected objects will be saved to multiple images inside a folder. In this case Path should be set to a folder. Note that this option can be used with all of the possible options in input.
2. Video: A video file with the detected objects highlighted with a red rectangle will be created. In this case Path should point to a video file that will be created after the test completes. Note that this option will only work if Video is selected in the input settings.



Cascade Classifier XML:
Scale Factor : 1.10
Minimum Neighbors : 3
Input Settings: Single Image [✓ Video File] Images in a Folder; Ignore empty frames ☐
Path :
Output Type: Cropped Images in a Folder, Result Images in a Folder [✓ Video]
Path :

Cascade Trainger GUI (Version 3.3.1)
File Help
Train Test Log Preview Cropper Help
Browse
Start

Fig. 16. Test tab

After clicking on Start button test will start. We see a progress bar and the following preview screen with results.

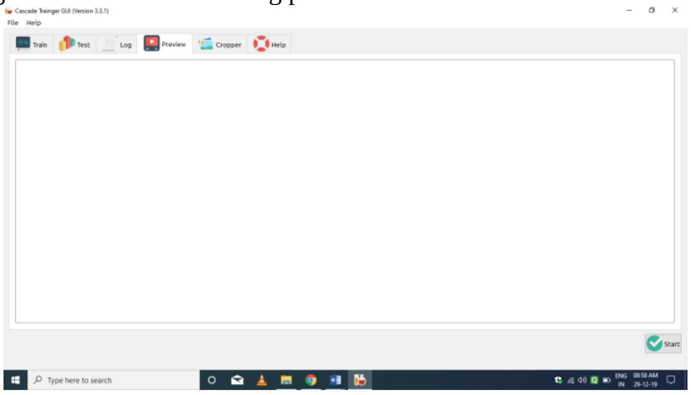

Cascade Trainger GUI (Version 3.3.1)
File Help
Train Test Log Preview Cropper Help
Start

Fig.17. Progress tab during Testing

After completion of the testing process check the output folder to analyze classifier accuracy. In the first time of the training we get some "false positive" as well as "false negative" errors in the detection. In that case, we need to re-train our classifier again by following all the above steps. But during retraining one thing we should do that we need to train the classifier on those "false positive" and "false negative" errors. This means during retraining we need to put "false positive" areas which actually are negative samples in the "n" folder in the classifier folder and "false negative" are actually face images that we need to put in the "p" folder in the classifier folder. And in this way we can reduce this error rate.

ACKNOWLEDGMENT

I would like to thank my guide, Dr. Suhas Patil, for his guidance and patience for explaining each and every concept. Thanks are due to Dr. Dipak Tatpuje (Head, Research Projects--Symbiosis International University), who's encouragement led me to write this paper. Also Thanks to Dr. Vidya Tatpuje (Secretary-VidyaDeep Foundation) whose mastery of the English language has systematically improved my writing of this paper.

REFERENCES

[1] Md. Iqbal Quraishi, J.P. Choudhury, Mallika De, “Image Recognition and Processing Using Artificial Neural Network”, 1st Int’l Conf. on Recent Advances in Information and Technology, (2012) , 978-1-4577-0697-4/12.

[2] AMER AL-RAHAYFEH AND MIAD FAEZIPOUR, Member, IEEE, “Eye Tracking and Head Movement Detection: A State-of-Art Survey”, IEEE Trans. In Transactional Engineering in Health and Medicine, VOLUME 1, pp. 2168-2372, 14 November 2013.

[3] Ming-Hsuan Yang, “Face Detection”, University of California, Merced, CA 95344.

[4] www.intel.in

[5] https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9