Face To Face: An accuracy-time weighted comparison of facial recognition methods and practical differences

Hirsh Guha
Princeton University
hguha@princeton.edu

Josh Cohen
Princeton University
jmc16@princeton.edu

Abstract

Object detection, and in particular facial recognition, has been an active and important area of research since the 1960s. Although humans recognize faces without effort or delay, recognition by a machine is still a non-trivial task. Facial recognition has broad applicability in a variety of fields, including user authentication, person identification, video surveillance, criminal investigations, data privacy, gaming, and photography. In this paper, we study three different methods for categorizing faces from a variety of images: Haar Cascade Classifiers, Histogram Oriented Gradients, and Convolutional Neural Networks. Using the FDDB dataset, we will examine how each of these models performs on various image types by generating bounding boxes, time, and accuracy statistics. We then use this to analyze the situations that favor one model over another, and discuss the trade-offs present in various scenarios. Lastly, we examine the use of principal component analysis(PCA) and Support Vector Machines(SVM) in creating generalizations and predictive capabilities beyond detection on a single image.

1 Introduction

In recent decades, tasks that have traditionally been performed by humans have increasingly been automated and are now performed by computers. There is good reason for this - computers are faster, more reliable, and can even outperform humans in certain tasks. Increasingly, researchers have attempted to include object detection, including facial detection, in this category. Facial recognition has wide ranging applications, including data privacy, user authentication, person identification, video surveillance, criminal investigations, gaming, photography, and many more. Facial recognition is still far from a solved problem, and we wanted to investigate and analyze several currently-used methods in order to answer and explore the following four questions and topics: 1) the various methods by which facial detection can be performed 2) how well those methods perform on various metrics 3) in what scenarios certain methods are superior to others and 4) what analysis can be done on recognized faces.

To this end, we will explore anc evaluate Haar Cascade Classifiers, Histogram Oriented Gradients, and Convolutional Neural Networks in order to ascertain the strengths and weaknesses of these methods and why those strengths and weaknesses matter in a real-world setting.

2 Related Work

We note that very little if any work has been done in comparing facial recognition methods against each other, and the primary work that does by Mondal et al. analyzes various methods at a high level, with no pipeline for automatic accuracy detection, and generalized results [11]. There is very little precedent for a large scale implementation of specifically facial detection comparisons, which is what we aim to provide.

The study of facial detection, or more broadly, object detection, has a long history, with papers on the study of human factors in image interpretations dating back as early as 1961 [16]. The three approaches we used are well-implemented in a real world setting, as we will discuss later, but they are also well discussed in the theoretical sense in object-detection literature.

The Haar cascade classifier is one of the most common implementations of facial recognition in use, mainly due to the readily pre-trained model available in the OpenCV package[8]. The model has training weights for various objects and body parts, such as the face, the eyes, a smile, a full body, and even a series of models for cat images [1]. This method has applications to more general recognition problems, but these problems can be far more difficult, as Bailing Zhang [17] studied. However, the ideas that will be used in this paper come from the paradigm-setting paper by Paul Viola and Michael Jones in which they first described the method of rapid object detection using a boosted cascade of simple features[15], though we will examine this further in Section 5.

Histogram Oriented Gradients(HoG) as a method is considered one of the faster object detection methods and therefore is commonly used in surveillance technology[4, 12]. Pang et al. found a way to use HoG in conjunction with Support Vector Machine algorithms to take what is one of the more accurate methods, and increase the speed through feature reuse and sub-cell based interpolation to efficiently compute the HOG features for each block [12].

Convolutional Neural Network (CNN) is a deep learning method which is well described by Albawi et al as a mathematical linear operation between matrices with multiple layers; including a convolutional layer, non-linearity layer, pooling layer and fully-connected layer [3]. This has use cases far beyond facial detection, yet in some cases may be the most accurate in this application[14].

3 Dataset

Our first dataset comes from the Face Detection Data Set and Benchmark (FDDB)[7], a collection of 5171 faces in 2845 images created by researchers at the University of Massachusetts - Amherst. These images were taken between 2002 and 2003, and are quite varied in setting and in number of faces. Additionally, they are labelled with (correct) face coordinates, making it relatively simple to compare the results of our models with the true values without needing to manually check each image.

For results via eigenvalues, we use the Labeled Faces in the Wild(LFW) dataset provided in scikit. For seven of celebrities, we can ensure that we get at least 70 labeled images of each. These celebrities include Ariel Sharon, Colin Powell, Donald Rumsfeld, George W Bush, Gerhard Schroeder, Hugo Chavez, and Tony Blair. Using dozens of images of each of them will be useful in performing some of the more classic machine learning analysis that we have done in previous homework assignments.

4 Methods

4.1 Models

We implemented the following three models of object detection: Haar Cascade Classifiers, Histogram Oriented Gradients, and Convolutional Neural Networks. In our implementation, we used two main python libraries: openCV, a library designed specifically for computer vision problems and containing a number of trained model weights, and dlib, a C++ machine learning toolkit with various high performance methods[1, 2]. Here we will describe how the three methods work:

Haar Cascade Classifer Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" [15] and that we will describe in more depth in Section 5. A cascade function is trained from numerous positive and negative images. It is then used to detect objects in other images.

Histogram Oriented Gradients The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy [6].

Convolutional Neural Networks In deep learning, a convolutional neural network(CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on the shared-weight architecture of the convolution kernels that shift over input features and provide translation equivariant responses. Our particular implementation via a trained Dlib model also implements Max-Margin Object Detection (MMOD), as an optimization over any detection method. [9].

Eigenvalues We adapt the method described in scikit's documentation on facial recognition example using eigenfaces generated by Principal Component Analysis and correctly predicted using Support Vector Machines [13]. We will break the images into a test set and training set, and use PCA to break images into 150 components to extract 150 ”eigenfaces” and measure the accuracy of predictions of these eigenfaces.

4.2 Evaluating the Models

The FDDB dataset comes with labels for each of the faces present in each image. However, making this usable and automated was nontrivial. First, we had to parse the annotations, which were simply given as a text file rather than a CSV file or JSON. Then, since the FDDB database gave bounding ellipses, we converted these to bounding boxes simply using the center and axes. The given ellipses were angled, yet we ignored the angles, both because all the angles were quite small and had little impact on the results, and because the location of a face is not exact (For instance, does the bounding box include the forehead, chin, and/or ears?). We were not aiming to reproduce the labels exactly but rather to test which faces were recognized, so we allow for some approximation.

Specifically, in order to compare the bounding boxes generated by our models with the true values, we did the following:

1. Find the closest predicted bounding box (comparing corners) to each true box, recording the (Euclidean) distances.
2. Find the smallest distance among the above and consider that pair of boxes. If the distance is < 50 pixels, remove these boxes from their respective sets, and count this as a correct image. Otherwise, we are done: all remaining faces are false positives or negatives.
3. If we did not finish in Step 2, repeat the above until there are no more boxes in one of the sets.

The idea with the above is that the bounding boxes fall into 3 categories: those correctly found (in which there are true and predicted boxes close to each others), a false positive (a predicted bounding box nowhere near a true one), and a false negative (a true bounding box nowhere near a predicted one). Since the boxes are not exact, we want to know if there is a pair that is likely to be a match. The true boxes each correspond to actual faces, so we want to see which (if any) of the generated boxes most likely corresponds to that face. We give leeway of 50 pixels (about 1/8 of the height of most images), since by manually looking at the outputted images, we found that to be a good dividing line between correct and incorrect pairings. Then, we know that of the remaining boxes (after step 3), all of the true boxes are false negatives (they had no corresponding predicted box), and all the predicted boxes are false positives (there was no corresponding true box).

4.3 Metrics

Our metrics for situational comparison between the facial recognition models are accuracy and time.

Measuring time is very simple but very important in many applications of facial recognition. For instance, if the HoG method was very slow on images with multiple faces, it would be a poor fit

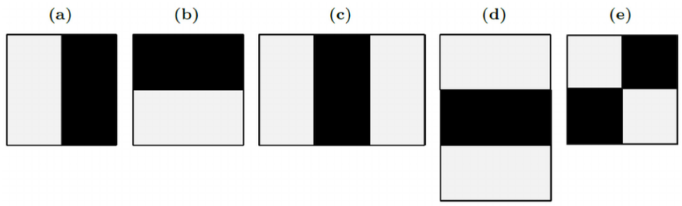

Figure 2: Various common patterns for Haar shapes

for use in a security camera trying to continuously recognize faces in images with dozens of people. Thus, we study and discuss the time taken by each method in various scenarios, as this information must be part of any useful analysis.

We compute accuracy to determine whether our methods will correctly identify faces. However, there are actually several pieces of information that we need for useful analysis. We need to know which faces were correctly identified, which faces were missed (false negatives), and which objects were incorrectly marked as faces (false positives). As described in the previous section, we can generate all of these metrics, since we use several heuristics to match the bounding boxes and determine if the fit is accurate or not. We did manually examine many of the resulting images, and did not find any misclassifications, which adds confidence to our approach and our 50px margin being an acceptable threshhold.

5 One Model in-Depth: Haar Cascade Classifiers

A cascade of boosted classifiers working with Haar-like features is a special case of ensemble learning known as boosting. Cascade classifiers are trained both on a few hundred sample images that contain the object we want to detect and on other images that do not contain those objects. In our case, we trained the model on hundreds of images of faces (though more generally, the same methods apply to any objects, such as cats, cars, houses, etc). The algorithm we implemented is quite popular due to being readily available with the OpenCV library in Python and allows for implementation of facial detection is very few lines of code. It is a method was discussed by Paul Viola and Micheal Jones in what is known as the Viola–Jones object detection framework, which has the following steps: [15]:

- Haar Feature Selection
- Create integral image
- Adaboost Training
- Cascading Classifiers

5.1 Haar Feature Selection

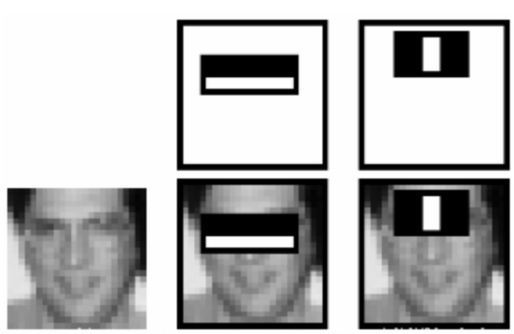

Figure 1: Haar features on a face, with additive (white) and subtractive (black) pixels

There are some common features that we find on most human faces, such as a dark eye region compared to upper-cheeks, or a bright nose bridge region compared to the eyes. These characteristics are called Haar Features. Figure 1 shows an example, the measuring the difference in intensity between the region of the eyes and across the upper cheeks. The feature value is computed by summing the pixel values in the black area(the grayscale color) and subtracting the pixels in the white area.

There are several types of rectangles that can be applied for Haar Features extraction, shown in Figure 2.

Then, we apply this rectangle as a convolutional kernel over our whole image. In order to train the best model, we apply all possible dimensions and positions of each kernel, but a simple 24*24 images would result in over 160000 features of a additive/subtractive pixels values, which is computationally intractable. Instead, once the good region has been identified by a rectangle, we compute the rectangle features using the integral image principle, which speeds up the operations significantly.

5.2 The Integral Image

The Integral Image is an intermediate representation which allows any rectangular sum to be computed simply, using only four values. Suppose we want to determine the rectangle features at a given pixel with coordinates (x,y). Then, the integral image of the pixel is the sum of the pixels above and to the left of the given pixel.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image.

Computing the entire integral image can be done using a recurrence, thus requiring only a single pass over the original image. Indeed, we can define the following pair of recurrences :

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

where $s(x, y)$ is the cumulative row sum and and $s(x1) = 0, ii(1, y) = 0$ [15].

5.3 Adaboost Training

Given a set of labeled training images (positive or negative), Adaboost is used to select a small set of features and train the classifier. Looking at our example of 160000 features, most are likely to be quite irrelevant. The weak learning algorithm that the boosted model is built on is designed to select the single rectangle feature which best splits the image. Now that the features have been selected, we apply them on the set of training images using Adaboost classification, which combines a set of weak classifiers to create an accurate ensemble model. As described in Paul Viola and Micheal Jones's Paper, this results in 6000 features instead of 160000, and an accuracy of 95% of faces correct located and marked, and a false positive rate of 1 in 14084[15].

5.4 Cascading Classifiers

In our case, most of the image consists of irrelevant information (ie, background pixels that are not faces). Therefore, it would be quite inefficient to give equal importance to every region of the image, and we should mainly focus on regions that are most likely to contain relevant information. Viola and Jones achieved this increased detection rate while reducing computation time by using cascading classifiers.

The key idea is to reject regions that do not contain faces while identifying regions that do. Note that, since we want to properly identify the face, we are very concerned with minimizing the false negative rate: regions that contain a face but are marked as not containing one.

These classifiers are simple decision trees: if the first classifier is positive, we move on to the second, and if the second classifier is positive, we move on to the third, and so on. Any negative result at some point leads to a rejection of the region as potentially containing a face. The initial classifier eliminates most negative examples at a low computational cost, and the following classifiers eliminate additional negative examples but require more computational effort. When training such a model, the variables are the the number of classifier stages, the number of features in each stage, and the threshold of each stage.

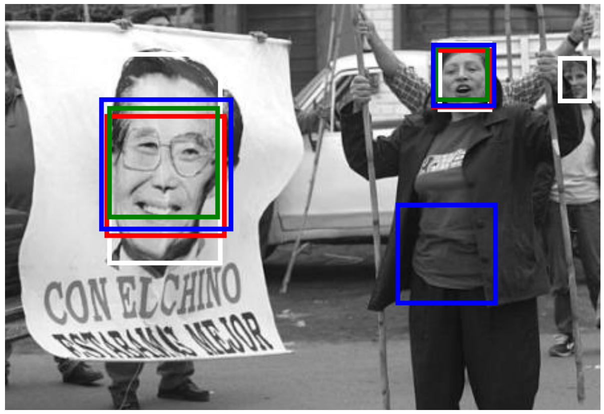

Figure 3: Example of classifiers: labels (white), HOG (red), CNN (green), Cascade (blue)

	Overall			1 Face in Image			2 Faces in Image			>2 Faces in Image		
Model	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.
HOG	.747	.987	.754	.940	.976	.963	.796	.993	.800	.566	.998	.567
Cascade	.658	.884	.720	.783	.845	.914	.687	.892	.750	.533	.933	.554
CNN	.783	.952	.815	.833	.904	.914	.869	.967	.895	.690	.993	.693

Table 1: Accuracy, Precision, and Recall based on number of faces in image

6 Expected Behaviour

In this project, we would like to be able to identify faces accurately using the various methods described above as well as to be able to identify the main strengths, weaknesses, and differences of each model. From our research in Section 2 on related work, we expect that HoG will be the fastest algorithm, followed by Haar Cascade Classifier and finally CNNs. However, we also expect that CNN should be the most accurate algorithm, with HoG and Haar Cascade Classifiers performing roughly equal, and having trouble with finer details, such as small faces or lesser contrast in the image. Finally, in order to understand facial recognition in the context of predicting labeled faces, we expect PCA to be able to break down the faces into meaningful eigenvalues that will roughly show simplified facial details, and from Precept 8, these demopoisitons will allow for data compression, and better visualization of general facial construction [10].

7 Results

The above tables show the results of the classifiers on the FDDB dataset, broken down both by the number of faces in the image as well as the true size of the face. Table 1 shows the overall accuracy, precision, and recall for each classifier, as well as those statistics for faces in image with 1 face, 2 faces, and more than 2 faces. Table 2 shows the accuracy of each classifier on small ($<$ 150 pixels) and large ($>$ 300 pixels) faces, where the size of a face is defined by the length of the diagonal in the labelled bounding box. We also show the total time each classifier took on the entire dataset. Finally, we show an example image in Figure 3. On this image, all models found 2 of the real faces, all missed a small one, and the Cascade classifier found a false positive.

	Accuracy - Small	Accuracy - Large	Time (s)
HOG	0.638	0.877	186
Cascade	0.603	0.774	107
CNN	0.721	0.671	13,673

Table 2: Accuracy on small and large images and total time taken

	Precision	Recall	F1-Score	Support
Ariel Sharon	0.71	0.38	0.50	13
Colin Powell	0.80	0.87	0.83	60
Donald Rumsfeld	0.89	0.63	0.74	27
George Bush	0.83	0.98	0.90	146
Gerhard Shroeder	0.95	0.76	0.84	25
Hugo Chavez	1.00	0.53	0.70	15
Tony Blair	0.97	0.81	0.88	36

Table 3: The classification report of the predicted versus true labels for the LFW dataset trained via SVM

To analyze faces using PCA and SVM by breaking the face images into eigenvalues, we take 1288 images, extract 1850 features from each, and extract 150 components via PCA. Those 150 decomposed faces are then used to train an SVM model that is then used to create Table 3 via scikit's classification report.

8 Discussion

In contrast to what we predicted in Section 6 on Expected Behaviour, Table 2 shows that the cascade classifier was the fastest, almost twice as fast than the next fastest, the HoG method. As expected, the CNN method was by far the slowest, taking almost 4 hours compared to 2-3 minutes for the other methods.

Overall, we see that CNN is the most accurate in general, yet this only gives a partial picture. HOG consistently had far fewer false positives (seeing an image where there is none) than the other 2 methods, and thus had a higher precision. Cascade classifiers, on the other hand, consistently had higher levels of false positives and false negatives (missing a true face). Thus, in general, we see a speed-accuracy tradeoff, with the fastest algorithm being the least accurate (as well as having low precision and recall) and the slowest being the most accurate.

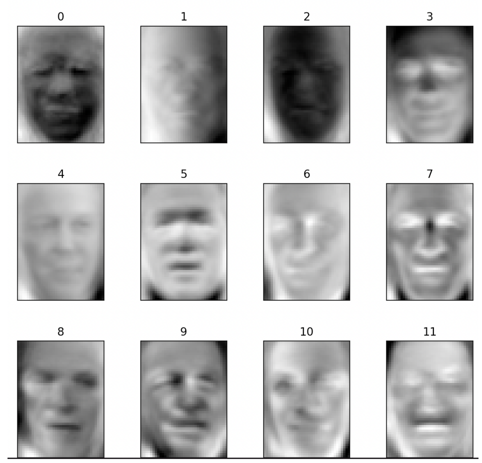

Figure 4: The first 12 eigenvalue components from the LFW image dataset

However, we wanted to delve deeper into the strengths of weaknesses of each classifier, so we also broke down the results by the number of faces in the image and by the size of individual faces (Ideally, we would have liked to include additional characteristics, such as partially obscured faces, angled faces, low-light, etc, but these features are not labelled on the dataset. We can calculate the number and size of faces quite easily from the labelled bounding boxes). Table 1 shows that both HOG and the Cascade classifier become significantly less accurate and find significantly more false negatives as the number of faces increases (the number of false positives, on the other hand, actually decreases). Interestingly, however, the CNN actually becomes more accurate with 2 faces in the image, and is far better on images with at least 2 faces than the other two. In fact, CNN does quite poorly with only 1 face relative to HOG, with lower accuracy, precision, and recall.

These results align with the data on small and large faces, presented in Table 2. While HOG and the Cascade classifier perform much worse on small faces than large ones, CNN does the opposite. These two features are not independent; we would expect images with many faces to have smaller ones, and vice versa. Thus, this may offer a potential explanation for why CNN performs much better on images with more faces - it is better on smaller faces. Naively, we would expect small faces and images with more people to be more difficult, and unfortunately the black box nature of CNNs makes it unclear why this case is different.

Thus, we see that various classifiers are best used in different situations. Cascade classifiers are the best choice if speed is paramount (say, for real-time facial detection and in situations where computation is limited). HOG is best in situations where there is only a single or large faces and where avoiding false positives is very important. And CNNs are the best in cases where there are many or small faces and time is not a limiting factor. Finally, we note that the accuracy was relatively poor overall, with almost all accuracy rates under 90%, and only a few above 80%. It is clear that there is still much work to be done in developing more accurate and more generally applicable models.

Lastly, we examine the results of our efforts in breaking down a series of images into its eigenvalues, and showing the top 12 of those in Figure 4. In Table 3, we show the precision, recall, f1-scores, and support as provided by the scikit classification report given the top 7 labels, along with the predicted values and true values for each. We note that these two aspects of facial detection can be used to augment each other. For instance, a common use case of facial detection is to find criminals by detecting facts based on mugshot or other images. This allows us create a more specified model - from the general model that was trained on faces, we could retrain the same model with a specific person's face - as we essentially can create a dataset via our facial detection methods. Then using that detected face, we would want to then use methods such as SVM to search through hundreds of hours of surveillance footage from different cameras in order to find a match on the suspect's face. However, many cameras produce images of varying quality, and the suspect may be wearing hairstyles, or facial fair, or a number of other variables that would make it hard for the models to detect. Using a method like PCA for dimension reduction can be useful for image compression, but the outputted eigenvalues are useful for constructing various predictions on how the suspect may have changed their look [10].

9 Conclusions

We were able to successfully implement three different facial recognition methods - Haar Cascade Classification, Histogram Oriented Gradients, and Convolutional Neural Networks - in a way that allowed easy and intuitive comparison between them on the same image set, based on a configuration file. In addition to simply being able to show a series of methods that can accurately detect faces on any common image, we were able to analyze situations where one method might be superior to another, as discussed in Section 8. Finally, we show eigenvalue decomposition via PCA and SVM on another image set in order to extend our project to its possible uses. Thus, we have shown analysis of a complete pipeline that begins with a single image, predicts the location of the face, and then can use that face to then locate other instances of that same face.

10 Future Work

First, we could extend this work to incorporate other models and forms of facial detection to get a more complete view of all the current methods available. This would help us to determine the best models for various types of images. For instance, we could investigate creating a You Only Look Once face detector or Single Shot Detector[5].

Second, delving deeper into specific features of images that make certain classifiers more or less effective would be useful. For instance, is there are difference when faces are in shadow or partially turned away from the camera? With a sufficiently-labelled dataset, this would help expand our discussion of the strengths and weaknesses of various classifiers.

Finally, one area of computer science that intersects with philosophy is the ethics of facial recognition. We recognize when discussing the relevance of facial recognition that it is not all positive. Facial recognition has become somewhat of a controversial topic in the mainstream as of late. Because of its use in various protests, or as a governmental tracking tool, the public is understandably wary of advancements in the field that present privacy concerns. Deep fakes are another area of worry, wherein a person's face can be grafted onto another person's body, or a person can be made to look as though they are doing something incriminating through entirely computer generated imagery, and it is these same methods that we explored that enable these malicious behaviours. An entire line of research can be done just into these ethical concerns, and what steps can be taken to minimize adversarial behaviour.

References

[1] Opencv python library documentation. *OpenCV*.

[2] *dlib C Library*, Mar 2021.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Muhammad Awais, Muhammad Javed Iqbal, Iftikhar Ahmad, Madini O Alassafi, Rayed Alghamdi, Mohammad Basheri, and Muhammad Waqas. Real-time surveillance through face recognition using hog and feedforward neural networks. *IEEE Access*, 7:121236–121244, 2019.

[5] Ambika Choudhury. Top 8 algorithms for object detection one must know, Feb 2021.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[7] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[8] Abid K. Face detection using haar cascades¶, 2013.

[9] Davis E King. Max-margin object detection. *arXiv preprint arXiv:1502.00046*, 2015.

[10] Sulin Liu and Xioyan Li. Precept 8: Dimension reduction: Pca, svd and nmf. *Precept 8*, page 5, March 2021.

[11] Sudipto Kumar Mondal, Indraneel Mukhopadhyay, and Supreme Dutta. Review and comparison of face detection techniques. In Mohuya Chakraborty, Satyajit Chakrabarti, and Valentina E. Balas, editors, *Proceedings of International Ethical Hacking Conference 2019*, pages 3–14, Singapore, 2020. Springer Singapore.

[12] Yanwei Pang, Yuan Yuan, Xuelong Li, and Jing Pan. Efficient hog human detection. *Signal Processing*, 91(4):773–781, 2011.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] M Sivaram, V Porkodi, Amin Salih Mohammed, and V Manikandan. Detection of accurate facial detection using hybrid deep convolutional recurrent neural network. *ICTACT Journal on Soft Computing*, 9(2), 2019.

[15] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[16] Joseph Zeidner. *Human factors studies in image interpretation: vertical and oblique photos*, volume 120. US Army Personnel Research Office, 1961.

[17] Bailing Zhang. Reliable classification of vehicle types based on cascade classifier ensembles. *IEEE Transactions on intelligent transportation systems*, 14(1):322–332, 2012.

11 Appendix: Roles and Responsibilities

11.1 Hirsh Guha

Hirsh handled the initial set up of the three machine learning models, Haar Cascade Classifiers, Histogram Oriented Gradients, and Convolutional Neural Networks as well as a configuration file that allowed us to easily manipulate the image data passed in, and the prepossessing. Hirsh also handled the set up of the PCA and SVM models that would result in the classification data from Table 3 and the eigenfaces in Figure 4. Lastly, Hirsh was responsible for completing the related work literature review, and diving deep into Viola et al's paper on cascading classifier in order to discuss one-model in depth.

11.2 Josh Cohen

Josh found the FDDB dataset, which met our requirements of being large enough, having varied types of images, and having labels to identify regions as faces. He then did the preprocessing of the data, such as converting the bounding ellipses into boxes, and wrote the parts of the code that enabled automatic accuracy, precision, and recall calculations based on the various image features we identified (number of faces and size of faces). Josh used the algorithms and infrastructure that Hirsh set up to run the models on all the data (which took about 4 hours each time) and calculate the relevant results and statistics given in this report, writing the associated information about these results and the relevant discussion.

All other relevant sections of this paper, including the poster, proposal, ideas, and code writing were worked on by both Hirsh and Josh together. We feel as though we both contributed equally to this project.