Comparative Analysis of Haar Cascades and Deep Learning Models for Real-time Face Blurring

Brian Justin Bishop II
Department of Computer Science
University of South Carolina Upstate
Spartanburg, South Carolina, USA
September 16, 2023
bjbishop@email.uscupstate.edu

ABSTRACT
In an era dominated by constant surveillance and the ever increasing importance of privacy, facial blurring has become more important than ever. This paper presents an analysis of Haar cascades and deep learning models, focusing on their efficacy in facial blurring for image processing measured in accuracy and the algorithms overall confidence. The digital age demands new techniques for privacy preservation, with facial blurring in images standing out as a big part of how this is done. Haar cascades are traditional and lightweight, and have long been utilized for face detection, whereas deep learning models, with their advanced learning capabilities, are emerging as powerful tools for facial detection tasks. I will evaluate both methodologies based on accuracy, confidence, and ability to analyze diverse image conditions. Initial results indicate that while Haar cascades provide a faster and easier solution, deep learning models demonstrate superior accuracy and versatility. Deep learning models seem to do especially well in challenging scenarios with varied lighting, poses, and occlusions compared to Haar Cascades where even a simple profile face requires extra work. The goal of this paper is to show the importance of selecting appropriate facial detection mechanisms based on specific use-case requirements and the ongoing battle everyday people have with their privacy.

Keywords
Haar Cascades, DLM, Deep learning model, Face blurring, AI, Face detection, Bounding, Neural Networks, Deep learning

1. INTRODUCTION
In today's interconnected digital age, cameras and recording devices permeate many facets of our daily existence. From surveillance cameras stationed in public spaces to the presence of smartphone cameras in everyone's pockets, it's undeniable that recording and photos are being taken everywhere. While this ability to capture moments offers a ton of great advantages, including documenting real-time events or preserving cherished memories, it also presents pressing challenges, especially concerning individual privacy and the safeguarding of personal data.

Face blurring in images has emerged as a solution to such privacy concerns. This technique uses automated detection through different algorithms to mask an individual's face in captured images, making sure their identity remains protected. This is crucial in a world where not everyone is comfortable with their likeness being shared, used, or potentially misused. Key motivations for the adoption of face blurring in images include: Privacy Preservation, Consent and Ethics, and Compliance with Regulations, among others. As technology and society continue to evolve, understanding and addressing these concerns becomes imperative.

Privacy preservation is still very important in the digital age. Large amounts of sensitive data are uploaded, shared, and viewed every day. As images circulate on platforms, whether social media, news websites, or digital databases, the potential for misuse or unintended exposure of an individual's identity increases. Blurring faces in images serves as a proactive measure to mitigate such risks. By making someone's face unrecognizable, this technique ensures that the individuals in the photos remain anonymous. This not only safeguards personal privacy but also reduces the potential for targeted cyber-attacks, doxing, and other malicious actions that capitalize on identifiable information.

In a world dominated by immediate content sharing, not all images captured are done so with the explicit consent of those featured. Whether taken in public places, during events, or even on accident as part of a background scene of someone's innocent photo, photos can be passed and viewed without the knowledge or approval of the subjects. Ethical considerations come to the forefront now. By blurring faces in images, content creators and viewers demonstrate a commitment to ethical practices, respecting the autonomy and wishes of individuals. It's a tangible way to ensure that, even in the absence of consent, a person's right to their own likeness is still respected and protected.

Over the years, various jurisdictions worldwide have implemented more and more tight data protection and privacy laws. These regulations, such as the General Data Protection Regulation

(GDPR) in Europe, mandate the protection of personal data, which includes images that can identify an individual. Non-compliance can result in hefty fines and legal consequences. Blurring faces in images is not merely an ethical choice but, in many contexts, a legal requirement. It ensures that organizations, businesses, and individuals align with stipulated regulations, avoiding potential legal ramifications while upholding the standards set forth by governing bodies.

With these concerts in mind the development of algorithms that hastily do this task instead of having to edit images by hand. One such algorithm is Haar Cascade. It is fast and quick to give you results however is not always the most reliable. A quick example shown in the following figure.

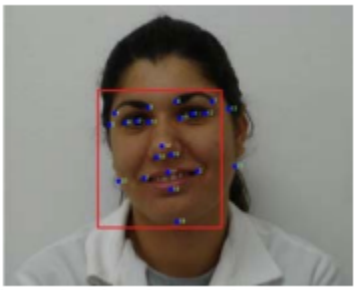


Figure 1: Face located with Haar classifiers [12]

Deep learning models on the other hand can use various different methods to achieve the same result. The only difference is they typically are more accurate but slower to give you an output. The reason for this is that typically they will use a multitude of different classifiers to achieve this result. In turn running slower but also being more correct in identifying a face.

2. LITERATURE REVIEW

Face detection is a fundamental task in regards to video with any kind of filter. It involves not only image processing and in some cases real time image processing but human-computer interaction. Facial detection has been a multi-year journey with new advancements every year. This section will provide details on different advancements in methods that will be discussed later such as neural networks and the Haar feature-based cascade classifiers.

The fundamentals of the Haar cascade algorithm have been steadily researched ever since its inception in 2001 by Paul Viola and Michael Jones in their paper titled, “Rapid Object Detection using a Boosted Cascade of Simple Features” [13]. A big outstanding factor in why this algorithm was such an advancement was it at the time was able to keep high frame rates without having to resort to different detection methods such as image differencing or skin color detection.

In their paper Viola and Jones describe how their algorithm achieved, “a frontal face detection system which achieves detection and false positive rates which are equivalent to the best published results”. While this was published in 2001 the Haar cascade algorithm is still widely used today as it still holds up all these years later. The process is done through evaluating features rather than individual pixels like previous algorithms. This causes the algorithm to be much faster as it doesn't need to evaluate every individual pixel. The algorithm uses rectangles enclosed in windows. It finds the sum of the pixels lying within the white rectangles and subtracts it from the sum of pixels in the gray rectangles (see figure 1) [13].

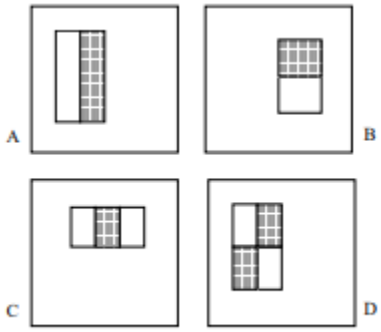


Figure 2: Rectangles enclosed in detection window [13]

Advancements have been made on the Haar cascade features however with more recent iterations moving from rectangles to instead measure edges and the area around a central point.

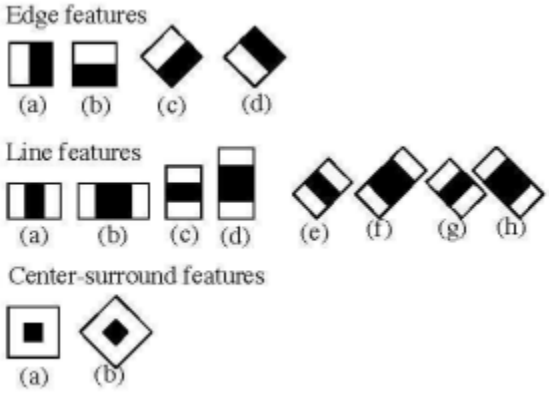


Figure 3: Haar-like features [1]

Deep learning models differ drastically in how they can be implemented compared to Haar Cascades implementation. Deep learning models have to have their own datasets in order to analyze and create their own predictions while Haar Cascades is based purely off already created compiled datasets into XML files that can be found on Github. In order to accurately state the difference between Haar Cascades base algorithm and a deep learning model, here is a quote from a separate research paper that combines three networks in order to better capture faces, "The cascade face detector proposed by Viola and Jones [2] utilizes Haar-Like features and AdaBoost to train cascaded classifiers, which achieves good performance with real-time efficiency. However, quite a few works [1, 3, 4] indicate that this kind of detector may degrade significantly in real-world applications with larger visual variations of human faces even with more advanced features and classifiers" [8]. This directly aligns with my results but that will be saved for a future section.

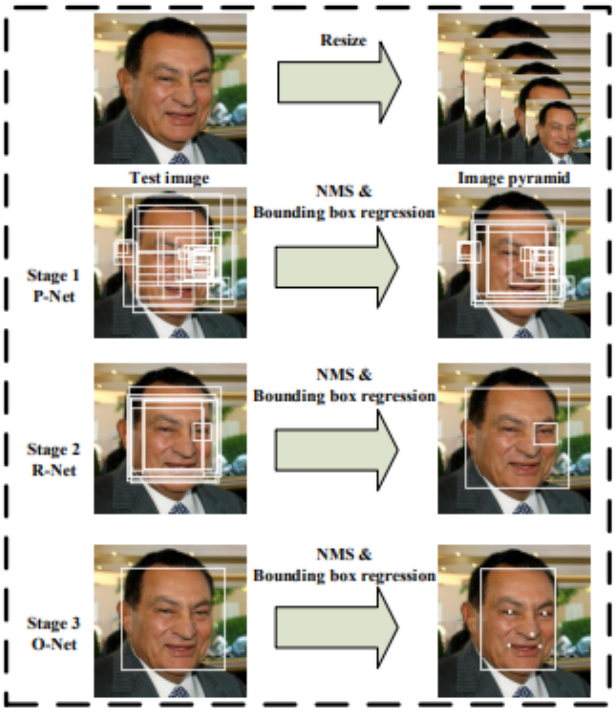


Figure 4: Results of three stage network [8]

In figure 3 you can see the results clearly as a rough overview of how through three stages of analysis of an image the algorithm devised can produce highly accurate results. Firstly a fast Proposal Network produces candidate windows. Stage 2 is next which is their Refinement Network. Finally their Output Network finishes with the last bounding box. Through three stages their network provides much better results with vastly less false positives however is of course computationally more intense.

3. METHODOLOGY

This methodology section delves into two prominent face detection techniques for still images: the Haar cascades method and the deep learning model approach. The Haar cascades method uses basic mechanics, which involves leveraging pre-trained classifiers. A crucial step in this method is the transformation of images into grayscale, optimizing the face detection process. This technique, while efficient, may have its limitations, especially when dealing with certain lighting conditions or orientations. On the other side, the deep learning model approach employs intricate neural network architectures trained on extensive datasets. This method showcases its strength in tackling complex scenarios like partial face obstructions or varied lighting conditions. Though powerful in accuracy, it demands more computational resources. Throughout this section, a side-by-side comparison between the two methods will provide insights into their respective strengths and challenges in different scenarios.

3.1 HAAR CASCADE

This study seeks to effectively detect and blur faces in static images, leveraging the power of Haar cascades and the OpenCV library in Python. The Haar cascade algorithm demonstrates the use of classifiers in OpenCV for facial recognition tasks. Initially, it loads trained XML classifiers for detecting frontal faces, profile faces, and eyes. An image is loaded and converted to grayscale which is often a common step in preprocessing images.

The 'detectMultiScale' method is used in order to detect faces at many different sensitivity levels. These levels are controlled by the 'minNeighbors' parameter. It detects faces (frontal and profile) in both the original and horizontally flipped images to account for left and right profiles. After detection, it adjusts the coordinates of detections from the flipped image to match the original image's coordinate system.

For each detected face, blur filters are added to the face region to anonymize it, and adds text to display the confidence level of detection. Additionally, it detects and blurs the eyes within the detected facial regions. This additional step of eye detection and blurring is specific to faces detected as frontal.

3.2 DEEP LEARNING MODEL

The deep learning model used was MTCNN. This model uses three stages of convolutional networks: the Proposal Network (P-Net), Refine Network (R-Net), and Output Network (O-Net). These networks work together to detect face regions and landmarks on a face such as eyes, nose and mouth. The model operates on an image pyramid to handle varying face sizes, scaling the image to multiple sizes and applying the P-Net at each scale, allowing it to detect faces of different dimensions.

The MTCNN detector is initialized and applied to an image that is first converted from BGR to RGB color space, aligning with MTCNN's input requirements. The detect_faces method of MTCNN performs face detection, providing bounding box coordinates and confidence scores for each detected face. The detected faces are then anonymized using Gaussian blur, a common technique for blurring images. Additionally, the

confidence scores are overlaid on the image to indicate the model's certainty about each detection.

3.3 ALGORITHM COMPARISON

Haar Cascades algorithm computes its rectangular features through integral images. From the Viola-Jones paper these images are calculated through, “The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive: ”

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

Figure 5: Integral Image formula [13]

To put the formula into something more tangible here's an image provided from Viola and Jones paper as well with an explanation.

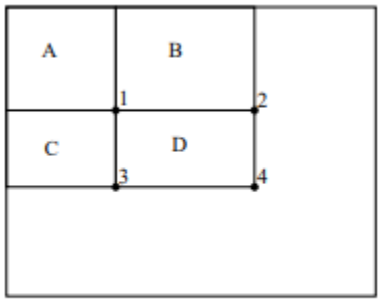
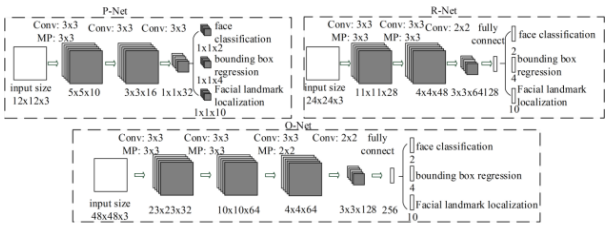


Figure 6: “The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A.The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4 + 1 - (2 + 3).” [13]

The deep learning model will use haar like algorithms but in turn implement other methods such as Convolutional Neural Networks (CNNs). The MTCNN model that is being used like stated above in 3.2 uses a 3 stage network in addition to the Haar cascade structure to aid it.



- Figure 7: Architecture of 3 stage network [8]

In this figure the MTCNN’s 3 stage structure utilizes max pool;ing and convolution in order to strengthen the detector during training.

4. IMPLEMENTATION

4.1 HAAR CASCADE

The implementation of the Haar Cascades and deep learning model algorithms is using Python as its language. This is mainly due to its extensive libraries and functionality when it comes to facial tracking. With libraries such as OpenCV and dlib it makes the integration of frameworks such as TensorFlow and PyTorch for deep learning models seamless. Python was also chosen as the primary language because of the vast community behind it that makes finding and researching topics hands down the smoothest process.

OpenCV (Open Source Computer Vision Library) was chosen as the main library to implement my Haar Cascade algorithm for one main reason, its open source nature allows for plenty of room to adapt and tweak code to different scenarios without the headache of worrying about it breaking anything in its functionality. Its set of image processing functions, coupled with its optimized performance, made it an ideal candidate for real-time applications, a crucial factor in facial tracking. Notably, the library incorporates pre-trained models like Haar cascades, facilitating immediate and straightforward facial detection without necessitating comprehensive training.

This leads to the next topic of the three main classifiers chosen to be implemented: frontal face, profile face, and eye. Coupled together, testing found that frontal face and the eye classifiers from the OpenCV Github were great for straight on images, however once any kind of side of the face was introduced the results were drastically worse. To correct this issue the profile face classifier was also implemented. One thing to note however is that by itself the classifier only detects left sides of faces meaning in order to detect someone facing the right the images had to be reversed, analyzed once more, and then the results put back onto the original image.

4.2 DEEP LEARNING MODEL

This algorithm uses a Python implementation that integrates face detection and anonymization using the Multi-task Cascaded Convolutional Networks (MTCNN) and OpenCV libraries, along with monitoring system resource usage.

The main function, ‘display_blurred_faces’, is designed to process a given image. It starts by loading the image using OpenCV and then converts it from BGR to RGB color format, as MTCNN requires RGB input. The MTCNN detector is then initialized to detect faces in the image. Once faces are detected, the script iterates over each face, retrieving the coordinates and dimensions of its bounding box and the associated confidence score. For each detected face, the corresponding region in the image is blurred using a Gaussian blur, a common technique for anonymizing facial features in images. The blurred region replaces the original face in the image. Additionally, the confidence score of each detection is overlaid onto the image as text.

After processing the image, the function measures the time taken for the entire operation and again checks the system's CPU and memory usage. The final image, with blurred faces, is displayed using OpenCV's ‘imshow’ function, and the program waits for a keypress before closing the image window and releasing resources.

5. EXPERIMENTAL SETUP

For the experimentation side of things there obviously had to be metrics at which I could compare the algorithms. To do this I used three main comparisons: Accuracy, Time to complete, and confidence.

Accuracy was the measure of how many faces were accurately blurred. For instance if an image had 12 faces but only 8 were blurred the accuracy would be 67%.

Time to complete is just as it sounds. It is how long the algorithms took to process and output an image. This was tested for all on the same system in order to give complete fairness to the algorithms in case the speed of a machine came into account.

Lastly, confidence was how confident the algorithm was in its own blurring of whatever was on the image. Below is an example from the Haar cascade algorithm that shows the confidence counter being displayed on top of the bounding boxes that are blurred.

- Figure 8: Output image from Haar cascade algorithm

5.1 HAAR CASCADE

My first datasets were the original OpenCV datasets for Haar cascade. This includes:

- haarcascade_frontalface_default.xml
- haarcascade_profileface.xml
- haarcascade_eye.xml

Figure 9: Caltech dataset [4]

These three datasets were already pre trained. They were trained on a dataset from Caltech called “Caltech 10,000 Web Faces” [4]. The dataset includes 10,524 human faces. The evaluation of such a dataset is vital to gauge its effectiveness and reliability. To assess the dataset, researchers often employ a variety of experimental protocols. Typically, the dataset is split into distinct training and testing subsets. A facial recognition algorithm is then trained using the training subset, after which its performance is

evaluated on the unseen testing subset. Various metrics, such as accuracy, precision, recall, and the F1 score, were used to quantify the algorithm's proficiency in detecting and recognizing faces. It's worth noting that, for a comprehensive evaluation, the dataset's diverse and challenging real-world conditions - like variations in illumination, pose, and expression - are considered. The images of people were gathered by searching google for common names and then using the images that came from that. In total this includes 10,524 faces on 7,092 images. The average resolution of the pictures is 304x312. Information on what datasets the other two classifiers were trained on is not available however.

These images are then annotated in order to train the algorithm on what exactly is and is not a face. For example this image below provides an example of what that would look like:

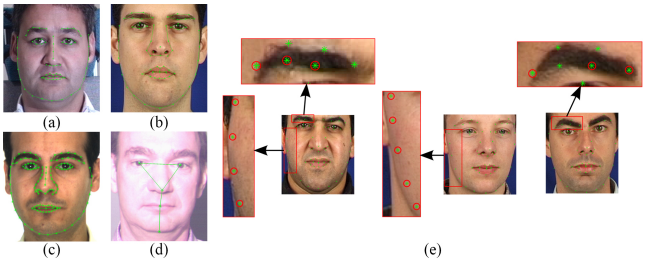


Figure 10: Annotated faces [4]

5.2 DEEP LEARNING MODEL

The MTCNN model was trained on the "Wider-Face" [14] dataset. This data set is a collection of 32,203 images and label 393,703 faces with, "a high degree of variability in scale, pose and occlusion as depicted in the sample images.". The MTCNN model was trained by collecting positives and negatives from this image set which gave it a good basis as to what was and was not faces.

- Figure 11: WIDER FACE collection [14]

6. RESULT ANALYSIS

The results for these two algorithms were all measured off of these four photos:

- Figure 12: Test image one

- Figure 13: Test image two

- Figure 14: Test image three

- Figure 15: Test image four

My results ultimately proved that the deep learning model was better than the Haar cascade model in almost every way. For example in my evaluation I found that on the tests for 4 different images these were the results of the Haar cascade:

HAAR CASCADE

Group Image #	Accuracy	Average confidence	False Positive	Time Taken (s
1	88%	68%	1	0.0571
2	100%	90%	3	0.1701
3	55%	73%	1	0.0838
4	46%	80%	1	0.0877

- Figure 16: Haar Cascade image set results

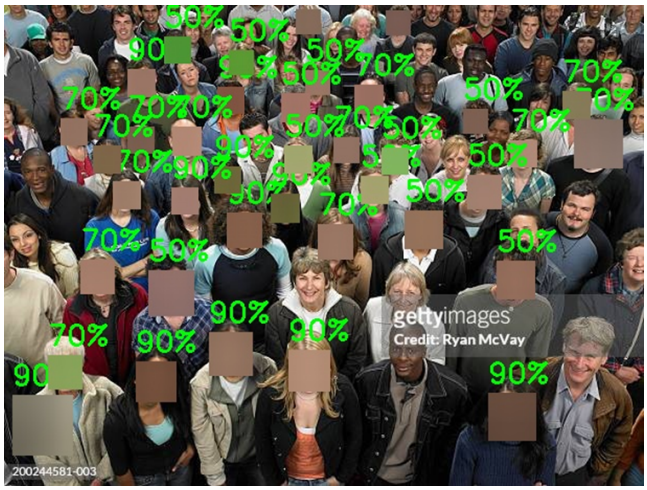
Now if we compare those results to the deep learning model's results on the exact same images you can see the difference in just about everything.

DEEP LEARNING

Group Image #	Accuracy	Average confidence	False Positives	Time Taken (s)
1	100%	100%	0	1.0551
2	100%	100%	0	1.4458
3	86%	91%	0	1.4827
4	92%	99%	0	1.6736

- Figure 17: Deep learning model image set results

As you can see the deep learning model vastly outpaces the Haar cascade algorithm in a multitude of ways. For instance in the accuracy category there are gaps of almost 100% on larger group photos such as image three. Where Haar cascade ultimately fell behind was the side profile of faces and large amounts of faces. The testing proved that on numerous occasions such as image three and four you can see where the average accuracy and confidence dropped on the Haar cascade algorithm. The deep learning model also gave zero false positives while the Haar cascade algorithm always had at least one. Below is a few test examples of how these two algorithms compared:



- Figure 18: Haar cascade test image three

- Figure 19: Deep learning model test image three

- Figure 20: Haar cascade test image four

- Figure 21: Deep learning model test image four

Credit where credit is due however the Haar cascade algorithm vastly outperformed the deep learning model in turns of speed being hundreds of times faster in processing speeds.

7. CONCLUSION

In conclusion the deep learning model is overall better. Its accuracy and confidence levels were overall higher leading to much better blurring all around of the faces on the images. Even in large groups such as test image three it easily blurred the majority of faces with negligible runtime difference. The Haar cascade algorithm however could still have uses in real time footage blurring as its runtime was significantly less in comparison to the deep learning model.

One thing of note however is that deep learning models are much more involved as far as training and development. Not only does the deep learning model implement the Haar cascade methodology in how it looks for faces but also on top of it adds three network classifiers on top. In development these things must be taken into account as not always will you need to yield such high results as to warrant this much work. The Haar cascade as it is is still not a bad algorithm to use for simple blurring of more straight on faces.

8. REFERENCES

[1] Ali, Alya'a R., and Ban N. Dhannoon. "Real time multi face blurring on uncontrolled environment based on color space algorithm." Iraqi Journal of Science, 2019, pp. 618–1626, https://doi.org/10.24996/ijs.2019.60.7.22.

[2] A. Sharifara, M. S. Mohd Rahim and Y. Anisi, "A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection," 2014 International Symposium on Biometrics and Security Technologies (ISBAST), Kuala Lumpur, Malaysia, 2014, pp. 73-78, doi: 10.1109/ISBAST.2014.7013097.

[3] A. Singh; H. Herunde; F. Furtado. "Modified Haar-cascade model for face detection issues". International Journal of Research in Industrial Engineering, 9, 2, 2020, 143-171. doi: 10.22105/riej.2020.226857.1129

[4] "Caltech 10K Web Faces." Perona Lab - Caltech 10k Web Faces, www.vision.caltech.edu/datasets/caltech_10k_webfaces/. Accessed 20 Oct. 2023.

[5] Devaux, Alexandre. "Face Blurring for Privacy in Street-Level Geoviewers Combining Face ..." ResearchGate, www.researchgate.net/publication/292848049_Face_blurring_for_privacy_in_street-level_geoviewers_combining_face_body_and_skin_detectors . Accessed 18 Sept. 2023.

[6] Gosper, Maree, et al. Web-Based Lecture Technologies: Blurring the Boundaries between Face-To ..., files.eric.ed.gov/fulltext/EJ809980.pdf. Accessed 18 Sept. 2023.

[7] Howse, Joseph. OpenCV 4 for Secret Agents: Use Opencv 4 in Secret Projects to Classify Cats, Reveal the Unseen, and React to Rogue Drivers. Packt Publishing, 2019.

[8] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[9] Lewis, Michael B. "Face Detection: Mapping Human Performance - Sage Journals." SageJournals, journals.sagepub.com/doi/10.1068/p5007. Accessed 18 Sept. 2023.

[10] Meenakshi, M. "Real-time facial recognition system—design, implementation and validation." Journal of Signal Processing Theory and Applications, 10 Nov. 2012, https://doi.org/10.7726/jspta.2013.1001.

[11] Opencv. "Opencv/Opencv: Open Source Computer Vision Library." GitHub, github.com/opencv/opencv. Accessed 19 Oct. 2023.

[12] R. Padilla, et al. Evaluation of Haar Cascade Classifiers Designed for Face Detection. 2910, Zenodo, Apr. 2012, doi:10.5281/zenodo.1058133.

[13] Viola, P., and M. Jones. "Rapid Object Detection Using a Boosted Cascade of Simple Features." *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, doi:10.1109/cvpr.2001.990517.

[14] "WIDER FACE: A Face Detection Benchmark." *Wider Face: A Face Detection Benchmark*, shuoyang1213.me/WIDERFACE/. Accessed 23 Nov. 2023.