

PAPER • OPEN ACCESS

## A web-based application for face detection in real-time images and videos

To cite this article: Mehul Arora *et al* 2022 *J. Phys.: Conf. Ser.* **2161** 012071

View the [article online](#) for updates and enhancements.

### You may also like

- [A Rotation-Invariance Face Detector Based on RetinaNet](#)  
Yuyang Xiong, Wei Meng, Junwei Yan et al.
- [Improving the accuracy of old and young face detection in the template matching method with Fuzzy Associative Memory \(FAM\)](#)  
Rafika Sari Sembiring, Syahril Efendi and Saib Suwilo
- [High-Precision Portrait Classification Based on MTCNN and Its Application on Similarity Judgement](#)  
Juan Du

 The Electrochemical Society  
Advancing solid state & electrochemical science & technology

# UNITED THROUGH SCIENCE & TECHNOLOGY

## 248th ECS Meeting Chicago, IL October 12-16, 2025 Hilton Chicago



## Science + Technology + YOU!

### Register by September 22 to save \$\$

[REGISTER NOW](#)

# A web-based application for face detection in real-time images and videos

Mehul Arora<sup>1</sup>, Sarthak Naithani<sup>1</sup>, Anu Shaju Areeckal<sup>1\*</sup>

<sup>1</sup>Department of Electronics & Communication Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Karnataka, India

\*Corresponding author: [anu.areeckal@manipal.edu](mailto:anu.areeckal@manipal.edu)

**Abstract.** Face detection is widely used in the consumer industry such as advertising, user interfaces, video streaming apps and in many security applications. Every application has its own demands and constraints, and hence cannot be fulfilled by a single face detection algorithm. In this work, we developed an interactive web-based application for face detection in real-time images and videos. Pretrained face detection algorithms, namely Haar cascade classifier, HOG-based frontal face detector, Multi-task Cascaded Convolutional Neural Network (MTCNN) and Deep Neural Network (DNN), were used in the web-based application. A performance analysis of these face detection algorithms is done for various parameters such as different lighting conditions, face occlusion and frame rate. The web app interface can be used for an easy comparison of different face detection algorithms. This will help the user to decide on the algorithm that suits their purpose and requirements for various applications.

**Keywords:** Web app, Face detection, Haar cascade, HOG, MTCNN, DNN

## 1. Introduction

The advances in computer technology has created a complex vision-based world. It led to the development of machines becoming intelligent, very importantly in computer vision. The broader aim of computer vision is to envision the world the way a human sees it through eyes. It could replace or partially replace humans in those tasks that are not physically viable for humans to reach or perform, the tasks that are repetitive in nature, or in tasks that have low error tolerance, such as applications in surveillance and security. The most basic paradigm for the applications is face detection, which later can be generalized in face recognition and object detection.

Face detection is a task which is very intuitive to human eyes. For computers, it is a challenging task, as there can be many false negatives, since face in 3D can be difficult to analyse in different lighting conditions, orientation of the face, occlusion and many other factors. Some techniques of face detection can only work when the front image of the face is present with no occlusion.

All these challenges inspired the development of many face detection techniques which are successful in different experimental conditions. In this paper, we have explored four algorithms for face detection. They are compared on the basis of their performance in different experimental setups. There are existing works on experimental comparisons of face detection algorithms [1-2]. However, this paper presents the performance analysis of different face detection algorithms, namely Haar feature-based cascade classifier, Histogram Of Gradient (HOG)-based frontal face detector, Multi-task Cascaded Convolutional Neural Network (MTCNN) and Deep Neural Network (DNN). Furthermore, we



developed a web-based application for face detection, which provides a simple dashboard to compare different face detectors on images, videos or using webcam.

The remaining paper discusses the different face detection algorithms in section 2, development of the web application in section 3, experimental results of face detection algorithms in section 4 and conclusion of the work in section 5.

## 2. Background

In this paper, four pretrained face detection algorithms are used, namely Haar cascade classifier, HOG-based frontal face detector, MTCNN and DNN.

Haar cascade classifier, also known as Haar Cascades, is as fast as a basic CNN model [3,4]. In this model, many features are extracted, out of which some dominant features are selected using Adaboost [5]. The Adaboost optimization helps to reduce the feature size significantly. These features are then grouped using a cascade of classifiers [6]. The grouped features are passed through a test window. If the test window passes all the test features, then it is detected as a face.

The HOG-based frontal face detector is majorly used for face landmark detection [7]. The detector is conceptually based on histogram of orientated gradients (HOG) [8] and linear SVM [9]. In HOG, features are described using the information on how the directions of gradients are distributed in the image. The distribution of gradient orientations are plotted on a histogram, which serves as the features.

MTCNN was introduced by Kaipeng Zhang in 2016 [10]. A 3-stage cascade structure comprising of CNNs are used. In the first stage, the CNN obtains bounding box regression vectors. In the second stage, the false positives are rejected and bounding boxes are calibrated by the CNN. In the final stage, facial landmark detection is performed.

DNN is a face detector available in the deep neural network module of OpenCV. It is a Caffe model which uses a ResNet-10 architecture [11].

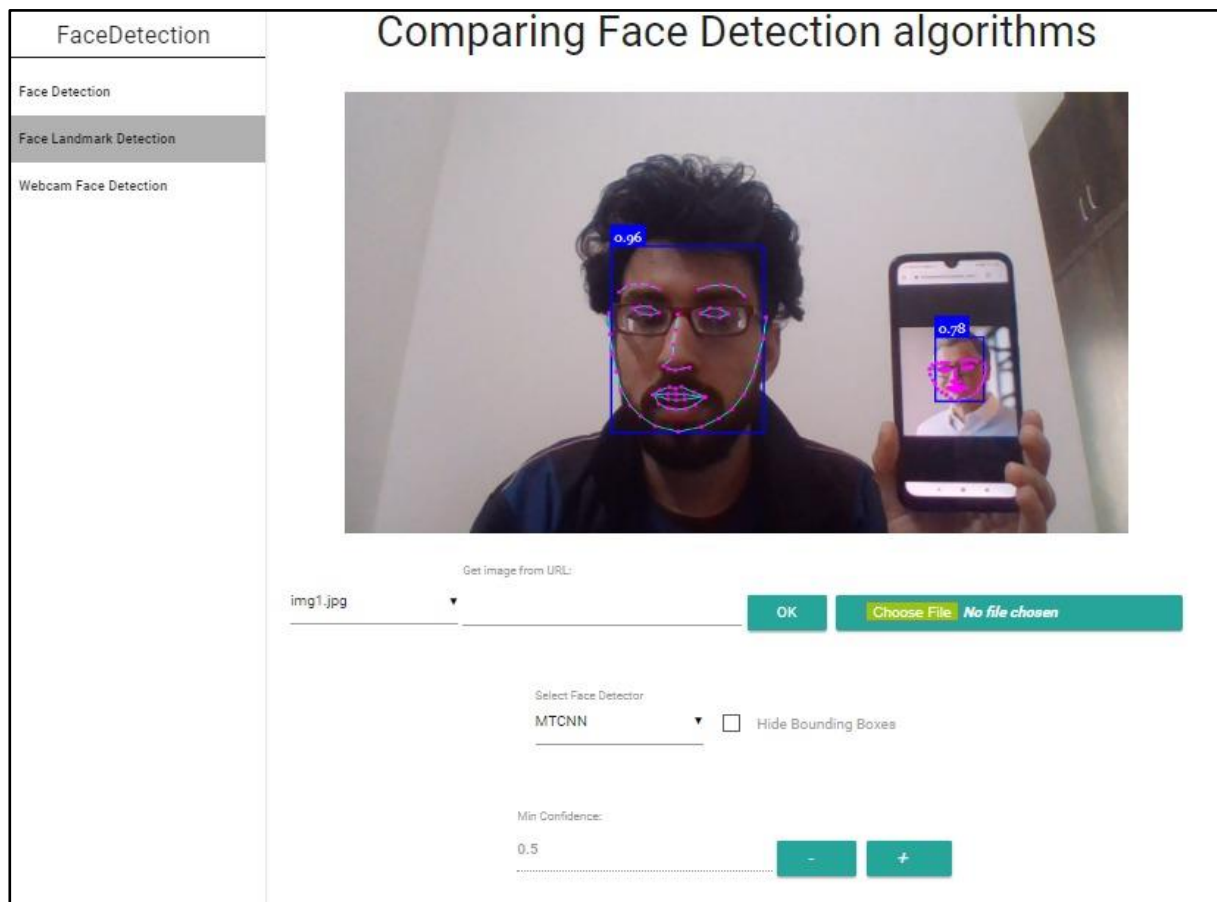
## 3. Implementation and Development of Web App

The novelty of our paper lies in the development of a web-based application with an easy-to-use user interface for the performance analysis of four different face detection algorithms. To the best of our knowledge, no work has been done on a web application that can be used to test various face detection algorithms in real-time images and videos.

### 3.1 Web app development

The web application is built using ReactJs and NodeJs. React is a frontend framework used for making the User Interface (UI) elements, whereas Node is a backend framework for handling the Application Programming Interface (API) calls and data processing. User can select different algorithms and can change different parameters using the UI. Upon selecting the algorithm and clicking on the button, an API request is triggered which is sent to the Node backend. The backend have API routes and all the functions and algorithm models. The API request from the frontend is routed by the backend router and an appropriate function is called. This function is responsible for communicating with the algorithm models, which then executes all the processing on the input image. After the processing is done, the image is sent back to the frontend as a string. This string is then converted into an image, which is displayed to the user.

Other helper libraries like Axios and Tensorflow.js are also being used. Sample images for the application are stored in the backend, and is made accessible to the user using the frontend UI. The homepage of the UI developed is shown in figure 1.



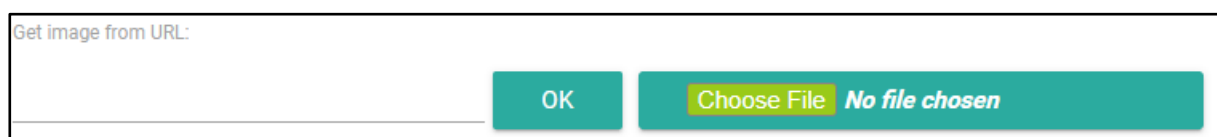
**Figure 1.** Homepage of the web-based application for face detection

### 3.2 Web app features

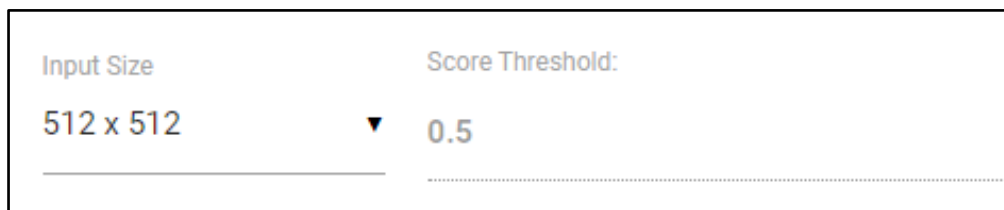
The idea behind the development of the web app is to display the simulated result live from the camera. The various features implemented in the web app are input Uniform Resource Locator (URL) of the images, resolution of input images, selection of face detection algorithm, minimum confidence level for the algorithm, estimation of frame rate and face extraction.

**3.2.1 Input Image.** The UI of the web app has a feature to enter the URL of the input image or browse the image from the folders in the computer, as shown in figure 2.

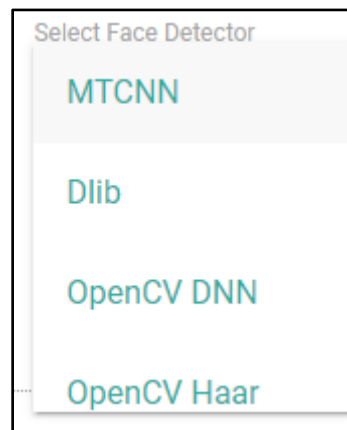
**3.2.2 Image Resolution.** The resolution of the input image can be changed from the dropdown menu, as seen in figure 3. The range of values available are 512x512, 416x416, 320x320 and 224x224. This UI feature will reduce or scale the size of the input image to the selected image resolution. It can be useful to simulate cases where the quality of the images cannot be guaranteed.



**Figure 2.** Input URL for the image

The image shows a UI section with two controls. On the left, under the label 'Input Size', there is a text input field containing '512 x 512'. On the right, under the label 'Score Threshold:', there is a text input field containing '0.5'. A small downward-pointing triangle icon is positioned between the two input fields. Below the inputs is a horizontal dotted line.

**Figure 3.** Input image resolution

The image shows a dropdown menu titled 'Select Face Detector'. The menu is open, displaying four options: 'MTCNN', 'Dlib', 'OpenCV DNN', and 'OpenCV Haar'. The 'MTCNN' option is currently selected and highlighted with a light blue background.

**Figure 4.** Selection of face detection algorithm

**3.2.3 Selection of face detector algorithm.** In this feature, a drop-down menu is given to select any one of the face detection algorithms, namely Haar cascade, HOG-based frontal face detector in Dlib, MTCNN and DNN. The UI can be observed in figure 4.

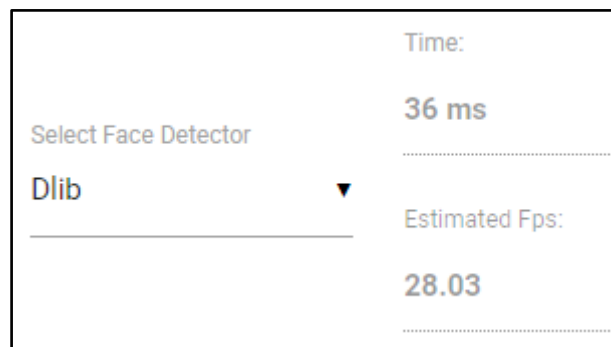
**3.2.4 Minimum confidence level.** Minimum confidence level for an algorithm can be incremented to decremented using the feature input, Min Confidence, shown in figure 5. It decides the cut-off score for which face is detected in the image. Min Confidence value ranges from 0 to 1. The default value is kept at 0.5.

**3.2.5 Frame rate estimation.** In the developed web app, a provision for estimation of frame rate in frames per second (fps) for the selected detection algorithm is provided. The UI is shown in figure 6.

**3.2.6 Face detection output.** Using this feature, all the faces in the input image is extracted and the count of the number of faces detected in the image is displayed. The UI for the face count and the extracted images from the input image is shown in figure 7.

The image shows a UI for the 'Min Confidence' level. It features a label 'Min Confidence:' followed by a text input field containing the value '0.5'. To the right of the input field are two teal-colored buttons: one with a minus sign '-' and one with a plus sign '+'. A horizontal dotted line is located below the input field.

**Figure 5.** Minimum confidence level for the detection algorithm



**Figure 6.** Checking the frame rate



**Figure 7.** Extraction of faces from the input image

#### 4. Results and Discussion

In this section, we discuss the experimental results of the face detection algorithms on a number of input images.

##### 4.1 Image data

The image data used for the performance analysis of face detection algorithms are images captured by two different web cameras and images available in the internet. We have assumed that there is low variance in distance between the person and the camera. The software used is Visual Studio Code on Intel Core i7 processor without any specialized hardware.

##### 4.2 Experimental set-up

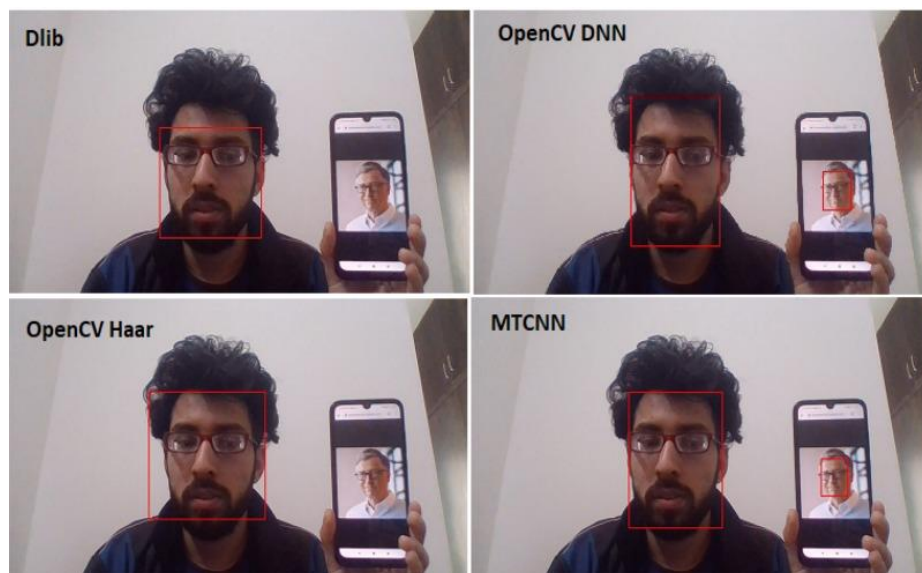
To analyse the performance of face detection algorithms, different experimental set-ups were used on images captured using web cameras. The experimental parameters used are varying lighting conditions, face occlusion and frame rate estimation. The visual comparison of the face detectors for these parameters in a webcam video taken by the author, are shown in figures 8-11.



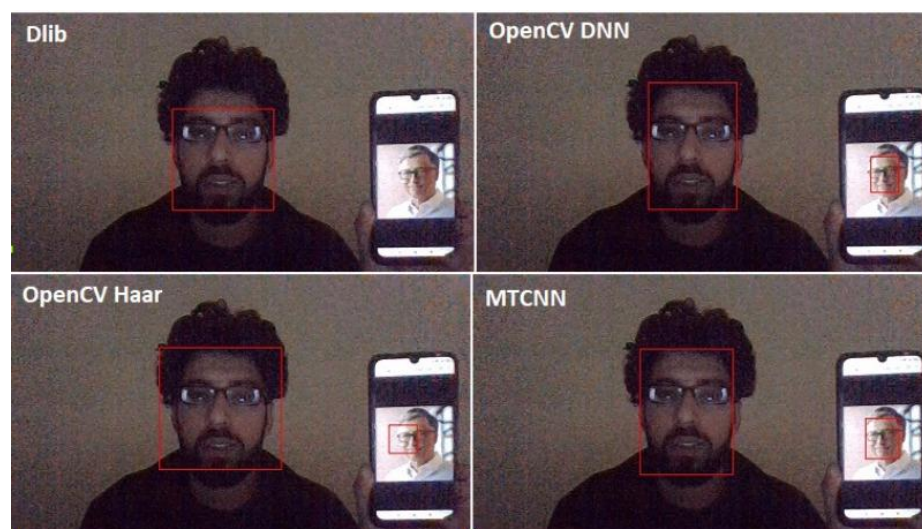
**4.2.1 Lighting condition.** This experimental set-up compared the performance of face detection algorithms in good and poor lighting conditions. In the good lighting condition, since there is a good contrast in the background and the face, feature extraction is simpler. When the background light is diminished, the contrast in the image is reduced, hence, making it harder to detect the edges, in turn making feature extraction to be tougher task. That is why different lighting conditions can make a difference in the performance of face detection algorithms.

In good lighting conditions, DNN performs the best closely followed by MTCNN. HOG-based face detector could not detect the faces when they are extremely small or extremely big. Haar cascade classifier was also not able to detect all the faces. A test comparison on an image taken using a web cam with a person holding a mobile phone with another face can be seen in figure 8.

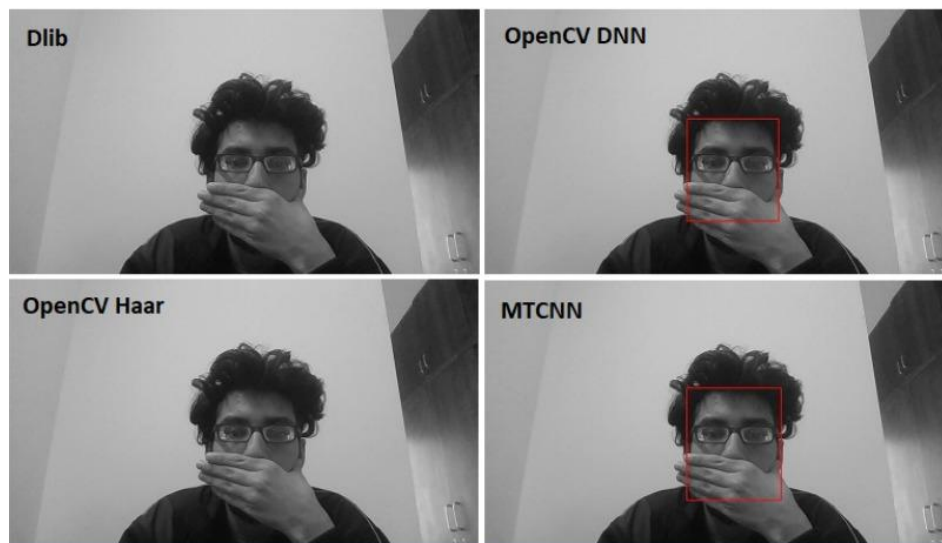
In poor lighting conditions, DNN and MTCNN gives the best results. HOG-based face detector could not detect some outlier cases. Haar cascade is able to detect even the small faces correctly. An example image can be seen in figure 9.



**Figure 8.** Comparison of algorithms in good-lighting conditions



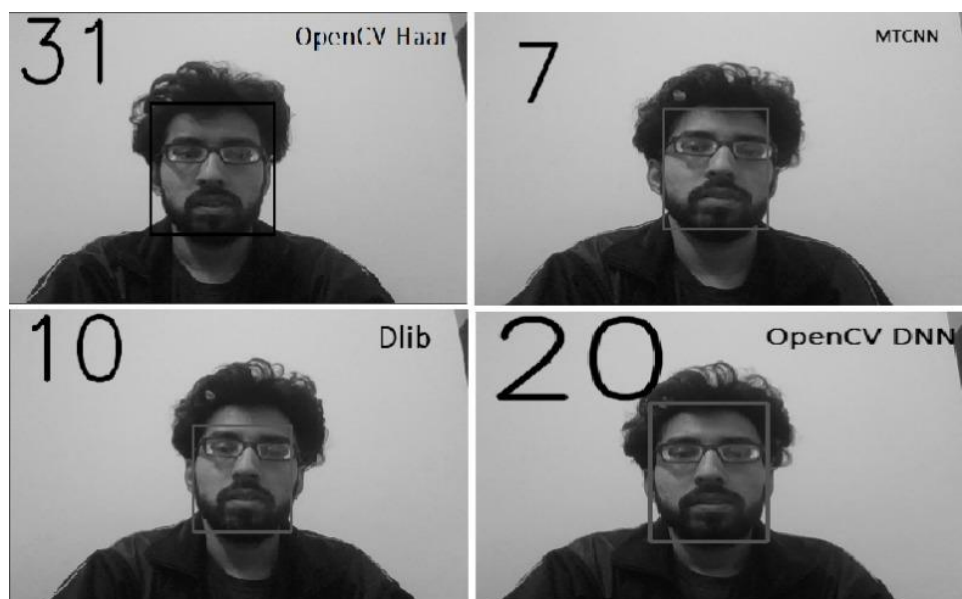
**Figure 9.** Comparison of algorithms in poor-lighting conditions



**Figure 10.** Comparison of algorithms during occlusion of face

**4.2.2 Face occlusion.** In this experimental set-up, images with occlusions of face was captured using web cameras. In the example image shown in figure 10, the face is partially occluded using the hand. For images with face occlusion, DNN and MTCNN performs the best, followed by HOG-based face detector. Haar cascade fails to detect faces in most of the face occlusion cases.

**4.2.3 Frame rate comparison.** In this experiment, the frame rate is estimated in fps for real-time videos captured using web cameras. A screenshot of one of the videos displaying the frame rate for all the four face detection algorithms is shown in figure 11. The frame rate of Haar cascade shows the best result, followed by DNN and HOG-based face detector. MTCNN shows a poor frame rate.



**Figure 11.** FPS comparison of algorithms



**Table 1.** Results obtained by testing 5 images on each face detection algorithm in different experimental set-ups: good lighting, low lighting, occlusion and frame rate

Number of faces detected for each condition							
	Good-Lighting Conditions (N=5)		Low-Lighting Conditions (N=5)		Occlusion (N=5)		Frames per second (Avg)
	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	
	N: number of faces = 20						
Dlib	5	0	4	1	3	2	19
OpenCV Harr	5	0	2	3	3	2	30
MTCNN	5	0	5	0	4	1	7
OpenCV DNN	5	0	4	1	4	1	23

Twenty tests are done for the performance analysis of the face detection algorithms. Table 1 shows the results of algorithms' performance on the count of faces correctly detected. MTCNN and DNN performs the best for the cases of varying light conditions and face occlusion. However, MTCNN has a low frame rate.





**4.2.4 Effect of input image resolution.** To analyse the effect of varying the image resolution on face detection output, four images obtained from the internet are tested for different resolutions such as 512x512, 416x416, 320x320 and 224x224. The score threshold is kept at 0.5 by default during testing. Table 2 shows the results of the face detection on varying input resolution for the HOG-based face detector in Dlib. It performs well for input resolutions of 512x512 and 416x416, and performs the worst for resolution of 224x224.

This work has explored the performance of different face detection algorithms on varying image capture conditions. A web-app has been developed for face detection in real-time images and videos. A few limitations of the work is that the image dataset used for the performance analysis is small. Only four face detection algorithms were implemented. The work needs to be validated on a larger dataset with various types of diverse images. However, the web app developed can find many promising applications.

#### 4.3 Future scope of the work

There are many promising applications that directly or indirectly use face detection. There are applications like online test monitoring, face unlocks, identify people on social media, identity validation and in security applications to catch shoplifting, detecting face peeking, etc. [12]. All these applications have their own requirements and constraints and have different environment for operation. Hence, the vendor can do the testing on which algorithm is needed for his/her application, without going deep into the technical details. This work can be further extended by providing a set of best-suited applications for each algorithm.

**Table 2.** Testing the effect of varying image resolutions on HOG-based face detector

	Image No.	512 x 512		416 x 416		320 x 320		224 x 224		Algorithm - Dlib
		TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	
	Image_01	1	0	1	0	1	0	0	1	Score Threshold - 0.5
	Image_02	1	0	1	0	0	1	0	1	TRUE -> All the faces in the image are detected correctly.
	Image_03	1	0	1	0	0	1	0	1	FALSE -> Not all the faces are detected.
	Image_04	1	0	1	0	1	0	0	1	

## 5. Conclusion

In this paper, we present a web app to compare four face detection algorithms. The web app has features for selecting different face detectors, changing the resolution of image, face extraction and observing the frame rate of the algorithms. Haar Cascades gave the worst result in all the experimental analysis. It also fails to detect the right position of face in many cases, hence giving a lot of false positives. HOG-based face detector and MTCNN gave significantly good results that are comparable. However, HOG-based face detector failed to detect faces in certain outlier cases. The disadvantage of MTCNN is that it gave the lowest frame rate. DNN has shown the best performance for all the test cases in all the parameters. The web app can be developed further to be used for customer-specific applications.

## References

- [1] Bajpai S, Singh A and Karthik KV 2013 An experimental comparison of face detection algorithms *Int J Computer Theory and Engineering* **5** p 47
- [2] Rath SK and Rautaray SS 2014 A survey on face detection and recognition techniques in different application domain *Int. J. Modern Education and Computer Science* **6** p 34
- [3] Chandrappa DN, Akshay G and Ravishankar M 2012 Face detection using a boosted cascade of features using OpenCV *Int. Conf. Information Processing* (Springer, Berlin, Heidelberg) pp 399-404
- [4] Mori K, Matsugu M and Suzuki T 2005 Face recognition using SVM fed with intermediate output of CNN for face detection. *MVA* pp 410-13
- [5] Ma S and Du T 2010 Improved Adaboost face detection *Int. Conf. Measuring Technology and Mechatronics Automation* (IEEE) **2** pp 434-37
- [6] Cuimei L, Zhiliang Q, Nan J and Jianhua W 2017 Human face detection algorithm via Haar cascade classifier combined with three additional classifiers *13<sup>th</sup> IEEE Int. Conf. Electronic Measurement & Instruments (ICEMI)* pp 483-87
- [7] Zhou H, Chen P and Shen W 2017 A multi-view face recognition system based on cascade face detector and improved Dlib. *MIPPR 2017: Pattern Recognition and Computer Vision 2018* (International Society for Optics and Photonics) **10609** p 1060908
- [8] Cerna LR, Cámara-Chávez G and Menotti D 2013 Face detection: Histogram of oriented gradients and bag of feature method *Proc. Int. Conf. Image Processing, Computer Vision, and Pattern Recognition* (WorldComp) p 1
- [9] Shavers C, Li R and Lebby G 2006 An SVM-based approach to face detection *Proc. 38<sup>th</sup> Southeastern Symposium on System Theory* (IEEE) pp 362-66

- [10] Zhang K, Zhang Z, Li Z and Qiao Y 2016 Joint face detection and alignment using multitask cascaded convolutional networks *IEEE Signal Processing Letters* **23** pp 1499-503
- [11] Ghenescu V, Mihaescu RE, Carata SV, Ghenescu MT, Barnoviciu E and Chindea M 2018 Face detection and recognition based on general purpose DNN object detector *Int. Symposium on Electronics and Telecommunications* (IEEE) pp 1-4
- [12] Chen CY, Lin BY, Wang J and Shin KG 2019 Keep others from peeking at your mobile device screen! *25<sup>th</sup> Annual Int. Conf. Mobile Computing and Networking* pp 1-16