The George Washington University

INDIVIDUAL FINAL REPORT

Machine Learning 1 - DATS 6202
Summer 2019

Gregg Berne Legarda

INTRODUCTION

This is a project that shows an implementation of a facial recognition algorithm using Haar cascades with gradient and Histogram of Gradients (HOG) algorithms. Haar cascades uses Haar-like features and detects edges and lines by using kernels (such as 3x3 pixels). When the algorithm detects a face like pattern, it classifies it as a face. In addition to Haar cascades, we added gradients to somewhat help with out facial recognition. Although a bit better, it still needs to train with a bigger dataset. HOG facial recognition algorithms on the other hand, uses gradients to detect and recognize faces. This is a little bit slower but effective in recognition and classification. Our project showcases both of these algorithms to help viewers compare the difference between the two. We also added some other features such as gradient view mode to show how gradients look on camera, implemented web scraping from Instagram profiles, as wells as Dash Plotly web application for a better user experience.

Our team of two shared work on everything and was there for each other for the whole process. From ideas, to coding, from papers, to presentations, and also doing calculations needed for the project. Both members were active in participating and the result was a good, clean working application.

DESCRIPTION OF INDIVIDUAL WORK

We basically did everything together, constantly having video chats on how everything is working. From ideas, to the coding to the presentation. For the ideas, it was mostly Cristina initiating. She has many ideas and we would talk about if it would work or not. When we started, the big issues were packages since one computer is Mac and one is Windows. We have different versions of codes but are pretty much the same since we would build up on what the other person added.

For the teamwork, we were working on Haar initially because it worked on both computers. HOG did not work on mine. Cristina discovered that it was more accurate in recognition, but HOG didn't work on mine due to space issues. So, I decided to wipeout and reformat my computer and eventually, HOG worked. In the first half, I was more involved. In the second half, on the application, she was making a lot of it work and we were both really happy with our result. Although there are slight variations on our current codes, mostly aesthetics and user interface issues, it is pretty much the same. I think we really worked well as a team since both of us are really involved in the participation.

For the coding, whenever we had problems, we would consult the other person if they can help with the code one person would push it, and the other person would look at how to debug it. We decided that we would claim and share 50/50 of the credit for work on the individual description part since there was really no "individual parts" we were coding it together the whole time and working late. Even with the cleaning up of the folders and cleaning up of the code, we

were both involved. In addition, the documentation was also done in equal parts. We are both very present on communication even on non-free times.

We were both doing so much that I think there was a mini competition on who's version is better, but we didn't want to compete and decided that we would build up and merge our codes and features every tsime we update. I really think we worked great as a team.

For the presentation, we discussed initially on what was going to be presented had to consult and talk to each other. Cristina and I added slides and recommendations on the presentation, and we rehearsed enough times to have confidence presenting in class.

SHARED VS INDIVIDUAL PORTION OF THE WORK

To begin with, we started playing around with image detection and made it work. We then moved on to live webcam face detection using Haar Cascades. When we moved to facial recognition, we added gradients to the Haar Cascades to make it work a little bit better. We then implemented another feature called web scraping. It takes the Instagram details of the person identified and trained through the algorithms.

We made a switched from Haar to HOG since we figured out it worked better at recognition with small datasets. We tried to do Convolutional Neural Network, but the problem was that it took so much computing power that image processing and training took more than 4 hours. It is too time consuming and we would need to upgrade our hardware to continue with this. We decided not to follow through this feature and stayed with HOG. We continually fixed bugs and error created through the process. We added another feature and put it in the Dash Plotly web application as a host for our user interface.

For Plotly, we added buttons for training, capturing images, HOG and Haar Webcam feeds, Gradient Demo, and a Stop button to stop the Webcam feed. After creating the buttons, we then implemented the corresponding files and classes when the button is clicked. This resulted in an output in the Web applicaton screen. Then, we made the training easier with a capture option for the user so they can also try and use our demo.

We then created separate HOG, Haar and gradient folders, cleaned up the code, and made other folders as needed. We also created a Readme.md file for the code. There are two versions, one can be run on Physhell and one can be run on Pycharm both basically do the same thing. We then created the group report and the group presentation together and checked the project outline if we were missing something.

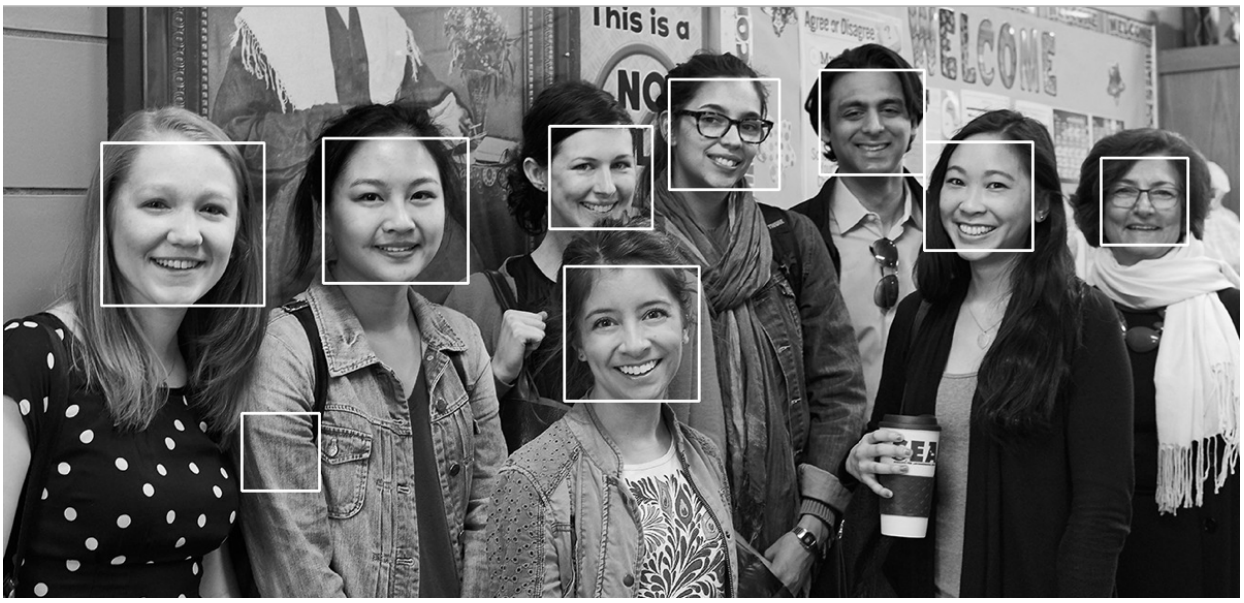
RESULTS

The result of the project is an application using a local host that can detect and recognize faces. The recognition depends on whether the model is trained to recognize a specific face. The project pulls information from the identified Instagram user's account and displays it on the web browser along with a box that detects the face.

The application we built is easy to use, and anyone can capture images and re-train the model to recognize their face and pull information from their Instagram account. They can also compare the speed and accuracy of the two models used in the project namely Haar and HOG. In addition, they can view the camera on gradient mode so that they can see how these two models view the environment to remove some bias.

Face Detection

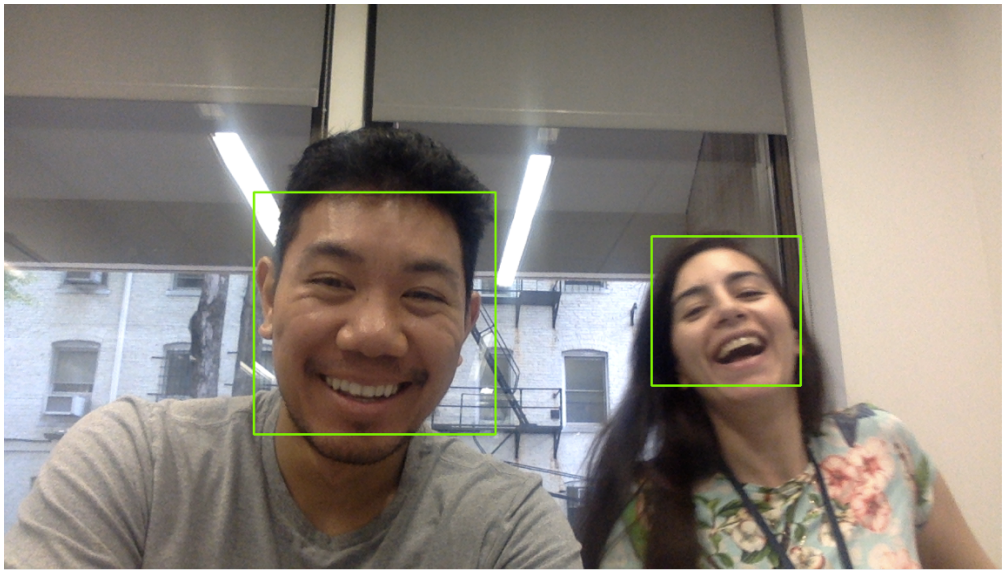
Image Detection



The face detection algorithm we used first is Haar cascades and this was our result. It is accurate enough to detect faces but there are false positive results as you can see on the bottom right side of the image. This is because Haar uses line and edge-based detection that differentiates between the intensities of the pixels. And if it fits a "face" pattern, it will classify it as a face.

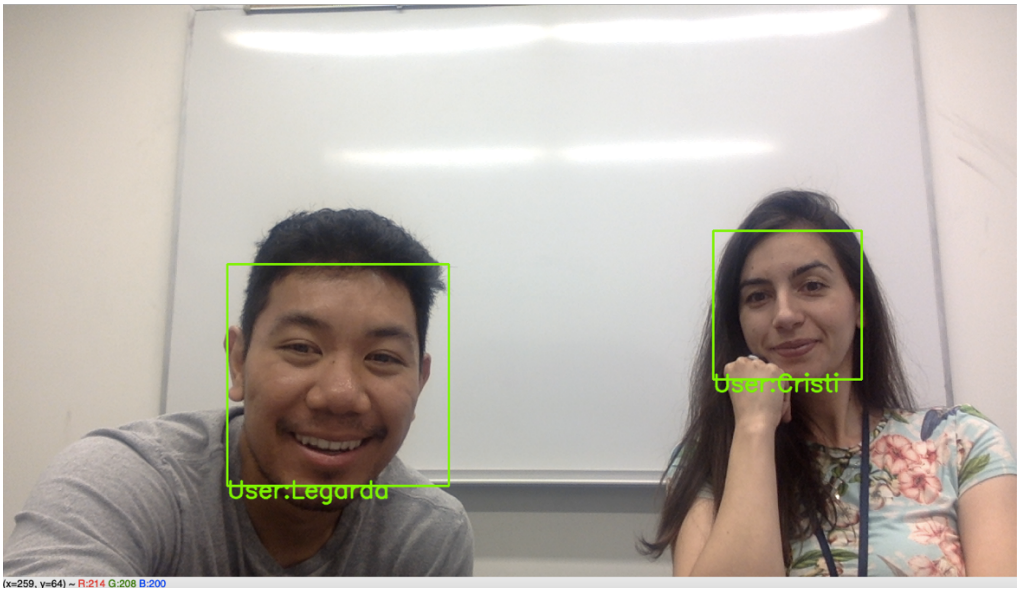
Live Face Detection

Haar Live Webcam Feed Detection



Live Face Recognition

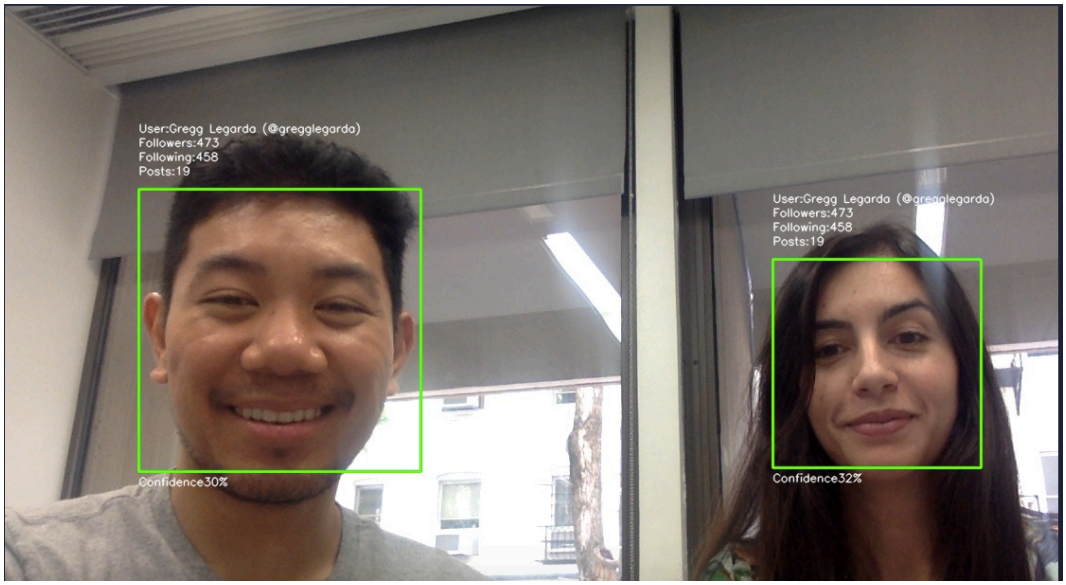
Haar Live Webcam Feed with Gradients



We played a little bit on the number of pictures in each target class, along with gradients, to have this result. I was about 20 pictures for Cristina Giraldo, and about 10 for Gregg Legarda. This is because the Algorithm had biases when different numbers of datasets were presented. Sometimes, it would be both “Cristi” and sometimes it would be both “Legarda”.

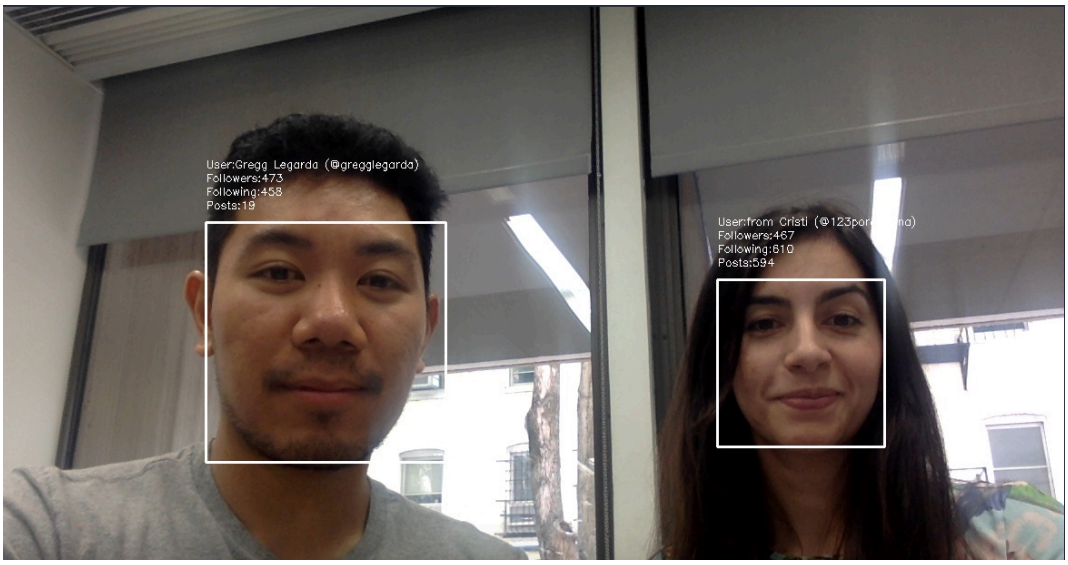
Facial recognition with web scraping

Haar Version



Haar Cascades was very fast and very good at detecting faces but was inaccurate. We made it work at some point, but we needed a bigger data set. At this point, we also have applied Instagram web scraping in the mix. It shows the followers, following, and posts of the user.

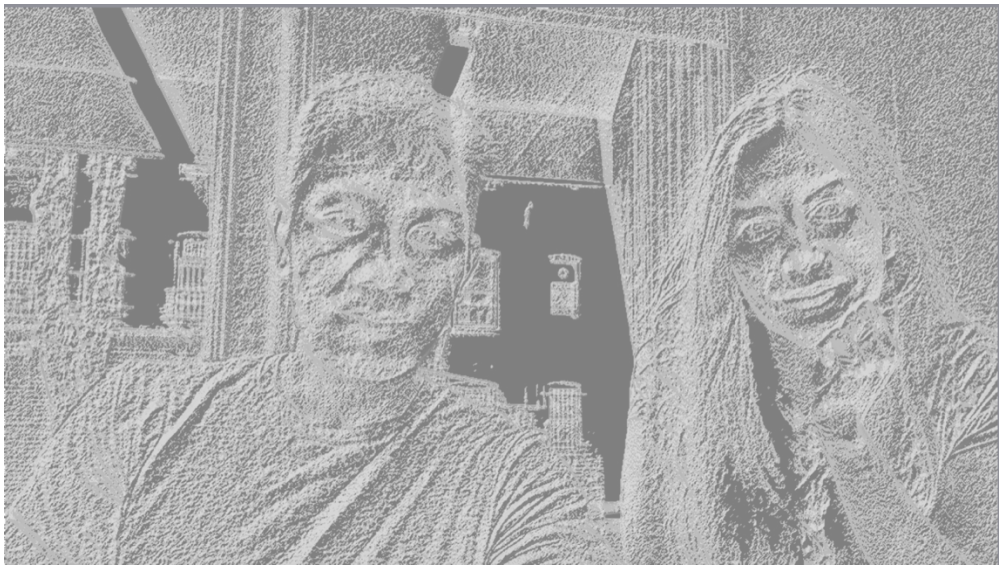
HOG Version



HOG on the other hand was slower but is more accurate, there were no problems with recognition.

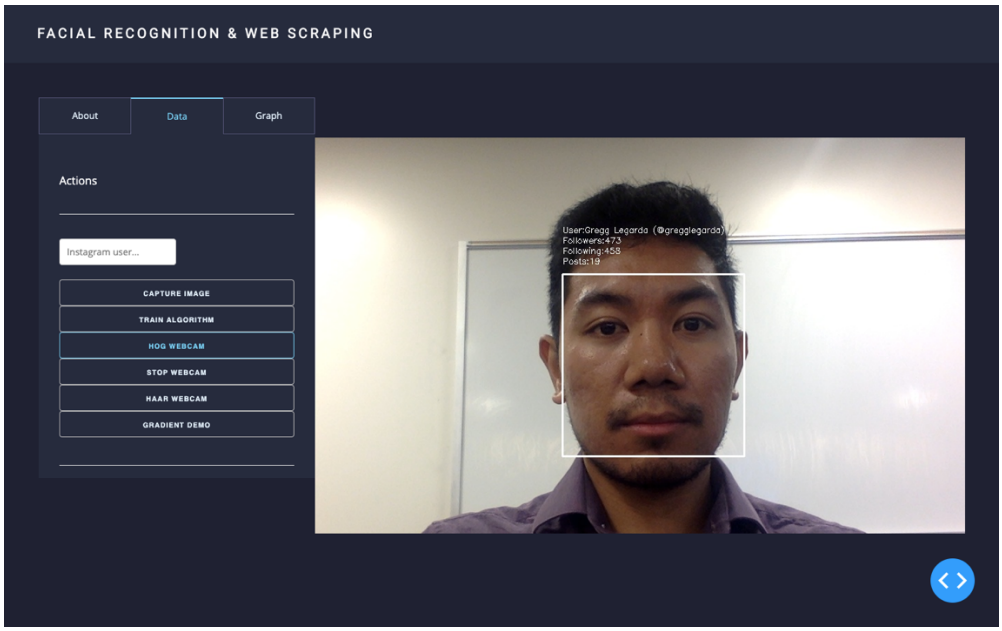
Other Added features

Gradient View



This is an example of implementing gradients in the image. We manually applied this to the Haar facial recognition to make it classify “Gregg” and “Cristina” correctly.

Dash Plotly Web Application

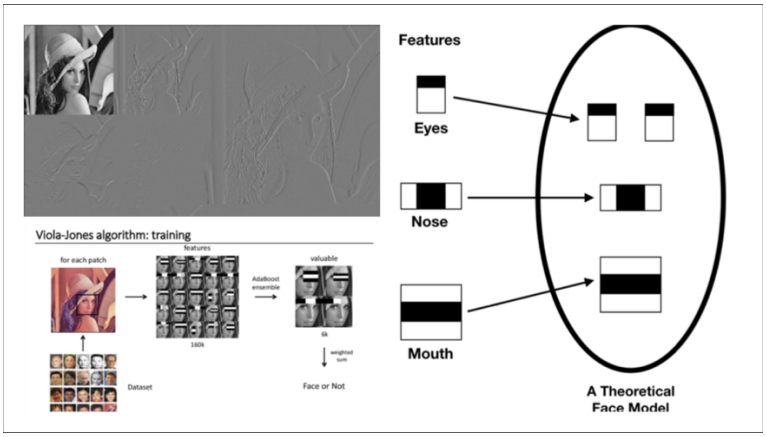


This web application allows the user to train the algorithms easier by capturing and adding their own images.

SUMMARY AND CONCLUSIONS

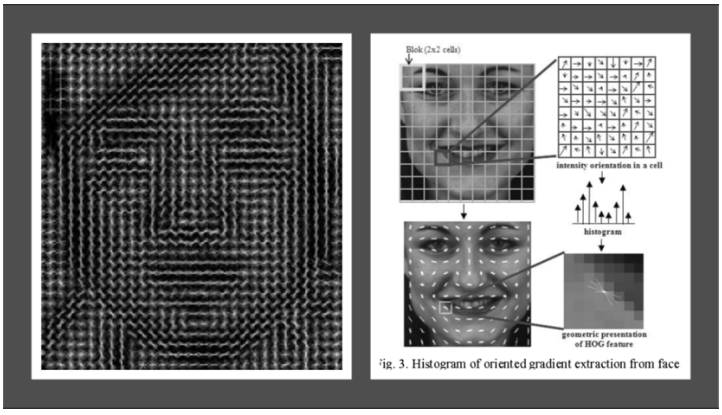
I learned about different types of facial recognition algorithms and their associated pros and cons. For Haar, the advantages are that it is faster than HOG. It is also great at face detection but sometime have more false positives than HOG. This is because it is edge and line based. It detects the differences in intensity of the pixels in the “key areas” of the face. Its disadvantage is, it is less accurate than HOG in face recognition. It can only be better, if we created a whole cascade that is trained but this will take a lot of computing power, a large database and is time consuming if we want an accurate recognition.

Haar Model



Similarly, HOG is also great at face detection and doesn’t over-compensate as much like Haar. It has a lower level of false positives for detection. For recognition, it is better than Haar because it is based on gradients rather than lines. The biggest disadvantage is that it is slower than Haar. We also touched up a little bit on Convolutional Neural Networks (CNN but we found out that it needed a lot of computing power and training data and we would need to run it in another server for training. The advantage of CNN is that it doesn’t use pretrained datasets and it learns as it is trained.

HOG Model



Some improvements that we can do if we have more processing capacity, is train individual cascades for recognition since this will both be fast and accurate. But our time and resources are limited so we decided to use a different route.

In the far future, I want to merge this into a drone or a portable camera to identify people. Since it can be connected to a database in the internet, it can be used to identify and pull information of people identified and can be used for law enforcement purposes.

CALCULATION OF THE PERCENTAGE

File Name	Created	Modied	Ours	Total	Copied
face_train.py	6	5	11	45	34
faces.py	14	21	35	59	24
gradient.py	23	2	25	29	4
capture_image.py	29	2	31	31	0
encode_faces.py	0	12	12	40	28
insta_info_scraper.py	75	2	77	98	21
app.py	265	18	283	381	98
Total	412	62	474	683	209
			Ours	0.693997072	0.69399707
			Copied	0.306002928	0.30600293

Group					
Total	209	Modified	62	Numerator	147
Total	209	Created	412	Denominator	621
				Total both copied	23.6714976

Individual					
Total	104.5	Modified	31	Numerator	73.5
Total	104.5	Created	206	Denominator	310.5
				Total Individual copied	11.8357488

REFERENCES

Docs.opencv.org. (2018). OpenCV: Face Detection using Haar Cascades. [online] Available at: https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html [Accessed 21 Aug. 2019].

Rosebrock, A. (2014). Histogram of Oriented Gradients and Object Detection - PyImageSearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ [Accessed 21 Aug. 2019].