University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

The Sensitivity of Facial Analysis Algorithms to Race and Gender

Mohammed Zeerak

April 8, 2021

Abstract

Facial detection bias between races and gender is a serious concern within the field of computer vision. This technology is seeing practical use in the real world, therefore any bias present could incur detrimental consequences for the targeted group. This project investigated a range of facial detection algorithms to understand if, and why there existed a bias towards race and gender. The Viola-Jones, HOG, MTCNN and RetinaFace algorithms were chosen to represent the variety of the face detection scene. They were evaluated against a novel dataset which was comprised of represented and underrepresented faces, to assess if a bias was present. It was found that differing levels of bias existed within the algorithms. The Viola-Jones and HOG algorithms showed slight bias due to their feature extraction and training data. The RetinaFace algorithm showed no observable bias, and the MTCNN algorithm showed the largest bias. This was a result of the datasets they were trained with. Analysis of the algorithms led to understanding that bias within face detection stems from two major components. The first being the design of the algorithms and the second being the bias present in the datasets the algorithms are trained on. A deeper look into the datasets showed that this dataset bias is introduced through sampling bias and a lack of mitigation through benchmark datasets.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Mohammed Zeerak Date: 6 March 2021

Contents

1 Introduction

1.1 Motivation

Detecting faces has been an interesting topic of research within the wider object detection and computer vision field. With the rise of processing power and machine learning, it is only in the last decade that there has been advancements and improvements such that facial detection is seeing practical use within many sectors and industries. These include biometrics, where the face is the primary method of security (Chang et al. (2005)), to health in which face detection is being used to help understand and diagnose specific syndromes (Rai et al. (2015)). With the increase in use of face detection technologies, there has also been an increase in the number of problems associated with it.

The most important of these being, that face detection accuracy is dependant on factors unique to an individual such as race and gender. This results in a bias. The issue is that when face detection is used in the real world, then its impacts could have negative and unfair consequences to those who are treated with more inaccuracy by the algorithms. Looking at biometric security, if the algorithm detecting the face performs inaccurately due to factors unique to the individual. Then in turn they are negatively impacted as the bias has caused them issues with ensuring the security of their device.

This is a growing area of research and many large companies including tech giants like IBM (Puri (2018)) and Microsoft (Roach (2018)), are trying to reduce and understand the bias present in face detection. With the growing concerns behind the use of these technologies, it is important for this bias to be researched and understood so that it can be mitigated and trust can be regained in the emerging technology.

1.2 Aim

This project intends to provide quantifiable results and research into a set of face detection algorithms to answer the projects research question. Does there exists a bias within the face detection scene towards race and gender? If so, then how has this bias been introduced? This will be done through evaluating 4 face detection algorithms which differ in structure and design. The algorithms help represent the computer vision scene through their own novel approaches to face detection. The quantifiable results produced will be analysed to assist in understanding the sensitivity behind the facial detection algorithms, towards race and gender.

1.3 Dissertation Structure

This chapter identified the motivation and aims of the project. The remainder of the paper will discuss the findings and experiments to answer the research question behind the project. The paper is structured as follows:

- **Chapter 2**: Background, discusses types of algorithm, details of algorithms, cases of bias and previous work.

- **Chapter 3**: Implementation, discusses how each algorithm was implemented along side the requirements.
- **Chapter 4**: 5025 Dataset, describes motivation behind the novel dataset and its design.
- **Chapter 5**: Evaluation, discusses evaluation strategy and evaluation metrics that are used to assess bias.
- **Chapter 6**: Results, details experiments results and quantitative data gathered from each algorithm.
- **Chapter 7**: Discussion, analysis of results relating to research question.
- **Chapter 8**: Conclusion, summarises project results and describes potential future work to be explored.

2 Background

This chapter will explain key face detection concepts, as well as describe the 4 main algorithms of this research project in detail. Finally some cases of bias existing and previous work on projects of a similar aim will be detailed.

2.1 Face Detection

Face detection is the earliest component of facial analysis or recognition algorithms. It is the stage in which the presence of a face in an image is detected, and a bounding box is placed around the detected face. Landmark coordinates are often also placed to denote the position of facial features such as eyes, nose and mouth. This step is critical and serves as the initial stage of many computer vision algorithms. It is important to distinguish that face detection and recognition are not synonymous. Recognition entails identifying who the face belongs to, whereas detection focuses on locating and classifying the presence of a face.

If you gave someone an image and asked them to classify whether the image contained a face, the task would be trivial. They would take the image and start searching for what they think a face looked like. To humans, faces are obvious. To search for them, we look for features that identify a face such as the presence of eyes and a nose. We look for shapes and edges which resemble that of a face, and use context to understand what is being seen; Are the eyes above the nose and the mouth below both? As humans we use pre-existing knowledge on what a face looks like to identify if we see one or not. Computer vision follows a very similar idea. The computer searches for features similar to how humans do, and uses context to distinguish faces. The pre-existing human knowledge of what classifies a face is encoded into the computer such that it can understand what features are, and search for them. Since images are very complex, intermediate representations are generated such that the only details that remain in the image are what the computer can understand. If the computer comes across features in the image that it knows classify a face, then it can confidently output that a face has been detected.

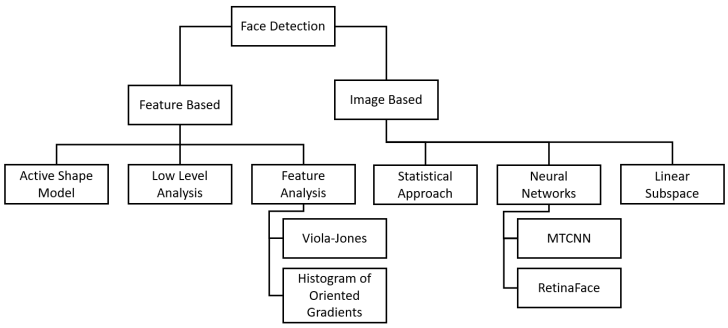

Figure 2.1: *Diagram of face detection approaches modified from: Face Detection: A Survey (Hjelmås and Low (2001)).*

2.2 Types of Face Detection

Face Detection: A Survey (Hjelmås and Low (2001)), details that face detection approaches can be placed into two distinct categories. These being feature-based methods and image-based methods, seen in figure 2.1. Both methods use a similar approach to detect faces by searching for features within an image. The difference comes from how the representation of a feature is encoded into the algorithms.

2.2.1 Feature-Based

Feature-based detection is form of face detection that uses handcrafted filters that look for features in the image. These filters are created manually and leverage the pre-existing knowledge into the facial domain to understand what features can be extracted from the image. The knowledge of the facial domain consists of understanding things such as the geometry of features, regions of brightness and other hidden structures that are present in faces. These shared characteristics of the face are encoded into a classifier that is trained to recognise them. If the classifier locates these features in an image. Then a face is detected. This type of face detection was a very popular method to detect faces as it was relatively easy to understand and implement. Many of the early detectors in the field of computer vision were feature-based detectors. They are still used today, combining with machine learning classifiers to become strong tools when it comes to face detection and classification. They are usually very efficient and accurate when the filters can match in the image. Although, due to their handcrafted design, they can fail spectacularly when the filters don't match. This results in a lack of robustness when using them in uncontrolled environments, where the faces could have different expressions, lighting and occlusion. These filters could be engineered to perform better in these circumstances but this becomes a difficult and complex task as the number of factors the filter has to be prepared for increases.

2.2.2 Image-Based

Image-based detection is another form of detection and is the category most of the current state-of-the-art detectors belong to. CNN and deep learning are both image based methods (Brownlee (2019)). In this method the face detection is accomplished by using a convolutional neural network, which in its base form is a deep learning algorithm. It can take a set of images and understand the hidden structures and encodings within them automatically. This is done through the use of the convolution layer, which is where the the input image is multiplied by a set of weights, these weights are the filter. By training the algorithm with a dataset of annotated faces, to minimise the euclidean distance between predicted coordinates and ground truth coordinates. The weights and therefore filter, are altered in accordance to this learning problem. This means the weights/filter would eventually learn the task and be able to predict coordinates for landmarks and bounding boxes with accuracy. Unlike handcrafted detectors where the filter has to be created depending on pre-existing domain knowledge. The CNN method allows a filter to be learned and optimised automatically using a dataset and learning problem (Brownlee (2017)). It is a very robust method and benefits from more data being fed into it to further improve its prediction accuracy. With the rise of faster GPU's, image-based methods dominate the face detection field.

2.3 Algorithms

Following are the details behind the 4 algorithms that were evaluated for this project. The algorithms differ in structure and design. They were chosen based on their novel approaches to face detection, as they contained a variety of components present in the computer vision scene.

2.3.1 Viola-Jones Algorithm

The Viola-Jones algorithm (Viola et al. (2001)), is a foundational feature-based algorithm. It is one of the earliest algorithms in the field and the simplest. The ability to detect faces accurately and efficiently is why it is still being used today as a baseline comparison algorithm.

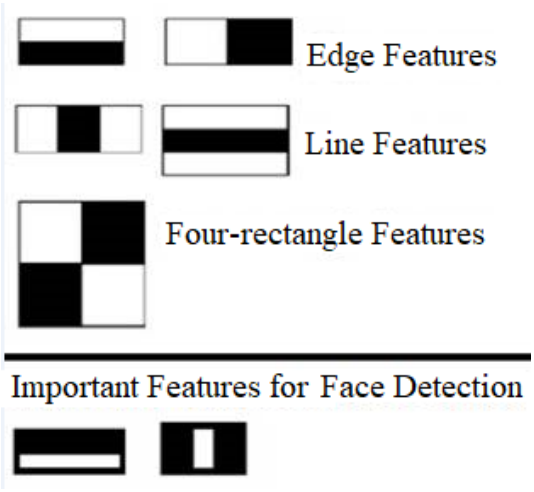

Figure 2.2: *Haar-like features taken from: Towards Data Science (Gupta (2019)).*

The algorithm works by using the pre-existing knowledge that features in a face vary in brightness. If the algorithm can find these features then it can classify the face. It does this by using haar-like features based on haar wavelets as described in Mallat (1989). These simple image features can be visualised as white and black adjacent rectangles as seen in Figure 2.2. They are calculated by taking the difference of pixel intensities of adjacent regions in a rectangle in the image. For example, in a face the region around the eyes would usually be darker than the eyes themselves resulting in a haar-like feature for the eyes. In the same way that the centre of the nose would be brighter than the region around it, resulting in another haar-like feature which represents the nose. This is described in figure 2.3, where the haar-like visualisations have been imposed over a face. This feature representation is what helps the algorithm understand what facial features it is seeing in an image.

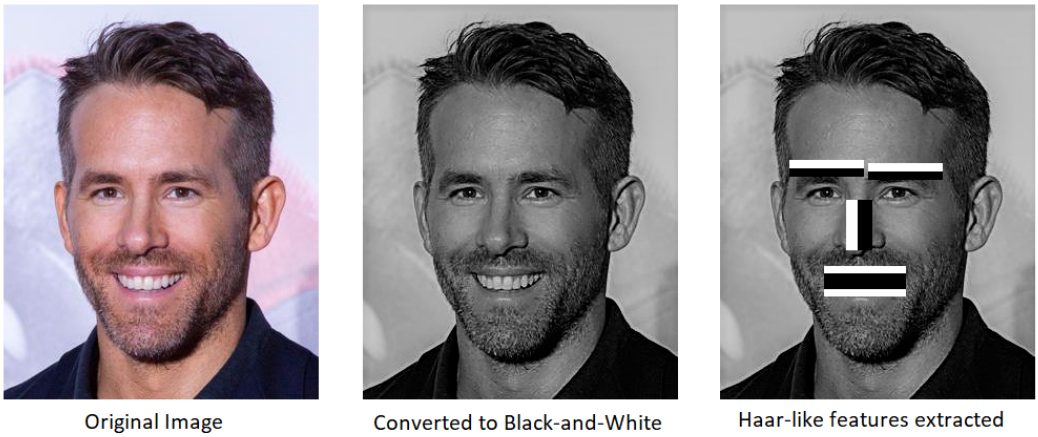

Figure 2.3: *Haar-like features imposed over face.*

Computing these features can be costly which is why the Viola-Jones algorithm uses an intermediate representation for the image, called the integral image. This allows for quick calculation of the haar-like features by having every pixel represent the sum of every pixel to its left and above it. To extract a feature, the 4 corners of a region in the integral image can be used to calculate the pixel intensity difference for the whole region. This is more efficient then adding up each pixels intensity separately in the regions and subtracting them.

To understand what haar-like features are the most important to classify a face, a strong classifier is created by combining weaker classifiers together through adaptive boosting. This works by starting at a weak classifier which used one feature to classify an image. This classifier was taken and another feature was added to it. The feature added to it was chosen based on how well it would compliment the previous classifiers feature. This meant that it was chosen based on the images the previous classifier got wrong, and what feature could best classify these images. This would increase the weight of the images the classifier had trouble with, and lead to optimising a strong set of features that classify a face.

The Viola-Jones algorithm uses a cascade of classifiers, where each classifier was trained with 9832 training faces and 10,000 non-face sub windows using the AdaBoost learning technique. The algorithm continually scans the integral image using a window looking for the most important general feature. If this haar-like feature is not present then the window can be rejected and classified as not containing a face. However, if the feature is present then the window moves to the next stage in the cascade, where another feature more specific than the previous is searched for. If this feature exists then the window again moves forward to the next stage. As the stages in the cascade continue, the haar-like features that are being searched for are more specific. If a window can go through every stage of the cascade without being rejected. Then every haar-like feature has been found, and the window contains a face. The cascade structure of the classifiers means that only the strongest candidate windows would reach the strongest and most computationally expensive classifiers. This method reduced the computation time drastically whilst improving the efficiency.

2.3.2 Histogram of Oriented Gradients

The histogram of oriented gradients (Dalal and Triggs (2005)) is a popular feature based detector available through the DLIB library, which is a machine learning and data analysis toolkit (King (2009)). The DLIB face detector can be broken down into 2 stages, a histogram of oriented gradients feature descriptor and a linear support vector machine classification algorithm.

The detector works by using a histogram of oriented gradients to represent an image. As a feature descriptor, its purpose is to represent the image in a form that is easier for the computer to understand. By transforming the image into its gradients, the excess information is removed that the algorithm doesn't require like colour and background. These oriented gradients describe the edges in an image through the changes in intensity.

To calculate the oriented gradients of an image. Two kernels are applied. These kernels output the respective horizontal and vertical gradients in the image according to the changes in intensity between the adjacent pixels around the target pixel. Using this information the gradient magnitude and orientation can be calculated for the horizontal gradient, g_x and vertical gradient, g_y.

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = arctan(g_y/g_x)$$

The pixels can be represented as a combination of oriented gradients with both magnitude (g) and direction (θ) associated with them.

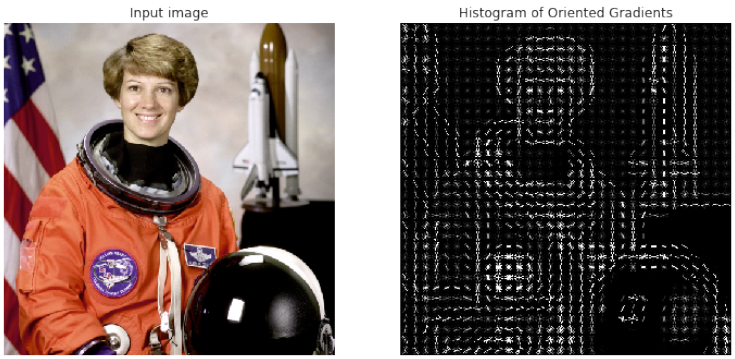

Figure 2.4: *Histogram of oriented gradients representation from: Opengenus IQ (Chopra (2017)).*

This results in an intermediary representation of the image with only its oriented gradients and magnitude as seen in figure 2.4. Here the image can still be realised even though it doesn't have its other characteristics like colour and background.

The next step is to generate the histogram of these oriented gradients within the image. This works by looking at 8x8 blocks of pixels in the image and calculating a histogram of the gradients and directions. Each histogram consists of 9 bins of 20^o. The bin is chosen depending on the value of orientation, and the value placed within the bin is dependant on the magnitude. This is to say that if a pixel is halfway between two bins, then its magnitude is split evenly for each bin. If the pixel is closer to one bin than the other, then one bin will have a larger share of the magnitude.

Often the 8x8 blocks are very sensitive to lighting differences as some might be brighter than others. To reduce this variation 4 8x8 blocks are used to create a single 16x16 block. From this block, the 36x1 vector representing the histogram will be normalised. For a 36x1 vector, V:

$$V = [x_1, x_2, x_3, ...x_{36}]$$

The normalisation factor k, would be the root of the sum of squares for each value:

$$k = \sqrt{x_1^2, x_2^2, x_3^2, ...x_{36}^2}$$

Resulting in the normalised vector being every value in vector V divided by the normalisation factor:

$$NormalisedVector = (\frac{x_1}{k}, \frac{x_2}{k}, \frac{x_3}{k}, ... \frac{x_{36}}{k})$$

This normalised vector would be calculated for all 16x16 grids in the image and combined to produce one large feature vector which would be trained with the SVM classifier to detect faces. If the classifier came across an image which resulted in a feature vector similar to a feature vector it had been trained to understand represents a face. Then the classifier would be able to classify that a face has been detected. This method uses the HOG descriptors as feature representations for the image, similar to how the Viola-Jones algorithm used haar-like features.

The landmark predictor present in the DLIB library is based on the algorithm from Kazemi and Sullivan (2014). The landmark coordinates are predicted through the use of a predictor trained on the 68 point i-BUG dataset (Sagonas et al. (2013)). An initial prediction of the 68 points are generated based on the mean of the training data scaled by the bounding box. This prediction is then altered by a slight amount and updated continuously based on adjacent pixel intensities to help regress the 68 points to their most accurate positions.

2.3.3 Multitask Cascaded Convolution Networks

The MTCNN algorithm is an image-based CNN algorithm which leverages interesting design to improve its accuracy and efficiency (Zhang et al. (2016)). It uses 3 different neural networks to perform bounding box calibration, face landmark localisation and face classification. The proposal network (P-Net) is used to propose bounding boxes and performs non-maximum suppression. The refine network (R-Net) is used to refine the bounding boxes through bounding box regression and NMS. The final output network (O-Net) is used to output the five facial landmarks present in the face. These 3 networks are visualised in figure 2.5

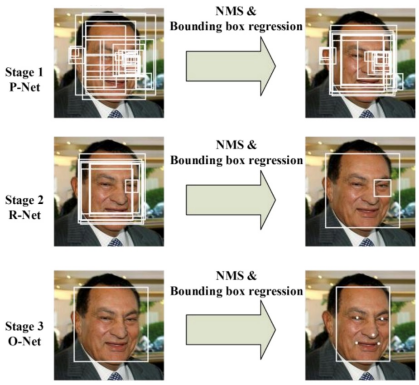

Figure 2.5: *MTCNN pipeline modified from: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks (Zhang et al. (2016)).*

The algorithm uses a sliding window to scan through the image and send 12x12 pixel sized regions to the first neural network, the P-Net. Although, before it can do this the image is first re-scaled to create an image pyramid. This is a series of the image all scaled to smaller sizes. The reasoning behind scaling the input image to different sizes is too reduce the complexity and allow the larger faces to be scaled down to a size that the scanning window can recognise, as well as maintain the larger image so that smaller faces within it can be identified. This is illustrated in figure 2.6, where the sliding window cannot recognise the original scale face as it is too large for the window. However, when the image is scaled down through the image pyramid. The sliding window is able to detect the presence of a face as it fits within the window size. This is an important step as it allows faces of different sizes the opportunity to be correctly recognised by the first network.

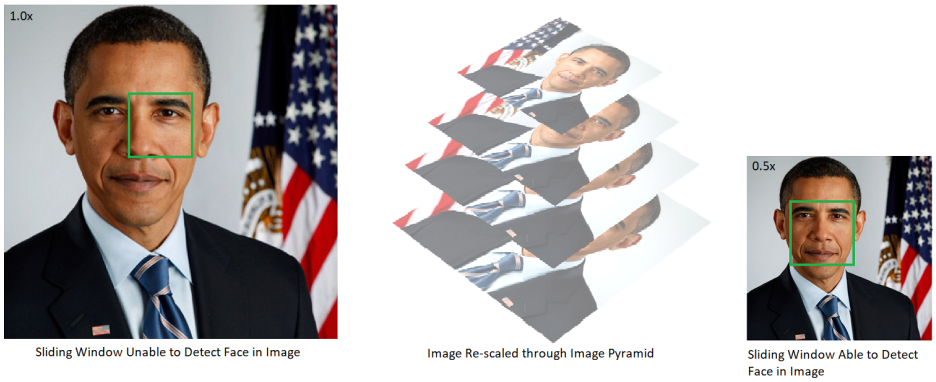

Figure 2.6: *Image pyramid re-scaling image for detection window.*

The P-Net proposes the first set of bounding boxes it detects in the image. Each box has a confidence score assigned with it, which relates to how confident the algorithm is that its successfully detected a face. To reduce the computational effort in the later networks, any bounding boxes with low confidence scores are removed. This reduces the number of candidates, but to further reduce them, non-maximum suppression occurs. This sorts the candidate bounding boxes by their confidence score and removes those candidates which have a large overlap with other higher confidence candidates. This means that the P-Net outputs a list of bounding boxes the network is most confident in. These are re-scaled and fed into the next network, the R-Net. This network performs similarly to the P-Net and looks to regress the bounding boxes to more accurate coordinates, as well as reduce even more candidates through NMS merge. The final network takes the bounding boxes returned by the R-Net and predicts the landmark locations and a confidence level for each box. Boxes with a lower confidence level are removed and NMS occurs. This should result in each face detected having a single bounding box and set of landmarks associated with it.

Each layer of the network is trained differently. The P-Net is trained using randomly selected images from the WIDER FACE (Yang et al. (2016)) dataset for classification and the CelebA dataset (Liu et al. (2015)) for landmark localisation. The R-Net is trained using the P-Net's detected faces from WIDER FACE for face classification and detected faces from the CelebA dataset for landmark localisation. The final network O-Net is trained similarly to the R-Net but both classification and landmarks localisation is trained on faces detected with P-Net and R-Net from WIDER FACE and CelebA respectively.

2.3.4 RetinaFace

RetinaFace is a single shot image-based detection method which has the goal of accurate detection of multiple faces in a image (Deng et al. (2020)). Single shot means that the method doesn't use the two stage architecture of proposal and refinement in which coordinates are proposed and rejected, but instead predicts directly from its feature maps through pixel-wise face localisation. This results in an efficient algorithm as there is no time consuming proposal stage.

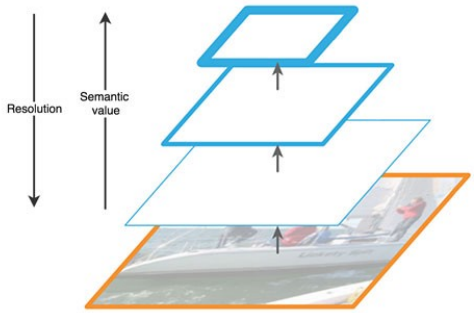

Figure 2.7: *Feature pyramid network semantic value and resolution from: Medium (Hui (2018)).*

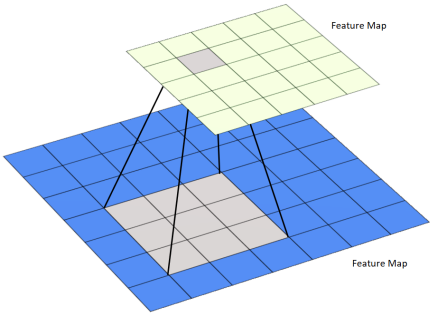

Figure 2.8: *Diagram showing how semantic information is increased through layers of pyramid.*

It uses a CNN method which utilises 3 main components. A feature pyramid network, context head module and cascade multitask loss. From these 3 components the most important to understand for RetinaFace is its feature pyramid network. The feature pyramid is similar to an image pyramid in only its shape. The difference is that in an image pyramid, the image itself is scaled to different sizes and features are extracted from each scale. This can be time consuming when it comes to training the algorithm and it often requires a large amount of memory for the multiple scaled images it has to store. A feature pyramid instead generates feature maps for the single scale image and then uses those feature maps to generate different sized feature maps as

seen in figure 2.7. A feature map is the output of applying a filter on an input image. Through figure 2.8, it can be seen that the structure of the feature pyramid network results in each layer taking its previous layer and reducing its size, but by doing so, also compressing the information from one feature map into the smaller output feature map. This means that each subsequent layer in a bottom-up approach has a lower resolution but more semantic information contained within it.

The RetinaFace algorithm uses these lower resolution, high level feature maps to detect faces. The layers with more semantic information can be used to reconstruct higher resolution feature maps which can then be used to make predictions of faces and landmarks. Due to the upsampling and downsampling, the location of faces within an image wont be precise. To solve this there are lateral connections between feature maps and their reconstructed counterparts at each layer. This helps the reconstructed layers with precise location and assists the detector in making more accurate predictions (figure 2.9).

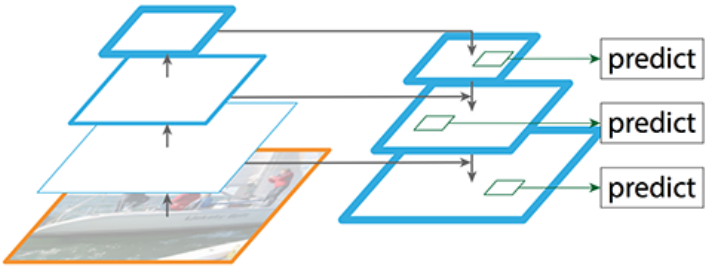

Figure 2.9: *Feature pyramid network upsampling and downsampling structure from: Feature Pyramid Networks for Object Detection (Li et al. (2019)).*

The RetinaFace algorithm is trained using the WIDER FACE dataset (Yang et al. (2016)), which contains 32,203 images and 393,703 annotated bounding boxes. The five facial landmarks of nose, eyes and mouth edges have been manually annotated onto this dataset to train the algorithm for landmark detection.

2.4 Bias Existing

There are many notable cases of facial analysis algorithms performing in a biased way and due to the rise of this emerging technology, these cases are becoming more common. In 2017 with Apple's new iPhone X launch, came their new Face ID feature which used faces as biometric security. Their implementation created issues as it was reported that in China the facial detection was under performing. It was unable to distinguish faces correctly, and would unlock the phone under falsely identifying someone as the correct user (Perillo (2018)). This problem was only evident in China indicating a bias towards race. Another example is Amazon's face recognition system called "Rekognition", mismatching members of the American congress to mugshots of criminals (Snow (2018)). This was criticised on the fact that 40% of the mismatched faces belonged to 20% of the people of colour. These disproportionate results indicated the underlying bias present in the company's system. At a larger scale the UK Home Office implemented a new passport checking service, in which the users face would be detected and compared against a database of faces to match faces to passports. The system failed as the facial detection didn't work upon very light or dark skin tone individuals (BBC (2019)). This was a technology that was approved and introduced into real world use which had a major bias present within it. It is evident there are many cases of this bias existing, and with the current use cases of face detection

in law enforcement and hospitals. The existence of bias would cause major repercussions for the targeted groups.

2.5 Previous Work

Since the mainstream introduction of computer vision technologies, only recently has there been dialogue in recognising the existence of bias within them. Therefore, there are not many published papers available that directly research this bias. However, there are a few pieces of literature that stand out.

2.5.1 Gender Shades

The Gender Shades paper (Buolamwini and Gebru (2018)), discusses an approach to evaluate bias present in facial analysis algorithms in regards to race and gender. They evaluated 3 commercial classifiers from Microsoft, IBM and Face++. They found that all classifiers performed better on male faces than female faces, and all classifiers performed better on lighter faces than darker faces. This led them to believe that single performance metrics are occlusive to the field and industry because the classifiers they tested all had accuracy values ranging from 87.9% to 93.7%. This meant that the classifiers could be described and marketed as accurate, but as their results indicated this accuracy broke down completely when it came to females and darker skin tones. They also recognized the lack of datasets available in which phenotypic features such as skin tone were labelled. Understanding that the lack of datasets causes a large training bias, they constructed the Pilots Parliamentary benchmark dataset. It consisted of 1270 parliamentary subjects that were evenly distributed in regards to gender and skin colour. This paper is important as it discusses many of the issues with bias and ultimately shows that it exists for even the largest tech companies algorithms. Being a recent paper from 2018 means that there is a lot of work left to do for facial detection to be used fairly in the wild.

2.5.2 IBM: Diversity in Faces

IBM's Diversity in Faces (Merler et al. (2019)), proposed a new unbiased dataset by providing a diverse large annotated face dataset to train and evaluate algorithms. In this paper diversity isn't reduced to only ethnicity or gender, which some other datasets conclude is representative. Instead diversity is put down to 10 different coding schemes which include gender, age, skin colour, facial symmetry and different craniofacial differences. This dataset looks to have an even distribution of faces according to their different coding schemes. They found that this approach to diversity would mean a largely unbiased dataset as diversity was being statistically measured to ensure that the different factors that made faces diverse were being evenly distributed. This paper also evaluated other popular datasets available. They found that these datasets like IJB-C (Maze et al. (2018)) and AgeDB (Moschoglou et al. (2017)), were heavily skewed towards containing a smaller percentage of darker skin tone faces and females. This emphasised the need for their fairer dataset. The paper made a large step in addressing the bias present in training data, and helping advance the study of fairer datasets.

2.5.3 Pride or Prejudiced?

Deep Learning for Face Recognition: Pride or Prejudiced? (Nagpal et al. (2019)), looked to develop a deeper understanding into the bias of deep learning face recognition systems, and if the bias existing in the deep networks was similar to that of the bias within the human brain. The key findings of their research included that when a deep learning network is trained upon data representative of real world situations, such as a lack of exposure to specific races/ages. Then the network would show in-group bias across these characteristics similar to how humans would.

Their experiments also indicated that in the case of two separate sets of images for two different races. That training a model on one race would always result in worst performance for images of the other. The paper analysed the bias existing with deep learning methods, and looked specifically into the datasets as opposed to the components of the network. This brought forward the conclusions that there are benefits in using larger datasets to improve performance. However, it is not a complete solution to remove bias in regards to race and age.

2.5.4 Role of Demographic Information

Face Recognition Performance: Role of Demographic Information (Klare et al. (2012)), looked at 3 more commercial face recognition algorithms and evaluated their performance against a dataset labelled with race, ethnicity, gender and age. The results of the study found that females in the black and younger groups were harder to recognise for every algorithm. They found that training the algorithms with images of faces that are well distributed across the groups is very important in reducing the difficulties for recognising the specific groups. This paper was published in 2012 and shares many relevant findings with the 2018 Gender Shades paper. This shows that within a six year gap there hasn't been a considerable improvement in reducing bias at a large scale, even though the problem has been evident for years.

3 Implementation

This chapter discusses the brief set of requirements and implementation details behind the 4 algorithms that were used for this projects research.

3.1 Requirements

When implementing the different algorithms there were certain requirements that had to be kept in mind. These requirements were formulated based on the research question. They describe the minimum required from the implementation, so that it can be evaluated to see if there exists a bias.

- Algorithms must provide bounding box coordinates.
- Algorithms must provide landmark locations.
- Open source implementations used must not differ from the training described in the original papers.

Requiring the implementations to provide bounding box coordinates and landmark locations is necessary as this is the first stage of detection, and the coordinates are required to indicate accuracy. Ensuring that the implementations used weren't trained on other datasets was also necessary, because evaluation of the algorithms training data would be harder if they were fine tuned with out explicitly stating so.

3.2 Notebooks

The algorithms were implemented within the Google Colab environment. This is a cloud based Python notebook that allows for quick execution of code and provides a GPU memory allocation for any machine learning tasks. Each algorithms implementation was self-contained in its own notebook as they used an assortment of libraries and modules which would've caused compatibility issues if present in the same notebook.

3.2.1 Viola-Jones Algorithm

The Viola-Jones algorithm is implemented using the OpenCV Python library (Bradski (2000)), which consisted of useful Python bindings that assisted in computer vision and more specifically, face detection tasks.

To run the algorithm, the pre-trained haar cascade classifier present in the OpenCV library is instantiated. Accessing the frontal face classifier allows the face detector to detect faces that are fully frontal facing because this is what the classifier has been trained for. The image is imported into the notebook through OpenCV. This imports the image as a colour image in the BGR colour mode. Since the classifier doesn't require colour, the image is converted to gray scale. To call the detector, the *detectMultiScale()* function is called with an image as a parameter. This returns an array of length 4 that contains the bounding box values. These values are bounding box's x and y coordinate of the top left point, along with the height and width. Using these

4 values, the coordinates of the 4 corner points on the bounding box can be obtained. The bounding box values for each image are placed into a dictionary and exported as a JSON file for further analysis.

The Viola-Jones algorithm doesn't output landmark locations because the filter is only trained to classify faces and output bounding boxes. There are smile classifiers and eye classifiers available through the OpenCV library, these could be used to get the location of landmarks. However, these filters only place a box around the feature instead of returning a coordinate point. The lack of precision and not being discussed in the original paper is why these filters weren't considered for the implementation.

3.2.2 Histogram of Oriented Gradients

The 68 point HOG detector implementation is heavily inspired from Adrian Rosebrock's blog (Rosebrock (2017)), in which he discusses a method to detect facial landmarks using DLIB, OpenCV and Python.

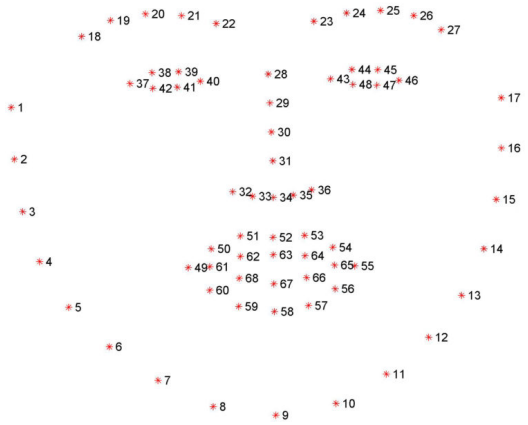

Figure 3.1: *Positioning of the 68 landmark coordinates from: Py Image Search (Rosebrock (2017)).*

The model being used has been trained on the iBUG dataset (Sagonas et al. (2013)), which consists of face images annotated with 68 points, these are visualised in figure 3.1. This model will predict coordinate points around the face and landmarks creating a mask. First the 68 point pre-trained model is downloaded and instantiated as the predictor through the dlib *shape_predictor function()*. The face detector is also instantiated using the *get_frontal_face_detector()* function. This function initialises the HOG + SVM detector that DLIB has altered to work with face detection as opposed to general object detection. Similar to the haar cascade classifier, the DLIB detector doesn't require the images colours, so each image is transformed into gray scale. The instantiated detector is called on an image, which returns a rectangle object that contains the 4 values to describe the bounding box points. The predictor is then called which takes the image and calculated bounding box, and returns the predictions on the 68 coordinate points within a shape object. The helper function transforms the shape object to a numpy array for easier manipulation.

The 68 point detector varies in point selection from the 5 point detectors. This is because the eye landmarks which sit in the centre of the eyes are not present in the 68 point predictions. These points are important because they are two of the five landmarks present in the other algorithms. The predictor does make predictions for points on the edges of the eye as seen by points 37,40,43 and 46 in figure 3.1. This means that an estimate of the eye landmarks can be generated. This estimate is created by taking the midpoint between the line connecting the edge points for each

eye. It should indicate the centre of the eye with reasonable accuracy. The nose and mouth landmarks are present in the 68 point model, so these are extracted from the 68 points predicted. The 5 landmark coordinates and 4 bounding box values are placed in a dictionary of structure seen in figure 3.2. This is then exported to a JSON file for further analysis.


```
{
    2: {
        "bounding_box": {"x": 35.0, "y": 118.0, "w": 186.0, "h": 186.0},
        "landmarks": {
            "left_eye": [91.0, 165.5],
            "right_eye": [169.5, 155.5],
            "nose": [128.0, 200.0],
            "left_mouth": [95.0, 241.0],
            "right_mouth": [186.0, 233.0],
        },
    }
}
```


Figure 3.2: *Dictionary structure of bounding box and landmark coordinates.*

3.2.3 Multitask Cascaded Convolution Networks

The algorithm is implemented through a Pytorch implementation of the pre-trained MTCNN face detection model available on Github (Esler (2020)). Using the infer notebook supplied on the repository provided a solid base to set the algorithm up. The implementation heavily followed the steps detailed within it.

First the MTCNN module is instantiated with its specified parameters. These include the minimum face size, image pyramid scaling factor, thresholds and margin. All the parameters besides image size remain at their defaults values. The image size parameter was set to 300. This was to ensure the best results, as the images it would be evaluated against would be of 300 width.

A loader object is then instantiated that is used to load in the images into the algorithms detection stage. For every image in the loader the *detect()* method is called on it, with the landmarks parameter set to true. This returns bounding box values, landmark locations and a probability value. The probability value is a number between 0 and 1 indicating how confident the algorithm is in its prediction. The bounding box values returned aren't of the standard form of x,y,w,h but instead are the x,y values of the top left and bottom right coordinates in the bounding box. To put these into the standard form of x,y,w,h. The height and width is calculated using the distance between the coordinates, and stored alongside the x and y coordinates of the top left point. The bounding box values along with the landmark locations are placed into a dictionary (figure 3.2), which is exported as a JSON file.

3.2.4 RetinaFace

The RetinaFace algorithm is implemented based on a PyTorch implementation on Github (Iglovikov (2020)). The detection is very simple and all that is required is the pre-trained model. This pre-trained model is trained on the WIDER FACE (Yang et al. (2016)) dataset as stated in the RetinaFace paper (Deng et al. (2020)). It is downloaded through the *get_model()* function and is instantiated as the model with the parameters of max size set to 300. The max size is set to 300 to improve the performance of the algorithm. Every image is of 300 width with varying height to maintain its ratio. This means that setting the max size to 300 allows the same model to be used for every image and provide consistent results. After the model is set, face detection can be carried out by calling the *predict_jsons()* function on each image. This returns a list of dictionaries for every face it detects. Each dictionary contains the bounding box coordinates of the top left

and bottom right points, as well as the landmark coordinates. The bounding box coordinates are used to generate the height and width, so that they can be stored of the form x,y,w,h in the output dictionary (figure 3.2), alongside the landmarks. This dictionary is then exported as a JSON file.

4 5025 Dataset

This chapter discusses the motivation and design methodology behind the novel 5025 dataset. With in depth discussion on design decisions and limitations of the dataset.

4.1 Motivation

Throughout the course of the project it was realised that a dataset would be required to evaluate the performance of the algorithms. The motivation was to obtain a dataset that had an evenly distributed number of different races and genders, such that the accuracy of the algorithm could be compared between the different groups within the dataset. This annotated dataset was called the 5025 dataset and consisted of the images seen in figure 4.1.

Figure 4.1: *Images of 5025 dataset.*

4.2 Design

4.2.1 Pre-existing or Novel

There were many considerations into the design of the dataset. The largest being whether to use a pre-existing dataset or compile a new one. Due to previous research being completed on understanding the bias within facial analysis algorithms, there existed datasets that contained faces of varying gender and race along with annotations labelling these aspects. These datasets were detailed in other papers such as Gender Shades (Buolamwini and Gebru (2018)) and FairFace (Karkkainen and Joo (2021)). Using these would be a valid approach as the annotations relate specifically to race and gender, which would allow the dataset to be split into distinct groups for evaluation. However, there is a lack of response in requesting many of these datasets which reduces their viability. There also exists difficulties coming from the variance of photos in the dataset. Often these images are taken in the wild, meaning that considerable effort would be

required to comb through the datasets for images that were of little variance. Therefore, to allow for more control of every aspect of the dataset, a novel one was created.

4.2.2 Dataset Diveristy

To ensure the dataset was diverse and valid for evaluating if a bias was present, the judgement was made that the two factors of race and gender would be combined evenly into two categories. Represented and underrepresented faces. These categories better described the different races and genders that could be affected by bias, whilst maintaining two distinct groups. Gender was limited to males or females, and was split equally between the two categories. Race however wasn't limited the same way. Looking towards a specific race or by extension skin tone, would result in many groups in the dataset. To overcome this, race was generalised by using the idea of representation. Representation would consider how well a race was represented in the available datasets, and it would be placed in the corresponding group. This allowed for multiple races to be categorised in the dataset under the two groups. Underrepresented faces were faces considered to be underrepresented in benchmark datasets, which resulted in African American, South Asian and East Asian faces present. Whereas represented faces contained those of mainly European and American Caucasian faces, which are prevalent as the majority of faces in benchmark datasets. By considering race not as discrete races but as a continuous spectrum, allowed multiple races to be categorised into each group. This ensured the dataset was diverse, whilst maintaining it's ability to assess if a bias was present with race and gender.

4.2.3 Collation of Images

There were multiple ways to collate the images. These being through Google Images, pre-existing datasets or manually capturing images. The latter would have involved taking photos of participants belonging to represented and underrepresented groups within a controlled environment. It would provide the most control over the images but due to ongoing Covid-19 pandemic, it wasn't viable. The second option would've been to comb through the datasets that were available and select faces belonging to a specific race or gender, such that there was an even distribution. The problem with this was that many of the datasets were not labelled with race or gender, which combined with their large number of images, made it difficult to automatically or manually extract images of specific race or gender from them. The option of Google Images was chosen to collate dataset photos. When choosing the photos, it was ensured that they were of creative commons licence and that the photos had relatively little factors that would impact them in comparison to others. These factors included expression, illumination, shadows, occlusion and pose, as described in Shinwari et al. (2019). Minimising the differences was important in ensuring a fair evaluation of each group in regards to only race and gender. The images were collated with the goal to ensure faces were fully frontal facing and showed little variance. To increase the number of available photos to select from, celebrities and public figures were specifically searched for, as they would have more photos available of them due to their significance. It was difficult to minimise the variance in the photos when capturing them from Google Images because the photos of the celebrities were taken under different conditions. To counteract this the photos were cropped and aligned manually. Faces showing neutral or smiling expressions and images that didn't have radically different lighting conditions were also specifically chosen to reduce variance. To further bring consistency to the dataset, the collated photos were resized so that every image was of 300 pixel width whilst maintaining the same aspect ratio. This ensured that the results wouldn't be skewed depending on the size of each face.

4.2.4 Technical Details

The dataset consisted of 48 distinct faces. Each image had a width of 300. This was decided by using the smallest width image and resizing every other image to match its width whilst

maintaining the same aspect ratio. There were 24 represented faces and 24 underrepresented faces. Both these groups contained an equal number of males and females at 12 each. Each image was labelled following the scheme:

ID_GENDER_GROUP

The ID was the face id ranging from 0 to 47, the gender was assigned either "m" or "f" indicating male or female and the group was assigned "m" or "r" indicating underrepresented or represented.

4.2.5 Annotations

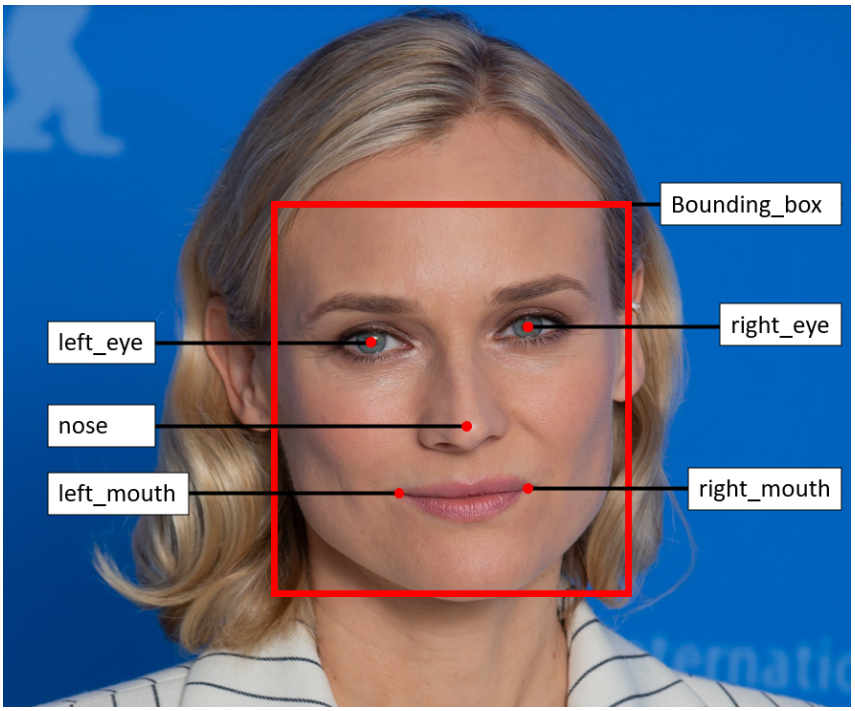

Figure 4.2: *Different landmarks annotated for each face.*

The dataset was annotated to provide a ground truth for the bounding box and landmarks. This consisted of annotating coordinates for the 5 landmarks and bounding box present for a face. The 5 landmarks consisted of the 2 eye landmarks, the nose landmark and the 2 mouth landmarks. The bounding box annotations were the x,y coordinates denoting the top left corner of the bounding box, along with the corresponding height and width in pixels. These annotations can be seen visualised in figure 4.2

They were generated manually through the use of MakeSense.ai (Skalski (2020)), which is an online tool that provided a graphical interface to place points on each image and export the resulting coordinates as a CSV file. Manual annotation was chosen because each point could be placed accurately. With only 48 images, this task viable and not time consuming. A collaborative method was considered that would allow participants to annotate the images, which would then be averaged and used as the ground truth. However, this method was not used because it was apparent that different participants had a differing ideas of what they would consider a landmark location and bounding box. Meaning the results would be inconsistent even when averaged. Therefore, manual annotation of the image was completed by one individual.

4.2.6 Limitations

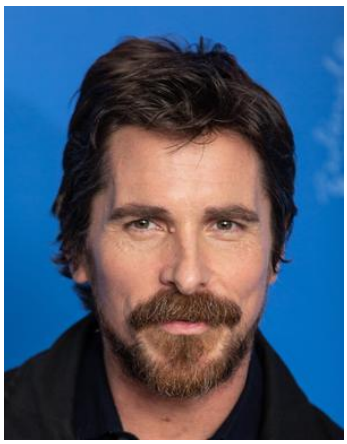

Figure 4.3: *Occlusion of face due to facial hair.*

Figure 4.4: *Expression variation due to smile.*

Figure 4.5: *Pose variation due to head rotation.*

Considerable effort was put in to the creation of the dataset such that it could provide a valid evaluation for the research into the algorithms bias. This however didn't ensure the dataset was perfect as it had many limitations. The size of the dataset was a major limiting factor. The relatively low number of images meant that the accuracy of the average results would suffer, as well as introduce a larger error margin. The other limitation was the variance present in the characteristics of the images. Images were selected to reduce variance but due to the selection method, it was difficult for every image to be standardised. Variance within the dataset consisted of slight pose variation in which some faces were not fully frontal facing (figure 4.5) as well as expression (figure 4.4), and occlusion due to facial hair (figure 4.3). These variables of the images were difficult to control and would impact the robustness of the results, when looking for a bias.

5 Evaluation

This chapter discusses the strategy used to evaluate the algorithms and assess if there exists a bias towards race and gender. It also details the metrics used and the figures generated.

5.1 Evaluation Strategy

The general methodology followed that of previous work such as Gender Shades (Buolamwini and Gebru (2018)) and InclusiveFace (Ryu et al. (2017)). Where an algorithmic framework was chosen and evaluated against a dataset containing an evenly distributed number of faces, of different genders and races. The performance of the algorithm on the different groups indicated if any bias existed towards the groups.

For the specific research question, the algorithms were chosen such that they represented the different aspects of the computer vision field throughout the last 20 years. This was done by using algorithms that had the old foundational hand crafted filter structure, and also using newer state-of-the-art algorithms that leveraged deep learning to detect faces. The dataset used was the 5025 dataset, which consisted of two groups, represented and underrepresned faces, that contained different races and genders. To understand if a bias existed within the range of algorithms, they were executed against the 5025 dataset and the resulting coordinate predictions were compared to the ground truth coordinates for each image. This comparison provided a strong indication to the accuracy of the algorithms against the represented and underrepresented groups. If an algorithm was more accurate for one group over the other, then a bias could exist towards race and gender.

5.2 Evaluation Metrics

The accuracy measure is calculated using the difference between the ground truth coordinates and the algorithms predicted coordinates. This involved taking the euclidean distance between the predetermined ground truth, and the predicted coordinates by the algorithms. This distance value on its own is not useful as a comparison metric because every face differs in size. To combat this discrepancy, each distance value for a face is normalised by the euclidean distance between the ground truth eye landmarks for that face. These normalised values are multiplied by 100 to produce larger numbers to work with. This metric called "Error Differences" is calculated for every annotation across the faces and algorithms. It shows the normalised distance between the ground truth coordinates and the algorithms predicted coordinates for a specified face.

The metric is used to assess the accuracy of the underrepresented and represented groups in the dataset, and generate the different figures detailed below.

5.2.1 Frequency of Error Differences at Different Thresholds

The frequency of error difference at different thresholds is calculated by counting the number of error differences above each error difference magnitude for both groups. The number of errors

indicates the accuracy of the group. If there is a large number of errors at the large thresholds then the group is inaccurate. The data is plotted as a line graph for both groups, in which a difference between the lines indicates a difference in accuracy.

5.2.2 Bounding Box Accuracy Histogram

The intersection over union explains a lot about the accuracy of the bounding boxes (Rosebrock (2016)). To calculate this, the area of overlap is divided by the area of union between the ground truth bounding box and predicted bounding box. A high percentage overlap of the predicted bounding box with the ground truth bounding box indicates a high level of accuracy. The percentage of overlap for each face and algorithm is placed into bins with intervals of 4%. This means that the bins contain the frequencies of certain magnitudes of overlap. Histograms are generated using these bins, which show the distribution of bounding box accuracy between both groups for every algorithm.

5.2.3 Landmark Error Difference Histogram

The landmark error difference histogram gives more detail into the performance of the groups by separating the performance of the algorithm by landmark. To calculate the histogram of error differences for each of the landmarks. The error differences between both groups for each landmark are placed into 7 equally sized bins ranging from 0 to 14 error difference. The bins capture the frequency of the error differences at specific magnitudes for the landmarks. Histograms are generated from the sets of bins for both groups. These histograms show the distribution of error difference for each of landmarks, across both groups and every algorithm.

5.2.4 Average Error Difference

The average values are calculated by using the landmark error differences of both groups for each of the algorithms. The error differences are added up for every landmark and are divided by the size of the groups, which is 24. This results in average error values for every landmark across the groups and algorithms. Each average value is used to create a difference metric between the groups. This is the average error difference of the underrepresented landmark subtracted by the average error difference of the represented landmark. If the value is positive then represented faces perform better, and if the value is negative then the underrepresented faces perform better. Since the average is being calculated, there is a corresponding uncertainty with it. The uncertainty is calculated through bootstrapping. Where the average results are taken of each landmark for multiple random subsets of the dataset. The standard deviation is calculated of these averages and gives an indication to the uncertainty of the average results for the landmarks.

5.2.5 Qualitative Image

To generate qualitative data each image within the dataset for the HOG, MTCNN and RetinaFace algorithms, has its bounding box and landmark locations imposed onto it. The ground truth is also imposed onto the images, so that visual comparisons can be made between the algorithms and faces. The Viola-Jones algorithm doesn't predict landmarks, so its visualisation only consists of the predicted bounding box and ground truth bounding box imposed onto the images.

6 Results

This chapter describes the results of the evaluation. The different metrics and figures generated are explained in detail.

6.1 Frequency of Error Differences at Different Thresholds

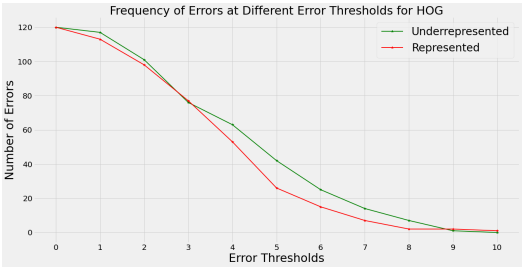

Figure 6.1: *Threshold graph for HOG error difference.*

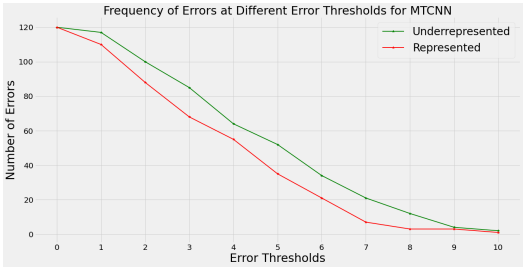

Figure 6.2: *Threshold graph for MTCNN error difference.*

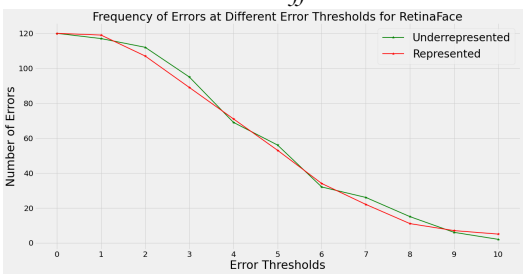

Figure 6.3: *Threshold graph for RetinaFace error difference.*

The 3 algorithms exhibit 3 differing trends between the represented and underrepresented groups in the graphs. The HOG algorithm (figure 6.1), shows a similar number of errors between the represented and underrepresented groups until the 3 error threshold. However, after this threshold there is consistently more errors for underrepresented faces than there are for represented faces. This differs from the MTCNN algorithm (figure 6.2), which shows a larger number of errors throughout every error threshold for underrepresented faces, with the largest difference of around 15 errors being between the 5 and 7 thresholds. The RetinaFace algorithm (figure 6.3), performs with more errors in total for each threshold in comparison to the other algorithms, but its performance between represented and underrepresented groups is very similar. With underrepresented faces showing only a slightly higher number of errors at the thresholds.

6.2 Landmark Error Difference Histogram

6.2.1 HOG

Figure 6.4: *Left eye histogram of error difference for HOG.*

Figure 6.5: *Right eye histogram of error difference for HOG.*

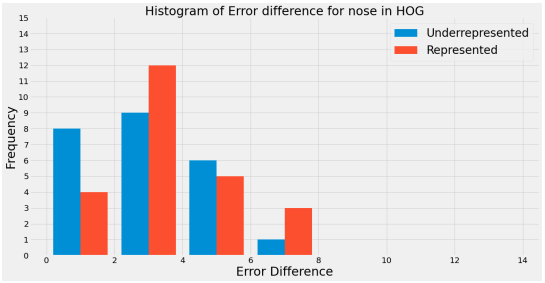

Figure 6.6: *Nose histogram of error difference for HOG.*

Figure 6.7: *Left mouth histogram of error difference for HOG.*

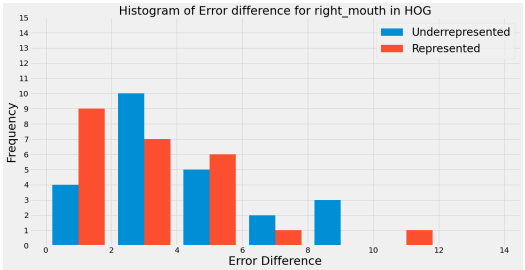

Figure 6.8: *Right mouth histogram of error difference for HOG.*

Figure 6.7 and figure 6.8 show that the left and right mouth landmark locations for the HOG algorithm have a significant difference between the groups. There are a larger number of errors before the 4 threshold for represented faces than there are for underrepresented faces. This difference being 32 errors in total for the represented group in both landmarks, and 24 for the underrepresented group. For every landmark there is a higher or equal number of errors present for underrepresented faces above the 8 magnitude than there are for represented faces. For both mouth landmarks at the 8 magnitude extreme, there are 5 errors present in total for underrepresented faces and 1 for represented faces. This indicates the underrepresented groups performing worst for the mouth landmarks. Although, for the eye landmarks the difference is of a much smaller magnitude. From figures 6.4 and 6.5, there are 16 errors in total for left and right eye landmarks of underrepresented faces below the 4 threshold, whereas there are 19 for represented faces. This shows represented faces performing slightly better for the eye landmarks. Looking at figure 6.6, underrepresented faces perform better than represented for

the nose landmark, as there more occurrences of smaller errors for underrepresented faces. This being 8 underrepresented faces under the 2 error threshold, compared to the 4 represented faces.

6.2.2 MTCNN

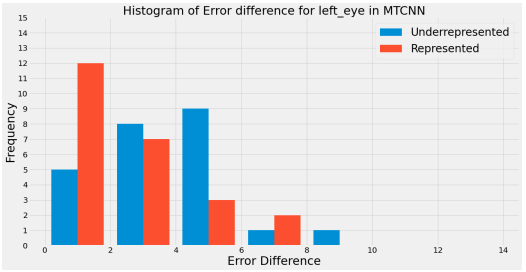

Figure 6.9: *Left eye histogram of error difference for MTCNN.*

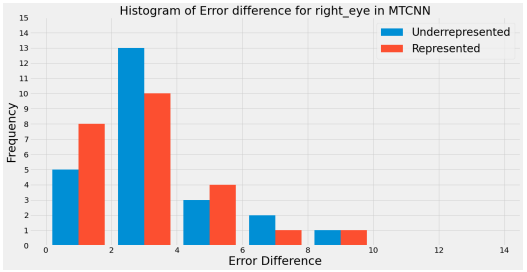

Figure 6.10: *Right eye histogram of error difference for MTCNN.*

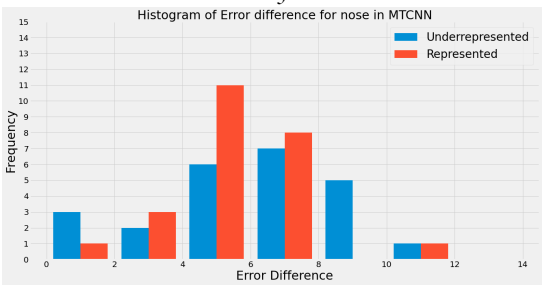

Figure 6.11: *Nose histogram of error difference for MTCNN.*

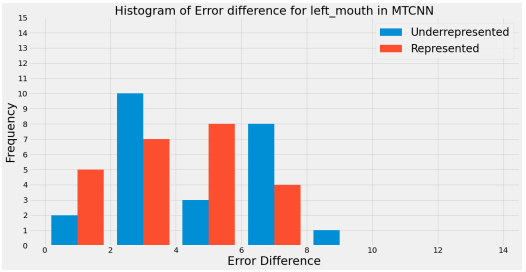

Figure 6.12: *Left mouth histogram of error difference for MTCNN.*

Figure 6.13: *Right mouth histogram of error difference for MTCNN.*

The MTCNN algorithm shows that there are more errors that are of a larger magnitude of above 6, for underrepresented faces in each landmark. With the left eye landmark in figure 6.9, not being consistent with this result, and instead having an equal number of errors between the groups. The nose landmark (figure 6.11), shows that after the 6 threshold there are 13 errors for underrepresented faces and only 9 for represented faces. However, the majority of the nose errors being 11 out of the possible 24 for represented faces occur between the 4 to 6 error difference magnitudes. This shows that even for represented faces the algorithm isn't entirely accurate. The left mouth (Figure 6.12), shows that under the 2 threshold the represented faces perform better with more errors at smaller magnitude with 5, compared to the 2 for underrepresented faces. Figure 6.13 for the right mouth, indicates the represented faces performing better than the underrepresented faces because there are more occurrences of errors at lower thresholds. At the higher thresholds for the mouth landmarks there are more errors for underrepresented faces.

The left mouth shows that there are 9 errors above a threshold of 6 for underrepresented faces, where as there are only 4 errors for represented faces. The same can be seen for the right mouth landmark which shows that there are 4 errors for represented faces and 7 for underrepresented faces. This means the mouth landmarks perform much worst for underrepresented faces. The right eye landmark (figure 6.10), shows that the performance between the groups is very similar but the represented group performs slightly better than the underrepresented. This is seen with there being 8 errors for represented and only 5 for underrepresented faces under the 2 threshold. The left eye landmark (figure 6.9), is the only landmark which shows that the represented faces performed equally with the number of errors above the 6 threshold, at 2 errors each. This equal performance is undermined by the higher number of errors for represented faces in the lower thresholds. Below 2 is where most of the represented errors are, with there being 12 errors compared to the 5 errors for underrepresented faces. This shows the represented faces performing better for the left eye landmark.

6.2.3 RetinaFace

Figure 6.14: *Left eye histogram of error difference for RetinaFace.*

Figure 6.15: *Right eye histogram of error difference for RetinaFace.*

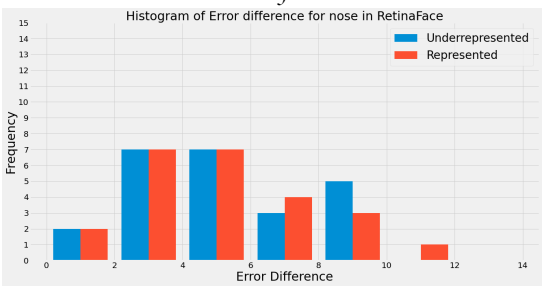

Figure 6.16: *Nose histogram of error difference for RetinaFace.*

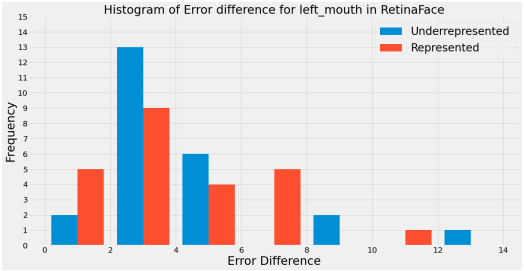

Figure 6.17: *Left mouth histogram of error difference for RetinaFace.*

Figure 6.18: *Right mouth histogram of error difference for RetinaFace.*

The RetinaFace algorithm has similar performance between both groups in each of the landmarks.

The left mouth landmark (figure 6.17), shows the largest difference between the groups under the 4 threshold, with there being 15 errors for underrepresented faces and 13 for represented faces. Figures 6.14 and 6.15 for the left eye and right eye both have a difference of 1 error, where there is 1 more error for underrepresented faces. The right mouth landmark seen in figure 6.18, also exhibits a difference of 1 error under the 4 threshold, but here there is one more error for represented faces. The nose landmark (figure 6.16), shows that under the 4 threshold the number of errors for represented and underrepresented faces are equal at 9 errors. There is no clear difference between the groups in any of the landmarks.

6.3 Average Error Difference

HOG	**Underrepresented**	**Represented**	**Uncertainty (+/-)**	**Difference**
left_eye_distance	5.048	4.411	0.246	**0.637**
right_eye_distance	4.282	4.434	0.131	-0.152
nose_distance	3.068	3.636	0.151	-0.568
left_mouth_distance	4.479	3.115	0.093	**1.364**
right_mouth_distance	4.19	3.186	0.436	**1.004**
total average	4.214	3.757	0.547	**0.457**

MTCNN	**Underrepresented**	**Represented**	**Uncertainty (+/-)**	**Difference**
left_eye_distance	3.778	2.574	0.196	**1.204**
right_eye_distance	3.606	3.045	0.132	**0.561**
nose_distance	5.935	5.472	0.392	**0.463**
left_mouth_distance	4.698	3.889	0.249	**0.809**
right_mouth_distance	5.053	3.716	0.42	**1.337**
total average	4.614	3.739	0.669	**0.875**

RetinaFace	**Underrepresented**	**Represented**	**Uncertainty (+/-)**	**Difference**
left_eye_distance	4.586	4.767	0.181	-0.181
right_eye_distance	4.19	4.422	0.311	-0.232
nose_distance	5.248	5.131	0.415	**0.117**
left_mouth_distance	4.164	4.194	0.319	-0.03
right_mouth_distance	6.148	5.705	0.344	**0.443**
total average	4.867	4.844	0.722	**0.023**

Table 6.1: *Average error values across algorithms for represented and underrepresented faces.*

The average difference values from table 6.1, described how much each algorithms performance leaned in relation to underrepresented or represented faces. The average values provided a metric that could be compared across both landmarks and algorithms for the entire dataset. The uncertainty values were calculated through bootstrapping by using the standard deviation of averages for multiple subsets of the results.

6.3.1 HOG

The largest average error difference for the HOG algorithm was at the left mouth landmark with a difference of 1.364. This difference was towards underrepresented faces being more inaccurate. The right mouth and left eye landmarks showed similar results but of a smaller magnitude, with right mouth having a difference of 1.004 and left eye with 0.637. It is important to point out that the left eye had the largest overall average error at 5.048 for underrepresented faces and

was the worst performing landmark for the HOG algorithm. The right eye and nose landmark showed differences leaning towards the represented faces performing worst. The right eye had a difference of -0.152 and the nose had a difference of -0.568. Overall the performance of the nose landmark was the most accurate for represented and underrepresented faces for the HOG algorithm. The average of all the landmarks in the represented faces was 3.757, with the average of underrepresented faces being 4.214, giving a difference of 0.457 towards underrepresented faces performing worst.

6.3.2 MTCNN

The MTCNN algorithm overall performed worst for underrepresented faces than it did for represented faces. All the landmarks have an average difference that is towards underrepresented faces performing worst, with the largest difference being the right mouth landmark. The right mouth landmark shows a difference of 1.337 between the represented and underrepresented averages. The left eye landmark also has a large difference between the averages of the two groups at 1.204. The nose landmark is the worst performing landmark in terms of averages for both of the groups, with an average of 5.935 for underrepresented faces and 5.472 for represented faces. Even though this landmark has the worst performance, it has the smallest difference between the two groups at 0.463. The left mouth landmark and right eye landmark show results consistent with the other landmarks, with differences of 0.809 and 0.561 respectively. The average for all the landmarks of represented faces is 3.739 whereas the average for underrepresented faces is 4.614 which gives a large difference of 0.875. This overall shows the underrepresented group performing worst than the represented.

6.3.3 RetinaFace

The RetinaFace algorithm has little difference between the represented and underrepresented faces. The largest difference is present in the right mouth landmark, for which the average error is 6.148 for underrepresented faces and 5.705 for represented faces, giving a 0.443 difference. The left mouth landmark shows the smallest difference between the groups at -0.03, this is also the smallest average difference in all of the algorithms. The right mouth landmark shows the largest difference between the groups at 0.438, where underrepresented faces perform worst. The right mouth landmark is also overall the worst performing landmark for both groups, with the right mouth for underrepresented faces having the worst average performance throughout all the algorithms at 6.148. The left and right eye landmarks both have slightly higher averages for the represented faces giving them a difference of -0.181 and -0.232 respectively. The nose landmark is the only other landmark besides the right mouth landmark where the underrepresented group performs worst than the represented group. Here the difference leans towards the underrepresented group performing worst but is still of a small magnitude at 0.117. The average error of all the landmarks is 4.867 for underrepresented faces and 4.844 for represented faces. This gives a small difference of 0.023 for the algorithm, which shows it performance is very similar between both groups across the landmarks.

6.4 Bounding Box Accuracy Histogram

Besides landmarks the algorithms also produced bounding boxes around faces that they detected. Looking at the overlap with the ground truth was a strong indication of the accuracy.

For the HOG algorithm in figure 6.19, it is evident that majority of represented faces are between 68% and 76%, with the majority of underrepresented faces between 76% to 80%. However, in the lowest percentages from 60% to 68% there are 5 underrepresented faces and 3 represented faces. At the other end of the spectrum above 88% there are no underrepresented faces but

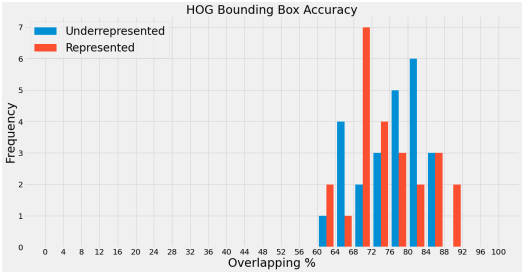

Figure 6.19: *Bounding box accuracy histogram for HOG.*

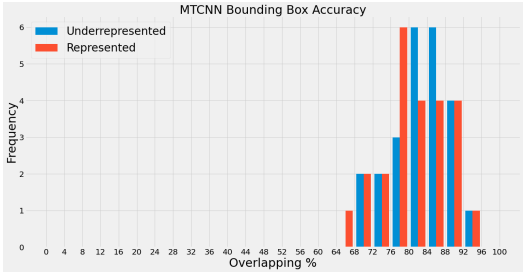

Figure 6.20: *Bounding box accuracy histogram for MTCNN.*

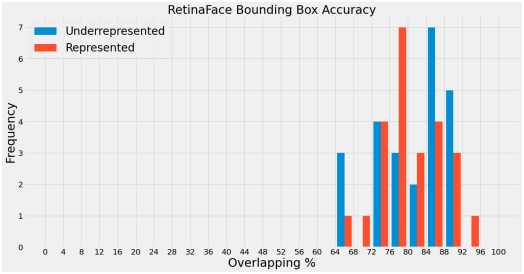

Figure 6.21: *Bounding box accuracy histogram for RetinaFace.*

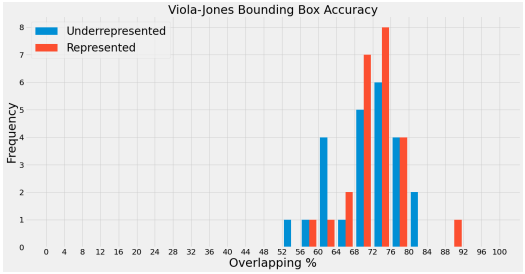

Figure 6.22: *Bounding box accuracy histogram for Viola-Jones.*

2 represented faces. This shows that at the extremes, underrepresented faces perform worst. MTCNN shows that in figure 6.20, the bounding box overlap of both groups is very high in the percentages and very similar. There is the same amount of faces with an accuracy above 88% for both groups. The difference is present between the 76% to 88% window, where represented faces perform slightly worst than underrepresented faces. Although overall the performance is very good for both groups. RetinaFace (figure 6.21), shows underrepresented faces performing better as there are 12 faces present above 84%, whereas there are only 8 for the represented group. Overall the performance for both groups is similar, with represented faces performing slightly worst. The Viola-Jones algorithm seen in figure 6.22, shows that the algorithm performs worst for underrepresented faces as there are 6 underrepresented faces under 64%, whereas there are only 2 represented faces. The majority of represented faces are in the accurate bounds of 68% to 80%, with their being 19 represented faces present in comparison to the 15 underrepresented faces.

7 Discussion

This chapter contains the analysis of the results from the evaluations. They are explained with respect to the research question of the project to understand if, and why there exists a bias towards race and gender.

7.1 Viola-Jones

This algorithm performs the worst in regards to its bounding boxes between both groups, although in its lowest percentages, there is a distinctly larger number of underrepresented faces than represented. This difference in performance could come from the fact that the the algorithm is an old foundational method. Its filter was created manually through the use of specific haar-like features. This feature representation is the core of the algorithm and is where the slight bias observed could be introduced. The haar-like features work by looking at the difference in regions of pixel intensity in the image. This helps the algorithm classify what features are being observed and if a face is detected. By using the difference in pixel intensities, the algorithm is looking directly at the difference in regions of brightness in an image. In represented faces of a lighter skin tone, this difference in brightness could be exaggerated and be easier for the classifier to understand. Whereas, in underrepresented darker skin toned faces, the difference between areas of intensity could be harder for the algorithm to distinguish as haar-like features. This would result in decreased accuracy with underrepresented faces, which is observed. However, it is important to state that there is a lack of information behind what images were used for the training of the algorithm. Therefore, it is viable that the bias could be introduced through the training data as well. An assumption can be made that because algorithm is one of the foundational methods in computer vision. That the goal early on in research wasn't to ensure fairness in the results but instead to just generate valid results. Meaning an unbiased dataset was probably not considered and in turn would be partly responsible for the slight bias seen.

7.2 HOG

The HOG algorithms performance after the 4 threshold indicates that it performs worst with underrepresented faces. By looking at the histograms and averages, it is clear to see that the majority of these large errors come from the mouth landmarks. This high difference could be the result of the algorithms landmark predictions being based on the bounding box generated through the HOG method. If the original bounding box prediction is inaccurate, then the likelihood of an inaccurate landmark prediction increases. The bounding box is generated through the use of HOG features which rely heavily on gradient differences to detect edges and distinguish faces. This could mean that since the gradients on a darker skin toned face differ from that of a lighter skin toned face, the algorithm could have trouble detecting the difference in gradient and the edges of the face, which would result in it being unable to place an accurate bounding box. This can be seen in Figure 7.1, the bounding box is inaccurate and over estimates on the width of the face. This stems from the intensity difference between the shadows present and the skin tone on the left side of the face, resulting in difficulty for the algorithm to distinguish and detect the edge.

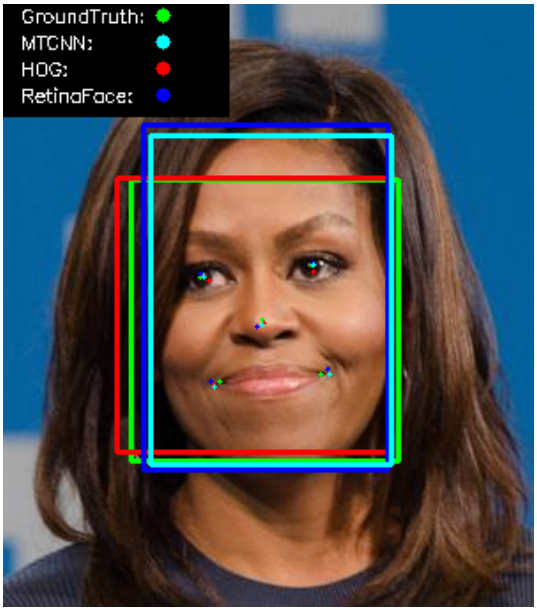

Figure 7.1: *Worst performing HOG face on average.*

This could be a valid reason and by looking at the bounding box overlap between the groups for the algorithm, it is clear that there is a slight difference between the accuracy of both groups. However, this difference is small, meaning that the bias is perhaps introduced somewhere else in the algorithm. The 68 point prediction is trained using the 68 point iBUG dataset (Sagonas et al. (2013)). It is comprised of 135 training images annotated with 68 points. Combing through the dataset, it can be seen that 119 out of the 135 images are of represented faces. This means that the model trained to make predictions, is trained with biased data and therefore would perform worst when it comes across faces it isn't trained with. This reasoning follows that of Nagpal et al. (2019), in which it is concluded that training with a specific group of data will result in better accuracy for that group, and in turn worst accuracy with any other groups that vary from the training source.

7.3 MTCNN

The MTCNN algorithm clearly shows in the threshold graphs, that at every threshold the underrepresented faces perform worst than the represented faces. Looking into the algorithms structure, it is hard to see how a bias like this could exist within the algorithm itself. The MTCNN algorithm uses 3 networks to get bounding box and landmark coordinates. The first two networks propose and refine the bounding box with the final output network outputting the 5 landmark locations. At the start of the algorithm, the image is resized to reduce its complexity and find larger faces. This stage could be where some biased is introduced as by reducing the complexity, the algorithm could have trouble in predicting where the bounding boxes would be. However, as evident in the bounding box histogram, both groups perform very well when it comes to bounding box overlap. The algorithm doesn't use handcrafted feature descriptors like the HOG algorithm therefore, it is difficult to say that the bias exists within how the algorithm functions. Instead it is more likely the bias exists within how the algorithm was trained.

Data bias isn't a new concept within the computer vision and machine learning fields. It is when there exists a weighting or bias within the data that a model is trained with, which results in the models behaviour itself being weighted or biased (Tommasi et al. (2017)). In terms of

the MTCNN algorithm, since the network used to predict landmark locations is trained on minimising the euclidean distance between the predicted landmarks and that ground truth landmarks. Then it is safe to assume that the training data plays a large part to why the algorithm performs in a biased way.

Figure 7.2: *Subset of avaliable attribites of CelebA dataset from: Large-scale CelebFaces Attributes (CelebA) Dataset (Liu et al. (2015)).*

The algorithms networks are trained for face classification using the WIDER FACE dataset (Yang et al. (2016)), and landmark detection using the CelebA (Liu et al. (2015)) dataset. The landmarks are where the difference in accuracy between the groups is present. Each landmark performed worst for underrepresented faces, with the mouth landmarks showing the largest difference. Therefore, it is important to understand the composition of the CelebA dataset, as it was used for landmark training. This is a dataset consisting of 202,599 faces of 10,177 unique celebrities. Although the dataset is annotated well in terms features in the image, the annotations don't specify attributes related to race. Instead they relate to attributes like expression or face structure, a subset of these attributes are seen in figure 7.2. This means that CelebA's annotations don't reflect human diversity well, as they fail to consider races, resulting in a biased set of annotations. A stronger reason for the algorithm performing with bias is due to the inherent sampling bias introduced by using celebrities faces in the dataset. Taking an example of Hollywood, it is generally understood that there is a lack of minority representation (Santhanam (2015)). This is something that will therefore be reflected in a dataset such as the CelebA dataset, which contains famous celebrities. It means the algorithm is better trained to handle represented faces than it is underrepresented faces. This is because it wasn't trained using underrepresented faces, which results in failure to accurately identify features for them, and causes the algorithm to perform with a bias.

7.4 RetinaFace

This algorithm shows very consistent results in terms of the performance of both groups. Looking at the experiments it is clear that a bias doesn't exist within the RetinaFace algorithm, as the number of errors for represented and underrepresented faces was very similar at every threshold. The average difference at all the landmarks was also very low leaning only towards underrepresented faces performing worst at the right mouth landmark. Although, even this difference was small compared to differences shown in the other algorithms. It is important to say that the average error value for the underrepresented faces in the RetinaFace algorithm was the worst between all the algorithms, but it showed the least bias as the average error value for represented faces was very similar. This indicated that the algorithm was inaccurate for both groups. The algorithm was trained using the Wider Face (Yang et al. (2016)) dataset, similar to that of the MTCNN

algorithm. The landmark annotations were manually annotated onto the WIDER FACE dataset, this allowed for both the face classification and landmark localisation to be trained using the same dataset. The unbiased performance could come from the fair dataset that was used to train the model. The collection method for the WIDER FACE dataset looks at different event categories like students or voters, and collates 1000 to 3000 images from the internet pertaining to these categories. This method of collection results in less of a sampling bias because the images aren't directly related to celebrities, in which there would be an inherent bias present. The spread of categories also helps include many different backgrounds and identities. However, this strong lack of bias could be a combination of both the fairer dataset and a component of the algorithm, the feature pyramid network. The FPN works by generating feature maps from the original image which contain rich semantic data, but are of a lower resolution. These maps can be used to reconstruct higher resolution layers with rich semantic information. However, due to the upsampling and downsampling of the feature maps, alot of the precise information can be lost. This means that features in images that could result in biased performance, have their complexity reduced, and the algorithm can make predictions on simpler feature maps it is better trained for. This could be a valid reason for the lack of bias, but since the algorithm is a deep learning algorithm, it is more likely that the dataset used to create the filter is responsible for the lack of bias.

7.5 Altering Images to Assess Dataset Bias

Figure 7.3: *Original underrepresented face.*

Figure 7.4: *Lower contrast underrepresented face.*

From the 3 algorithms it is clear to see that the MTCNN algorithm showed the most bias with its underrepresented faces performing worst compared to the represented faces. This bias seems to exist through the dataset the algorithm is trained on. For the next set of experiments, the focus was to see what factor in the underrepresented faces caused them to perform worst than the represented faces, and if the faces could be augmented such that the performance of the MTCNN algorithm could be improved. This would assist in assessing the dataset bias present.

To do this a subset of the dataset was used which had errors larger than a magnitude of 7 for the left and right mouth landmarks. These images were chosen because the mouth landmarks were the worst performing landmarks for the MTCNN algorithm. From this subset of 11 faces, 2 were represented males, 3 were underrepresented males and 6 were underrepresented females. The factor decided upon was altering the contrast of the images so that the subset of images more closely resembled the contrast of the training data. Through doing this the difference between the brightest and darkest parts of the image were reduced, this resulted in a lighter skin tone on the faces and the shadows became less pronounced. Other factors to alter such as brightness and

sharpness were considered but after some testing it was found that the changing these factors impacted other parts of the image outwith the face. Every image of the subset was taken and its contrast decreased through Microsoft's image editor. This reduction in contrast led to the images being lighter in tone, with less difference between the highlights and shadows of the image. The change is visible from figure 7.3 to figure 7.4.

id (gender/group)	face	feature	original	altered	difference
1_m_m		left_mouth	7.92	2.384	5.536
		right_mouth	8.615	3.002	5.613
4_m_m		left_mouth	6.948	2.356	4.592
		right_mouth	8.075	3.077	4.998
6_m_m		left_mouth	7.48	3.214	4.266
		right_mouth	5.682	4.939	0.743
13_m_r		left_mouth	3.507	5.642	-2.135
		right_mouth	9.871	10.8	-0.929
20_m_r		left_mouth	7.417	6.915	0.502
		right_mouth	4.179	4.274	-0.095
27_f_m		left_mouth	6.436	3.196	3.24
		right_mouth	7.292	6.484	0.808
29_f_m		left_mouth	2.001	1.308	0.693
		right_mouth	11.784	11.306	0.478
30_f_m		left_mouth	8.909	8.147	0.762
		right_mouth	0.941	1.186	-0.245
32_f_m		left_mouth	2.409	3.808	-1.399
		right_mouth	7.529	8.607	1.078
33_f_m		left_mouth	5.515	5.81	-0.295
		right_mouth	7.898	7.217	0.681
34_f_m		left_mouth	7.931	16.856	-8.925
		right_mouth	5.789	14.123	-8.334

Table 7.1: *Error difference results from changing contrast.*

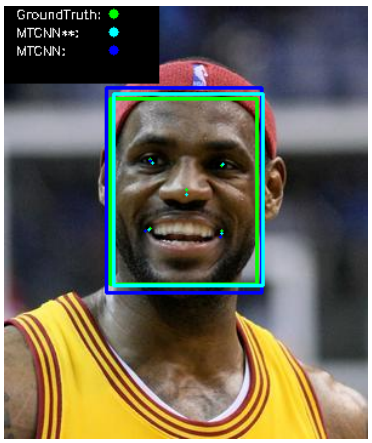

Figure 7.5: *Lebron James altered performance for MTCNN algorithm.*

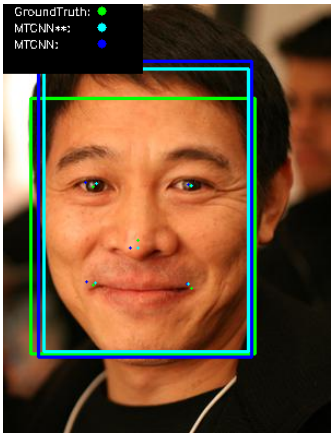

Figure 7.6: *Bruce Lee altered performance for MTCNN algorithm.*

Running this subset of images through the MTCNN algorithm generated interesting results detailed in table 7.1. The 3 male underrepresented faces were improved drastically by this change.

Looking specifically at figure 7.5, we can see that the left eye error of 7.92 was reduced to 2.384 and the right mouth error of 8.615 was reduced to 3.002. The worst performing face of the algorithm was figure 7.6, and by reducing the contrast of the face, the algorithm more accurately placed the landmarks. The left mouth and right mouth landmarks of 7.48 and 5.682 were reduced to 3.214 and 4.939 respectively. However, out of the 6 underrepresented women only 2 showed improvement on both landmarks, 3 showed improvement on a single landmark, and 1 performed worst on both when the images were altered. These results conclusive for the 3 underrepresented male faces were less so for female faces. This could be due to multiple factors outwith the experiment, such as facial expression and ground truth accuracy.

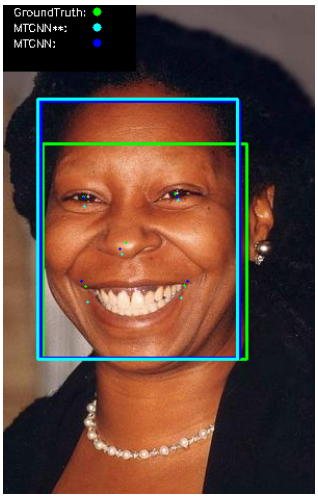

Figure 7.7: *Whoopi Goldberg altered performance for MTCNN algorithm.*

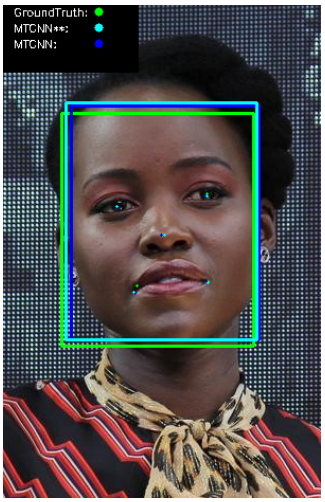

Figure 7.8: *Lupita Nyong'o altered performance for MTCNN algorithm.*

For example figure 7.7, showed a large smile which could effect the accuracy. This is because the algorithm could have a lack of training with expressions at this contrast, where the difference between the teeth and the lips is less pronounced. The ground truth accuracy also affected the results of the experiment. With certain faces like figure 7.8, the ground truth was slightly inaccurate. This means that the accuracy calculated could've shown improvement for both landmarks, but since the ground truth wasn't accurate, this improvement wasn't reflected properly. Even with these results for females, it is still evident that changing the contrast did overall have a positive influence in the algorithms performance. This further emphasises that a bias does exist within the data the algorithm was trained on, as by augmenting a subset of the images to resemble closer to the represented faces. The algorithm was able to provide a better performance for the majority of them.

7.6 Components of Bias

The results do indicate a bias existing within the MTCNN, and to a lesser degree that HOG and Viola-Jones algorithm. However, the algorithms have different reasons for why the bias could exist and by looking at these it is clear to distinguish two main components that can introduce bias. This is the bias introduced by the design of the algorithm, and the bias introduced through the dataset the algorithm is trained on. The design of the older algorithms like HOG and Viola-Jones are part of the former category because of their manually constructed filter and feature descriptors. The feature descriptors used for these algorithms could be encoded to use information from the image which varies between represented and underrepresented faces,

like pixel intensity. This would mean the feature extracted for a represented face, might not be extracted for a underrepresented one causing the filter to behave differently between the groups. The MTCNN algorithm introduces bias through the latter category as the filter isn't directly created with pre-existing domain knowledge which could introduce bias, but instead the filter is created automatically using the learning task and training data. This means that there is more emphasis on the training data being the root cause for bias being introduced in the CNN methods. This is seen by improvement to the results after altering the contrast of the images to resemble the training data faces.

7.7 Dataset Bias

Dataset bias is the prevalent bias present in CNN methods. There are two reasons it exists. Its introduction through sampling bias, and its lack of mitigation through benchmark datasets.

7.7.1 Sampling Bias

Dataset sampling bias occurs when collating images for a dataset. It is the result of collecting images in a way such that the images don't have an equal probability to be selected. The most common form of sampling bias present in datasets is related to sampling celebrities. This is exhibited in many datasets such as the CelebA (Liu et al. (2015)), VGGFace2 (Cao et al. (2018)) and PubFig (Kumar et al. (2009)) datasets. Specifically looking at the popular evaluation dataset, VGGFace2. It consists of 3.31 million images. The images were collated from Google image search and contain varying pose, age and most importantly ethnicity. From the 3.31 million images, there are only 9131 distinct faces. These distinct faces were chosen from an initial list of 500,000 public figures from the the Freebase knowledge graph (Bollacker et al. (2008)). The list was reduced in size by removing those names which didn't have enough credible photos along with them. This meant that the list then consisted of only those public figures who garnered the most fame. This was weighted towards represented faces more than underrepresented faces as looking at popular figures in terms of actors in Hollywood movies, there are more represented actors than their are underrepresented actors by a massive margin (Santhanam (2015)). Therefore, it is reasonable to assume that even though this dataset looks to include more ethnicities, it still has proportionally more represented faces which results in biased training. However, the sampling bias present in datasets doesn't only come from celebrities, as the bias can be introduced indirectly when the dataset only looks at a specific category of person. The PPB dataset from Buolamwini and Gebru (2018), looks to reduce bias for race and gender by using images of parliamentary members around the world. They succeed in reducing this bias present in the dataset but indirectly introduce another bias. Since all the faces are of parliamentary members, there is a large age bias present because the dataset doesn't contain any young faces. This shows the difficulties in using any specific category to create a dataset, as sampling bias can easily and indirectly be introduced.

The Diversity in Faces paper (Merler et al. (2019)), analysed a number of widely available and popular face datasets. It went through each dataset and looked at the number of males, females and type of skin tones present in them. The findings were conclusive and showed that for every dataset considered, there was a higher percentage of lighter skin tone faces than darker skin tones (Table 7.2). The Pubfig dataset described above does result in a very uneven skew of skin tone as only 18% of the dataset contains underrepresented faces with a darker skin tone. This is due to the sampling bias. The same is seen with the CelebA dataset, which the MTCNN algorithm uses to train its landmark localisation with. It is one of the least diversified datasets with only 14.2% of images being of darker toned underrepresented individuals. It is important to take these results with a level of uncertainty because the Diversity in Faces paper has not yet been published, and the methodology behind how the results were gathered isn't entirely explained. However, these

results do indicate a massive skew in the skin tone diversity of these datasets, which provides reasonable evidence to a sampling bias being present.

	Gender		Skin Color/Type	
Dataset	**Female**	**Male**	**Darker**	**Lighter**
LFW (Sagonas et al. (2013))	22.5%	77.4%	18.8%	81.2%
IJB-C (Maze et al. (2018))	37.4%	62.7%	18.0%	82.0%
Pubfig (Kumar et al. (2009))	50.8%	49.2%	18.0%	82.0%
CelebA (Liu et al. (2015))	58.1%	42.0%	14.2%	85.8%
UTKface (Zhang et al. (2017))	47.8%	52.2%	35.6%	64.4%
AgeDB (Moschoglou et al. (2017))	40.6%	59.5%	5.4%	94.6%
PPB (Buolamwini and Gebru (2018))	44.6%	55.4%	46.4%	53.6%
IMDB-Face (Wang et al. (2018))	45.0%	55.0%	12.0%	88.0%

Table 7.2: *Distribution of gender and skin color/type for seven prominent face image datasets From: Diversity in Faces (Merler et al. (2019)).*

7.7.2 Benchmark Datasets

The Labelled Faces in the Wild dataset (Sagonas et al. (2013)) is a popular dataset amongst algorithms and frameworks as it is a benchmark dataset used to evaluate accuracy and robustness. A reason for its popularity is that it contains images of faces in unconstrained environments, with differing pose, expression and shadows. An assumption would be that a popular dataset used to evaluate the robustness of face detection performance "in the wild" would be representative of the real world, but in reality this dataset has many issues. These being that the dataset lacks representation of minorities as well as women, kids and the elderly. Benchmark datasets like this are a large reason to why bias exists within face detection algorithms. They aren't being used to train the algorithms directly but they are being used to evaluate them. This means that potentially biased algorithms are being evaluated against a dataset that doesn't consider bias. This results in high accuracy scores for these algorithms and indicates to others that they are accurate. This is occlusive to the real underlying issues and indirectly increases the bias present. Since there is no evaluation of the bias, it is not being mitigated and algorithms continue to use the same biased datasets resulting in biased performance.

7.8 Research Question

To answer if there exists a bias within the face detection field is simple as both the results and research indicates that a bias exists in the scene towards both race and gender. This is evident from the difference in accuracy between the represented and underrepresented faces. To answer why there exists a bias is a more complicated question. There doesn't exist a bias within all the algorithms, therefore it isn't as straightforward as finding one factor across every algorithm that could introduce the bias. Instead each algorithm has to be specifically analysed to understand how the bias exists. Through doing so it is seen that bias can be introduced into the algorithms in two ways. One is through the design of the algorithm and the other is through the dataset which the algorithm is trained on.

The Viola-Jones and HOG algorithms showed bias through both their design and dataset. The algorithms feature descriptors look directly at differences in brightness which could vary between groups, resulting in an algorithmic bias present in the design of the feature descriptors. The dataset bias was introduced through using datasets that lacked underrepresented faces. In the

case of Viola-Jones this was assumed due to its foundational nature, and in the case of HOG. The 68 point iBUG dataset (Sagonas et al. (2013)), only consisted of only 12% underrepresented faces. The RetinaFace and MTCNN algorithms both showed sensitivity to bias through mainly the dataset they were trained on, instead of their design. The RetinaFace algorithm showed no bias due to being trained with the WIDER FACE dataset (Yang et al. (2016)), that contained a wide distribution of different genders and races. This was achieved through the sampling methodology, which looked at different events to ensure a wider range of genders and ethnic backgrounds. It was thought that the feature pyramid network present could reduce bias, but since the algorithm is a deep learning algorithm. It was likely that the dataset used to create the filter was responsible for the lack of bias. The MTCNN algorithm showed major bias due to being trained on the inherently biased CelebA dataset (Liu et al. (2015)). This bias was the result of sampling bias present when collating the images. Altering the worst performing images for the algorithm to match the characteristics of the training dataset, resulted in improved performance which emphasised the presence of dataset bias. The dataset bias present in the majority of the algorithms comes from two main factors. Its introduction through sampling bias and its lack of mitigation through benchmark dataset bias. These two factors create a cycle where algorithms using biased datasets are being evaluated as accurate by biased benchmark datasets. This results in the algorithms and datasets being perceived as accurate and implemented, even though there exists a bias.

It is clear that the image-based methods of MTCNN and RetinaFace introduce bias, or lack of bias through their dataset. Whereas the feature-based methods show more sensitivity to bias through their design. However, this isn't exclusive as analysis of the Viola-Jones and HOG algorithm indicate that dataset bias is still present.

7.9 Limitations of Results

The results and evaluation of the algorithms had limitations associated with them. The limited size of the dataset meant that the results gathered had a higher variance within them. This variance affected the strength of the analysis and reasoning on why a bias existed as it was based on a small sample size. The analysis behind the feature-based algorithms poor performance is also not as strong as it could be. Further testing could've been done to see if altering the intensity of pixels would affect the feature extraction for the HOG and Viola-Jones algorithm. This would provide stronger evidence to the bias being present in the design of the algorithms.

8 Conclusion

8.1 Summary

This paper provided research into facial detection algorithms to investigate if and why there existed a bias towards race and gender. A novel dataset called the 5025 dataset was compiled consisting of an equal number of genders and representations for faces. Each algorithms performance was measured against represented and underrepresented faces within the dataset. The difference between the performance of the algorithms with these groups indicated if a bias was present. The results showed that a slight bias existed within the Viola-Jones and HOG algorithms due to their feature descriptors and training data. The MTCNN algorithm showed the largest bias due to its dataset, and the RetinaFace algorithm had the smallest difference between the groups, indicating that a bias wasn't present. This was because of its training data. The dataset factor of face detection was researched and it was found that the bias existed due to its introduction from sampling bias, and its lack of mitigation through biased benchmark datasets.

8.2 Future Work

Due to the scope of the project along with the complexity of the face detection algorithms, there were many factors that were unable to be isolated and tested. Therefore, this leaves much room for further work, which includes:

- Expand on face detection field by evaluating other algorithms with differing features like the R-CNN family of algorithms (Girshick (2015)) or YOLO framework (Redmon et al. (2016)).
- Evaluate the algorithms on a larger dataset comprising of an evenly distributed number of represented and underrepresented faces.
- Isolate the results by gender to get insight into specifically the gender bias instead of the race and gender bias combined.
- Retrain the MTCNN algorithm using a fairer dataset to evaluate if there is any improvement against reducing the bias present.

These are the main directions that future work could be taken in. All try to improve the understanding behind the existence of bias, and work on bringing more robustness to the results. However, the most important consideration for any proposed future work is to continue a dialogue within the face detection field about the existence of bias, as it is only with understanding why the bias exists that it can be mitigated and fairness can be ensured.

A Appendices

A.1 Bounding Box and Landmark Predictions on Faces

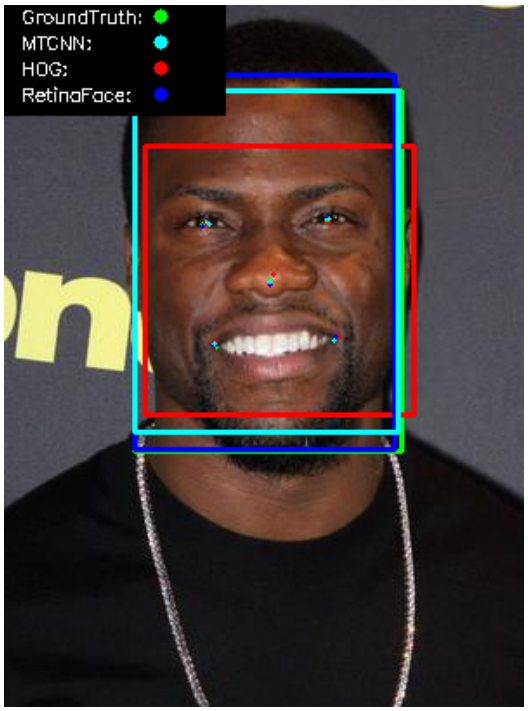

Figure A.1: *0_m_m performance across algorithms.*

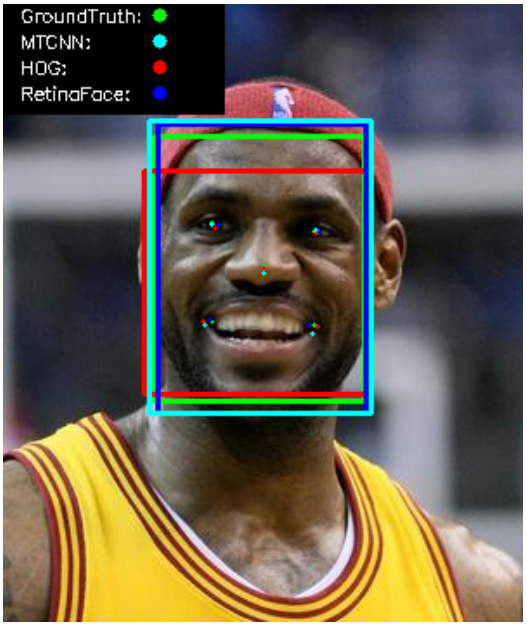

Figure A.2: *1_m_m performance across algorithms.*

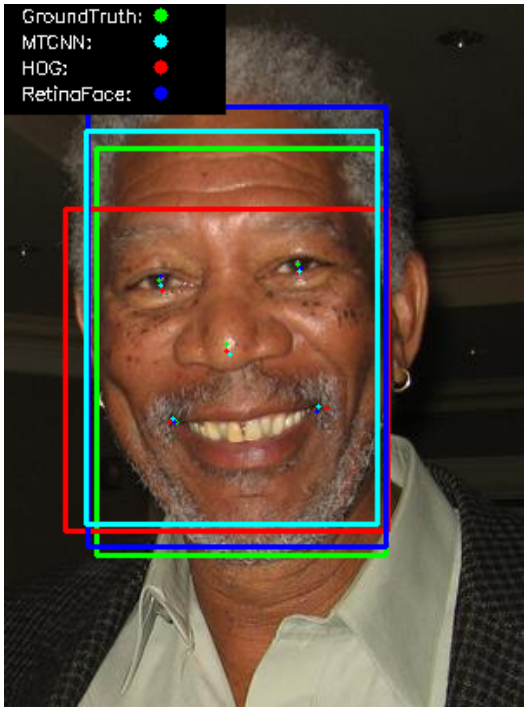

Figure A.3: 2_m_m performance across algorithms.

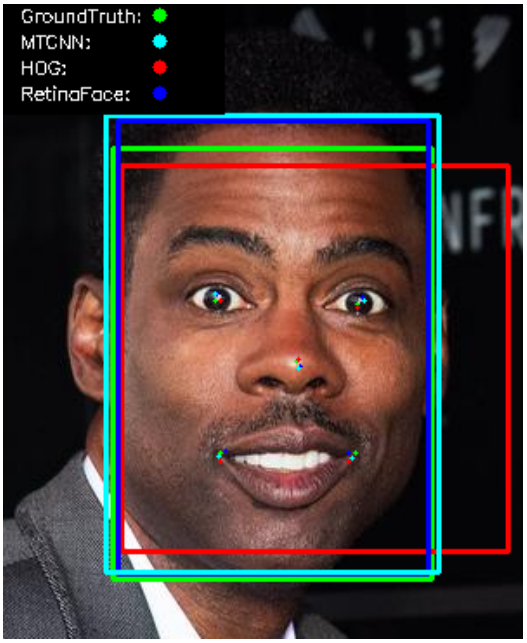

Figure A.4: 3_m_m performance across algorithms.

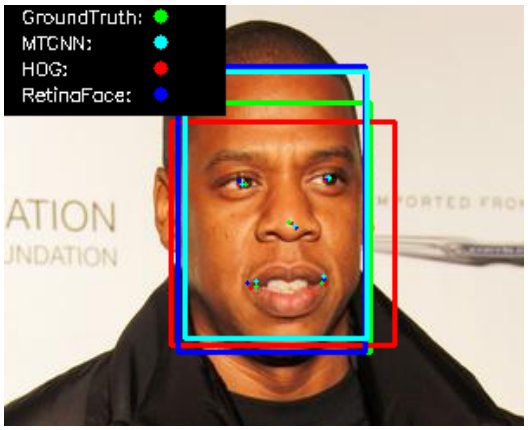

Figure A.5: 4_m_m performance across algorithms.

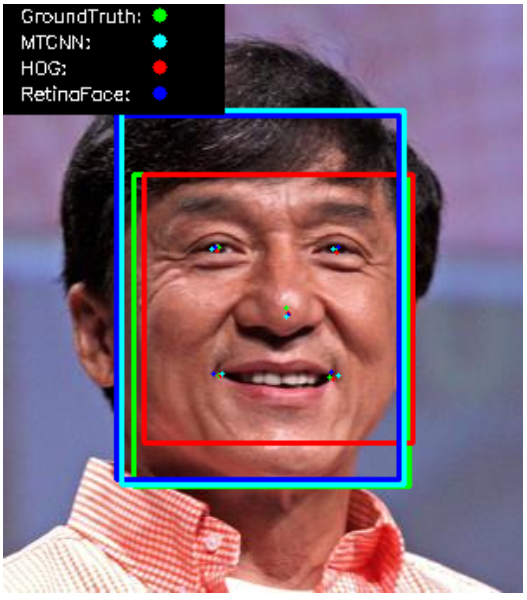

Figure A.6: 5_m_m performance across algorithms.

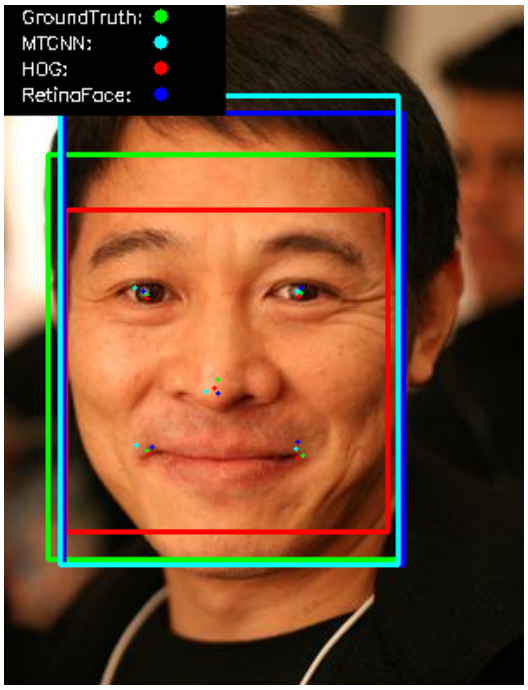

Figure A.7: 6_m_m performance across algorithms.

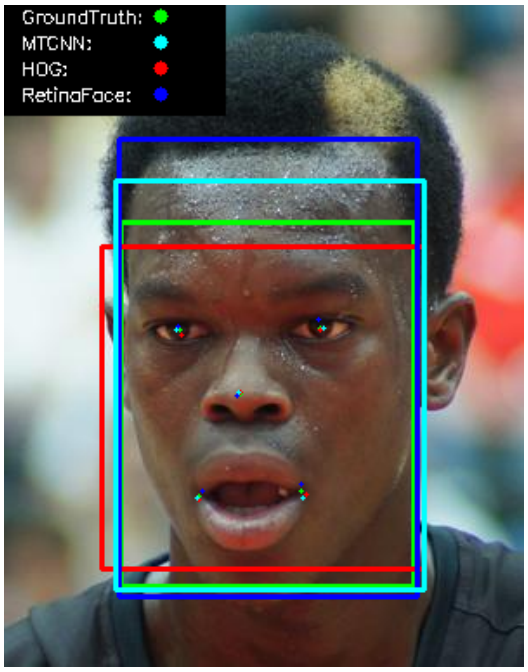

Figure A.8: 7_m_m performance across algorithms.

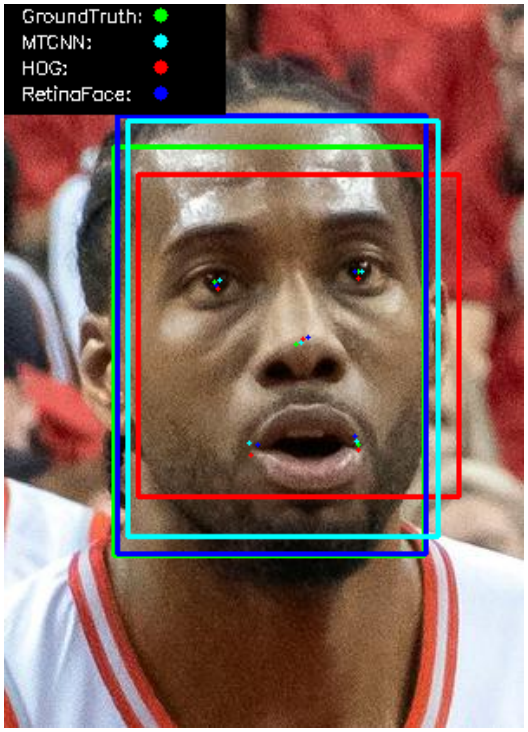

Figure A.9: 8_m_m performance across algorithms.

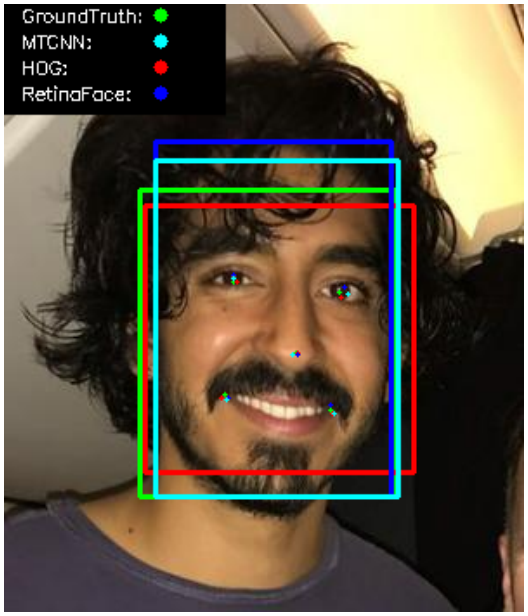

Figure A.10: 9_m_m performance across algorithms.

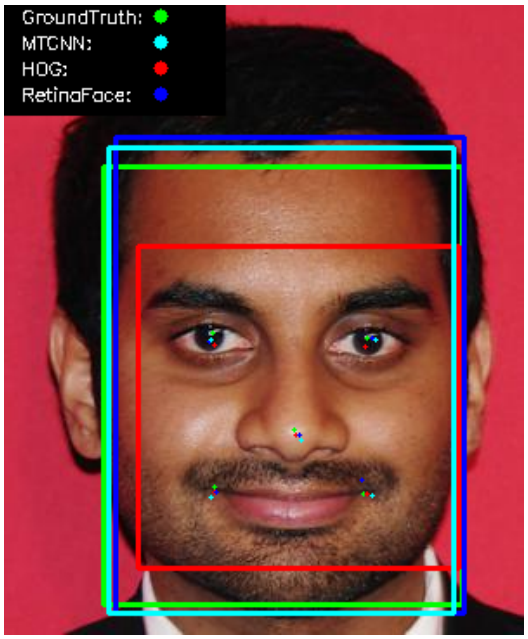

Figure A.11: *10_m_m performance across algorithms.*

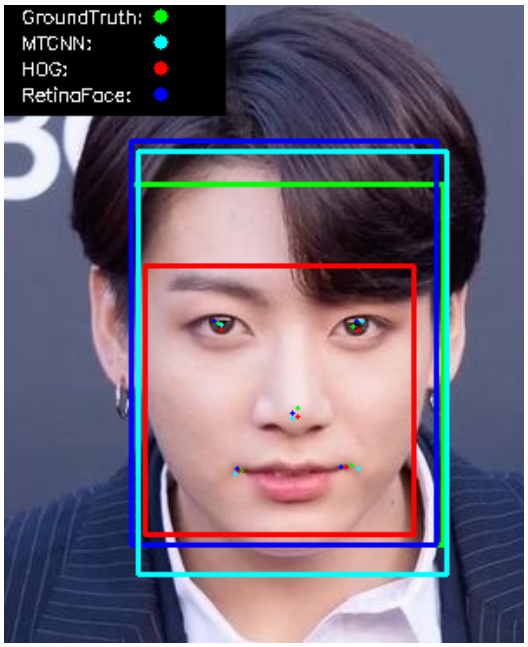

Figure A.12: *11_m_m performance across algorithms.*

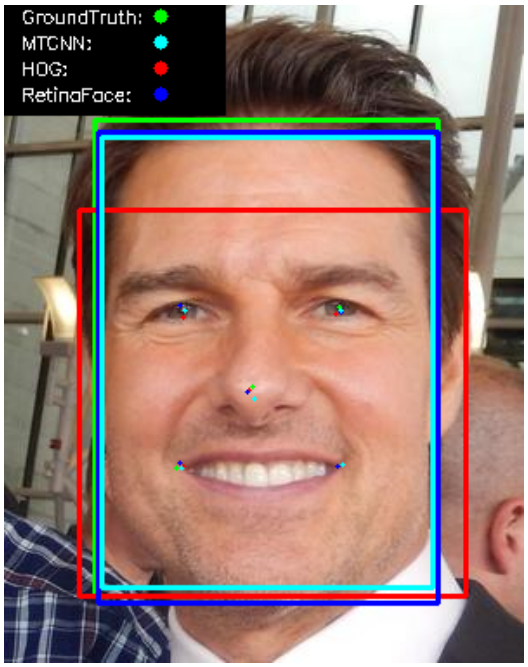

Figure A.13: *12_m_r performance across algorithms.*

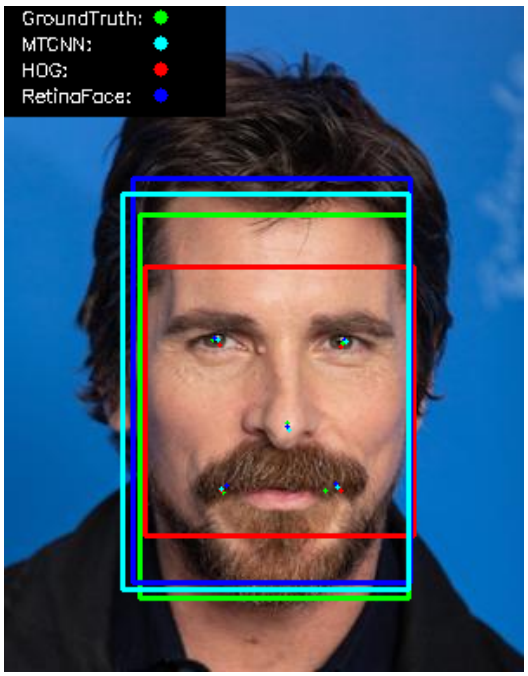

Figure A.14: *13_m_r performance across algorithms.*

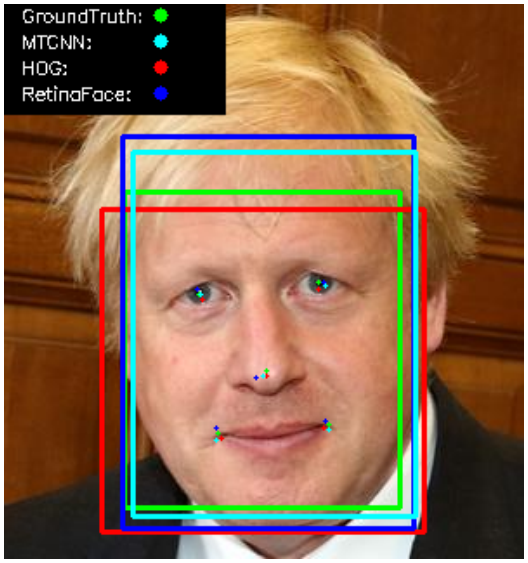

Figure A.15: *14_m_r performance across algorithms.*

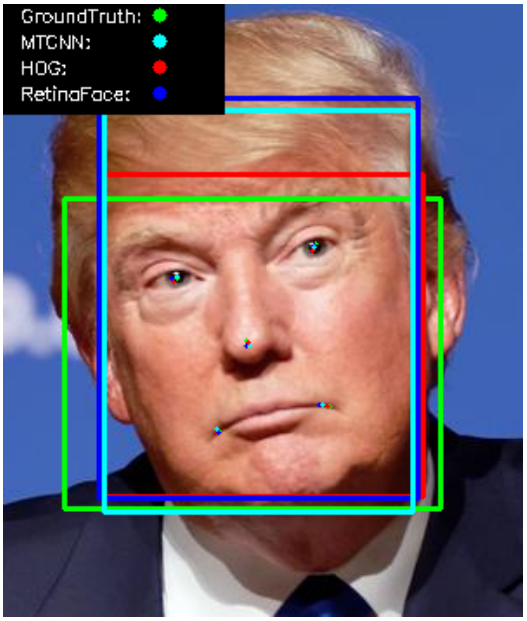

Figure A.16: *15_m_r performance across algorithms.*

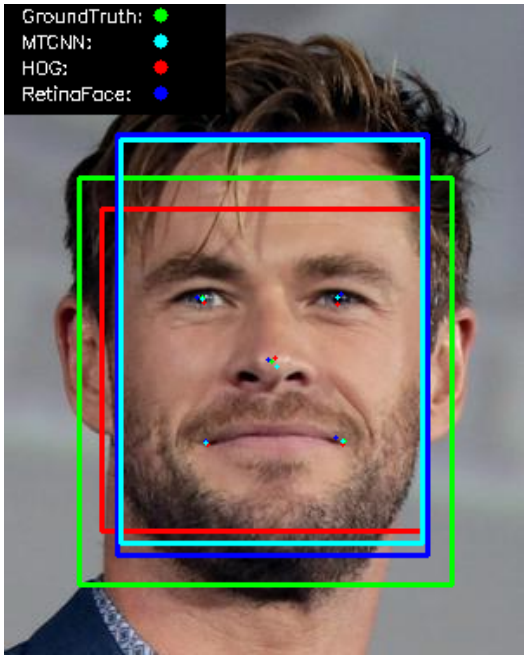

Figure A.17: *16_m_r performance across algorithms.*

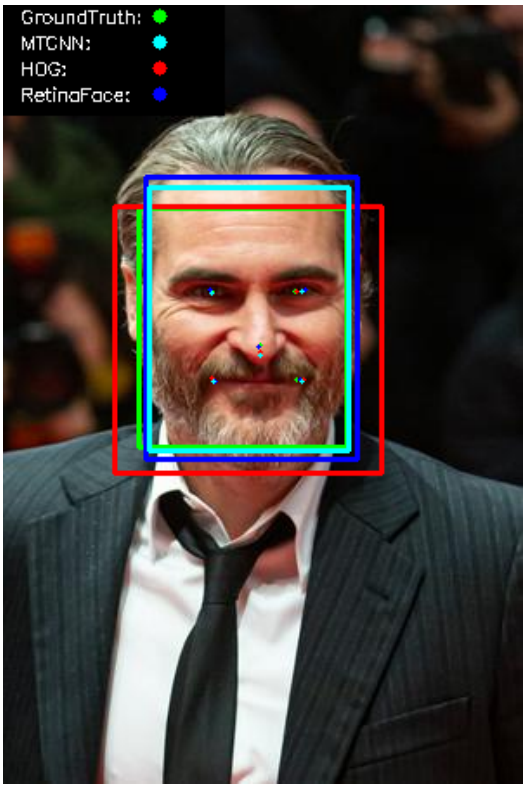

Figure A.18: *17_m_r performance across algorithms.*

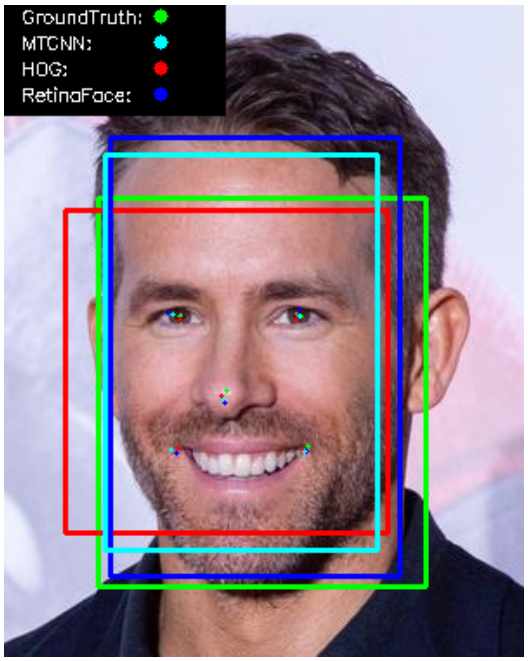

Figure A.19: *18_m_r performance across algorithms.*

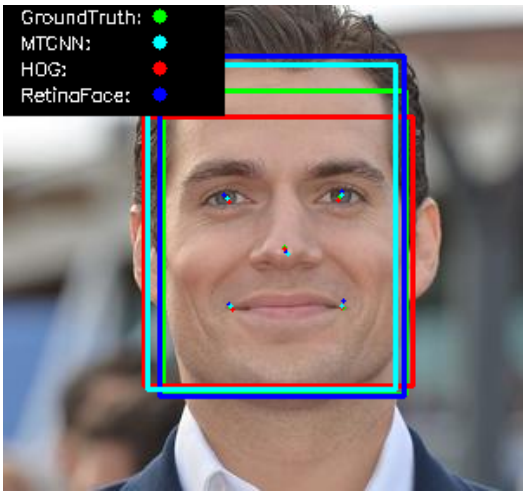

Figure A.20: *19_m_r performance across algorithms.*

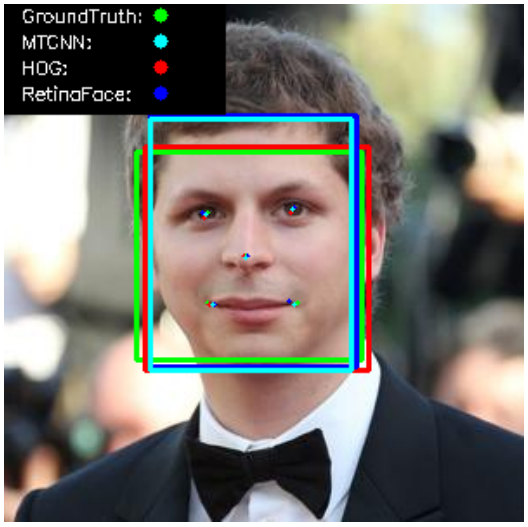

Figure A.21: *20_m_r performance across algorithms.*

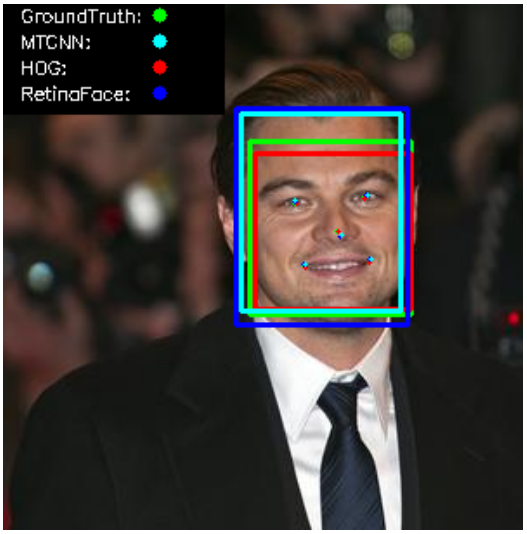

Figure A.22: *21_m_r performance across algorithms.*

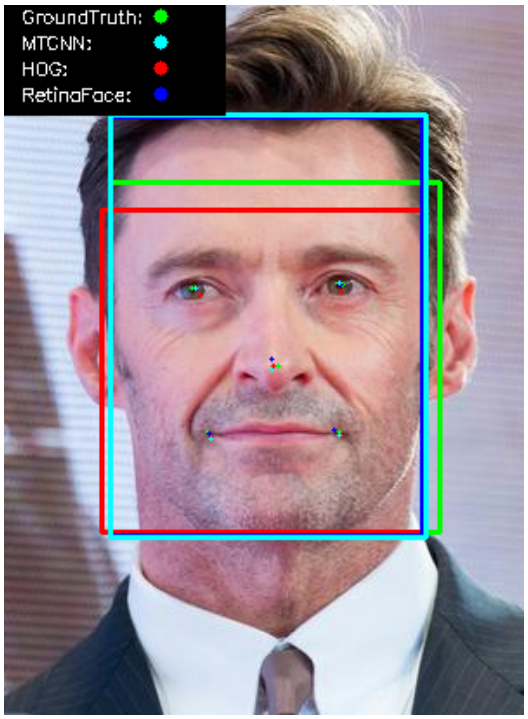

Figure A.23: 22_m_r performance across algorithms.

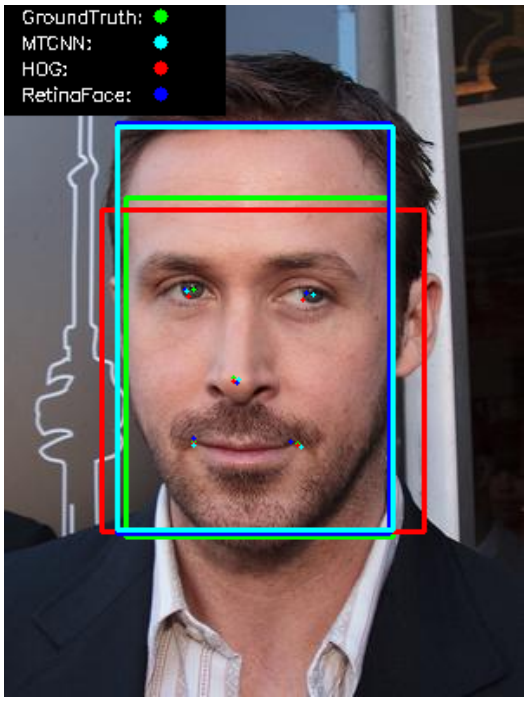

Figure A.24: 23_m_r performance across algorithms.

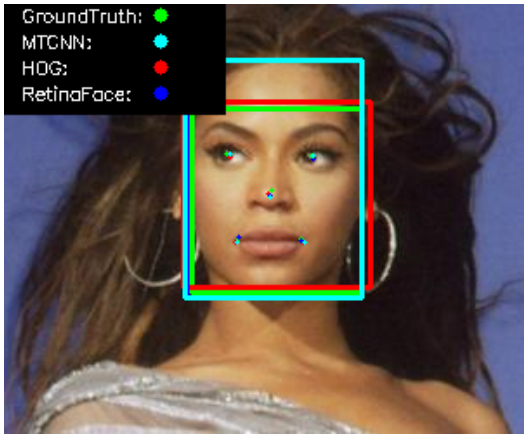

Figure A.25: 24_f_m performance across algorithms.

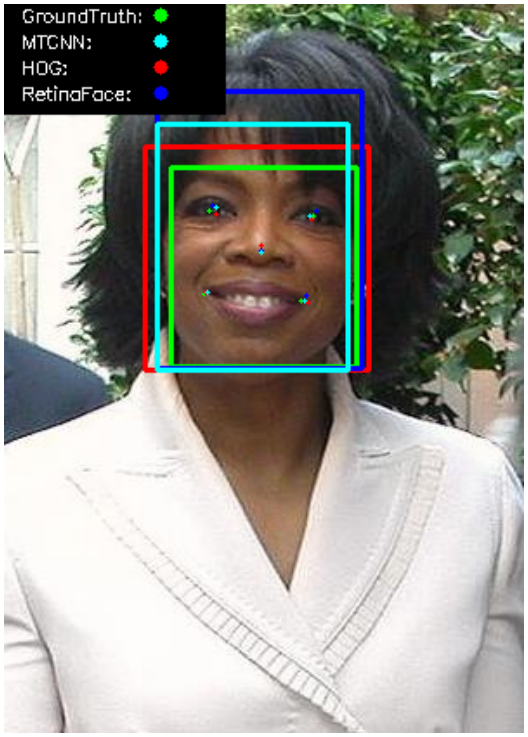

Figure A.26: 25_f_m performance across algorithms.

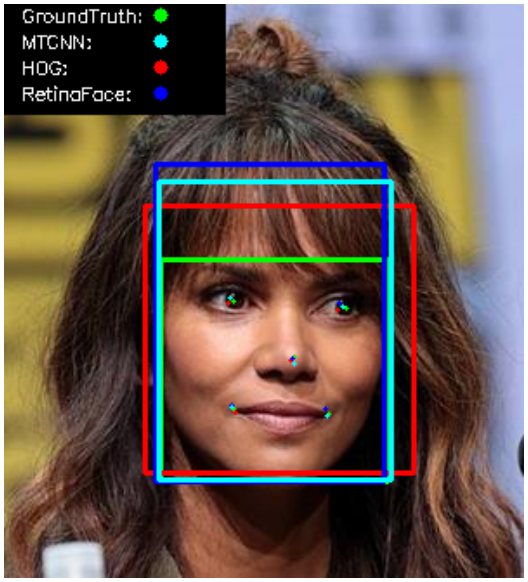

Figure A.27: *26_f_m performance across algorithms.*

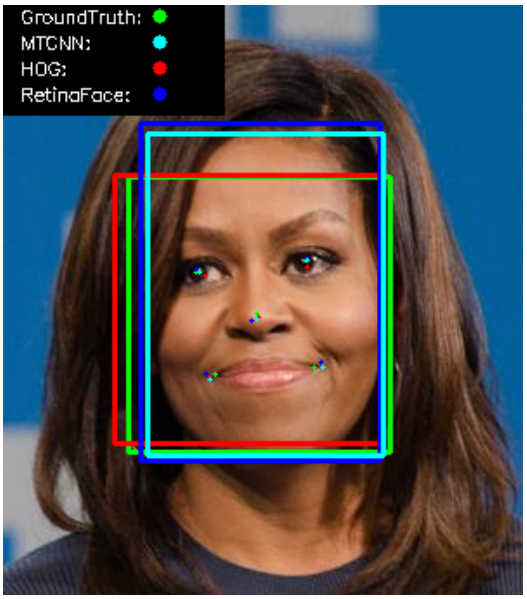

Figure A.28: *27_f_m performance across algorithms.*

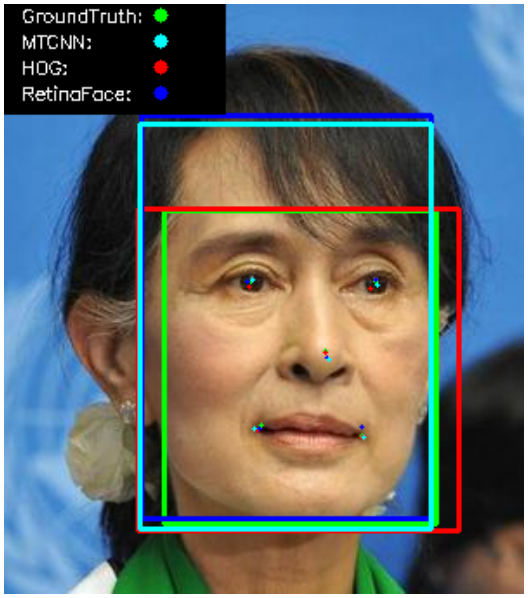

Figure A.29: *28_f_m performance across algorithms.*

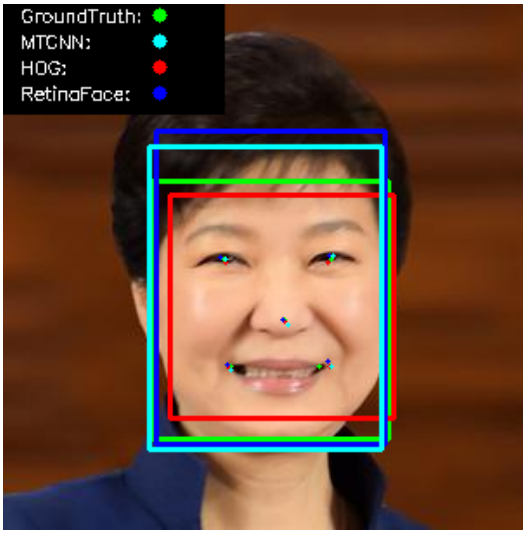

Figure A.30: *29_f_m performance across algorithms.*

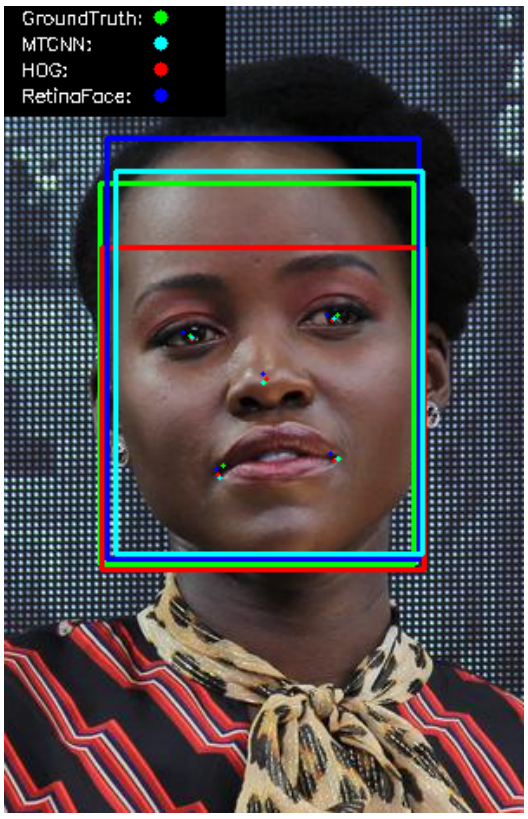

Figure A.31: *30_f_m performance across algorithms.*

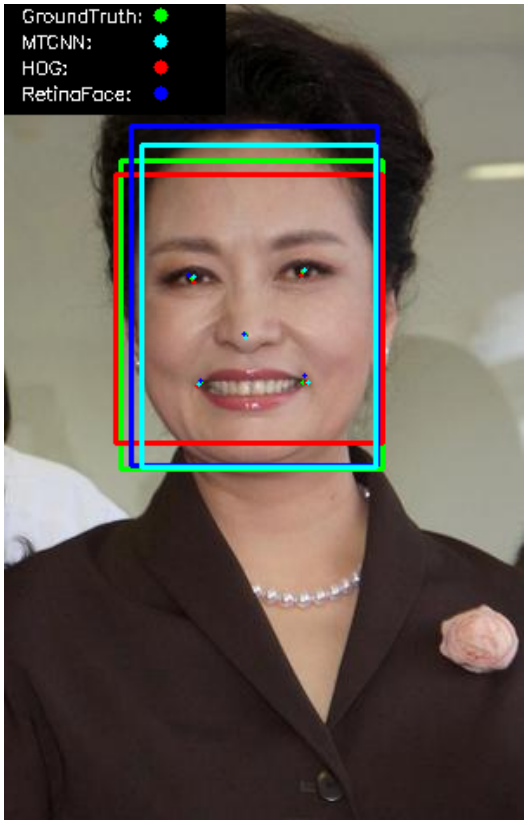

Figure A.32: *31_f_m performance across algorithms.*

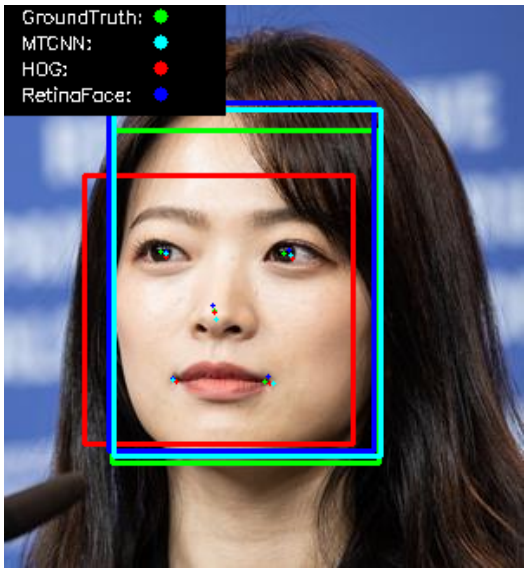

Figure A.33: *32_f_m performance across algorithms.*

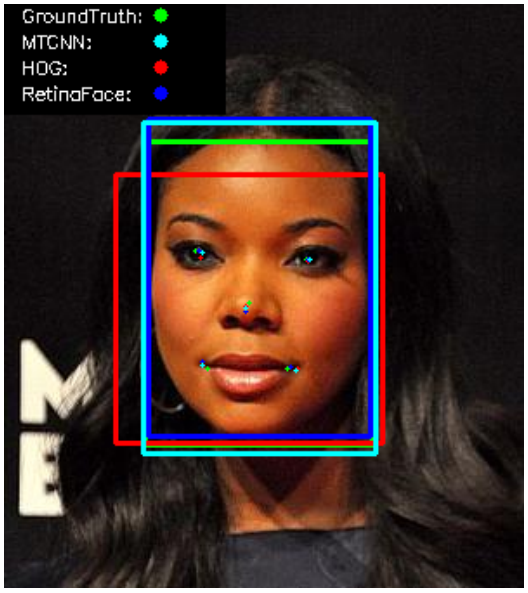

Figure A.34: *33_f_m performance across algorithms.*

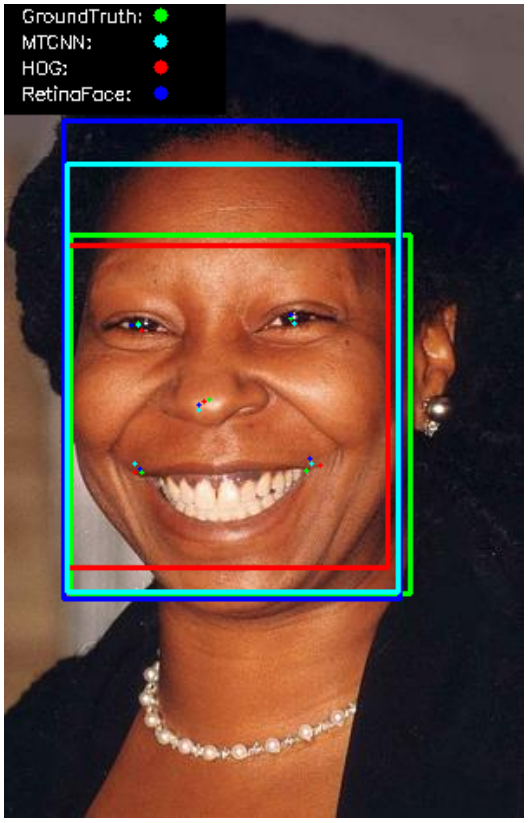

Figure A.35: *34_f_m performance across algorithms.*

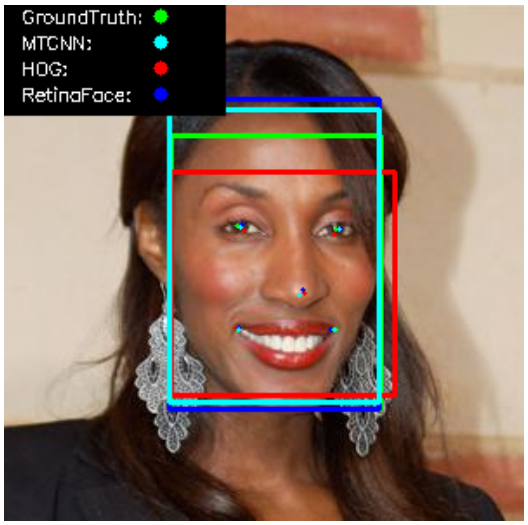

Figure A.36: *35_f_m performance across algorithms.*

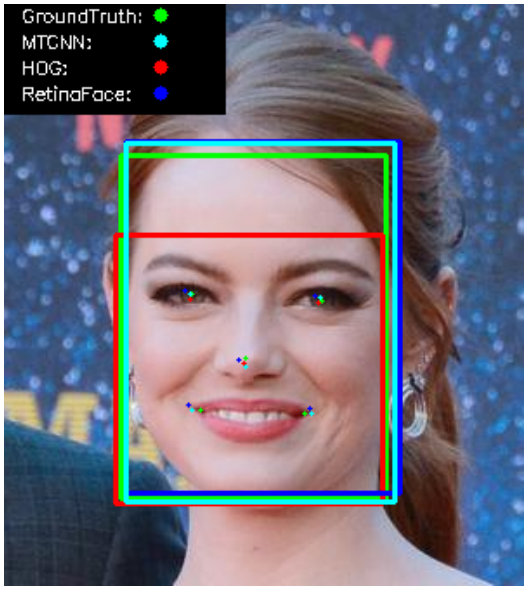

Figure A.37: *36_f_r performance across algorithms.*

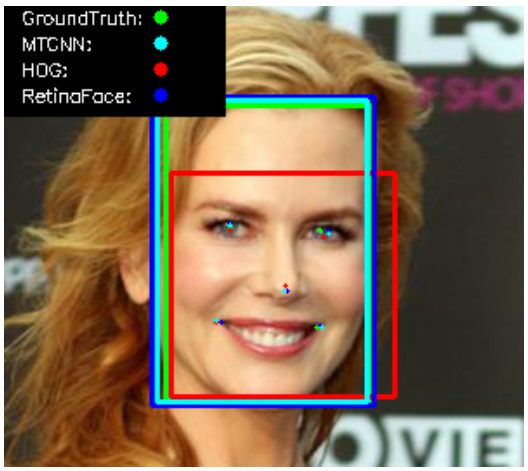

Figure A.38: *37_f_r performance across algorithms.*

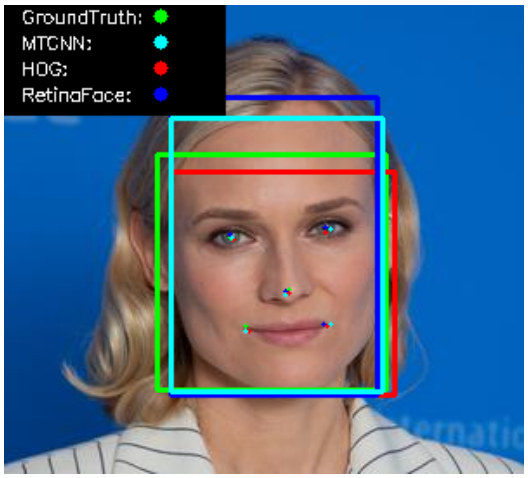

Figure A.39: *38_f_r performance across algorithms.*

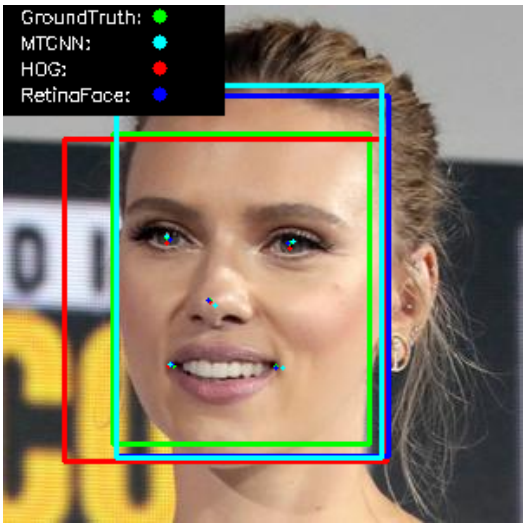

Figure A.40: *39_f_r performance across algorithms.*

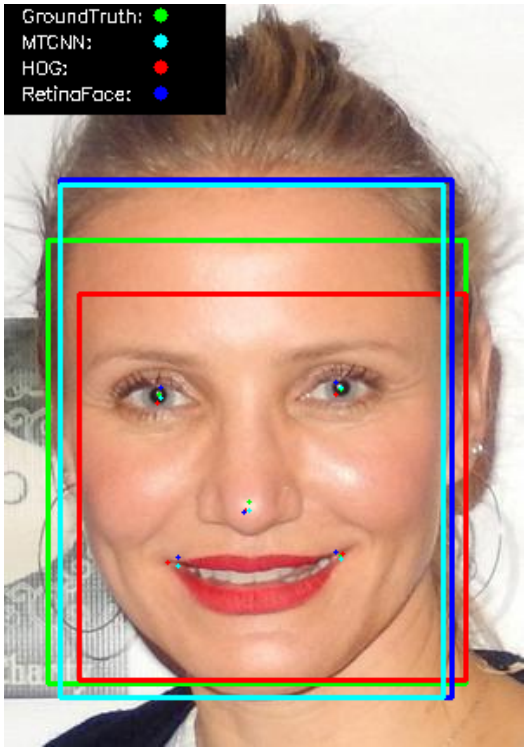

Figure A.41: *40_f_r performance across algorithms.*

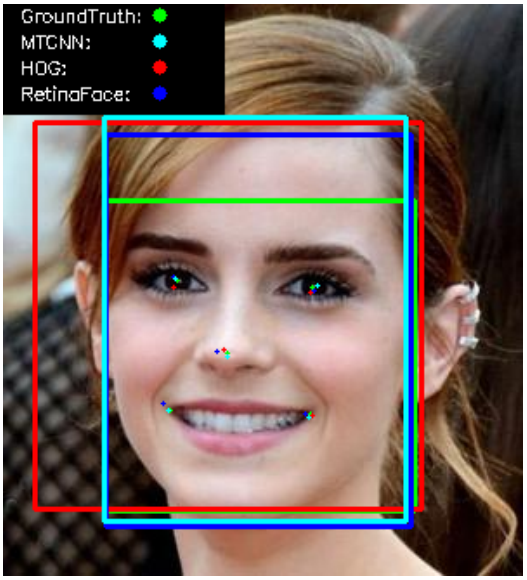

Figure A.42: *41_f_r performance across algorithms.*

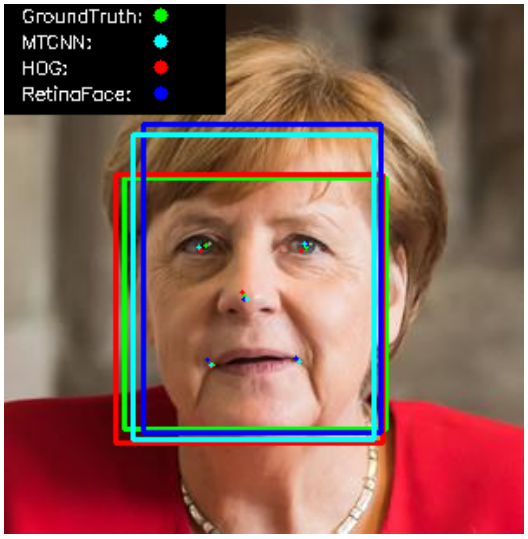

Figure A.43: *42_f_r performance across algorithms.*

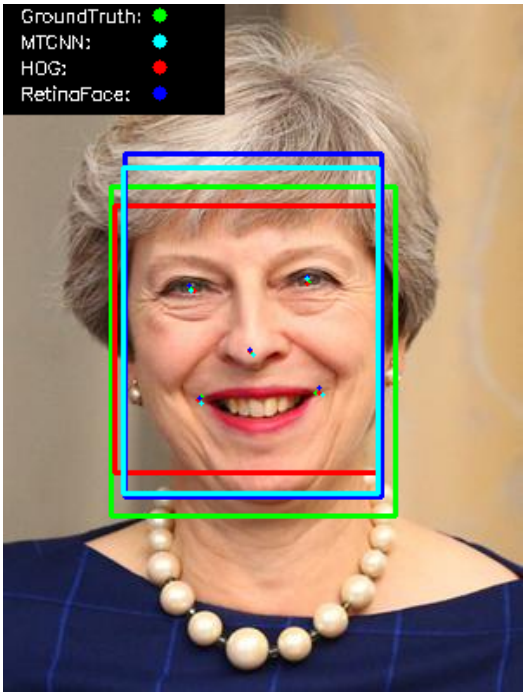

Figure A.44: *43_f_r performance across algorithms.*

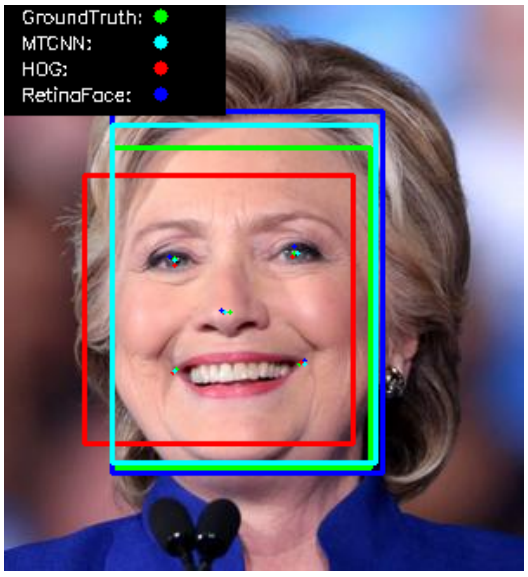

Figure A.45: *44_f_r performance across algorithms.*

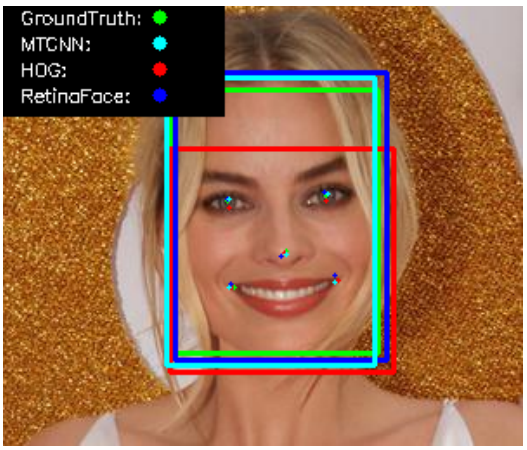

Figure A.46: *45_f_r performance across algorithms.*

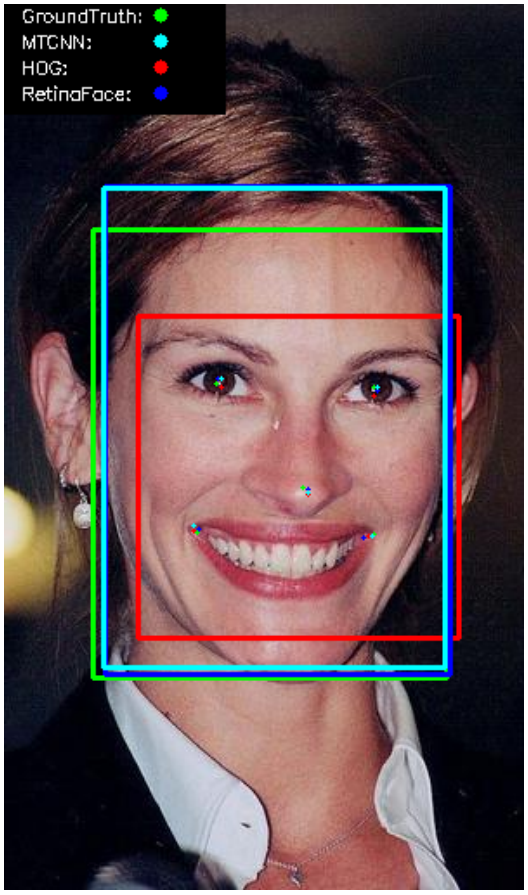

Figure A.47: *46_f_r performance across algorithms.*

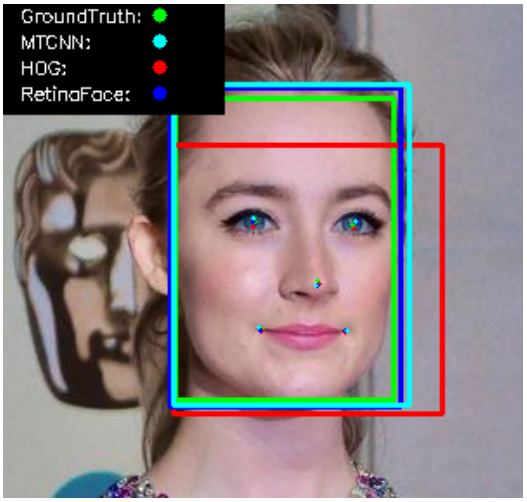

Figure A.48: *47_f_r performance across algorithms.*

A.2 Viola-Jones Bounding Box Predictions on Faces

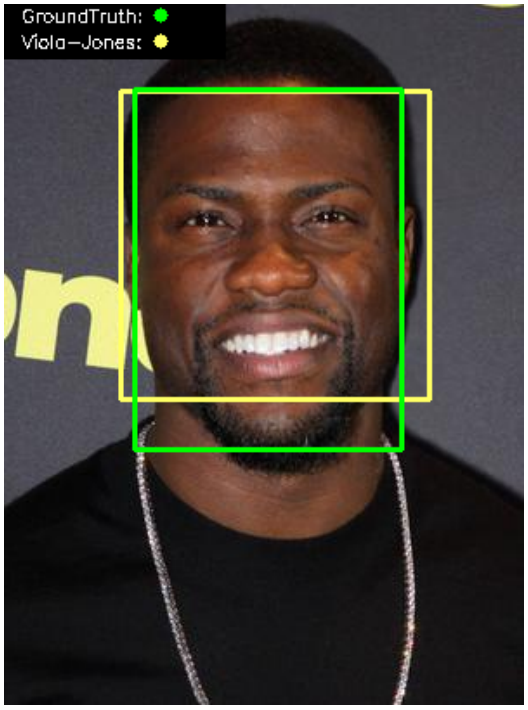

Figure A.49: *0_m_m bounding box performance for Viola-Jones.*

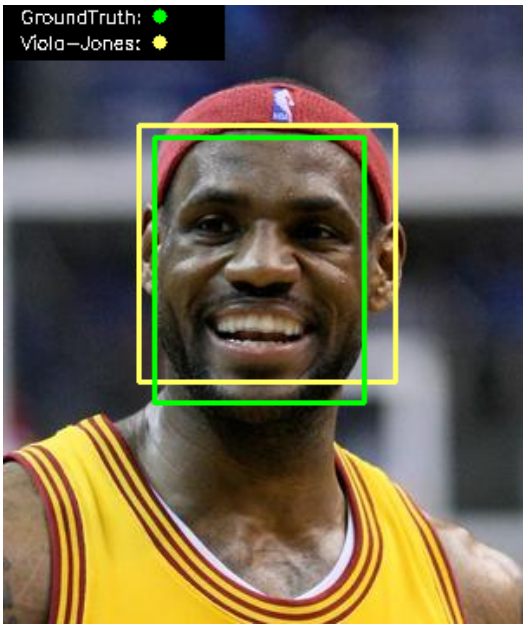

Figure A.50: *1_m_m bounding box performance for Viola-Jones.*

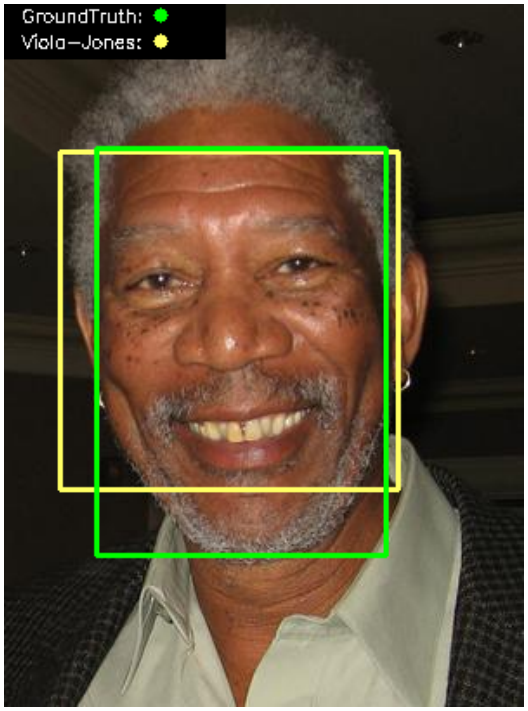

Figure A.51: 2_m_m bounding box performance for Viola-Jones.

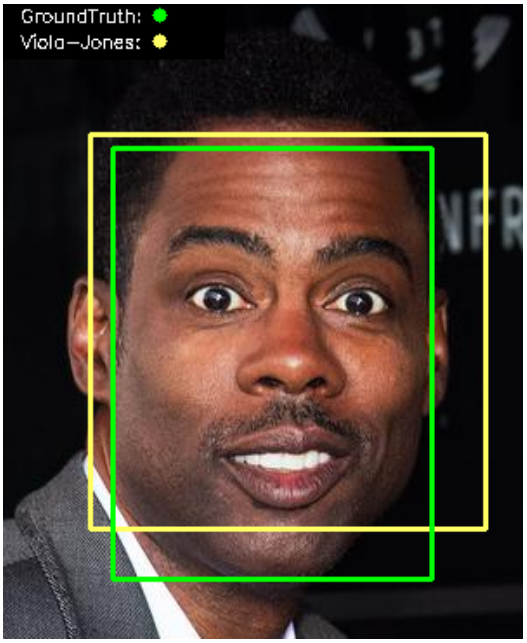

Figure A.52: 3_m_m bounding box performance for Viola-Jones.

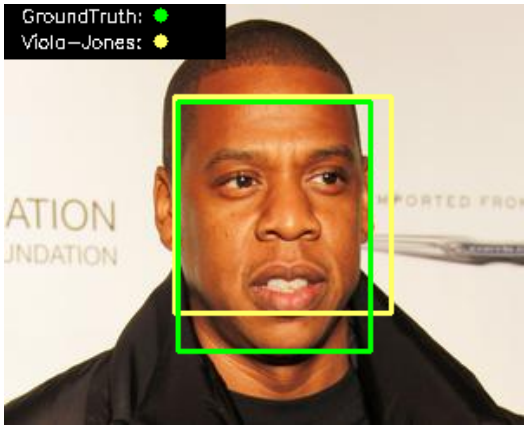

Figure A.53: 4_m_m bounding box performance for Viola-Jones.

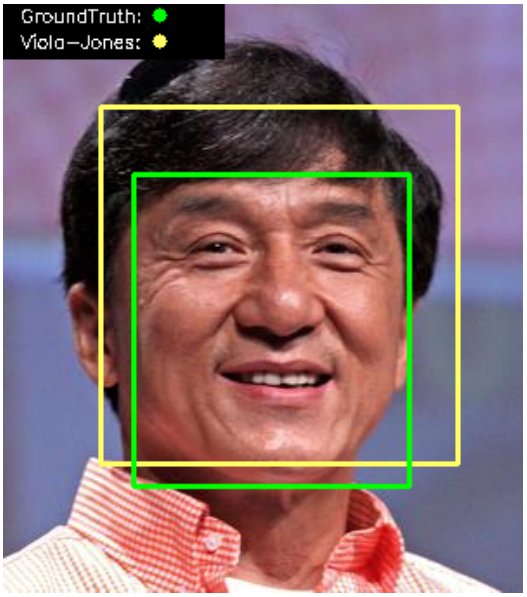

Figure A.54: 5_m_m bounding box performance for Viola-Jones.

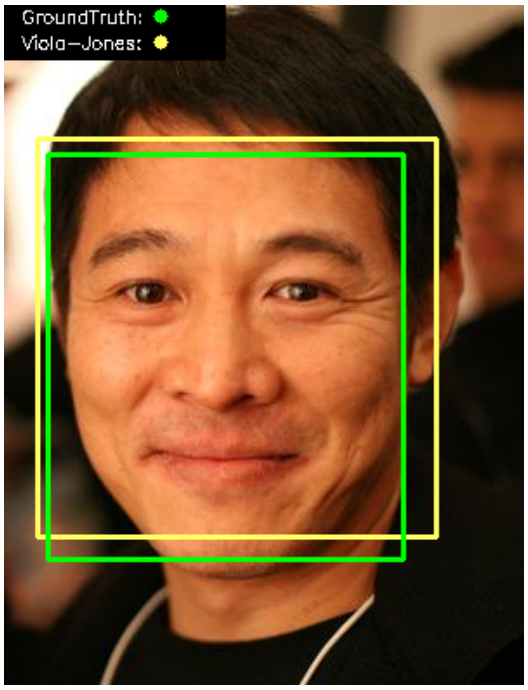

Figure A.55: *6_m_m bounding box performance for Viola-Jones.*

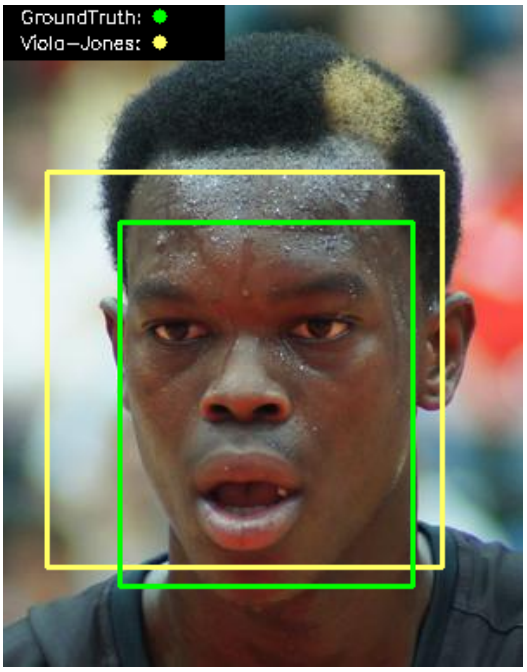

Figure A.56: *7_m_m bounding box performance for Viola-Jones.*

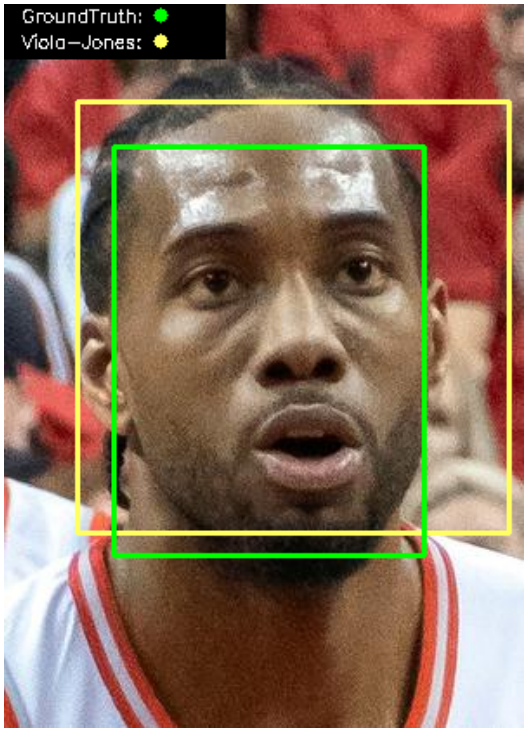

Figure A.57: *8_m_m bounding box performance for Viola-Jones.*

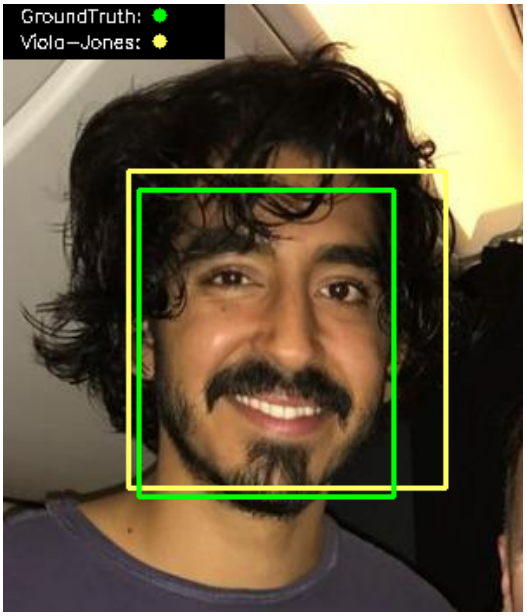

Figure A.58: *9_m_m bounding box performance for Viola-Jones.*

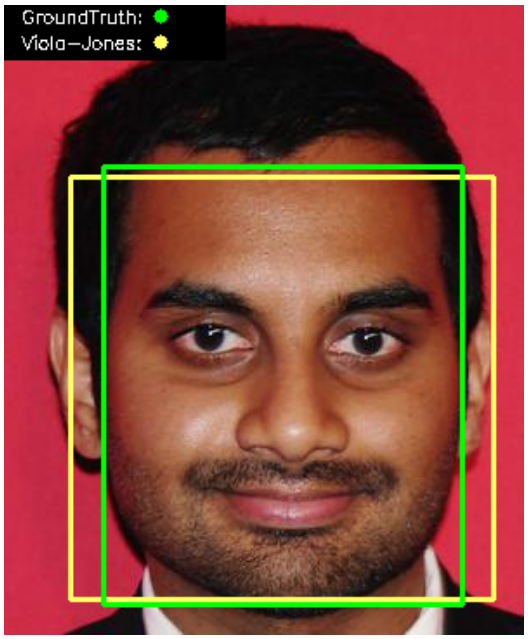

Figure A.59: 10_m_m bounding box performance for Viola-Jones.

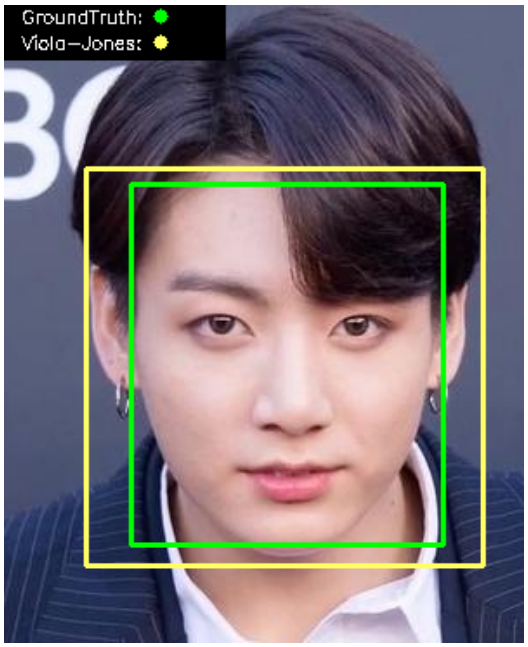

Figure A.60: 11_m_m bounding box performance for Viola-Jones.

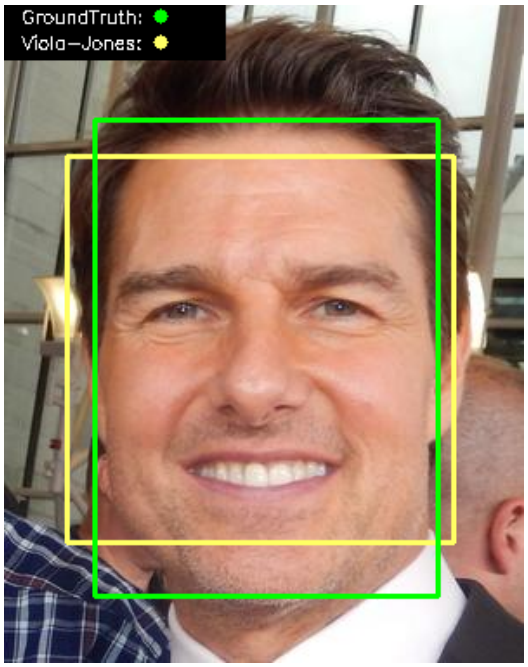

Figure A.61: 12_m_r bounding box performance for Viola-Jones.

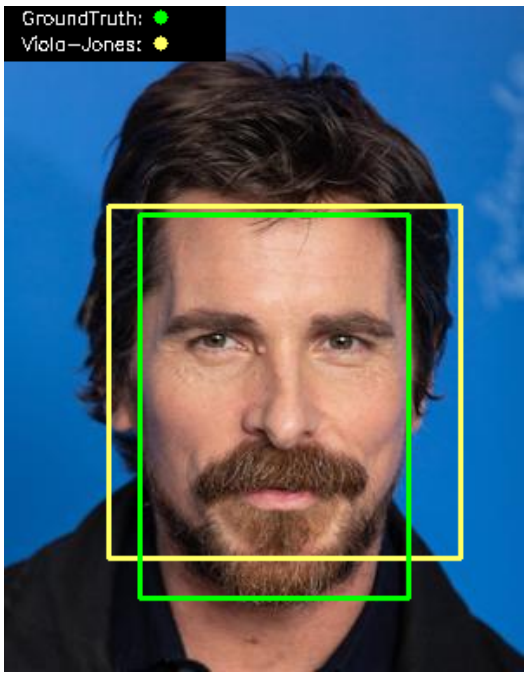

Figure A.62: 13_m_r bounding box performance for Viola-Jones.

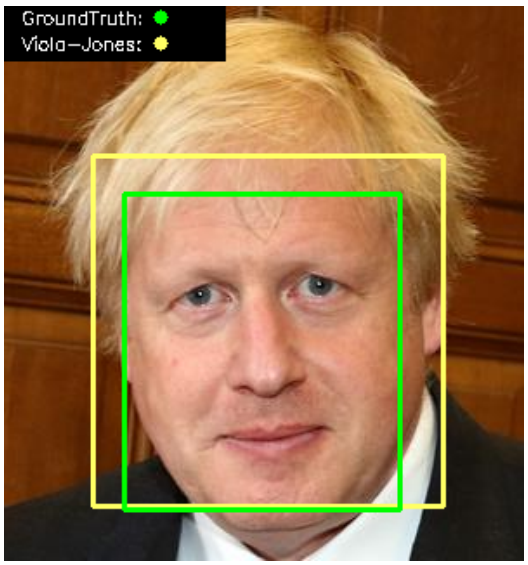

Figure A.63: *14_m_r bounding box performance for Viola-Jones.*

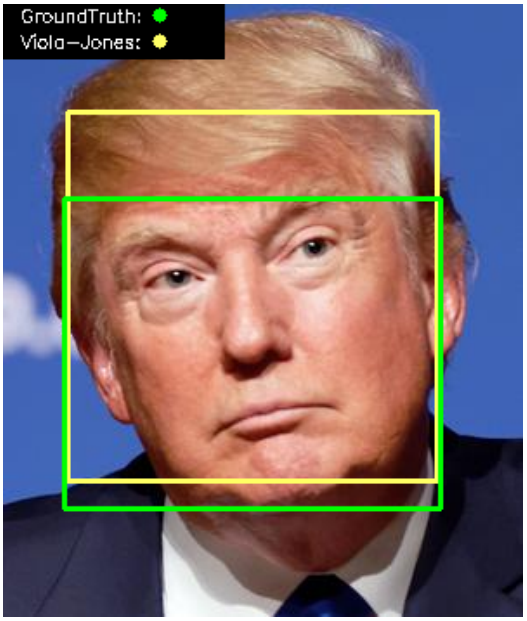

Figure A.64: *15_m_r bounding box performance for Viola-Jones.*

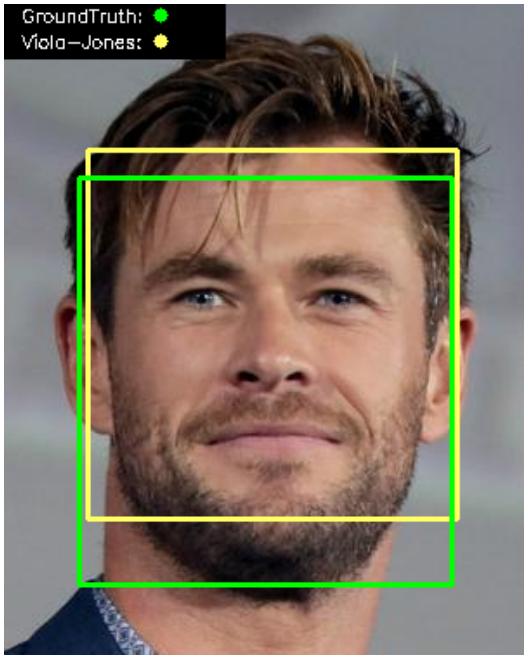

Figure A.65: *16_m_r bounding box performance for Viola-Jones.*

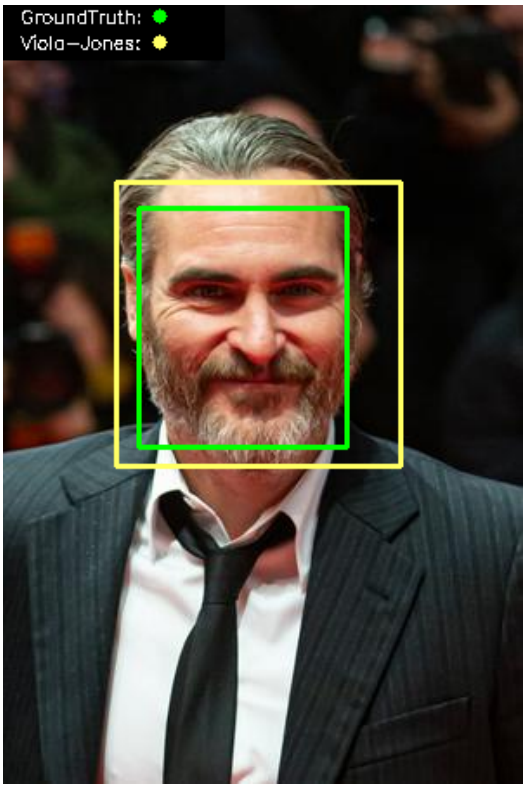

Figure A.66: *17_m_r bounding box performance for Viola-Jones.*

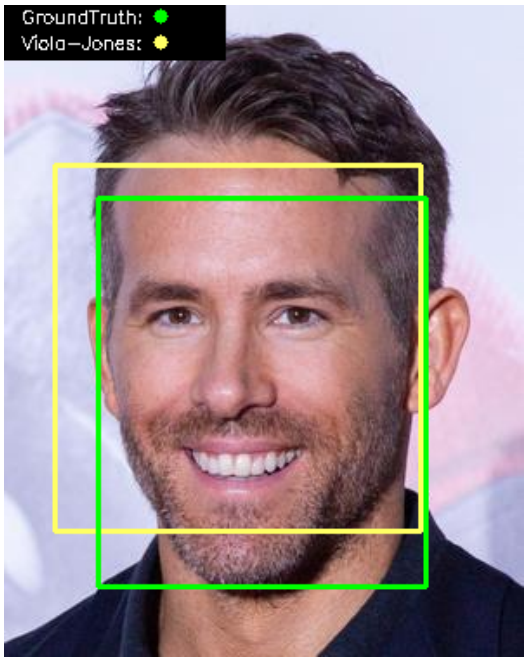

Figure A.67: 18_m_r bounding box performance for Viola-Jones.

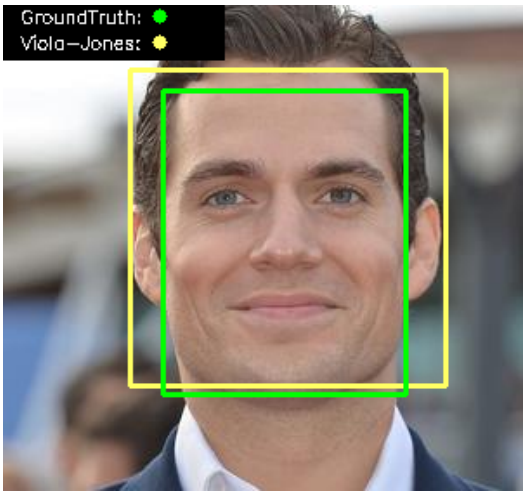

Figure A.68: 19_m_r bounding box performance for Viola-Jones.

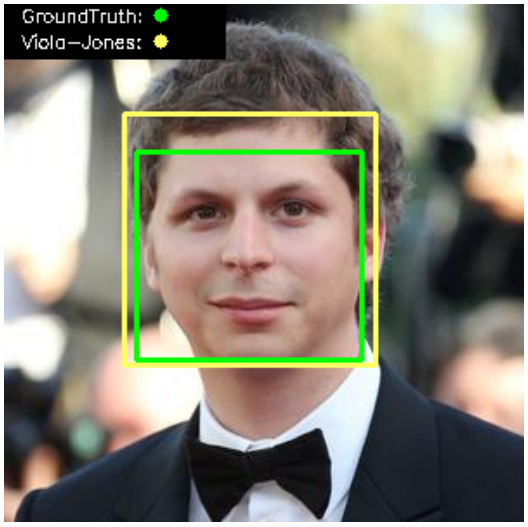

Figure A.69: 20_m_r bounding box performance for Viola-Jones.

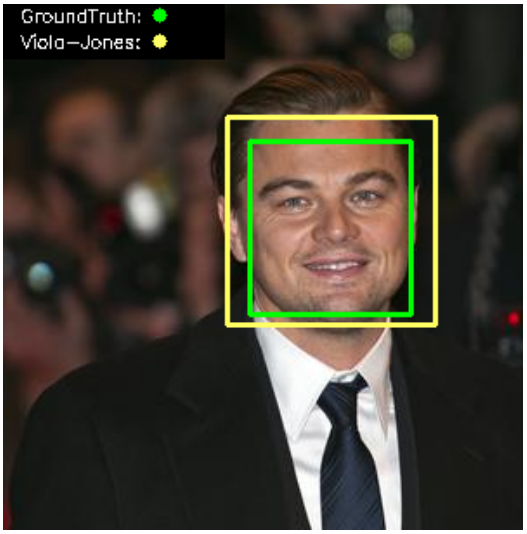

Figure A.70: 21_m_r bounding box performance for Viola-Jones.

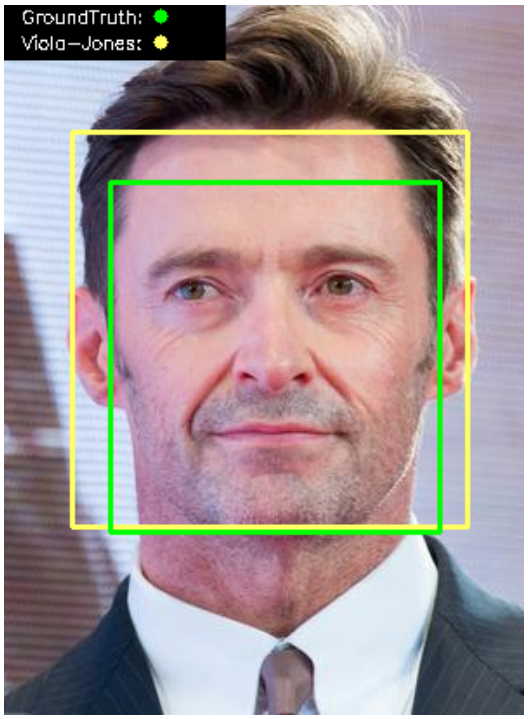

Figure A.71: *22_m_r bounding box performance for Viola-Jones.*

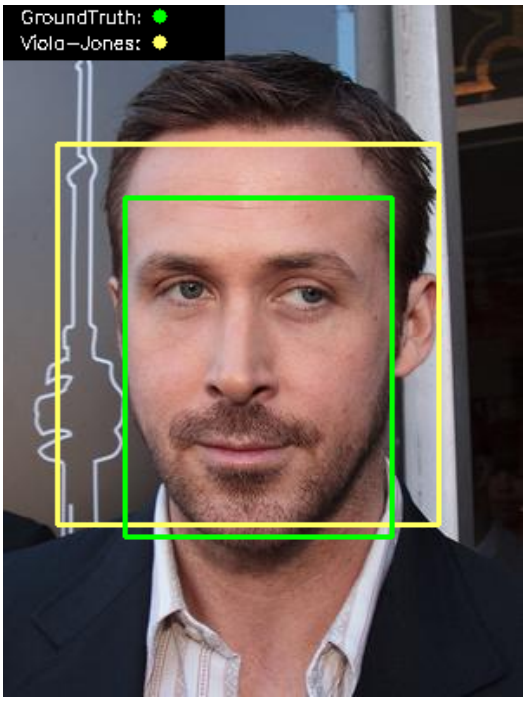

Figure A.72: *23_m_r bounding box performance for Viola-Jones.*

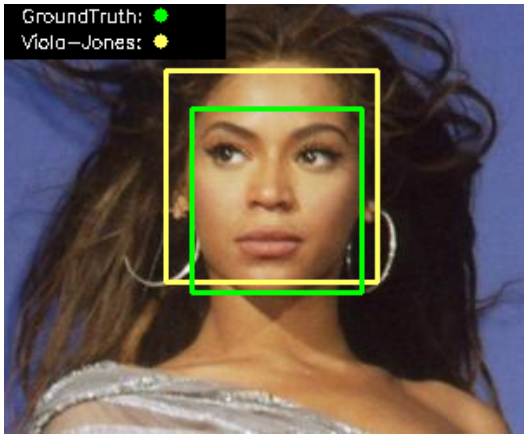

Figure A.73: *24_f_m bounding box performance for Viola-Jones.*

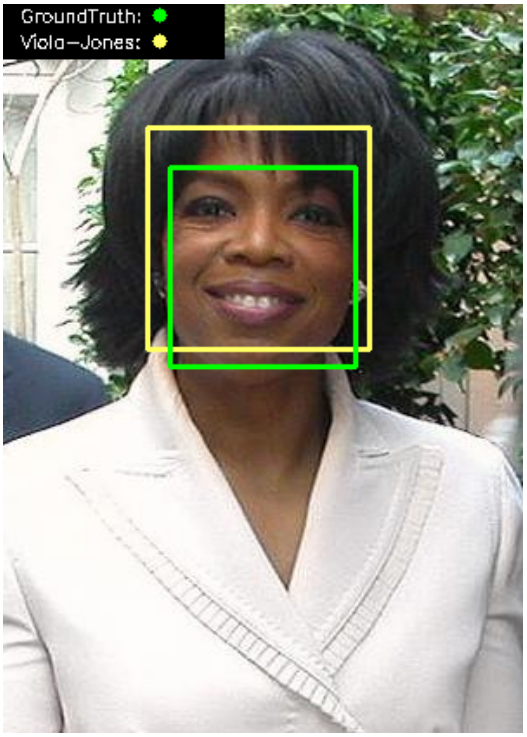

Figure A.74: *25_f_m bounding box performance for Viola-Jones.*

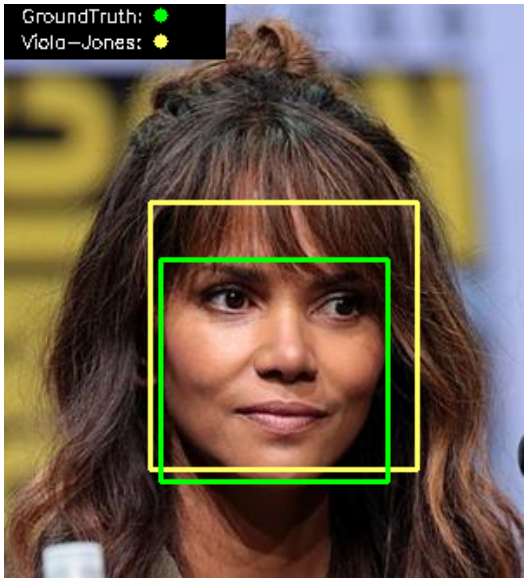

Figure A.75: *26_f_m bounding box performance for Viola-Jones.*

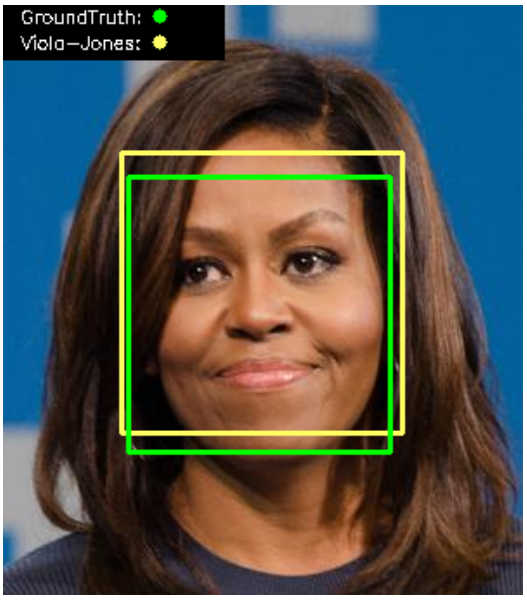

Figure A.76: *27_f_m bounding box performance for Viola-Jones.*

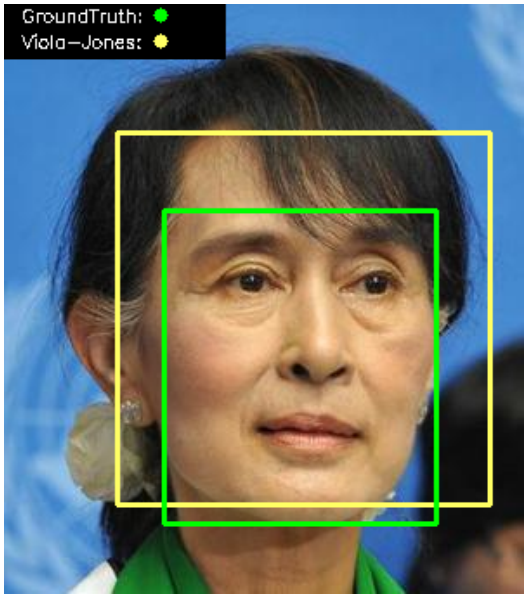

Figure A.77: *28_f_m bounding box performance for Viola-Jones.*

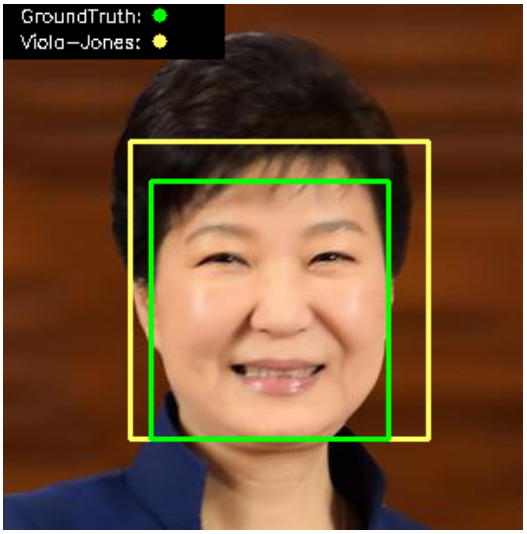

Figure A.78: *29_f_m bounding box performance for Viola-Jones.*

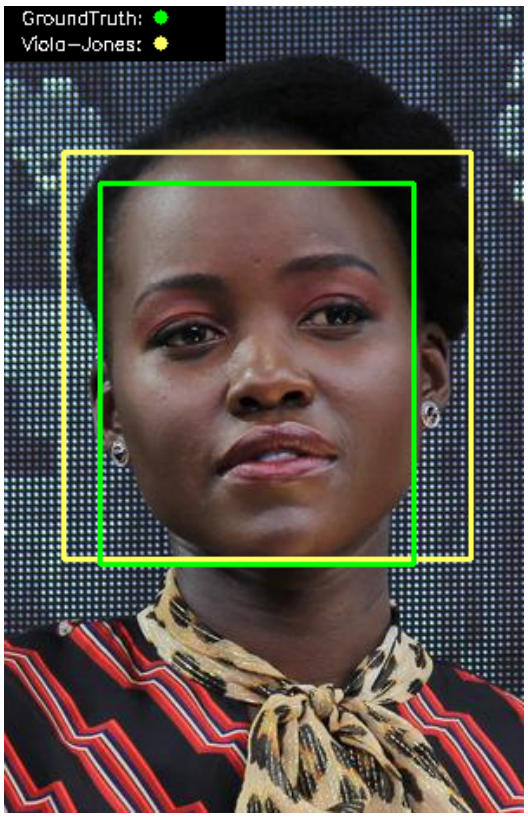

Figure A.79: *30_f_m bounding box performance for Viola-Jones.*

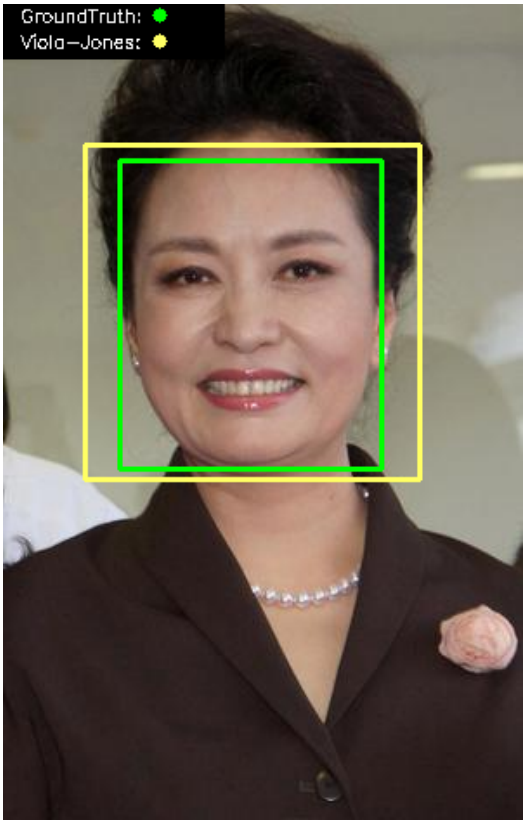

Figure A.80: *31_f_m bounding box performance for Viola-Jones.*

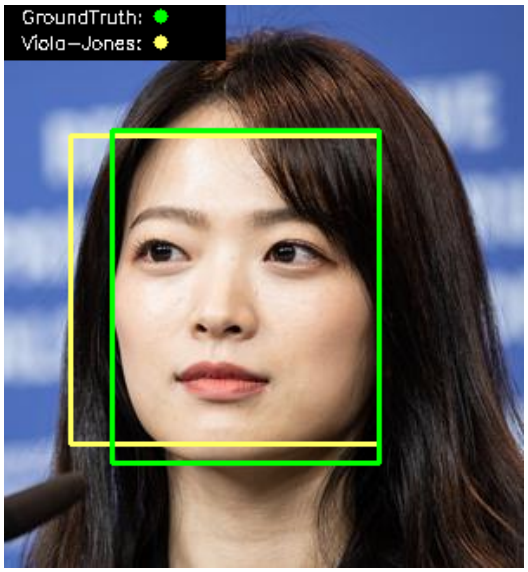

Figure A.81: *32_f_m bounding box performance for Viola-Jones.*

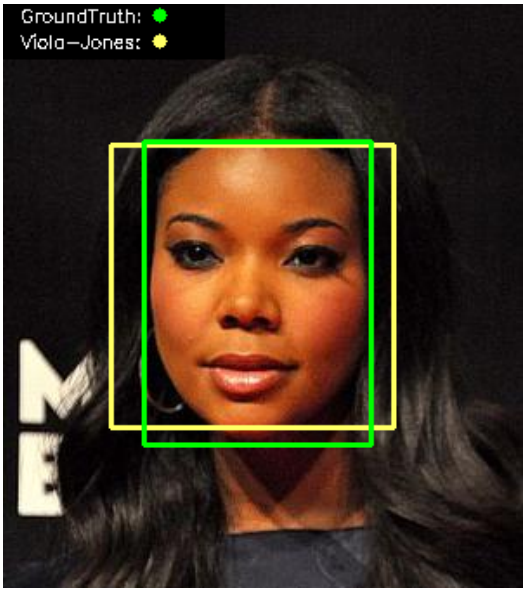

Figure A.82: *33_f_m bounding box performance for Viola-Jones.*

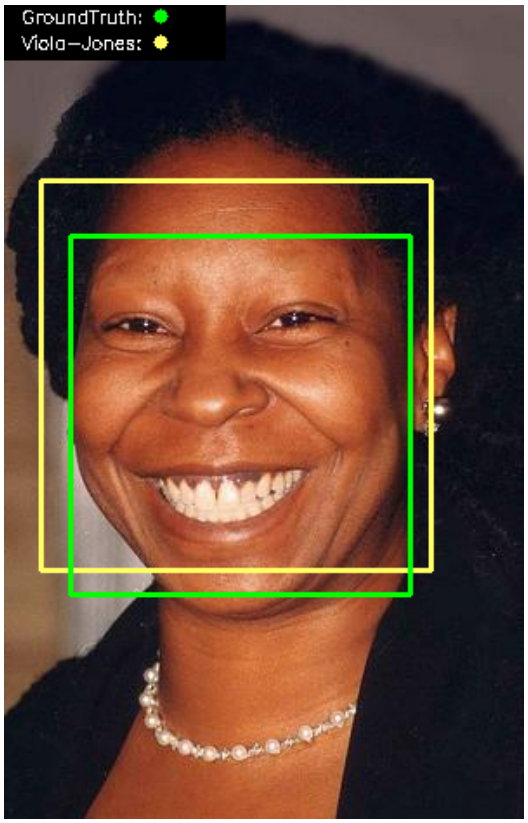

Figure A.83: *34_f_m bounding box performance for Viola-Jones.*

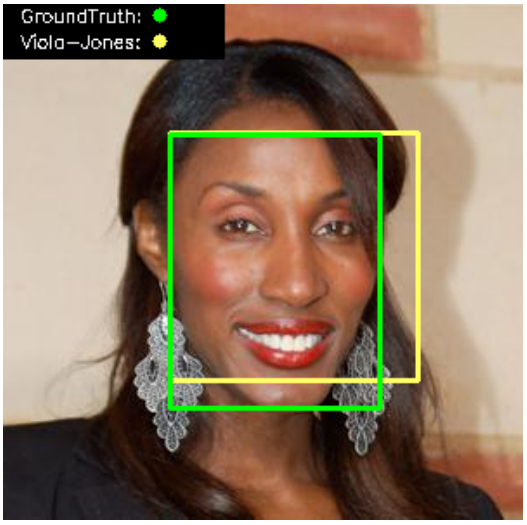

Figure A.84: *35_f_m bounding box performance for Viola-Jones.*

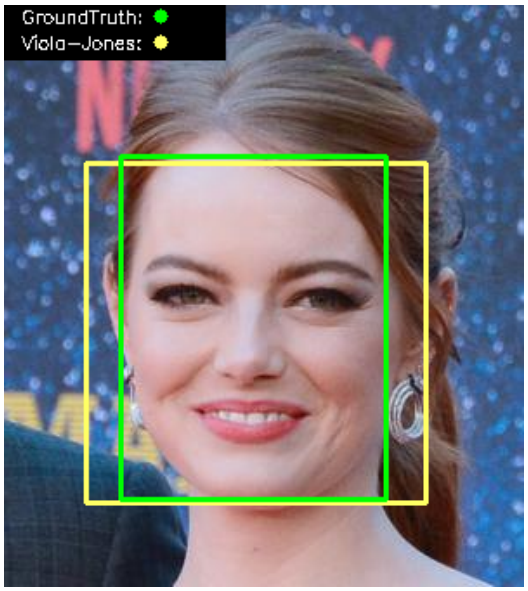

Figure A.85: *36_f_r bounding box performance for Viola-Jones.*

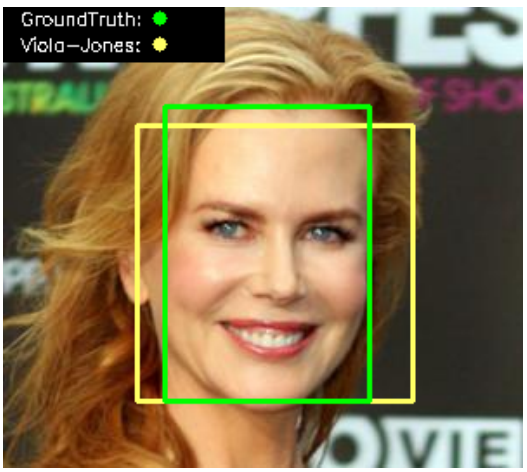

Figure A.86: *37_f_r bounding box performance for Viola-Jones.*

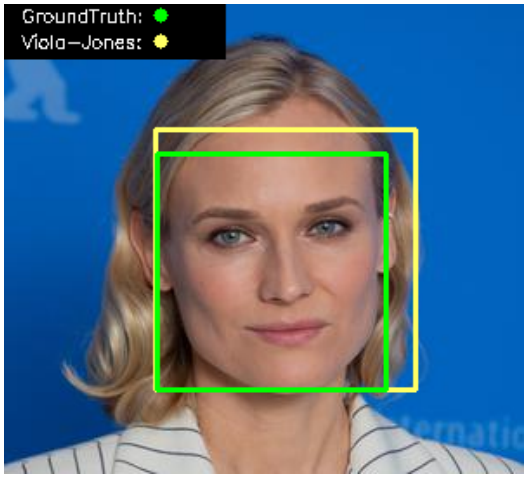

Figure A.87: *38_f_r bounding box performance for Viola-Jones.*

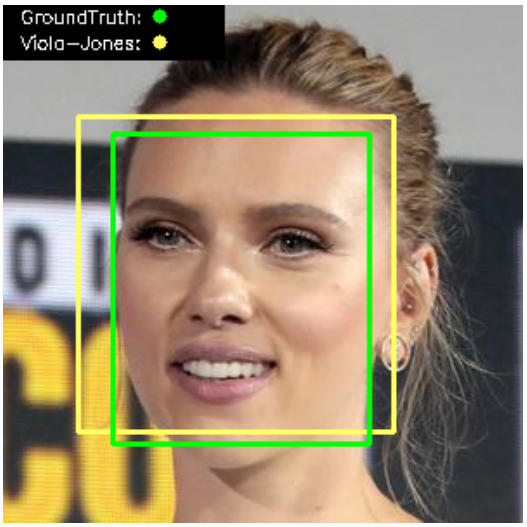

Figure A.88: *39_f_r bounding box performance for Viola-Jones.*

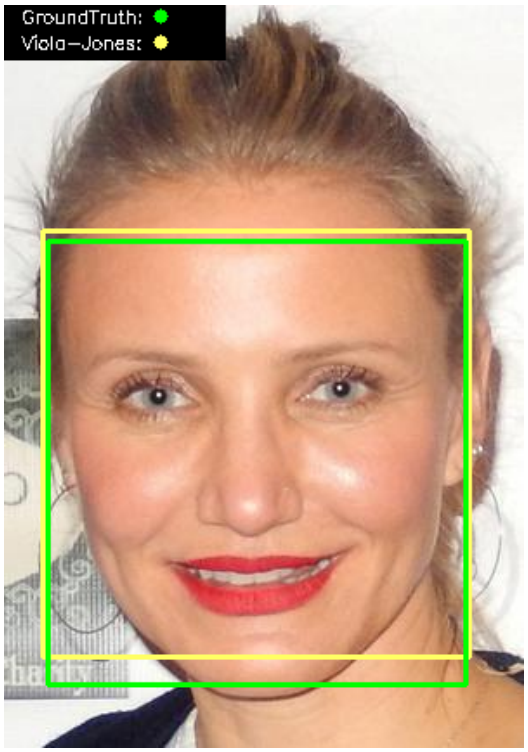

Figure A.89: *40_f_r bounding box performance for Viola-Jones.*

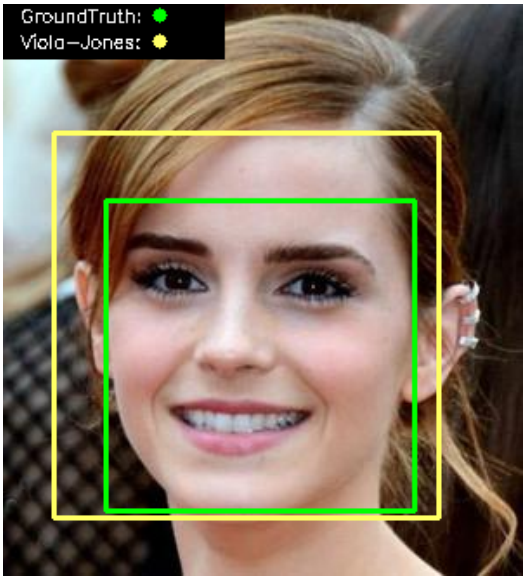

Figure A.90: *41_f_r bounding box performance for Viola-Jones.*

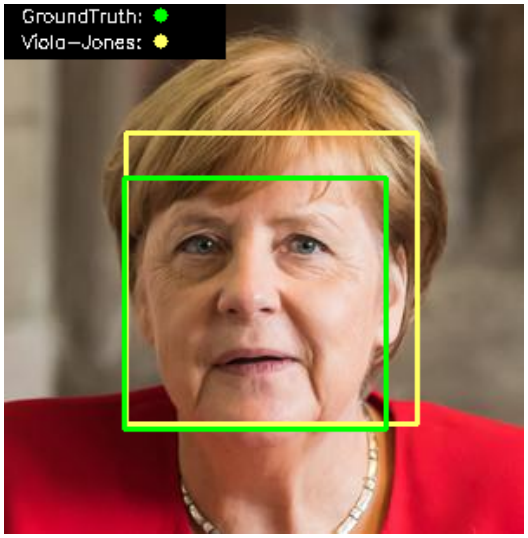

Figure A.91: *42_f_r bounding box performance for Viola-Jones.*

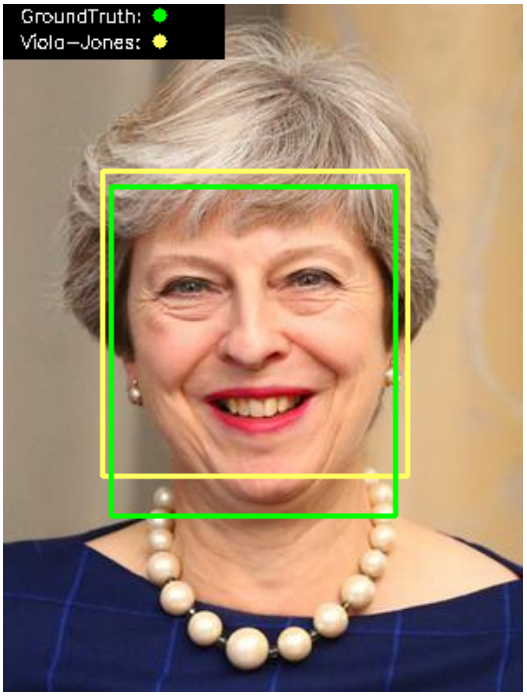

Figure A.92: *43_f_r bounding box performance for Viola-Jones.*

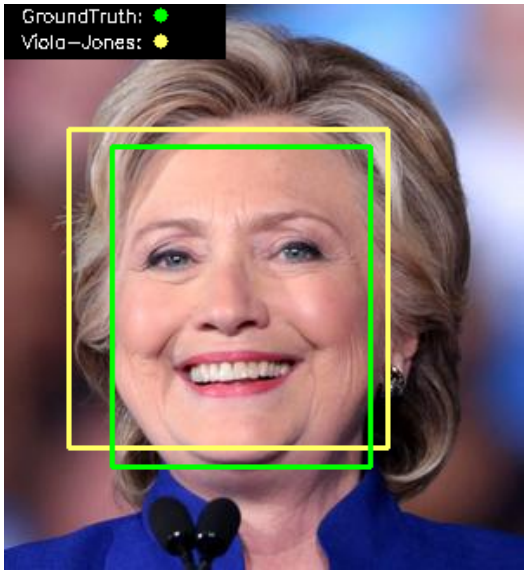

Figure A.93: *44_f_r bounding box performance for Viola-Jones.*

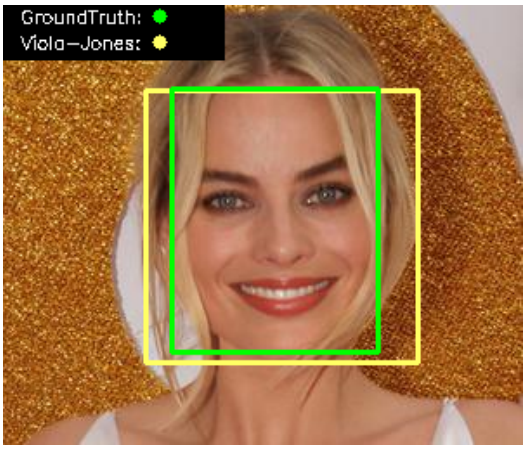

Figure A.94: *45_f_r bounding box performance for Viola-Jones.*

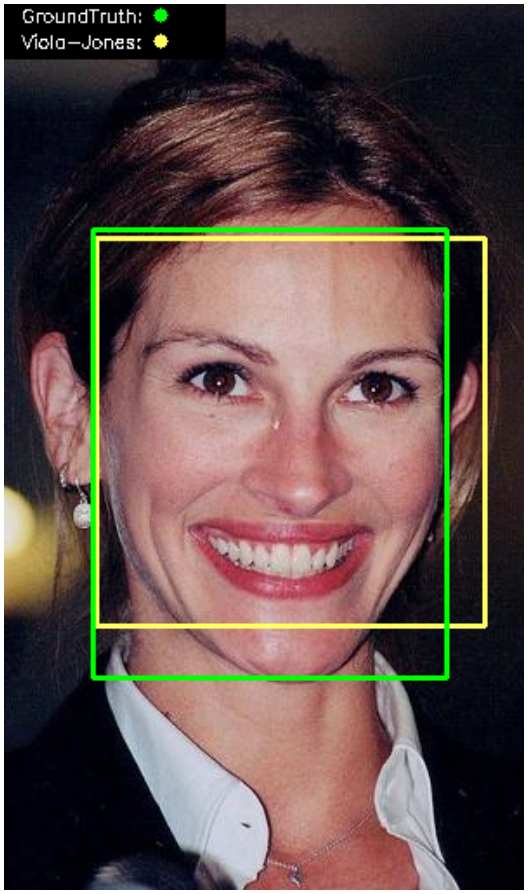

Figure A.95: *46_f_r bounding box performance for Viola-Jones.*

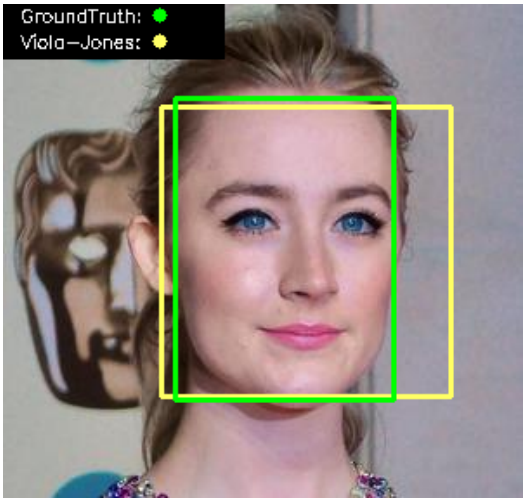

Figure A.96: *47_f_r bounding box performance for Viola-Jones.*

A.3 MTCNN Contrast Altered Predictions on Faces

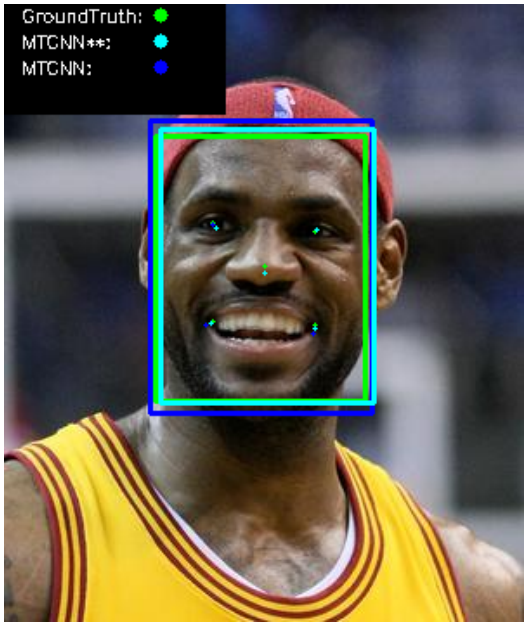

Figure A.97: 1_m_m MTCNN contrast altered performance.

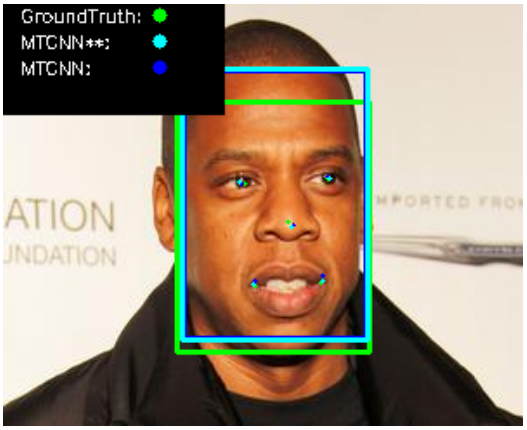

Figure A.98: 4_m_m MTCNN contrast altered performance.

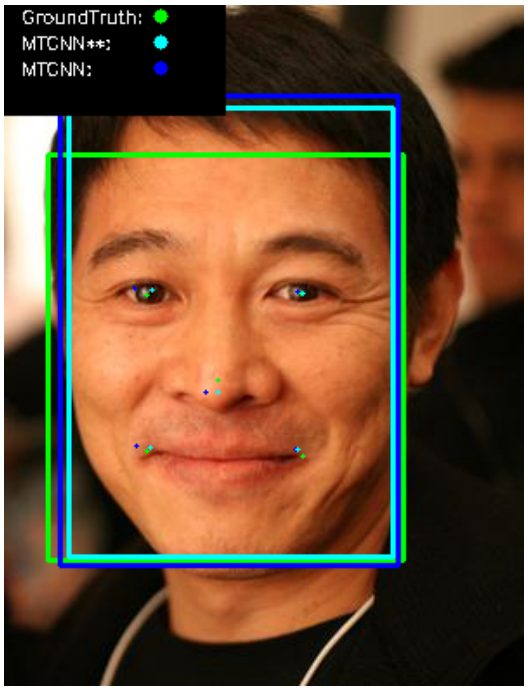

Figure A.99: 6_m_m MTCNN contrast altered performance.

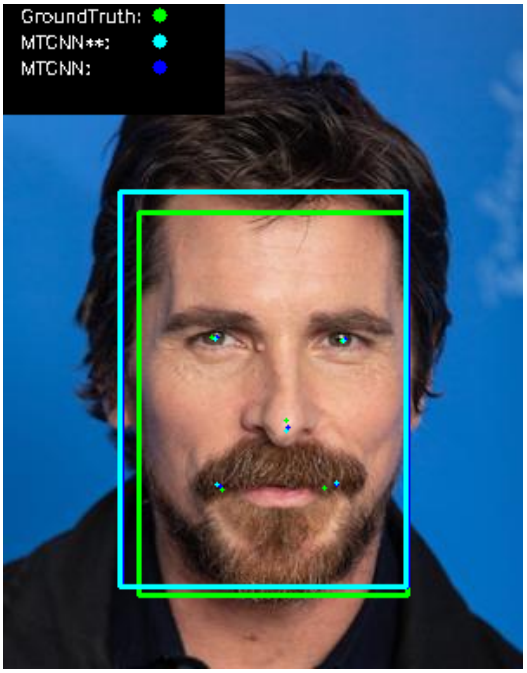

Figure A.100: 13_m_r MTCNN contrast altered performance.

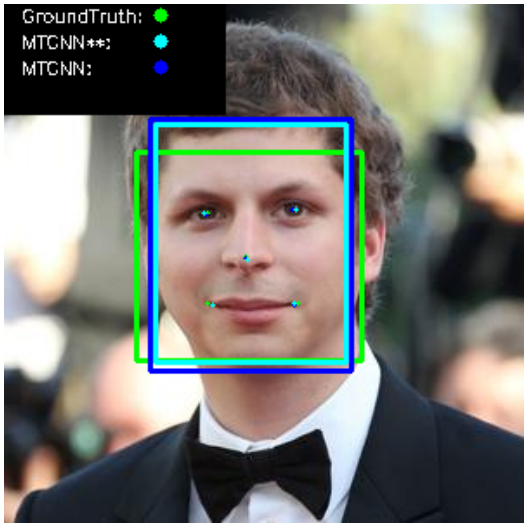

Figure A.101: *20_m_r MTCNN contrast altered performance.*

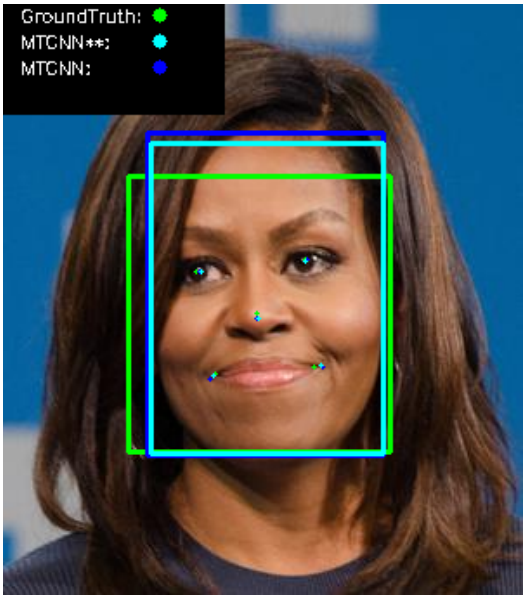

Figure A.102: *27_f_m MTCNN contrast altered performance.*

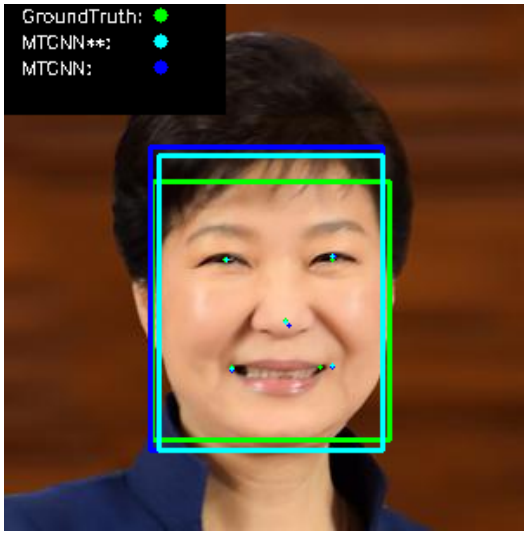

Figure A.103: *29_f_m MTCNN contrast altered performance.*

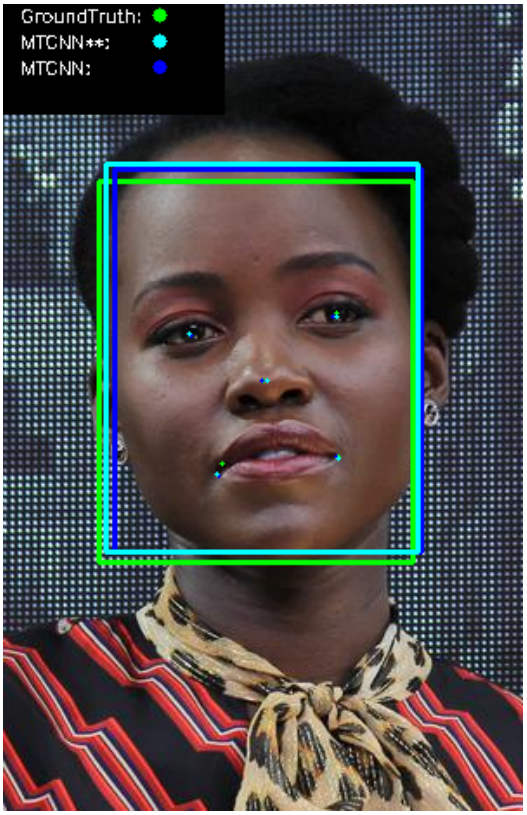

Figure A.104: *30_f_m MTCNN contrast altered performance.*

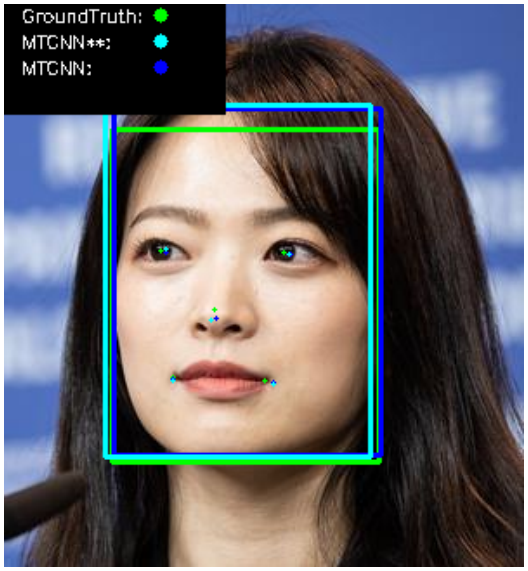

Figure A.105: *32_f_m MTCNN contrast altered performance.*

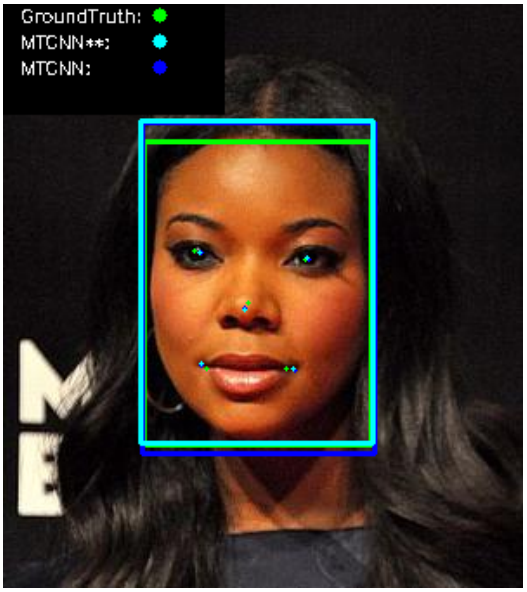

Figure A.106: *33_f_m MTCNN contrast altered performance.*

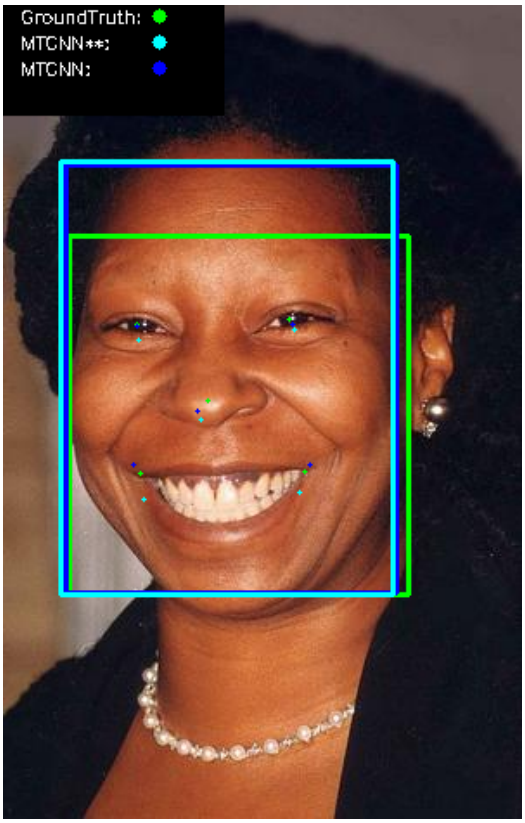

Figure A.107: *34_f_m MTCNN contrast altered performance.*

Bibliography

BBC. Passport facial recognition checks fail to work with dark skin, 2019. URL https://www.bbc.co.uk/news/technology-49993647. Accessed on 10.02.2021.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

J. Brownlee. How do convolutional layers work in deep learning neural networks?, 2017. URL https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. Accessed on 05.11.2020.

J. Brownlee. A gentle introduction to deep learning for face recognition, genders, 2019. URL https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/. Accessed on 05.11.2020.

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In S. A. Friedler and C. Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. PMLR.

Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, 2018. ISBN 9781538623350. doi: 10.1109/FG.2018.00020.

K. I. Chang, K. W. Bowyer, and P. J. Flynn. An evaluation of multimodal 2d+3d face biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):619–624, 2005. doi: 10.1109/TPAMI.2005.70.

E. Chopra. Using histogram of oriented gradients (hog) for object detection, 2017. URL https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/. Accessed on 05.02.2021.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I:886–893, 2005. doi: 10.1109/CVPR.2005.177.

J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5202–5211, 2020. doi: 10.1109/CVPR42600.2020.00525.

T. Esler. facenet-pytorch. https://github.com/timesler/facenet-pytorch, 2020.

R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

R. Gupta. Breaking down facial recognition: The viola-jones algorithm, 2019. URL https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999. Accessed on 04.02.2021.

E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001. ISSN 1077-3142. doi: https://doi.org/10.1006/cviu.2001.0921.

J. Hui. Understanding feature pyramid networks for object detection (fpn), 2018. URL https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c. Accessed on 08.11.2020.

V. Iglovikov. retinaface. https://github.com/ternaus/retinaface, 2020.

K. Karkkainen and J. Joo. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1548–1558, 2021.

V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.241.

D. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 07 2009. doi: 10.1145/1577069.1755843.

B. F. Klare, M. J. Burge, J. C. Klontz, R. W. Vorder Bruegge, and A. K. Jain. Face recognition performance: Role of demographic information. *IEEE Transactions on Information Forensics and Security*, 7(6):1789–1801, 2012. ISSN 15566013. doi: 10.1109/TIFS.2012.2214212.

N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009. ISBN 9781424444205. doi: 10.1109/ICCV.2009.5459250.

X. Li, T. Lai, S. Wang, Q. Chen, C. Yang, and R. Chen. Weighted feature pyramid networks for object detection. *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pages 1500–1504, 2019. doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015. doi: 10.1109/ICCV.2015.425.

S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. doi: 10.1109/34.192463.

B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney, and P. Grother. Iarpa janus benchmark - c: Face dataset and protocol. In *2018 International Conference on Biometrics (ICB)*, pages 158–165, 2018. doi: 10.1109/ICB2018.2018.00033.

M. Merler, N. Ratha, R. S. Feris, and J. R. Smith. Diversity in faces. *arXiv preprint arXiv:1901.10436*, 2019.

S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou. Agedb: The first manually collected, in-the-wild age database. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1997–2005, 2017. doi: 10.1109/CVPRW.2017.250.

S. Nagpal, M. Singh, R. Singh, and M. Vatsa. Deep learning for face recognition: Pride or prejudiced? *arXiv preprint arXiv:1904.01219*, 2019.

R. Perillo. iphone x face id unable to tell two chinese women apart, 2018. URL `https://www.eteknix.com/iphone-x-face-id-unable-tell-two-chinese-women-apart/`. Accessed on 10.02.2021.

R. Puri. Mitigating bias in ai models, 2018. URL `https://www.ibm.com/blogs/research/2018/02/mitigating-bias-ai-models/`. Accessed on 15.01.2021.

M. Rai, N. Werghi, H. Muhairi, and H. Alsafar. Using facial images for the diagnosis of genetic syndromes: A survey. *2015 International Conference on Communications, Signal Processing, and Their Applications, ICCSPA 2015*, 04 2015. doi: 10.1109/ICCSPA.2015.7081271.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

J. Roach. Microsoft improves facial recognition technology to perform well across all skin tones, genders, 2018. URL `https://blogs.microsoft.com/ai/gender-skin-tone-facial-recognition-improvement/`. Accessed on 15.01.2021.

A. Rosebrock. Intersection over union (iou) for object detection, 2016. URL `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`. Accessed on 15.11.2020.

A. Rosebrock. Facial landmarks with dlib, opencv, and python, 2017. URL `https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`. Accessed on 08.10.2020.

H. J. Ryu, H. Adam, and M. Mitchell. Inclusivefacenet: Improving face attribute detection with race and gender diversity. *arXiv preprint arXiv:1712.00193*, 2017.

C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark Localization Challenge. *Proceedings of the IEEE International Conference on Computer Vision*, pages 397–403, 2013. doi: 10.1109/ICCVW.2013.59.

L. Santhanam. Out of 30,000 hollywood film characters, here's how many weren't white, 2015. URL `https://www.pbs.org/newshour/nation/30000-hollywood-film-characters-heres-many-werent-white`. Accessed on 06.03.2021.

A. Shinwari, A. Balooch, A. Alariki, and S. Abduljalil Abdulhak. A comparative study of face recognition algorithms under facial expression and illumination. pages 390–394, 02 2019. doi: 10.23919/ICACT.2019.8702002.

P. Skalski. make-sense. https://github.com/SkalskiP/make-sense, 2020.

J. Snow. Amazon's face recognition falsely matched 28 members of congress with mugshots, 2018. URL https://www.aclu.org/blog/privacy-technology/surveillance-technologies/amazons-face-recognition-falsely-matched-28. Accessed on 10.02.2021.

T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. A deeper look at dataset bias. *Advances in Computer Vision and Pattern Recognition*, (9783319583464):37–55, 2017. ISSN 21916594. doi: 10.1007/978-3-319-58347-1_2.

P. Viola, M. Jones, et al. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001. doi: 10.1109/cvpr.2001.990517.

F. Wang, L. Chen, C. Li, S. Huang, Y. Chen, C. Qian, and C. C. Loy. The devil of face recognition is in the noise. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 780–795, Cham, 2018. Springer International Publishing.

S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5525–5533, 2016. doi: 10.1109/CVPR.2016.596.

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016. ISSN 10709908. doi: 10.1109/LSP.2016.2603342.

Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4352–4360, 2017. doi: 10.1109/CVPR.2017.463.