An Evaluation on the Performance of Face Detection Algorithms with Respect to Light Quality

Alex Purser
Falmouth University
Falmouth, Cornwall
ap276620@falmouth.ac.uk

***Abstract*—This paper presents an evaluation of different face detection algorithms in Python with respect to light quality. Light quality is a vital factor in successful face detection and needs to be considered when developing or selecting algorithms to be used in real world applications. The main objective of this experiment is to determine which face detection algorithm has the greatest overall performance throughout different increments of light quality. To evaluate this, the Yale Face Database was used, which contains 15 subjects each with 3 images involving light direction. The brightness of these images was then altered from –200% to 500% and used to calculate the performance of each algorithm. The algorithms selected were implementations of two main libraries: OpenCV and Dlib. The algorithms are titled: Haar Cascades, CNN, HOG, Face_recognition, MTCNN, and HOG MultiScale. Results demonstrate that Haar Cascades yielded the highest performance, showing the best average score, but had a smaller range of functionality with darker brightness levels compared to the other algorithms. These findings slightly add to the existing body of knowledge by further elaborating on a key functionality point that determines the accuracy of face detection algorithms, helping to display its importance and necessity.**

***Index Terms*—Face Detection; OpenCV; Dlib; HOG; CNN; MTCNN; HOG_MultiScale; Light Quality; Brightness; Haar Cascades.**

I. Introduction

Face detection systems are prevalent throughout modern technology, spanning various technologies such as computer vision, security, and biometrics. A study from CyberLink states that 176 million Americans use face recognition, and of those, 75% use it daily [1]. As a result, face detection and recognition algorithms must provide accurate and reliable functionality for users. However, these algorithms can be influenced by environmental factors such as light quality, present in the users environment.

The main objective of this study is to accurately evaluate the performance of six commonly used Python-based algorithms, using images subject to different light qualities in order to determine which algorithm is the most suitable for real-world applications. The images for this project are from the Yale Face Database [2] [3] and all of the images used have been subject to brightness changes of 25% from –200% to 500% (100% being the baseline) to test the algorithms range of functionality.

The six algorithms being compared are:

- **Haar Cascades**: A Python implementation of the OpenCV library.
- **Histogram of Oriented Gradients (HOG)**: Implemented in the Dlib library, HOG is a technique widely used for object detection.
- **Convolutional Neural Networks (CNN)**: Another implementation of the Dlib library, CNN utilises deep learning for face detection.
- **Face_Recognition**: A wrapper for the Dlib library, offering an API for face recognition tasks.
- **Multi-Task Cascaded Convolutional Neural Network (MTCNN)**: An advanced approach utilising multiple neural networks for face detection.
- **HOG_MultiScale**: Another implementation of Dlib which primarily is used for object detection.

The paper starts with background information including research findings from other studies and implementations of the algorithms. Progressing on to the methodology for the experiment, followed by results and analysis. Finally concluding the paper with legal and ethical considerations, and future work.

II. Background

A. Related Work

With face detection, the most crucial factor is accuracy; its ability to correctly identify and recognise faces. Without suitable accuracy, many problems can arise such as failed biometrics; potentially granting access to unauthorised users, or even preventing access for the authorised users. This is supported by William Crumpler in saying: "Further accuracy gains will continue to reduce risks related to misidentification, and expand the benefits that can come from proper use" [4].

One issue that can affect accuracy is the environment that the user is in, more specifically the light quality. In near perfect conditions, face detection algorithms perform the best, but it is uncommon for users to always be in these conditions. In the instance of CCTV for example, William Crumpler stated: *"Remote identification systems tend to have lower accuracies compared to verification systems, because it is harder for fixed cameras to take consistent, high-quality images of individuals moving freely through public spaces"* [4].

It is widely believed that irregular lighting environments prove the most troublesome for face detection, typical environments such as at night or outside in the sun, but is this truly the case?

A 2022 study completed by YouGov on '*The Impact of screens, lights, and noise on sleep*' shows that 61% of all users use their phones within 1 hour before going to sleep [5]. In this situation, it is vital that face detection successfully identifies the user, otherwise their access is restricted. Comparing these algorithms will provide an answer for which Python algorithm should be used to withstand this weakness.

Few sources can be found about the effects of light quality on face detection algorithms, but, of these sources one stood out showcasing interesting research into this field. A study by S.V. Viraktamath et al., on '*Face Detection and Tracking using OpenCV* explores the usage of OpenCV for face detection and the making of an automatic real-time face detection and tracking [6]. This study was deemed successful and provides great insight into the capabilities of face detection algorithms, which can improve upon already existing technologies such as CCTV.

B. Libraries

In terms of Python face detection, there are two main libraries: OpenCV and Dlib.

1) OpenCV: OpenCV (Open-Source Computer Vision Library) is an open-source library that contains more than 2500 optimised algorithms used for face and object detection, identification, and recognition. OpenCV was created by Intel, and in 2016 was supported by Willow Garage and the computer vision startup Itseez which intel acquired [7].

2) Dlib: Dlib is another popular library for face recognition. As defined by unogeeks.com *"Dlib is a popular toolkit for machine learning that is primarily used for computer vision and image processing tasks. It is written in C++ but has Python bindings, making it easily accessible from Python code"* [8]. In Python, Dlib has multiple implementations, but the three most used are HOG, CNN, and Face_recognition.

C. Algorithms

To start with, it is important to understand which algorithms are being used for this project. Below are brief definitions of each algorithm and the process behind them.

1) Haar Cascades: The OpenCV algorithm chosen for this project is Haar Cascades. Haar cascades, initially introduced by Viola and Jones in 2001 [10], involves a systematic sliding of a window across an image, followed by the identification of features such as edges and lines, and concluding with classification from the AdaBoost algorithm. An example of this can be seen in Figure 1. Typically, Haar cascades are known for prioritising swiftness over accuracy.

2) HOG: In no order, HOG is the first implementation of the Dlib library. HOG was developed by Dalal and Triggs in 2005 [11] and is a widely used feature descriptor. A feature descriptor combines computer vision and machine learning (ML) techniques for image processing. In short, feature descriptors like HOG summarise key information about an image by analysing gradient intensities in images, such as prominent features like edges, making it easier to infer key information. [12]

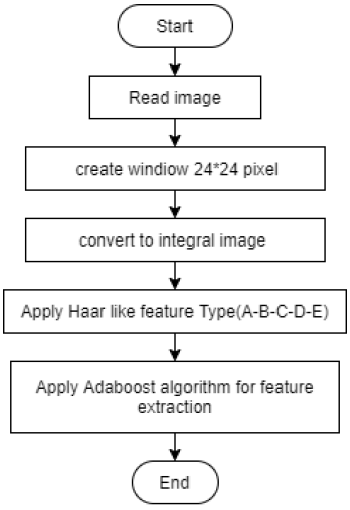

Fig. 1. Workflow of Haar Cascades [9]

3) CNN: CNN is the second implementation of the Dlib library. CNNs are machine learning models designed for analysing visual data in images. CNNs work by processing the image into three main layers: Convolutional, Pooling, and Fully Connected. In short, the Convolutional layer identifies key features, the Pooling layer reduces complexity, and the Fully Connected layer classifies inputs. Finally, functions such as ReLU (Rectified Linear Unit) and SoftMax are used on each layer to ensure greater accuracy [13].

4) Face_Recognition: Face_recognition is a Python package that wraps dlibs face recognition functionality into an easy-to-use API. This API can be used in Python and is one of the easiest functions to implement for face detection [14].

5) HOG MultiScale: The final implementation of Dlib for this experiment, known as HOG MultiScale, utilises the HOG multiscale detector technique for object detection. This method involves the analysis of gradient orientations within image sections to generate descriptors, identifying objects at various sizes. By using a sliding window approach across multiple scales of the image, it can detect objects of different sizes while proving resistant to changes in appearance [15].

6) MTCNN: MTCNN is an advanced face detection model utilising three networks (P-Net, R-Net, O-Net). It starts by generating candidate bounding boxes in the first stage, refining them in the second, and then extracting facial landmarks in the third. MTCNN is widely used for accurate face detection in various conditions such as face recognition [16].

III. Methodology

In this study, the effect light quality has on face detection algorithms is being tested, and from this, determining which yields the best average result while maintaining a good range of functionality. This is being investigated by the algorithms against a database of face pictures, each modified with a brightness change.

A. Data Preparation

The images in this study are from the Yale Face Database which contains 165 GIF images with 15 subjects. The Yale Face Database is publicly available and downloadable from Kaggle.com [17] subject to copyright. Each subject has 11 images of themselves subject to different lighting angles, glasses, no glasses, emotions, and different expressions. In this database each subject has 3 images where they have a different lighting direction labelled *"right light" "left light"* and *"centre light"*. Implementing lighting direction adds another level of complexity when testing the algorithms; seeing how well they work when the subjects face is not lit up centrally. Sorting through this database gave me 45 images useful for this study.

However, this quantity of pictures is not significant enough to create a fair test. This is stated by a study from Jeevan Singh and Vishal Kumar: *"By taking more and more photos of each person a, you can take better decision on database and improve the facial recognition accuracy, especially in different angles and lighting conditions"* [18]. Using Pixlr [19], the brightness of the images were changed in increments of 25%, ranging from –200% to 500% brightness, totalling 29 sets of images. While light quality is commonly associated with dark conditions such as night or dim lighting, to further test the algorithms functions, brighter environments have been added.

B. Algorithm Setup

The program is designed to incorporate new algorithms by storing them within their own function. The program has been designed this way to ensure modularity and greater organisation. These functions are integrated into the main flow of the program, only called when they are selected. This ensures a fair comparison of the algorithms, as each is structured and executed uniformly within the program. The program consists of three primary functions: 'menu,' 'prepare_image,' and 'results.' Within the 'menu' function, user inputs are managed to determine the correct speed, desired algorithm, and brightness level. The 'prepare_image' function calls the corresponding algorithm function, passing the current image as the parameter. Subsequently, the algorithm function attempts to detect faces within the image and produces the results, which are then processed by the 'results' function for printing and storage. Within each algorithm function, the algorithms have been changed marginally from their corresponding tutorial. The program can be found here: https://github.falmouth.ac.uk/AP276620/COMP213-AI-2202628.

C. Computer Specifications

While the computer specifications have no impact on the performance of each algorithm, the time taken by each algorithm is impacted by this and is documented in the results section. The specifications used were:

- **CPU**: Intel i7-8700 @ 3.20 GHz
- **RAM**: 16 GB DDR4
- **Operating System**: Windows 10 Home

D. Experimentation

Upon startup, the user is prompted with questions regarding their preferred processing speed, algorithm to use, and brightness level. For speed, the choices consist of 'i' for instant, 'q' for quick and 's' for slow. In the instant mode, the user specifies their desired algorithm, which after being entered, executes through each brightness setting, presenting only the outcomes. In quick mode, the user is prompted to input both the desired algorithm and the brightness level, with the program producing the results of this combination. Subsequently, in slow mode, the user is prompted with the same questions, with the program displaying each image individually, featuring a distinctive red box around any faces detected.

E. Storing the Results

To efficiently store and manage data, an export function was implemented that relocates the data to a CSV file called Results.csv.

One problem that occurred was with MTCNN and the output log. MTCNN prints each step and how long it took to complete proving beneficial for displaying performance. Unfortunately, the terminal deletes printed lines after reaching a certain amount, which in turn meant some of the scores were deleted from the terminal. This was a problem as it made reading the data difficult, haltering progress with data analysis. By implementing the new CSV file, the results are automatically updated with each run time of the program if instant mode is selected. From this, the results were easily exportable to Excel, allowing for visual graph generation to use for this report.

IV. Results

A. False Positives

To ensure accuracy within the results, utilising slow mode allowed me to check for any potential false positives. Unfortunately, HOG_MultiScale did highlight a few false positives, especially within the brighter ranges from 400% to 500%. Half of the results identified were false positives, and it is unclear whether the successful positives were indeed correct or a mishap as often, HOG_MultiScale identified multiple 'faces' within an image containing a singular subject. This is presumably due to the nature of HOG_MultiScale being designed for object detection rather than face detection.

For cohesion, HOG_MultiScale's results will be excluded from the analysis. Implementation of HOG_MultiScale should not be based upon the results shown due to its unsuitability for face detection.

B. Algorithm Performance

The main objective of this study was to determine the optimal algorithm for Python face detection, considering two key factors: performance and the range of functionality. The average score indicates the cumulative score across all variations of brightness levels, normalised by the number of increments. Meanwhile, the range of functionality indicates the

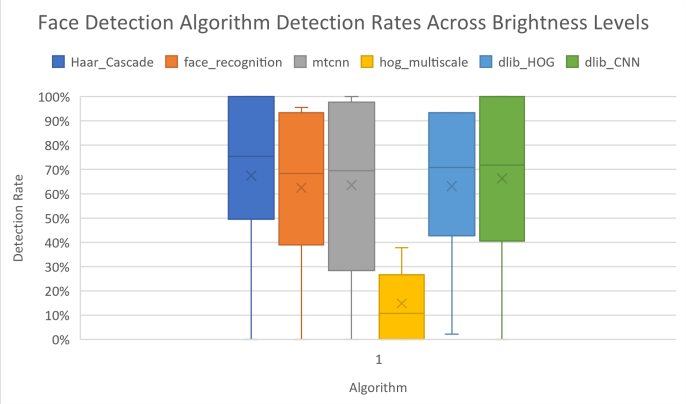

Fig. 2. Each Algorithms Detection Rates Throughout the Brightness Levels

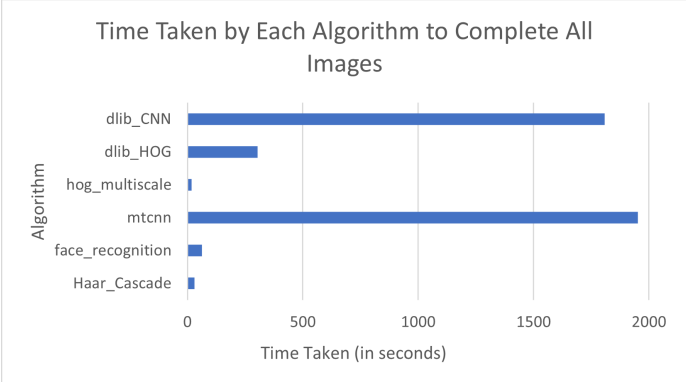

Fig. 3. Graph for Time Taken for Each Algorithm to Complete the Full Test. (Shorter is better)

breadth of brightness increments of which the algorithms could successfully detect faces. Figure 2 illustrates that these face detection algorithms all performed very similarly, following a pattern of trend. But from this, we can still identify differences in the algorithms performances, leading to a result.

Algorithm	Average Score (out of 45)
Haar Cascades	30.45
Face recognition	27.59
MTCNN	26.66
HOG Multiscale	3.97
HOG	28.55
CNN	28.97

TABLE I
AVERAGE SCORES

1) Average Score: When evaluating the performance of an algorithm, average score plays a crucial role. A high average score shows that the algorithm has great functionality. Oppositely, a low average score indicates limitations and unreliability.

From the data presented in Figure 2, it is evident that both Haar Cascades and CNN yielded the greatest performances, achieving a perfect score, but what was their averages?

The average score of each algorithm was calculated and can be seen in Table I. From this, Haar Cascades demonstrated the highest average score at 30.45/45, closely followed by the other algorithms ranging between 26.67 and 28.97.

2) Range of Functionality: Figure 2 shows that Haar Cascades was able to detect the most faces in brighter environments, scoring the highest from 300% to 500%, besides one fluctuation where HOG scored the best at 400% and both tied from 425% to 450%. However, HOG on the other hand, proved the greatest in darker environments, having the highest score from –75% to –125%, even successfully detecting 1 face at –125%, being the only algorithm to detect a face at this level of brightness.

C. Algorithm Speed

Another crucial factor is the speed at which each algorithm operates. Speed holds important significance in face detection; the faster an algorithm can determine accurate results, the better. To determine this, the duration taken from each algorithm spanning each brightness was recorded. Among the six algorithms evaluated Haar Cascades was the fastest with 29.92 seconds. The results can be seen in Figure 3 or in Table II.

Algorithm	Time Taken (in seconds)
Haar Cascades	29.92
Face recognition	63.05
MTCNN	1952.93
HOG	303.67
CNN	1808.87

TABLE II
TIME TAKEN FOR EACH ALGORITHM TO COMPLETE THE FULL TEST

V. ANALYSIS

To officially determine which algorithm performed the best overall, a culmination of average score and range of functionality is required.

A. Suitability

As mentioned in the results section, HOG_MultiScale has been excluded due to its high rate of false positives. This is because Hog_MultiScale typically is not suited for face detection, but more so object detection.

In terms of images, while the images utilised in this study provided insight into light quality and light direction, the subjects in the images were standing still, facing the centre of the camera. Excluding the brightness change, these images are 'ideal conditions' for face detection and do not fully test the functionality of these algorithms.

B. Algorithm Performance and Reasoning

From the results given in Figure 2, it is evident that Haar Cascades and CNN performed the best overall as these were the only two algorithms to achieve perfect scores. In terms of consistency however, HOG prevailed, having the smallest fluctuations across its scores. This was closely followed by Haar Cascades and Face_recognition.

After looking further into the individual results, it was discovered that both CNN and Haar Cascades had the same

number of perfect scores, just 2 cycles ajar; CNN from 25% to 275% and Haar Cascades from -25% to 225%. From this we can determine that for the majority of their perfect sprints, CNN performed better in brighter environments, and Haar Cascades performed better in darker environments. Needing to determine which of CNN and Haar Cascades outperformed the other, we need to factor in the average score. Haar Cascades attained the highest average score achieving 30.45, compared to CNN scoring 28.97, the second highest score.

From Figure 2 we can determine that HOG and Haar Cascades were the two algorithms that had the largest range of functionality, both spanning the same range, just at different ends of the spectrum. HOG proved to be the greatest in darker instances, successfully detecting 1 face at -125%, 50% darker than Haar Cascades, but did not uphold this throughout the rest of the cycles. Haar Cascades proved the greatest in brighter environments, scoring the highest

While Haar Cascades lacked the same performance as CNN in brighter instances lack functionality at darker instances compared to that of HOG, it provided the best face detection at brighter instances, outperforming the rest.

VI. Conclusion

Through analysis, it is evident that Haar Cascades, an algorithmic implementation of the OpenCV library, proved to be the greatest face detection algorithm overall: scoring the highest average, spanning the joint greatest range, and performing the fastest. Interestingly, this goes against a 2022 study from Sameer Aqib Hashmi on '*Face Detection in Extreme Conditions*'. While this paper uses different images subject to different changes, from the images included, the changes typically comprise of angles, subject quantity and lighting. The results from this study show that MTCNN scored the highest attaining a near perfect score of 99.83%, outshining Haar Cascade at 94% [20]. From these results, this difference suggests that while MTCNN typically outperforms when handling irregular conditions, it may not perform as well with light quality changes.

However, in answering the main question of this study: determining which algorithm is the most suitable for real-world applications, Haar Cascades does not prevail. The main reason for this is due to Haar Cascades being outperformed within darker environments. While the brightness change from Haar Cascades' lowest point compared to the rest was only 25%, this difference could prove important in real world applications.

A. Legal Ethical Social Professional Sustainable Issues

Face detection, being the first step towards more advanced technologies in this field, can often create legal and social issues. One of these social issues comes from a newer technology called deepfakes. As defined by Merriam-Webster: deepfakes are "an image or recording that has been convincingly altered and manipulated to misrepresent someone as doing or saying something that was not actually done or said" [21]. The introduction of this technology has caused uproar throughout the world, as people can be seen speaking or acting in videos that are purely programmatically generated. As stated by Morgan Currie for the *Edinburgh Impact*, *"The deep-learning software used to make deepfakes has become cheap and accessible, raising questions about the potential for abuse"* [22].

Another social issue present with face recognition is impersonation: posing as someone else to fraud the target [23]. Impersonation, one of the many potential cyberattacks, is a prevalent issue with face recognition technologies. With the advancement of face detection and recognition, attackers could potentially use images or pictures to try bypass the device login system, granting unauthorised access.

A prevalent legal issue with face detection and recognition stems from compliance with the GDPR. The GDPR denotes how user information should be managed [24]. Storing user data must comply within the rules of the GDPR, which each company using this technology will have to consider. As face detection and face recognition advance, more implementations of this technology will appear, each requiring further GDPR considerations. Incorrect management of user data can cause many issues for companies, from lawsuits to penalties and so forth.

VII. Future Work

To further improve upon the quality of this study, utilising a personal database of faces from myself and peers would prove better from a legal and ethical standpoint. Unfortunately, due to time bounds this was not possible as the images would need developing and consent forms would be required. Consequently, this explains the usage of the Yale Face Database for this study.

Although the results from this study do provide an insight into the accuracy and reliability of the selected algorithms, its contribution to the existing knowledge around light quality in face detection is minimal. Nevertheless, with further progression in this field, greater possibilities emerge. Applications such as improved CCTV and security camera performance, heightened reliability with biometric logins and potentially doors that can unlock by detecting the users face [25]. Furthermore, as face detection advances, identifying faces in even more irregular environments becomes feasible, encompassing varied lighting environments such as minimal, maximum, angled, and even tinted lighting.

VIII. Project Repository

For the repository of this project, click the link below. https://github.falmouth.ac.uk/AP276620/COMP213-AI-2202628

References

[1] FaceMe, "FaceMe: Face of the Future Report." [Online]. Available: https://www.cyberlink.com/faceme/insights/whitepaper/general/731/faceme-facial-recognition-report

[2] O. Belitskaya, "Yale Face Database." [Online]. Available: https://www.kaggle.com/datasets/olgabelitskaya/yale-face-database

[3] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[4] "How Accurate are Facial Recognition Systems – and Why Does It Matter? | Strategic Technologies Blog | CSIS." [Online]. Available: https://www.csis.org/blogs/strategic-technologies-blog/how-accurate-are-facial-recognition-systems-and-why-does-it

[5] M. Dinic, "The YouGov Sleep Study: Part four - The impact of screens, lights, and noise on sleep | YouGov," Jun. 2022. [Online]. Available: https://yougov.co.uk/health/articles/42947-yougov-sleep-study-part-four-impact-screens-lights

[6] V. S.V, M. Katti, A. Khatawkar, and P. Kulkarni, "Face Detection and Tracking using OpenCV," *The SIJ Transactions on Computer Networks & Communication Engineering*, vol. 04, no. 03, pp. 01–06, Jun. 2016. [Online]. Available: http://www.thesij.com/TableOfContentArticleDetails.aspx?JournalID=8&IssueID=196

[7] G. Boesch, "What is OpenCV? The Complete Guide (2024)," Dec. 2023. [Online]. Available: https://viso.ai/computer-vision/opencv/

[8] A. Unogeeks, "Dlib Python," Aug. 2023. [Online]. Available: https://unogeeks.com/dlib-python/

[9] S. Patil, S. Trivedi, J. Jani, S. Shah, and P. Kanani, "Digitized railway ticket verification using facial recognition," May 2021, pp. 1556–1563.

[10] P. Viola and M. 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*, vol. 1. Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I–511–I–518. [Online]. Available: http://ieeexplore.ieee.org/document/990517/

[11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893. [Online]. Available: http://ieeexplore.ieee.org/document/1467360/

[12] S. Ranjan Rath, "Image Recognition using Histogram of Oriented Gradients," Jun. 2020. [Online]. Available: https://debuggercafe.com/image-recognition-using-histogram-of-oriented-gradients-hog-descriptor/

[13] IBM, "What are Convolutional Neural Networks? | IBM," 2024. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks

[14] A. Rosebrock, "Face detection with dlib (HOG and CNN)," Apr. 2021. [Online]. Available: https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/

[15] ——, "HOG detectMultiScale parameters explained," Nov. 2015. [Online]. Available: https://pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/

[16] A. Jawabreh, "Explore the most advanced deep learning algorithm for face detection," Sep. 2023. [Online]. Available: https://medium.com/the-modern-scientist/multi-task-cascaded-convolutional-neural-network-mtcnn-a31d88f501c8

[17] Kaggle, "Kaggle.com." [Online]. Available: https://kaggle.com

[18] J. Singh, V. Kumar, B. Tech, and B. Tech, "Face Detection and Tracking Using OpenCV : A Survey," vol. 8, no. 6, 2020.

[19] Pixlr, "Batch Photo Editor: Free Bulk Image Editing Online | Pixlr." [Online]. Available: https://pixlr.com/batch/

[20] S. A. Hashmi, "Face Detection in Extreme Conditions: A Machine-learning Approach."

[21] M. Webster, "Definition of DEEPFAKE," Feb. 2024. [Online]. Available: https://www.merriam-webster.com/dictionary/deepfake

[22] D. Ramage, "What are deepfakes and how are they impacting society?" Mar. 2022. [Online]. Available: https://impact.ed.ac.uk/opinion/what-are-deepfakes-and-how-are-they-impacting-society/

[23] H. Rhim, "What Are Impersonation Attacks? | Baeldung on Computer Science," May 2023. [Online]. Available: https://www.baeldung.com/cs/impersonation-attacks

[24] "A guide to the data protection principles," Jul. 2023, publisher: ICO. [Online]. Available: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/

[25] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, Present, and Future of Face Recognition: A Review," *Electronics*, vol. 9, no. 8, p. 1188, Aug. 2020, number: 8 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/9/8/1188