PYTHON BASED POSE AND SKIN COLOR IMPROVED FACE RECOGNITION SYSTEM

by

Akku Merium Mathew

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2020

Committee Members:

Semih Aslan, Chair

William Stapleton

Damian Valles

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

DEDICATION

This M.Sc. thesis is delightfully dedicated to the Almighty God for his guidance throughout my thesis, my dear husband (Allan Babu Kurian) without whom this achievement could never be accomplished and my beloved parents (Mathew Cherian & Beena Mary Mathew) for their endless support and prayers.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my committee chair and thesis advisor Dr. Semih Aslan, for his continuous guidance and encouragement, without his persistent help this thesis would not have been possible.

I am grateful to my committee member, Dr. William Stapleton, for his encouragement, valuable feedback, and guidance for completion of this thesis.

I would like to convey my special thanks to my thesis committee member, Dr. Damian Valles and graduate advisor, Dr. Vishu Viswanathan for their attention, time, guidance, and exquisite support throughout the research.

Last but not least, I would like to express my utmost gratitude to the God Almighty, who gave me strength and wisdom to overcome my hurdles and to believe in my dreams. I would also like to thank my husband Allan Babu Kurian, who believed in me, supported me to accomplish my dreams and taught me to cherish every small milestone towards my goal. Without his help I would never have completed my thesis.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
CNN	Convolution Neural Networks Abbreviation
CPU	Central Processing Unit
HSV	Hue Saturation Value
GAN	Generative adversarial Networks
StyleGAN	Style Transfer Generative Adversarial Network
RGB	Red Green Blue
MTCNN	Multi-Task Cascaded Convolution Network
OpenCVDNN	OpenCV Deep Neural Network
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
LBPH	Local Binary Pattern Histogram
ID	identification
SVM	Support Vector Machine
KNN	K Nearest Neighbors
AdaIN	Adaptive Instance Normalization
CLAHE	Contrast Limited Adaptive Histogram Equalization
GPU	Graphics Processing Unit

ABSTRACT

Face recognition has been an active research topic in the field of computer vision. Face recognition is currently being used for applications such as law enforcement, security, and video surveillance. Variations in pose and skin color are the two main factors that affect the accuracy of a face recognition system. The aim of this research is to develop a pose invariant and skin color invariant face recognition system for surveillance and law enforcement applications. In this research, we tackle the problem of pose variations and skin color separately.

To understand the effect of pose on face recognition, we initially created a face recognition system for variation in head poses from -90 degrees yaw angle to 90 degrees yaw angle. From our study, we found that the accuracy of a face recognition system drops significantly from a yaw angle of 45 degrees to 90 degrees and -60 degrees to -90 degrees. To combat this problem, we identified two solutions. Firstly, we propose skin segmentation in HSV color space as a preprocessing step to improve accuracy for non-frontal poses from 45 degrees to 90 degrees and -60 degrees to -90 degrees. Secondly, we propose using a Style Transfer Generative Adversarial Network (StyleGAN) generated frontal image for face recognition to improve accuracy for non-frontal poses from 45 degrees to 75 degrees and -60 degrees to -75 degrees. The experimental results demonstrate that both methods have significantly improved the accuracy of non-frontal poses.

To study the effect of skin color on face recognition, we created the ‘Celeb-Skin’

dataset. The dataset contains 480 images of male celebrities of different races such as White, Asian, and African American. From testing the face recognition system on the Celeb-Skin dataset, we found that the accuracy of White faces was comparatively higher than African American and Asian faces. In this research, we propose that creating a training dataset based on skin color and grouping the celebrities based on their skin color would improve the accuracy of dark-skinned faces. Experimental results show that our technique has improved the accuracy of dark-skinned faces by 7.38 percent.

I. INTRODUCTION

Background and Motivation

Face Recognition is the process of identifying an individual from an input face image. Currently, face recognition is used in applications such as law enforcement, biometrics, and video surveillance. The accuracy of the face recognition system for a surveillance system should be high. Some of the major factors that affect the accuracy of a face recognition system are variation in poses, illumination, and skin color. This research aims at improving face recognition accuracy for variations in pose and skin color.

For surveillance and law enforcement applications, a frontal image will be available as a gallery image and the image acquired might be a profile or non-frontal image. In such cases, a normal face recognition system fails to recognize faces correctly. Therefore, it is substantial to build a pose invariant face recognition system for law enforcement and surveillance applications. In the past decades, a lot of research has been conducted in improving the performance of face recognition for pose variations and the researchers have found ways to improve the performance of face recognition algorithms for non-frontal poses by training their algorithms with large amounts of data. The main problem with the current face recognition system is racial bias. Current face recognition technology is less accurate on dark skin faces. Therefore, using such face recognition technology for law enforcement applications would put people of color at risk. From a study conducted to identify the performance of various commercial face recognition technology provided by global tech companies on different races and gender, it was found that the technology worked fine 99 percent of the time for white men. However,

with darker skin, face recognition technology failed often. Another study conducted by a researcher at MIT on the three most leading commercial face recognition software [1] found that the error rate for dark-skinned women is 35%. If the government plans to employ face recognition to create a centralized database for national security reasons, it is crucial to improve the performance of face recognition technology for dark-skinned faces, to treat all people equally, and to prevent future risks. Therefore, it is important to develop a reliable face recognition system that works for all skin colors.

Problems

Face recognition is an active research topic in computer vision. Recently Deep convolutional neural networks produced state of art results on benchmark datasets of face recognition. Some of the examples of deep convolutional neural architectures (CNN) for face recognition are FaceNet, VGGFace, and VGGFace2. These architectures have been trained with millions of face images and with vast computation resources. For example, VGGFace2 is trained on a dataset of 3.31 million images of 9131 subjects with an average of 362.6 images per subject with large variations in pose, age, gender, and ethnicity. The CNN architectures can handle pose variations with a very less error rate. Numerous research in face recognition with deep CNN architectures has almost solved the pose challenges in face recognition. The demerit with deep learning architectures is that it requires a lot of training data and huge amounts of computational resources. In this research, we are focusing on face recognition with less training data per person and no need for additional computational resources than a normal CPU.

Variations in Pose has been a major challenge in traditional face recognition approaches. For surveillance applications, the face image is mostly captured in an

uncontrolled environment. Images captured in an uncontrolled environment have varying illumination and faces might not be actually facing the camera. Depending on how the head of the face is facing the camera, the pose of the face can be categorized into two, frontal and non-frontal. A frontal pose is a front view of a face image whereas a non-frontal pose is a side view of a face image. Classic face recognition algorithms like the Eigenface algorithm and the Fisherface algorithm work well for frontal pose images. Accuracy of a traditional face recognition algorithm degrades for non-frontal poses. Matching a non-frontal probe image with a frontal gallery image induces differences in an Appearance-based model like Eigenface and consecutively reduces the accuracy. Hence, the accuracy of face recognition systems for non-frontal poses has to be improved.

Skin color is another important problem faced by current face recognition technologies. Face recognition technologies are less accurate with dark skin faces compared to white skin faces. Face recognition technology provided by leading companies such as Amazon, IBM, and Microsoft are used by lawmakers to find criminals. MIT's Ms. Buolamwini conducted a study to examine the performance of the three most leading face recognition technologies on gender and race. Her research found that all three companies performed better on males than females, and also on lighter-skinned faces than darker-skinned ones. The systems performed worst on darker females with Microsoft recording an error rate of one in five (20.8 percent) and IBM and Megvii's Face++ more than one in three [1] (34.7 percent and 34.5 percent respectively). The error rates for most leading face recognition technologies are shown in Figure 1. The higher error rate makes the minorities most vulnerable to dark skin bias in face

recognition applications. The reason for the dark skin bias is the lack of diversity in training images and benchmark datasets. The existing face recognition technologies are often trained and tested on a non-diverse dataset [2]. A popular open-source face dataset, 'Labeled Faces in the wild', was estimated to be 83.5 percent white [2]. A system trained mostly on white faces would have lower accuracy on black faces. Therefore, it is very important to develop a face recognition system whose performance is independent of the color of skin.

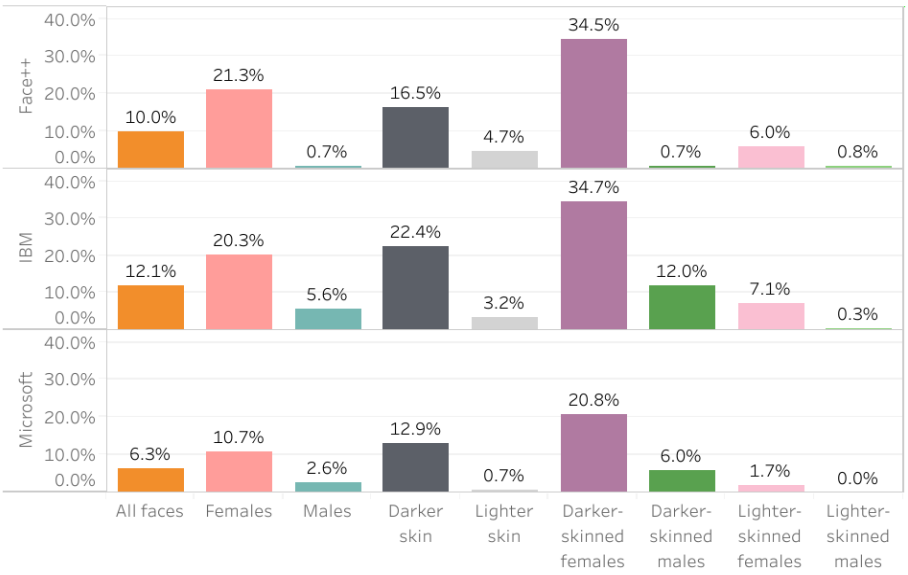


Figure 1. Error rates of facial recognition systems by skin color and gender [1]

Solutions

A good face recognition system must be pose-invariant and skin color-invariant. The accuracy of traditional face recognition algorithms like the Eigenface algorithm for non-frontal poses can be improved by separating the skin pixels from non-skin pixels of the image. The process of separating skin pixels from non-skin pixels is known as skin segmentation. Performing skin segmentation as an image preprocessing step before face detection would improve the performance of the face recognition system for non-frontal

poses. The experimental results show that skin segmentation in HSV color space has improved the performance of face recognition systems of non-frontal poses. Another solution to improve the accuracy of non-frontal poses is by creating a frontal view from a non-frontal image. In this research, we have created a frontal view from a non-frontal image by using a state of art architecture called Style Transfer Generative Adversarial Network (StyleGAN). A Generative Adversarial Network (GAN) creates a fake image from an actual image. The problem with GAN is that we lack control over the style of the synthetic image generated. To have control over the style of the synthetic image generated and to create photo-realistic images, researchers created a different architecture known as StyleGAN. The architecture of the StyleGAN model provides control over the style of generated images at various levels of detail. This helps us to generate synthetic images with variations in pose, age, expression, and gender. We use StyleGAN to generate a frontal view from a non-frontal image. The StyleGAN generated frontal image is then used for face recognition. The experimental results show that using StyleGAN generated frontal view image for face recognition instead of the non-frontal face image improves accuracy for non-frontal pose significantly.

Skin color is another major challenge faced by both the current state of art face recognition algorithms and traditional face recognition algorithms. The face recognition technology is more inaccurate on dark skin faces. Research has been conducted to examine the reasons for the lower accuracy of face recognition technology on dark skin faces. The main reason for the dark skin bias is the less diverse training datasets. In this thesis, we aim to improve the performance of traditional face recognition algorithms such as Eigenface algorithms on dark skin faces by creating separate training datasets based on

the skin tone of a face image. We use skin segmentation and K-means clustering to extract the most dominant skin tone in a face image. The skin tone is extracted in the form of RGB values. Skin color is dependent on illumination [3] and therefore RGB values of the skin tone of the same person vary based on the lighting conditions in which the image was captured. Therefore, we group the face images of different people based on their range of RGB values. Based on the RGB values, the skin tones are classified into white, brown, dark brown, and black. Our experimental results demonstrate that creating training datasets based on the skin tone of a person significantly improves the accuracy of dark skin tones.

Thesis Focus

The objectives of the thesis research are (i) to develop a pose-invariant face recognition system adaptable to yaw variations up to 90 degrees (ii) to implement a reliable face recognition system that improves face recognition accuracy for dark skin faces. Research has been conducted to improve the performance of face recognition systems for variation in poses. However, there is some research gap in identifying at what face angles the face recognition system starts to perform worse. While dealing with non-frontal faces, selecting an appropriate face detection method is important. Previous researchers have not addressed this question. In this research, we improve the performance of face recognition systems for non-frontal poses by skin segmentation. This technique has not been used in any prior research to improve face recognition performance on non-frontal poses. In addition to skin segmentation, we also propose the use of StyleGAN generated face image for face recognition. There is no previous research that has used StyleGAN generated images for face recognition.

In addition to investigating the effect of pose on face recognition, our research also conducts a study on the effect of the color of the skin on face recognition with traditional face recognition algorithms. Researchers have tried to improve the performance of face recognition technologies on dark skin faces through creating a diverse training dataset and training specifically on a particular race/ethnicity. In our research, we propose that creating training datasets based on skin tone would improve face recognition. Most researchers consider skin as a non-reliable parameter for face recognition due to illumination variations. But in this research, we provide solutions on how skin tone can be made a reliable parameter for face recognition. We also compare how our technique performs with other solutions offered by researchers such as diverse training datasets and training on a specific race.

The contributions of this thesis are:

1. A detailed study on the effect of pose on face recognition and to identify at what face angles does the face recognition accuracy starts to drop significantly
2. Identifying the best face detection model for extreme non-frontal poses
3. Improvement of face recognition accuracy for non-frontal poses by performing skin segmentation as an image preprocessing step before face detection
4. Improvement of face recognition accuracy for non-frontal poses by using a synthetically generated frontal view from a non-frontal image for face recognition through StyleGAN
5. A detailed study on the effect of skin color on the performance of face recognition systems
6. Improvement of accuracy of dark skin faces by creating a training dataset based

on the skin color of face images

Organization of the Thesis

The thesis is arranged into nine chapters:

a) CHAPTER I: introduction of the work has been presented

b) CHAPTER II: describes the previous work conducted in face recognition

c) CHAPTER III: studies the effect of the pose on face recognition and delineates the methods employed to develop the machine learning model for face recognition

d) CHAPTER IV: discusses the improvement of face recognition in non-frontal poses

e) CHAPTER V: studies the effect of skin color on face recognition and demonstrates methods employed to develop a machine learning model for face recognition

f) CHAPTER VI: describes the enhancement of face recognition for dark skin face

g) CHAPTER VII: explains and discusses the results of the initial study on the effect of pose on face recognition. This chapter also illustrates the results of improvement of face recognition accuracy for non-frontal poses

h) CHAPTER VIII: demonstrates the results of the initial study on the effect of skin color on face recognition and analyzes the results. This chapter also elucidates the results of the improvement of face recognition accuracy for dark skin faces.

i) CHAPTER IX: presents the conclusion of the work and future scope of the work described

II. LITERATURE REVIEW

Face recognition has been an active research topic in the field of computer vision. Based on the representation of features in an image, face recognition is mainly classified into two approaches, the Appearance-based approach, and the Feature-based approach. In the Appearance-based approach, the pixel intensity values of a face image are used as features whereas, in the Feature-based approach, the features are derived from the intensity data and correspond to the location of facial features such as mouth, eyes, nose, etc. [4]. Traditional face recognition approaches like Principal Component Analysis based Eigenface algorithm, Linear Discriminant Analysis based Fisherface algorithm falls under the category of Appearance-based approaches. The Eigenface algorithm is an information theory approach that decomposes face images into a small set of characteristic feature images known as Eigenfaces. These eigenfaces are projected on an Eigenface space and faces are then classified based on the distance between the individual projections [5]. Here the images are captured in a controlled environment and poses are mostly frontal. The accuracy of most of the traditional face recognition algorithms decreases for changes in pose, illumination, and expressions [6]. Eigenface being an appearance-based approach, only considers the 2D facial appearance of images. The 2D appearance of faces changes along with different poses due to the non-planar geometry of the face. The accuracy of the Eigenface algorithm and Fisherface algorithm degrades for changes in pose due to the inconsideration of 3D alignment of a face in feature extraction.

Variations in pose have been a major problem for face recognition technologies. Numerous research [6] [7] [8] [9] [10] [11] have been performed to tackle this problem.

Gross et al. proposed an appearance-based algorithm to deal with the pose variations by estimating Eigen light fields from input images. The input image is then matched with the gallery image by comparing the ELF Coefficients [7]. In the recognition stage, the face can be recognized even when there is only a single image of each person in the gallery [7]. This method was able to address the pose issues successfully compared to the Eigenfaces approach.

Reconstructing a frontal view from a non-frontal face image is another method for handling the pose variation in face recognition. Blanz and Vetter [8] created a 3D Morphable Model (3-DMM) model for reconstructing a frontal view and produced impressive results in pose invariant face recognition. However, the 3D model construction is computationally expensive. Chai et al. [6] proposed a Local Linear Regression based (LLR) method to generate a frontal view from a non-frontal view. This method was able to handle pose variations of yaw ±45 degrees. Reconstruction of frontal view by 3D pose normalization was proposed in [9]. The frontal face is reconstructed by 3D pose normalization using 2D facial feature points. A 2D input image is projected onto a 3D face model to create a textured 3D model. A 3D rigid transformation is performed to normalize the input face into a frontal pose. This research produced the best results for pose variations of ±45-degrees yaw and ±30 degrees pitch angles. Changxing et al. [10] delineate a pose invariant face recognition system that handles the pose ranges of ±90 degree yaw angle. Here a frontal face is obtained using 3D pose normalization and the face matching is performed at a patch level than at a holistic level. For extreme poses, 3D pose normalization results in self occluded faces, and only the unoccluded portion of the face is considered, and patch wise matching is performed with the gallery image. Another

way to handle extreme poses is described in [11]. This research uses 3D pose correction and only considers the half profile face for self-occluded faces. The face recognition was implemented using ResNet architecture with 28 layers. This technique [11] outperformed the state of art techniques for extreme poses and is easier to implement. The above-mentioned studies [9] [10] [11] have used 3D pose normalization to construct a frontal view from a non-frontal image.

The advent of Generative Adversarial Networks (GAN) has led to many studies in frontal view synthesis. Frontal view synthesis is the process of generating a frontal view from a non-frontal image. The only difference is that the image is synthetically generated. Over three years there are several GAN based architectures created for frontal view synthesis. Face Frontalization Generative Adversarial Networks (FF-GAN) is a 3D Morphable Model conditioned face frontalization network to generate neutral head pose images [12]. Pose weighted Generative Adversarial network (PW-GAN) creates photorealistic frontal views for large pose variations [13]. PW-GAN considers both 3D face geometry and uses pose normalization results to help GAN learn self-occluded regions. The Two Pathway Generative Adversarial Network (TP-GAN) recovers photorealistic and identity preserving frontal view image from a profile image. TP-GAN generates a photo-realistic frontal view from a non-frontal image, and it works well for extreme poses with yaw angles greater than 60 degrees. The research [14] found that the synthesized frontal image improved the face recognition performance for non-frontal poses. The main advantage when comparing 3D pose normalized image and TP-GAN generated frontal view image is that even for extreme pose variations, the faces are photo-realistic and preserve identity. Hung et al. [15] propose a Dual Attention

Generative adversarial network (DA-GAN) for photo-realistic face frontalization by capturing both contextual dependencies and local consistency during GAN training. DA-GAN architecture outperformed TP-GAN for generating photo-realistic images for large pose variations. DA-GAN generated frontal view image gave high face recognition accuracy for extreme poses of 90 degrees [15]. Our research is using StyleGAN architecture proposed in [16] generated face image and we move the latent representations generated by StyleGAN in the latent directions of a pose. This technique preserves the identity of the person to some extent. However, when incorporated with Eigenface algorithms, the StyleGAN generated frontal image offers improvement in face recognition accuracy for non-frontal poses up to 75 degrees.

Skin color is another factor that plays a significant role in face recognition. In a study conducted by Joy Buolamwini to identify the performance of three leading face recognition technologies such as Microsoft, IBM, and Face++ on different races and gender, it was found that the technology identified white people correctly 99 percent of the time but failed often to identify dark faces [1]. Training on non-diverse datasets is the primary reason for the racial bias of the face recognition system [17]. If black people are underrepresented in benchmark datasets, then the face recognition system will be less successful in identifying black faces [13]. This problem can be solved by using a diverse training set [17]. Researchers found that even a representative dataset could not solve the dark skin bias in face recognition technology. One of the reasons why face recognition technology is less successful on dark skin is that black skin reflects light in different wavelengths compared to white skin.

A study on the effect of skin color on face recognition is presented in [2]. In this

study, training specifically on a black face improved the accuracy of black faces by 2 percent. i.e., the accuracy of black faces can be improved by having a different training set with black faces alone. Especially in biometric applications, the 1-1 verification and identification are not performed automatically. There would be an officer who manually identifies whether a face is black or white. For example, if the individual identified on the probe image is black, then the algorithm specifically trained for black images will be used for training. During the testing stage, the face with the highest matching score will be selected as the face identified. This would improve the recognition accuracy of black faces. Training on a specific race outperforms the usage of a diverse training dataset in the face recognition algorithm [2]. In cases where the demographics are not known, generic algorithms trained equally on all cohorts were used. Creating a specific dataset based on the race of face image requires one to identify the race from a face image. Sometimes human identification can be wrong too because one cannot identify the ethnicity of a person just from his/her skin color. Our research differs from existing works by creating a specific training dataset based on skin color. The dominant skin color can be extracted from an image with the help of skin segmentation and K-means clustering. Based on the RGB values of the skin color extracted, the face images are grouped into four skin shades namely white, brown, dark brown, and black. Our technique has improved the accuracy of dark skin faces significantly.

III. EFFECT OF POSE ON FACE RECOGNITION

Face recognition has been a challenging research problem in computer vision. A face recognition system identifies a face from a background image and recognizes the person in the image. Firstly, a face recognition system detects a face/faces in an image and then transforms the face/faces into a feature vector. A feature vector is like a face print and contains important features of the face image. Once a feature vector of detected faces is obtained, the face recognition system compares it with a face database of known faces. The face with the least distance from the feature vector of the detected face will be the closest match to the detected face. A face recognition system can return the name of the person or an image that matches the detected face depending on the application. Most of the current face recognition systems work well on frontal pose images. However, with variation in poses, the face recognition technologies fail to recognize people correctly. For example, assume that the face detected is non-frontal and we compare the detected face with a known face database containing frontal images of known faces. When a face recognition system compares the feature vectors of detected non-frontal faces with feature vectors of known frontal faces in a database, the system might not recognize the person correctly. Even though the images are of the same person in different poses, the system finds the distance between the feature vectors of frontal and non-frontal pose larger than the threshold set by the system. It is easy for humans to identify a person from different poses, however, computers cannot. Hence, it is very important to study the effect of pose on face recognition and find ways to improve the performance of face recognition systems on non-frontal poses.

The effect of poses on face recognition is studied in this section. The pose of the face can be described as the orientation and rotation of the head with respect to the position of the camera. The head pose angle is classified into three types namely, yaw angle, pitch angle, and roll angle as shown in Figure 2. Yaw angle is the rotation of the face along the Y-axis of the camera plane, for instance, when a person turns their head towards the left or right relative to the position of the camera. Pitch angle is the rotation of face along the X-axis of the camera plane, e.g., turning the face-up or down from the camera whereas the roll angle is the rotation of face along the Z-axis of the camera plane. For the study of the effect of head pose, specifically variations in yaw angles on face recognition, we build a simple face recognition system as shown in Figure 3. A simple face recognition system consists of stages such as image acquisition, face detection, feature extraction, and face recognition.

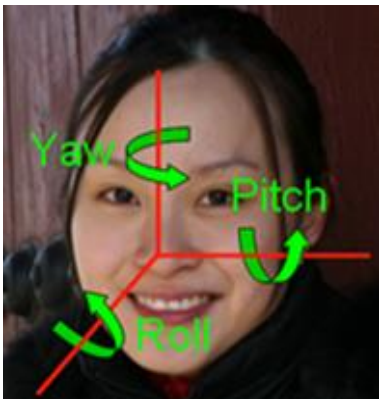


Figure 2. Head pose angles

Figure 3. Block diagram of a face recognition system

Image Acquisition

To study the effect of pose on face recognition, the images are captured from a mobile camera setting. To obtain different poses, markers were put in the whole room for various yaw angles from -90 degrees to +90 degrees in 15-degree increments. Each marker corresponds to a pose at a particular yaw angle maintaining the pitch angle at 0. Post-it was used as markers. The whole set of post-it covers a half-sphere in front of the person. To obtain the face in the center of the image, the person was asked to adjust the chair to see the device in front of him. After this initialization phase, we asked the person to stare successively at 13 post-its. At each post-it positions, the person was asked to keep the head in alignment with the position of the post-it. The person was allowed to move his eyes. A video of 30 seconds was captured for every angle from -90 degrees to 90 degrees yaw angle in 15-degree increments. Each video consists of images with the same yaw angle but with varying expressions and emotions, for example with eyes open, while talking, smiling. showing teeth, eyes looking at various points but maintaining the head at the same yaw position, etc. This video was converted into images using the Python Image Library (PIL). Capturing a video was easier compared to capturing images at various expressions separately.

Figure 4. An example of a frontal pose with a yaw angle of 0 degrees

Figure 5. An example of a non-frontal pose with a yaw angle of -90 degrees

The examples of the frontal pose at 0-degree yaw angle and non-frontal pose at -90 degrees yaw angle are shown in Figure 4 and Figure 5 respectively. The training and testing images were randomly selected from the images generated from the video. The images captured were of size 1920x1080 pixels. The images were captured for 13 poses from -90 degrees yaw angle to +90 degrees yaw angle in 15-degree increments, i.e. -90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, and 90 degrees. Our intuition is that in a known face database like a state database containing photos for every person in the state, images captured for the need for state identification cards are frontal images with a neutral pose. It is important to understand how the face recognition system behaves when the lawmakers have to match a face image in the database with a non-frontal image of a person. We are trying to replicate this scenario by having a training dataset of only frontal images at a yaw angle of 0 degrees and testing dataset with poses from a yaw angle of -90 degrees to 90 degrees.

Face Detection

Face detection is the primary step in face recognition. Face detection is an application of object detection in computer vision. An object detection algorithm detects objects in an image whereas a face detection considers a face as the object to be detected.

The face detection model locates a face in the image and the located face is then converted into a face feature vector and later sent to the face recognition model to identify the detected face. Face detection plays an important role in face recognition. If the face detector cannot detect faces in the image, then we cannot recognize the face. For example, Haar cascade classifiers cannot detect non-frontal faces in an image and we cannot recognize the non-frontal face in an image. Hence, it is important to select a suitable face detector model to ensure face recognition for poses in the range of -90 to 90 degrees yaw. Some of the commonly used face detector models are Haar cascade classifier, OpenCV Deep Neural network face detector (OpenCVDNN), Multi-task Cascaded Convolution Networks (MTCNN), and Histogram Of Gradients - dlib. In our research, we are working with Haar cascade face classifier, MTCNN, and OpenCVDNN face detector. In this section, we conduct a study on various face detectors and how it performs on non-frontal poses.

Haar Cascade classifier. Haar cascade classifier was proposed by Paul Viola and Michael Jones in [18]. Haar cascade classifier was the first real-time face detection model that offered high detection accuracy. The classifier was trained with an Adaboost learning algorithm on face images and non-face images. The classifier uses Haar features to extract facial features from the image. Haar feature value for each feature is calculated as the difference between the sum of pixels in a white rectangle from the sum of pixels in a black rectangle. Examples of Haar feature representation are shown in Figure 6. The paper states [18] that there are almost 160000 haar features in a 24x24 image window. To calculate the difference between the sum of black pixels from the sum of white pixels for 160000 Haar features is tedious. The researchers proposed an integral image to calculate

Haar feature value faster. An integral image calculates Haar feature value from just 4 pixels. If the accumulated Haar feature value is within the threshold limits set by the classifier, then that Haar feature is considered as a feature of a face. A number of Haar features contribute to the selection of a face. Out of 160000 features in a 24x24 window, only some Haar features would be relevant to face. The most relevant features are selected by Adaboost training. Weak classifiers with Adaboost learning would select the most important Haar features to classify faces. The classifier is known as weak because one Haar feature alone cannot classify a face. The set of weak classifiers reduced 160000 features to 6000 features after Adaboost learning in [18]. Searching each image window for 60000 features of a face is a time-consuming process.

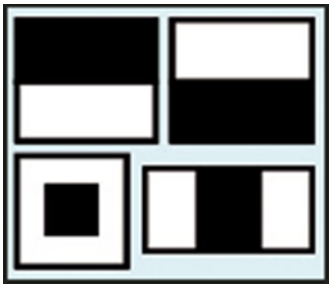


Figure 6. Examples of Haar-like features

The researchers proposed a Cascade of classifiers that checks 6000 features through different stages to make the face detection faster. The research [18] used 38 stages where 1, 10, 25, 25, 50 features were checked in the first five stages. If a window fails at the first stage then we discard it and the window which passes through all the stages of the cascade classifier contains a face region in it. Similarly, the cascade classifier is applied at different image windows successively. Finally detecting the entire face. The flow diagram of a Cascade classifier is shown in Figure 7.

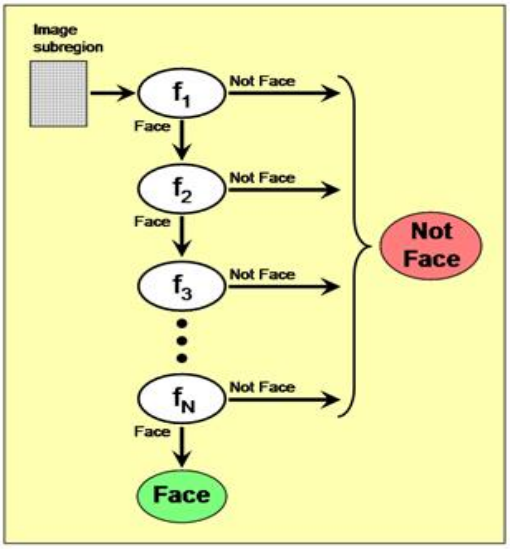


Figure 7. Flow diagram of a Cascade classifier [19]

OpenCV is a computer vision library and OpenCV has a pre-trained Haar cascade classifier for face detection. The pre-trained classifier for frontal face detection is available as an XML file 'haarcascade_frontalface_default.xml'. The first step of the OpenCV Haar cascade face detection is converting a color image to grayscale. Scale factor and minimum neighbors are parameters that control the accuracy of the Haar cascade classifier in the OpenCV library. We have used a scale factor of 1.2 and the number of neighbors as 5. The face detection is performed with the help of a function 'detectMultiscale'. This function returns the coordinates of the face detected. Using the pre-trained face classifier helps us to directly test on face image than going through the hassle of training the model for detecting faces. The disadvantages of the Haar Cascade Classifier are that it does not work well in non-frontal poses and has a high number of false positives.

Multi-task Cascaded Convolution Neural Networks (MTCNN). The MTCNN is a deep learning-based face detector that offers the state of art results in face detection. MTCNN proposed by Kaipeng Zhang et al. in [20] is one of the most popular deep

learning-based face detection models. MTCNN can also detect facial landmarks such as eyes and mouth. The MTCNN is a three-stage cascaded convolutional neural network. The MTCNN is made of 3 convolution networks namely Proposal Network (P-Network), Refine Network (R-Network), and Output Network (O-network). The first stage is a P-Network which proposes candidates face regions in an image. The proposed face region is given as input to the R-Network. The R-Network filters the bounding boxes in the face region. Finally, the bounding box regions are fed to the O-Network. The O-Network proposes facial landmarks in the face detected. The MTCNN is known as Multitask cascaded convolution network because it performs three tasks mainly face classification, bounding box regression, and facial landmark localization.

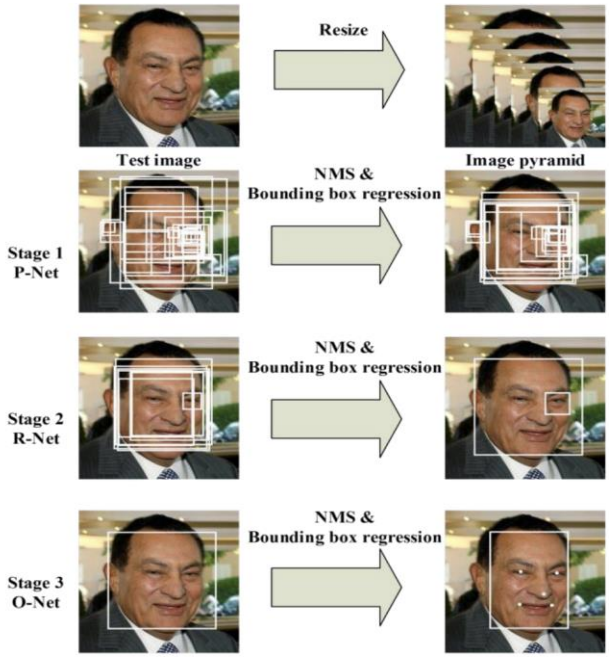


Figure 8. The pipeline of the MTCNN face detector [21]

The advantage of the MTCNN face detector is that it can detect non-frontal poses better compared to the Haar cascade classifiers. Another advantage of the MTCNN is that it comes with facial landmark detection, which is helpful for head pose estimation.

MTCNN architecture is complex to implement. So, we can make use of open-source implementations of the MTCNN face detector. The MTCNN face detector can be installed as a Python library 'MTCNN'. The MTCNN library contains the pre-trained model of the face detector. In Python, an instance of the MTCNN network can be created by calling the 'MTCNN()' constructor [21]. Once an instance of MTCNN is created, we can call the function 'detect_faces' to detect faces in an image. The function returns a dictionary containing the coordinates of the face detected, the confidence of the face detected, and coordinates of the location of eyes, nose, and mouth. This is how we detect faces with the MTCNN face detector.

OpenCV Deep Neural Network (OpenCV DNN). The OpenCV DNN face detector model is based on a Single-shot Multibox detector and uses ResNet-10 architecture [22]. This model is available in OpenCV 3.3 version. OpenCV has provided TensorFlow and Caffe implementation of the model. This model is the best compared to the Haar cascade classifier and MTCNN face detector because it can handle extreme pose variations and can detect faces in different scales.

Feature Extraction

Once we detect a face, the next step is to extract features. Before feature extraction, it is important to do some data preprocessing. Firstly, we resize all the detected faces to 500x500 pixels. After resizing, we apply standardization. Standardization of the detected faces is achieved through the 'StandardScaler' function in Scikit-learn. 'StandardScaler' standardizes features by removing the mean and scaling to unit variance [23]. In standardization, we find the mean of each feature and the standard deviation of each feature in the training data. Then we standardize the training data by

subtracting the sample mean of each feature from the sample feature and then divide the feature difference by standard deviation. The test data is then transformed according to the standardization of training data. Machine learning models will behave badly if they are fed data which is not standardized. Most machine learning models make predictions based on the distance calculated between actual output and predicted output. The reason there is a higher distance between the actual and predicted value is just that they were measured on different scales. Therefore, it is essential to standardize the data before feeding to your machine learning model.

Feature extraction plays a significant role in machine learning models. The accuracy of the algorithm depends on the method of feature extraction. In this research, we use Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Local Binary Pattern Histogram (LBPH) for feature extraction.

Principal Component Analysis (PCA). The feature of an image is the most important information of the image. The computer stores images as pixel values in the range of 0-255. If the pixel intensity is 0, then it has no information and if the pixel intensity is near to 255, it is white and contains information. The detected faces are of size 500x500 pixels and contain 250000 feature values where each pixel intensities are considered as features. If we have 100 training images, each image will have 250000 feature values, and then it would be computationally expensive. This is when we use dimensionality reduction. The Principal Component Analysis is a dimensionality reduction technique that extracts the features in such a way that it finds the direction of maximum variance in high dimensional data and projects on to a new subspace with equal or fewer dimensions than the original one. Principal components of the new

subspace are orthogonal to each other as illustrated in Figure 9. The PCA is performed with the help of the 'PCA' function in the Scikit-learn. PCA is sensitive to outliers therefore it is important to standardize the data before performing PCA dimensionality reduction.

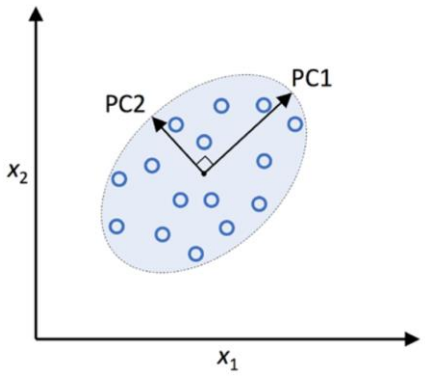


Figure 9. PCA feature representation [24]

PCA is an unsupervised method i.e. it does not require any class labels during the training stage. The important thing to remember is that PCA considers the entire training dataset and calculates k eigenfaces for the entire training dataset instead of calculating PCA for each image in the training set separately. Steps of PCA are

1. Standardize the d dimensional dataset where d is the number of features
2. Construct the covariance matrix
3. Obtaining eigenvalues and eigenvectors of the covariance matrix
4. Sort the eigenvalues in descending order to rank the eigenvectors
5. Select k eigenvectors which correspond to k largest eigenvalues, where k is the dimensionality of the new feature space
6. Construct a projection matrix W from the top k eigenvectors
7. Transform the d dimensional input dataset X using the projection matrix W to obtain the new k dimensional feature space [24]

Linear Discriminant Analysis (LDA). Linear Discriminant Analysis (LDA) is another method of feature extraction. LDA finds the feature subspace that optimizes the class separability. LDA is a supervised method and hence uses the class labels for feature extraction. Both PCA and LDA are good feature extractors. The concept of LDA for a two-class problem is shown in Figure 10. Class 1 samples are represented by circles and class 2 samples are represented by crosses.

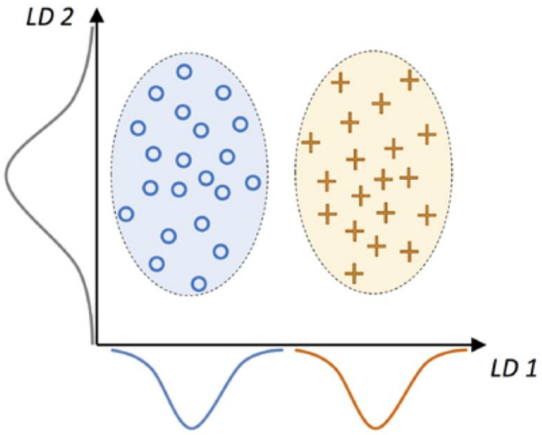


Figure 10. Linear Discriminant Analysis for a 2-class problem [23]

LDA is more useful when the classification is a multi-class classification problem than a binary classification problem. Similar to PCA, LDA is also calculated on the entire training dataset and it requires the data to be standardized before applying LDA. In LDA, we can have a maximum of c-1 linear discriminants where c is the number of classes, i.e. for a four-class classification problem, the maximum number of linear discriminants one can obtain is three. The steps of performing LDA are:

1. Standardize the d-dimensional dataset
2. Compute the d dimensional mean vector for each class
3. Construct the between-class scatter matrix S_B and within-class scatter matrix
4. Compute the eigenvectors and corresponding eigenvalues of the matrix $S_W^{-1}S_B$

5. Sort the eigenvalues in decreasing order to rank the corresponding eigenvectors
6. Choose the k eigenvectors that correspond to the k largest eigenvalues to construct a dxk dimensional transformation matrix W
7. Project the samples onto the new feature subspace using transformation matrix W

Local Binary Pattern Histogram (LBPH). Local Binary Pattern (LBP) descriptors compute the local representations of texture by comparing each pixel with its surrounding neighborhood of pixels. For each pixel in the grayscale image, a neighborhood of size r surrounding the central pixel is selected. LBP value is calculated for the central pixel as shown in Figure 11 and stored in the output 2D array with the same width and height as the input image [25]. Similarly, the LBP value is calculated for all pixels. Finally, we create an LBP representation of the image. A histogram is calculated from the LBP 2D array as illustrated in Figure 12. The calculated histogram contains information about the most important features. To account for variable neighborhood size, parameters 'p' and 'r' are introduced. Parameter 'p' represents the number of points in the neighborhood and r represents the radius of the circle. If there are 24 points and a radius of 8, the final histogram obtained will have 24+2 features. For LBP, a histogram is calculated for every single image rather than for the entire training dataset. LBPH for each image is independent of the LBPH of another image. During the testing phase, a histogram is obtained, and the system tries to compare the histogram of the test image with the histogram of each training image. The face recognition system based on LBPH compares two histograms and returns the image with the closest histogram. Depending on the system, the system also returns the name of the matched histogram.

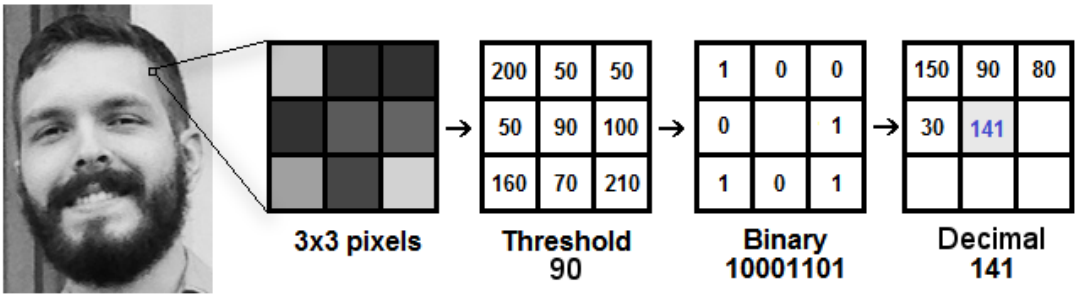


Figure 11. Applying LBP operator on an image [26]

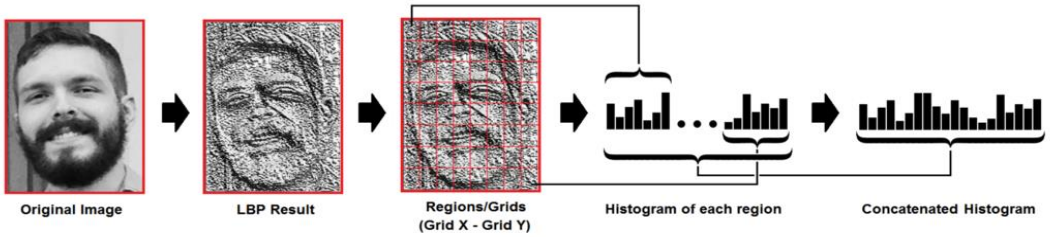


Figure 12. Creation of Local Binary Pattern Histogram from an image [25]

Face Recognition Model

The dataset for face recognition is divided into 'Train' and 'Test' dataset. The face detector model detects faces. The detected faces are resized into 500x500 pixels before passing it to the face recognition model. The face recognition model is a machine learning classifier that recognizes faces into Face ID classes. Here, we have used 3 classes such as 'FaceID1', 'FaceID2', and 'Unknown' class. The flow diagram of the face recognition model is shown in Figure 13. In the training phase, the extracted features are fed to a machine learning model. The model is then trained with the extracted face features. Ten percent of the training data is used for validation. We also perform hyperparameter tuning and retrain the model to improve the performance of the classifier. Once the machine learning model is optimized, the model is ready for test data. For the testing phase, the process repeats like before, the only difference is that instead of classifier training and hyperparameter tuning stages, the model directly predicts the test

image as ‘Face ID1’, ‘FaceID2’, or ‘Unknown face’.

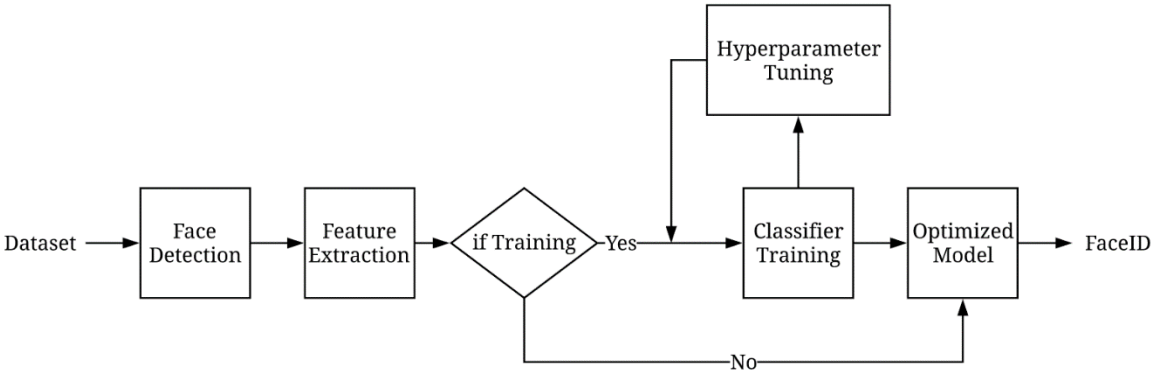


Figure 13. Block diagram of the face recognition model

Classifier selection and training. There are several classifications approaches available for machine learning such as Logistic Regression, Support Vector Machine, Decision Tree Classifier, Random Forest Classifier, and K Nearest Neighbors (KNN) classifier. Most face recognition approaches use the KNN algorithm with Euclidean distance and SVM for face recognition. In this research, training, and testing of the model was performed using SVM and KNN. The optimal model will be chosen based on performance.

SVM is a machine learning classifier that maximizes the margin between classes. The margin is defined as the distance between the decision boundary and the training samples that are closest to this decision boundary. The performance of SVM classification depends on the ‘C’ parameter. C parameter controls the penalty for misclassification. Large values of ‘C’ mean a large penalty for misclassification whereas, with small values of ‘C’, the penalty for misclassification would be less. The ‘C’ parameter of the SVM model controls the width of the margin and therefore tunes the bias-variance trade-off. The advantage of SVM is that it can work for both linear and non-linear classification. SVM can perform non-linear classification with the help of a

kernel [23]. Kernelized SVM creates non-linear combinations of the original features to project them to a higher dimensional space through a mapping function where the features are linearly separable. This separates two classes utilizing a linear hyperplane which becomes a non-linear decision boundary if we project it back to the original feature space.

After selecting the model, the model is trained with the help of the 'fit' method in Scikit-learn. Fit method loops over all individual samples in the training set and updates the weights according to the classifier rule. In this case, supervised training is used i.e. the model is trained with both data features and the class labels. During the training stage, class labels are predicted, and the training error is calculated. The learning of the model is dependent on the cost function. In the case of SVM, learning is dependent on the objective function which is the maximization of the margin whereas in KNN the features are learned by memorizing the training data set itself without depending on the objective function. The KNN selects the sample nearest to the training sample based on Euclidean distance. If the neighbors have a similar distance, then the classifier would choose the class label that comes first in the training dataset. The K parameter denotes the minimum number of nearest neighbors to include in the majority of the voting process. If k=1, then the predictions are not stable.

Hyperparameter tuning. After training the model, the performance of the model on unseen data has to be tested. This can be performed by cross-validation. Cross-validation helps to find the best model for the face classification problem. In this research, K fold cross-validation with 15 folds is used to evaluate the model. K fold cross-validation randomly splits the training dataset into K folds without replacement,

where K-1 folds are used for model training, and 1-fold is used for performance evaluation. The average scores are calculated out of the K folds. K fold cross-validation finds the optimal hyperparameters that provide a satisfying generalization performance. The optimal hyperparameters are determined by performing a Grid Search on the training data with K number of folds. After finding the optimal hyperparameter values, the model is retrained on the complete training dataset and the final performance is obtained using the unseen test dataset.

Optimized model. The best model is determined after retraining the model with optimal hyperparameters. The model will be tested on unseen data. We test the face recognition system for 13 poses from a yaw angle of -90 degrees to 90 degrees in 15-degree increments. The performance of the model will be evaluated based on the performance metrics such as accuracy, precision, recall, and F1 score. The prediction accuracy is the sum of correct predictions to the total number of predictions. Precision is a metric that helps to determine what proportion of positive identifications was actually correct. Recall indicates what proportion of actual positives was identified correctly. Often a combination of precision and recall is used and is called an F1 score.

IV. PROPOSED METHODOLOGY-POSE

Improvement of Face Recognition by Skin Segmentation

Skin is an important feature of a face image and the processing of skin is faster than other features. Skin color is different for every person and the variation in skin color is due to the difference in brightness of the image and not the chrominance. The distribution of skin color is clustered in a small portion of color space. Therefore, we can segment the image by skin segmentation. Skin segmentation separates skin pixels from non-skin pixels. Skin segmentation removes the background information of the image and hence reduces computation complexity. Therefore, skin segmentation can be considered as a preprocessing stage of face detection to reduce computational complexity. Skin segmentation is performed by choosing a proper color space and selecting a skin color model.

Color space. Skin color is sensitive to illumination, ethnicity, camera characteristics, etc. [27]. Hence it is important to choose an appropriate color space model that is robust to variations in illumination and ethnicity. Color space is a mathematical model that represents color information using three or four different color components. RGB, HSV, YCbCr are different color spaces available for skin detection. Studies on various color spaces [3] indicate that YCbCr is the best color space for skin detection. The separation of chrominance and luminance components makes YCbCr standout for skin detection. Luminance is stored in Y; Chrominance information is stored in Cb and Cr. YCbCr is used for skin detection in [26] and [28]. YCbCr can be calculated from RGB as shown below

$$Y=0.299R+0.287G+0.11B \quad (1)$$

$$Cr = R-Y \quad (2)$$

$$Cb = B-Y \quad (3)$$

HSV is another color space that works well for skin segmentation. RGB components of the color of an object in a digital image are correlated with the amount of light hitting the object and therefore image description is difficult. Whereas in HSV, image descriptions are more relevant [29]. A skin detection model in HSV color space is proposed in [30]. Hue varies from 0 to 1.0, from red through yellow, green, cyan, blue, magenta, and back to red [31]. As saturation varies from 0 to 1.0, color varies from shades of gray to fully saturated, and as V values vary from 0 to 1, the brightness of colors increases. The hue component in HSV is in the range of 0 to 360 degrees.

Skin color model. After choosing a proper color space, we have to specify the skin color model that segments skin and non-skin pixels. Different methods such as explicit skin-color space thresholding, histogram model, Gaussian classifiers, elliptical boundary model MLP classifier, Maximum entropy classifier, etc. have been proposed for skin segmentation [27]. Here we use skin color thresholding for selecting the skin pixels. The color space converted image is given some threshold range so that the values within the range would be turned to white and can be considered as skin pixels. The pixels that fall out of range are considered as background information or non-skin pixels would change to black. As a result, a binary image with skin pixels in white and non-skin pixels in the black color is formed.

Choosing an appropriate threshold value for skin color will affect the performance of skin segmentation. People have different skin tones; White people have lighter skin

tones and African American and Asian people have darker skin tones. The threshold value should be selected in such a way that it matches the skin color distribution of the person. Otherwise, skin segmentation for one skin color will not work for another skin color.

Binary morphological operations. The resultant binary image after thresholding will have more false-positive skin regions in the image [30]. To eliminate the false positives of skin detection, we pass the binary image through an elliptical structural kernel that performs a number of iterations of dilation and erosion. We can adjust the kernel size to get a clearly segmented image. Dilation adds pixels to the object boundaries whereas erosion removes pixels on object boundaries. The number of pixels added or removed is determined by the size and shape of the elliptical structural kernel.

Figure 14. Skin segmented image

After morphological operations, the output image is smoothed with a Gaussian blur to remove noise. Finally, the input image is multiplied with the masked binary image to obtain a skin segmented image. The skin segmented image will only contain the skin pixels and the background pixels will be in black. An example of a skin segmented image is shown in Figure 14. Skin segmentation reduces the resolution of the image and makes the resultant image easier for computations.

Proposed Methodology

The flow diagram of the pose improved face recognition is shown in Figure 15. The output of face detection is shown in Figure 16. In the proposed methodology, we first perform skin segmentation in HSV color space by giving threshold values according to the skin tone of the person. The skin segmented image is converted to grayscale before applying face detection. The skin segmentation removes the non-skin pixels from the image and the face detection is applied only on the skin pixels in the image.

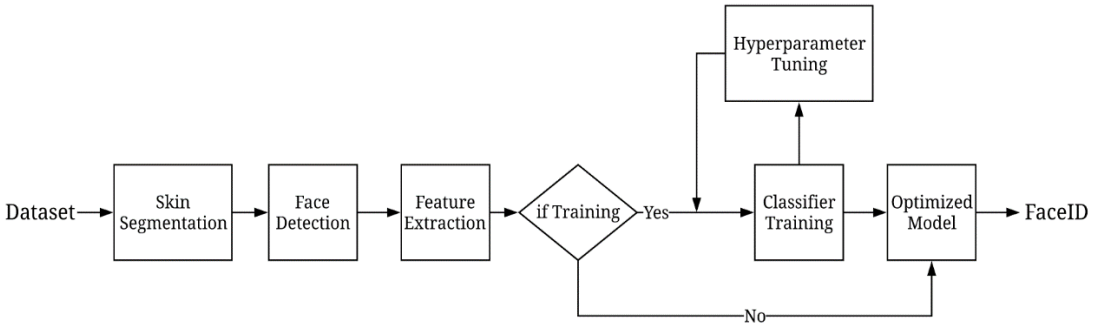


Figure 15. Proposed methodology

Figure 16. The detected face on a skin segmented image

Once the face is detected, we extract face features with the help of PCA. The PCA features extracted from the skin segmented face is used to train the classifier. We also perform hyperparameter tuning to determine the optimal parameters of the machine

learning model. Once the model is optimized, the model is ready for testing. For testing, first, we separate skin pixels in the image and then we apply face detection. PCA features are extracted from the detected face and are fed to the machine learning model to predict the FaceID. This is how our proposed methodology works. The proposed method just adds a preprocessing stage to the normal workflow of the face recognition model. Experimental results show that the proposed methodology significantly improves the face recognition accuracy for non-frontal poses.

Improvement of Face Recognition through Frontal View Synthesis

In this section, we propose another methodology to improve face recognition by generating a frontal view from a non-frontal image. We aim to recognize a non-frontal face by comparing the non-frontal face with a known frontal face database. Our model will be trained on frontal faces and will be tested on both frontal and non-frontal faces. We assume that we manually identify a pose as frontal or non-frontal before testing. If the face is frontal, then we use the model discussed in Figure 13 for face recognition. If the testing face is non-frontal we will use the model proposed in Figure 17.

The flow diagram for training the face recognition model is illustrated in Figure 18. Our model is trained on frontal faces. The faces will be first detected by the face detector and the features are extracted using Principal Component Analysis (PCA). The classifier will be trained on the extracted features from the training data. After training, the model is optimized with the help of hyperparameter tuning. To find the optimal parameters of the model, the optimization of the model is performed through Grid search. Once we obtain the optimal parameters of the model, the model is ready for testing.

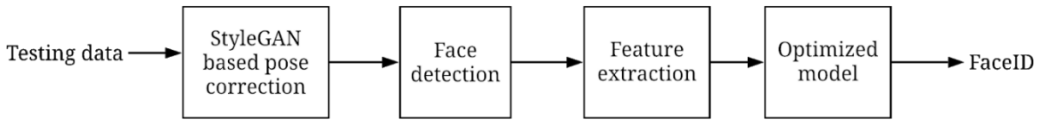


Figure 17. Face recognition with pose correction-testing phase

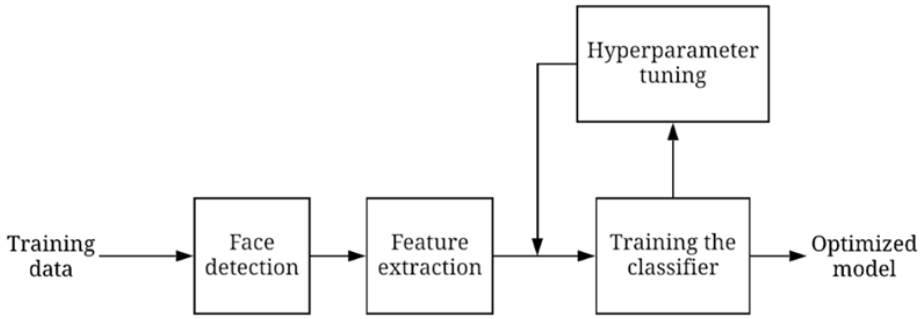


Figure 18. Face recognition with pose correction-training phase

When we have a non-frontal image as a test image, we feed the non-frontal image to a Style Transfer Generative Adversarial Network (StyleGAN). The StyleGAN architecture generates a synthetic image that looks exactly like the actual image. We then apply pose corrections on the StyleGAN generated image. The generation of a frontal pose from a non-frontal pose with StyleGAN will be explained in detail later. The StyleGAN architecture generates a frontal view and the generated frontal view is given as input to the face detector. Once the face is detected, the most important features are extracted through PCA. The PCA features extracted from the non-frontal face will be given as input to the optimized model. The optimized model now predicts the FaceID of the non-frontal face.

Generative Adversarial Network (GAN) architecture. A deep neural network is a type of machine learning network whose connections resemble the neural connections that exist in the human brain. Some of the commonly used deep neural

networks for image classification are Convolutional neural networks, Deconvolution network, and Generative Adversarial Network (GAN). StyleGAN is a type of GAN. GAN is an unsupervised deep learning network that automatically discovers the underlying structure of input data and creates new examples that could have been drawn from the original dataset [32]. GAN consists of two neural networks, a generator, and a discriminator. The generator generates image from a random latent vector. The Discriminator discriminates whether the generated image is a real image or fake image. A generator is a deconvolution network that can generate images from a latent vector whereas a discriminator is a convolution network that generates feature vectors from the input image. A random noise vector, mostly Gaussian distribution, is given as input to the generator. The generator generates images from this latent vector. The generated image is then fed to a discriminator. A discriminator is initially trained with real images so that it can discriminate between real and fake images. When the generator feeds the fake image to the discriminator, the discriminator would classify the image as a fake image and will give feedback to the generator to update its latent vector in such a way that the image generated looks like a real image. When the generator fools the discriminator, the generator is rewarded, and the discriminator is penalized, and model parameters are updated. The discriminator and generator compete with each other in such a way that the generator tries to generate a more realistic image, and the discriminator tries to discriminate it as a fake image. After sufficient training there comes a stage where the discriminator can no longer say whether the image generated is real or fake. An example of GAN architecture is shown in Figure 19. Some of the applications of GAN are creating super-resolution images, creating art, image to image translation, etc. The

drawbacks with traditional GAN are that we do not have control over the properties or style of the image generated. For example, if we have to make small changes in the generated image, like changing the hair color, trying to change the latent representation by a little would have changed the color of eyes or changed the shape of the nose resulting in a completely different image. This process where we cannot separate features from the latent representation is known as feature entanglement.

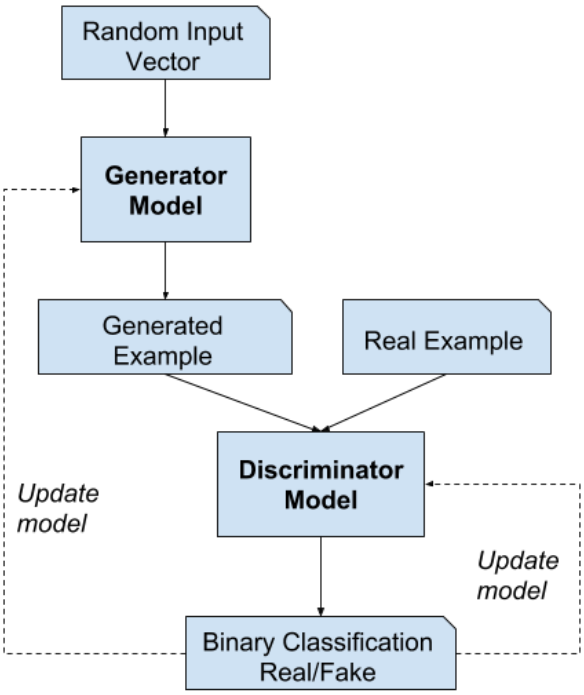


Figure 19. Example of GAN architecture [31]

Style Transfer Generative Adversarial Network (StyleGAN) architecture. StyleGAN is an extension to the GAN architecture developed to overcome the problem of feature disentanglement faced by GAN models. The most important property of StyleGAN is the ability to control the style of the image at different levels of detail. StyleGAN overcomes the problem of feature entanglement through an additional mapping network that maps the input latent vector into an intermediate latent vector.

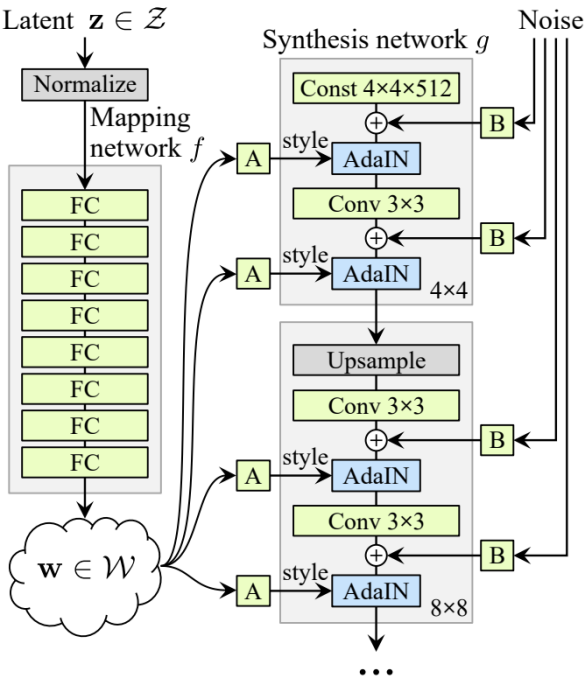


Figure 20. A StyleGAN generator model architecture [32]

The architecture of the StyleGAN generator model is illustrated in Figure 20. In GAN, a random noise vector is fed to the generator network whereas in StyleGAN, the input latent vector is mapped to an intermediate latent vector and this intermediate latent vector is fed to the generator network. Another major difference of StyleGAN from GAN is that in StyleGAN we inject the latent vector back to the generator through Adaptive Instance Normalization (AdaIN) and in GAN, the input latent vector will not be used further after the initial stage. Since adaptive instance normalization induces changes in style in every convolution layer, there is no need to use a random noise latent vector as the input of the generative model. The generator model of StyleGAN uses a learned constant the input of 4x4x512 tensor. Noise is injected pixel-wise at each AdaIN layer to provide stochastic variations in the image generated. The StyleGAN learns progressively from a lower resolution. Firstly, the model starts with generating 4x4 pixels images and when the model is stable, then it progresses to learn and generate 8x8 pixels images. This

process repeats until the model generates 1024x1024 pixels. The use of different style vectors at different points of the synthesis network gives control over the resultant image at different levels of detail [33]. The blocks of layers in the synthesis network at lower resolutions such as 4x4 pixels and 8x8 pixels control high-level styles such as pose, gender, age, etc. The layers of 16x16 pixels and 32x32 pixels control facial features and hairstyles and blocks from 64x64 to 1024x1024 control color schemes and fine details.

StyleGAN-based pose correction. StyleGAN architecture is used to generate images. We have a query image, and we aim to generate synthetic images that look exactly like the query image. Firstly, we look for faces in the image and crop the face region in the image. Then we align the faces by centering the nose and making the eyes horizontal. The aligned images are then fed to a pre-trained Resnet encoder to generate an estimate of the latent representation of the actual image. The Resnet generated latent vector will be given as the input to the pre-trained StyleGAN. The pre-trained StyleGAN [34] can generate a face image from the estimated latent vector. Now the problem is that the generated face image would not look like the actual query image. To make our generated image look like the query image, we use a pre-trained VGG16 network to extract face features from the query image and the generated image. The L2- loss in semantic VGG space is calculated as the difference between the feature vectors. We aim to minimize the distance between the feature vectors and to minimize the loss. We optimize the model by updating the initial latent vector estimate of the query image. The latent representations will be further optimized by stochastic gradient descent. The model converges when the generated image looks exactly like the query image i.e. when L2-loss in the semantic ‘VGG’ space between generated image feature vectors and the query

image feature vector is the minimum. Optimization is performed only for the latent representation we want to obtain. Now we have the actual latent representations, which when given to a StyleGAN will generate an image that looks exactly like the query image. The generation of the optimal latent vector from a query image is shown in Figure 21. The flow diagram for the generation of the optimal latent vector is shown in Figure 22.

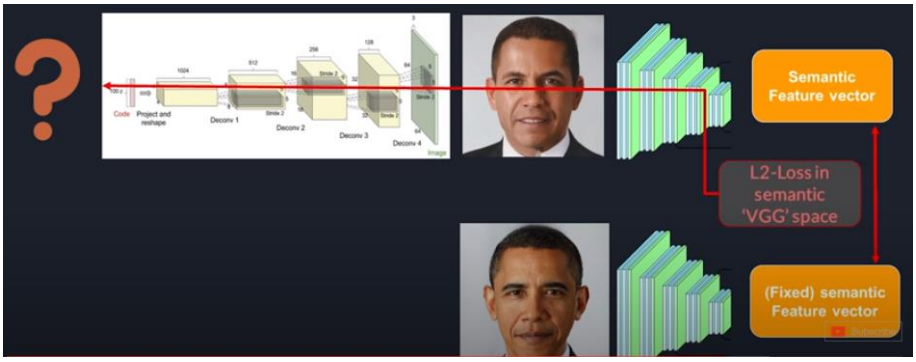


Figure 21. Generation of the optimal latent vector

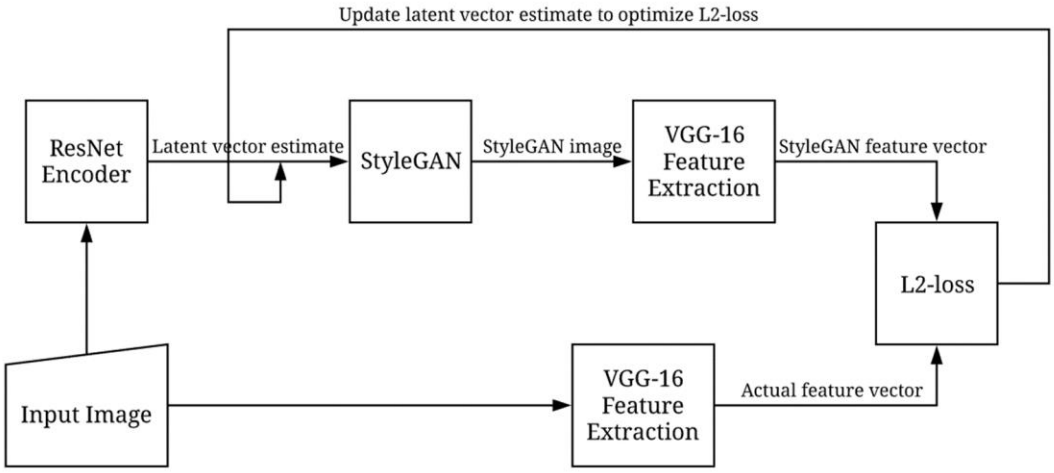


Figure 22. Flow diagram for the generation of the optimal latent vector

Once the latent vector is optimized, the latent vector can be transformed according to our wish. We find the direction of head pose yaw in the latent space and can

move the optimized latent vector in the direction of head pose yaw in the latent space and can transform it back to the image using the generator. We use a pre-trained head pose yaw latent direction available in [35] to transform the optimized latent vector to a latent vector in head pose yaw direction. The optimal latent vector is moved in the pre-trained directions of the head pose yaw. The pre-trained model was trained in the Flickr-Faces-HQ dataset to obtain the latent directions of the head pose yaw. The new latent vector will be the sum of the product of latent direction and coefficient and optimal latent vector. The coefficients indicate the direction in which we move the optimal latent vector. There are positive and negative coefficients. For example, in the head pose yaw latent directions, a coefficient of -1 means the movement of the face towards the right, +1 means the movement of the face towards the left, and 0 means the actual pose. We adjust the coefficients from -4 to +4 to generate frontal faces from non-frontal faces.

V. EFFECT OF SKIN COLOR ON FACE RECOGNITION

Skin color plays an important role in face recognition. Dark-skinned faces are one of the major challenges faced by the current face recognition system. The reasons for low face recognition accuracy for dark skin faces are less diverse training dataset and optical perspective of black color. Creating a diverse dataset could not solve the problem with dark skin. Researchers state that black color reflects light in different wavelengths compared to white faces, which makes it difficult for the current face recognition technology to correctly classify dark faces. In this section, we explain the creation of the 'Celeb-Skin' dataset for finding the impact of skin color on face recognition. We also explain the machine learning model created to find the effect of dark skin on face recognition. Our study found that dark-skinned Asians have a higher misclassification rate when compared to African American and White faces. The results of the initial study will be discussed in the results section.

Creation of the 'Celeb-Skin' dataset

The Celeb-Skin dataset contains images of 11 male celebrities. The images were collected from the web. The faces in the dataset are frontal. The dataset contains a total of 480 images in the dataset. Female celebrity faces were not collected because the face recognition accuracy of females was less compared to males. One of the main reasons that females have less accuracy than male faces is because of changes in hairstyle, jewelry, makeup, etc. There were other variations in the images of female celebrities other than skin color. We do not want our face recognition model to be tested with more than one variable parameter. We aim to find the effect of skin color on face recognition. So, we tried to discard other features like pose, gender, etc. We choose celebrities in

various skin tones and different races. We chose 4 celebrities from African American and White races, and 3 celebrities from Asian and 1 Asian person who is not a celebrity. Our dataset, therefore, contains an equal distribution of African American, Asian, and White faces. The classes of the Celeb-skin dataset are shown in Figure 23. The images shown in Figure 23 were collected from [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46].

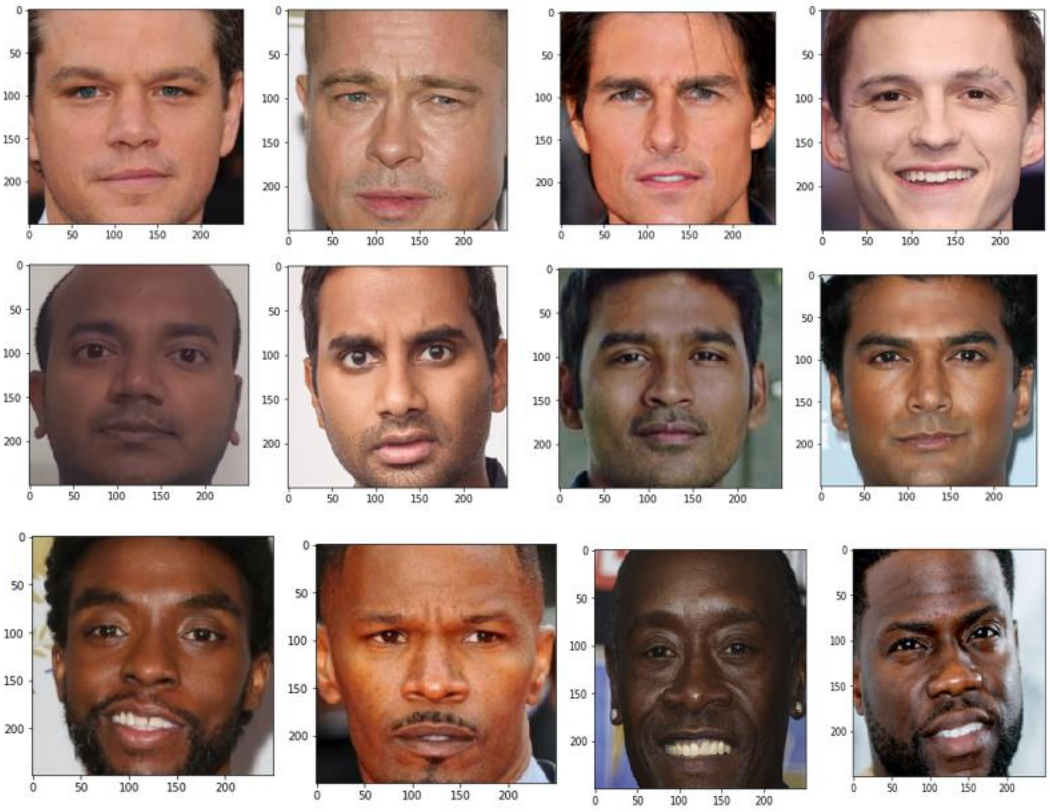


Figure 23. Example of classes in the Celeb-Skin dataset

Study on the Effect of Skin Color on Face Recognition

In this section, we discuss the development of machine learning models to find the effect of skin on face recognition. The initial results will be explained later in the results section. The flow diagram for the machine learning model to identify the effect of skin color on face recognition is shown in Figure 24. We use the 'Celeb-Skin' dataset to study the effect of skin color on face recognition.

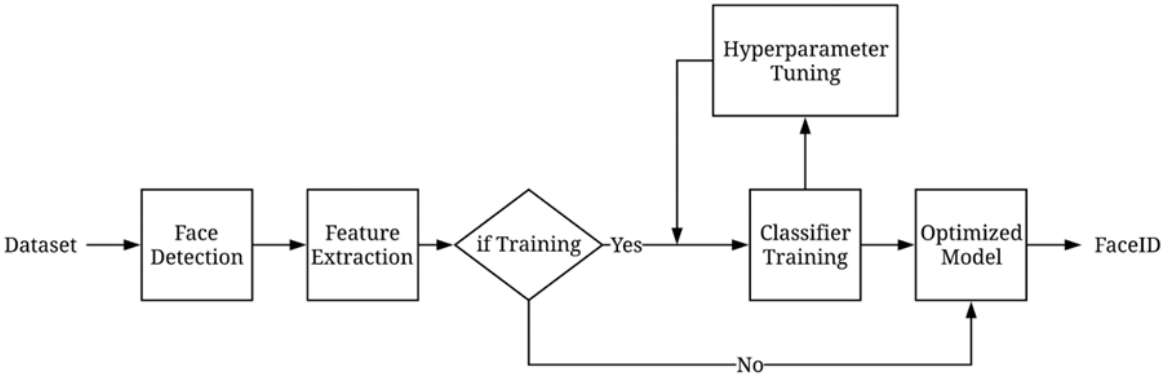


Figure 24. Flow diagram of the face recognition model

During the training phase, images are converted to grayscale, and faces are detected with the help of the Haar Cascade Classifier. We use a Haar cascade classifier because the dataset only contains frontal faces. Once the faces are detected, the detected faces are resized into 500x500 pixels. The features are extracted from the detected face using PCA and LDA. Once the features are extracted, the machine learning classifier, here we use a support vector machine, is trained on the PCA or LDA features. The PCA reduces the dimension from the 250000 features to a number of PCA components. Choosing the optimal number of PCA components would improve the accuracy of the model to a great extent. The model is optimized by hyperparameter tuning performed through grid search. Optimal hyperparameters would lead to the development of an optimized model that is ready to test unseen data.

During the testing phase, we perform operations such as converting to grayscale, resizing images to 500x500 pixels, feature extraction and finally the optimized model predicts the FaceID based on the PCA features extracted. The training data and test data must have the same dimensions. Test data should be transformed according to training data. After testing, we found that face recognition accuracy of dark Asians is less than white and African American faces.

VI. PROPOSED METHODOLOGY-SKIN

In this section, we propose the improvement of face recognition accuracy of dark skin faces. From our initial testing on the Celeb-Skin dataset, we found that dark Asians have the least recognition accuracy and are often misclassified with African American faces. We improve the accuracy of dark-skinned faces by specifically creating a dataset according to the skin tone. We create different training datasets for people in different ranges of skin tone. The classes whose skin tone (RGB values) falls in the same range falls is considered as one category. The skin tone range is determined by extracting the RGB values of the dominant skin tone. The flow diagram for creating separate datasets according to skin tone is shown in Figure 25.

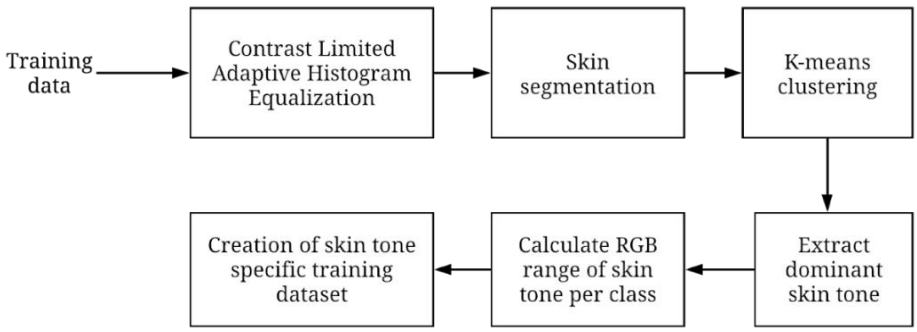


Figure 25. Creation of different training dataset based on skin tone

The Celeb-Skin dataset contains images collected from the web. The images have different illumination conditions, and the skin color of the person is dependent on the lighting conditions in which the image is captured [3]. To reduce the effect of illumination on skin tone, we perform Contrast Limited Adaptive Histogram Equalization (CLAHE). Skin segmentation is performed on the illumination corrected image to separate skin pixels from the background pixels. The skin segmented image contains only the skin portion of the face image. K-means clustering is performed to cluster the skin

tones of the image. The cluster gives information about the RGB pixel values of the skin tone in that cluster. Similarly, we obtain the RGB values of all the images in the same class and calculate the range of RGB values for that class. Based on the range of RGB values obtained, the entire training dataset is divided into 4 groups namely white, brown, dark brown, and black.

Contrast Limited Adaptive Histogram Equalization (CLAHE)

Histogram equalization is an image preprocessing step performed to improve the contrast of images. Histogram equalization improves the contrast of images by spreading out the most frequent intensity values. Adaptive histogram equalization (AHE) solved the global contrast improvement problem in histogram equalization by computing several histograms for different sessions of image. Adaptive histograms use these histograms to redistribute the lightness values of the image. AHE improves the local contrast of the image and enhances the definitions of edges in each region of an image. CLAHE was developed to prevent the overamplification of noise in adaptive histogram equalization.

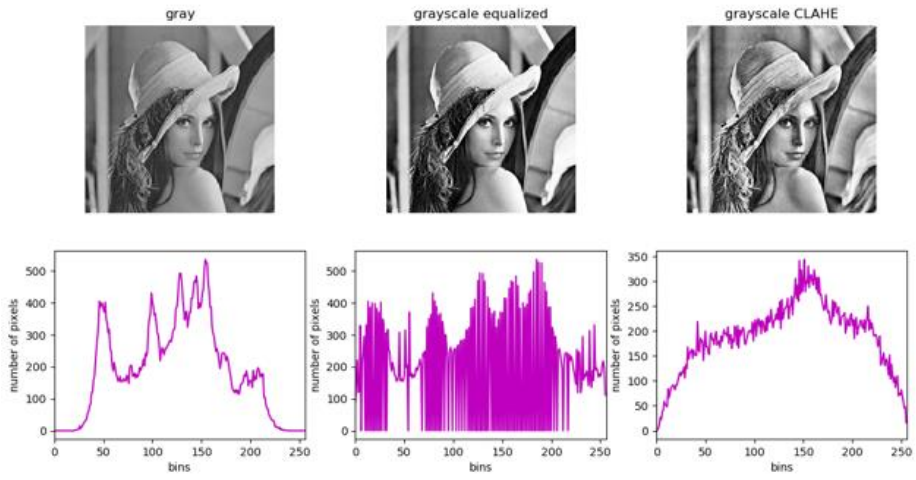
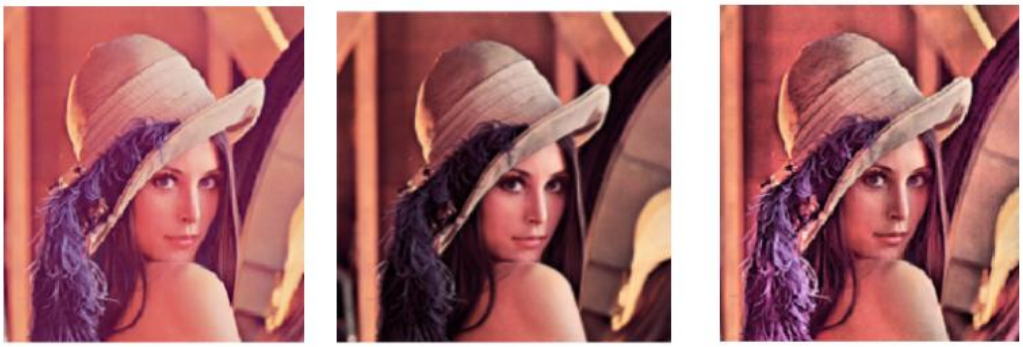


Figure 26. Histogram representation of AHE and CLAHE [35]



(a) Actual image (b) Histogram equalized image (c) ClAHE corrected image

Figure 27. Examples of histogram equalized and CLAHE corrected image

Histogram representation of histogram equalization and contrast limited histogram equalization is shown in Figure 26. An example of how an image changes with histogram equalization is shown in Figure 27. CLAHE offers higher performance when compared to Adaptive histogram equalization for image processing applications.

The Celeb-Skin dataset consists of images collected from the web. These images were captured under different lighting conditions and there is a variation in illumination. The skin tone of a person depends on the lighting conditions in which the image is captured. If there is much light when the image is captured, even a dark Asian person would have similar RGB values to a white person. We observed that RGB values of the same person vary significantly over different images. When we extract the RGB values, the range of RGB values for white and dark Asian almost fall in the same range. Computers finding that dark Asian has skin tone equivalent to a white person, due to the illumination conditions, is not good. To reduce the effect of illuminations on skin tone we used CLAHE. CLAHE is performed on the Value (V) component of HSV color space. Only the Value component of HSV space contributes to luminance or brightness, so we equalize the histograms in the value channel. CLAHE cannot be performed on RGB

because the luminance component cannot be separated in RGB color space. After equalizing the value component of the image, equalized value components are merged with hue and saturation components of the image resulting in a CLAHE corrected image. The hue and saturation values of the original image are kept intact and only the value component of the image changes after CLAHE. After performing CLAHE, the classes were grouped based on the range of RGB values.

Skin Segmentation

Skin segmentation is performed on the illumination corrected image. Skin pixels are separated from non-skin pixels in the image by applying thresholding in HSV color space. The pixel values within the threshold range would turn to white and all other pixels are turned to black. Thresholding results in a binary image. Choosing an optimal threshold value is important to properly segment the skin region. Threshold values for white and dark skin tones are different. The optimal value of the HSV range is figured out through trial and error. The optimal threshold value can separate skin and non-skin pixels for both black and white faces. The binary image is then smoothed with a Gaussian blur to remove noise. Finally, the input image is multiplied with the binary image to obtain the skin segmented image. The skin segmented image only contains the skin parts in the image and all the background pixels will be black. The dominant skin tone is extracted from the skin segmented image through K- means clustering.

K-means Clustering

K-means clustering is an unsupervised learning algorithm that finds groups of data points with similar features. This group of data is known as a cluster. K-means clustering groups data into K clusters as shown in Figure 28. K-means algorithm aims to

choose centroids that minimize the within-cluster sum of squares [37]. To extract the dominant skin tone, the number of clusters k is chosen as 2. This would return the two most dominant skin tones in the skin segmented image. The steps of K-means clustering are the following:

1. Choose k number of clusters
2. Choose random k number of data points as initial centroids
3. Assign each sample to its nearest centroid
4. Create new centroids by taking the mean value of all the samples assigned to the previous centroid [37]
5. Repeat steps 3 and 4 until the difference between the old centroid and the new centroid is less than a threshold and until the centroid does not move [37]

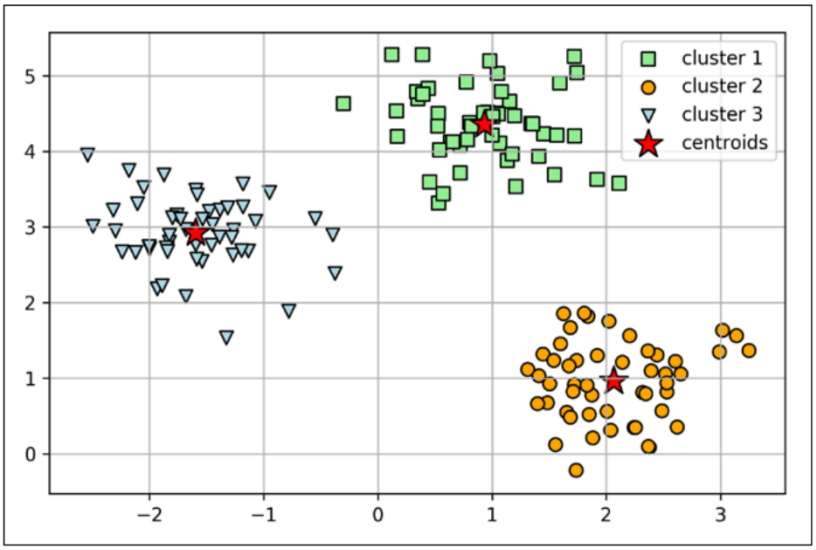


Figure 28. K-means clustering [24]

We use ‘sklearn.cluster.KMeans’ in Scikit-learn to group the skin segmented image into different clusters. The Scikit-learn model takes the number of clusters as input and predicts the cluster in which the data would belong to. We can also find the centroids

of each cluster index. While calculating the dominant skin tone, 2 clusters were used, and the model returned the cluster index and the centroid coordinates of each cluster index. The centroid coordinates of each cluster are the mean RGB values of each cluster.

Extraction of the Dominant Skin Tone

The K-means clustering algorithm groups the skin segmented image into 2 clusters and the RGB values of each cluster are obtained. To calculate the dominant skin tone, the number of data points in each cluster, and the total number of data points in both clusters are found. Then the percentage of data points in each skin cluster is calculated. The cluster with the highest percentage of data points is the dominant skin tone. The RGB values of the dominant skin tone can be obtained from the centroid estimates of the cluster with the highest percentage of data points.

Calculation of the RGB Range Per Class

The RGB values of the dominant skin tone are extracted for every image in the training dataset. We aim to extract the RGB value range of a person. For example, we have to calculate the RGB range of ‘Brad Pitt’, for that we extract RGB values of every image in the ‘Brad Pitt’ class of the training dataset. The RGB values are not constant for a person because the RGB values are dependent on the lighting conditions in which the image is captured. Therefore, we calculate the RGB range for each class in the training dataset. The range of RGB values is larger for each class because the images are captured in different lighting conditions. If the images were captured in a controlled environment with fixed lighting conditions then the range of RGB values is within ±5 pixels.

Creation of the Skin Tone Specific Training Dataset

After extracting the range of RGB values for each class in the training dataset, the classes with almost the same RGB range are arranged into one group. Based on the RGB range obtained, the training dataset was divided into 4 training datasets namely white, brown, dark brown, and black. An example of the training datasets based on a specific skin tone is shown in Figure 29.

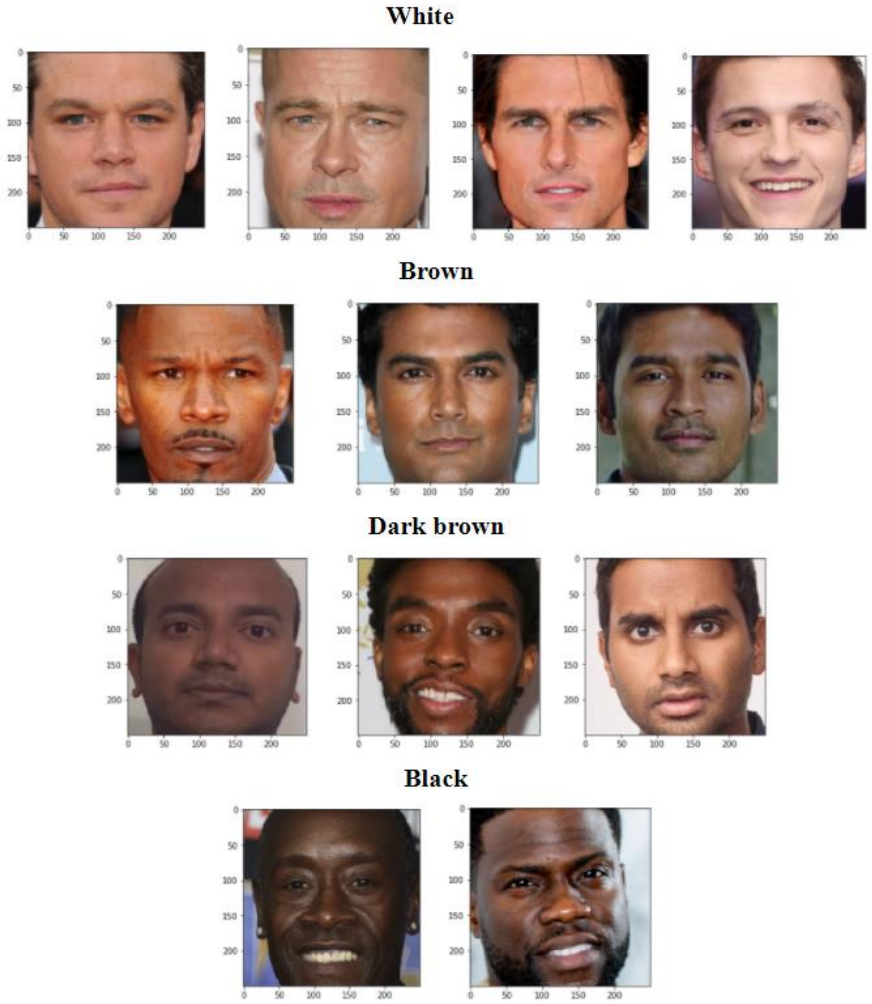


Figure 29. Training datasets specific to skin tone

When we obtained the range of RGB values for different skin tones, we found that there is an overlap in the upper range of black and lower range of dark brown.

Similarly, there is an overlap in the upper range of dark brown and the lower range of brown skin tones. In such cases, we only consider the lower ranges of each skin tone, specifically lower R channel ranges, to group the training dataset according to skin tone. For example, the range of RGB values for the black training dataset is R: 95-120 B: 60-90 and B: 40-80, and the RGB for the dark brown training dataset is R: 105-130, G: 60-100, B: 40-70. We can see that there is an overlap of R: 105-120 in both black and dark brown skin tone. So, we are only considering the lower range RGB values i.e. 95,60,40 for black and 105,60,40 for dark brown. In this way, we can separate the classes in the training dataset into 4 groups.

Face Recognition Model

The face recognition model is similar to the model discussed in Figure 24. The only difference is that instead of one single training dataset we have 4 training datasets according to skin color. We assume that the test dataset contains only the classes in the training dataset. If the test dataset contains faces that are not training datasets, then the model should predict the image as unknown and we have to match the test image with other training datasets to find the right match. Face detection is performed by the Haar cascade classifier and PCA features extracted from the faces are given to train the model. The model is optimized by hyperparameter tuning. The optimized model is tested on PCA extracted features of the test faces. We found that separately training on specific skin tone has significantly improved the accuracy of dark Asians.

VII. RESULTS AND DISCUSSION-POSE

The implementation described above was tested in Jupyter notebook using the Scikit-learn package in a PC with Intel® core™ i5-8250U. The programming was performed in Python language. Table 1 indicates the details of the test environment.

Table 1. Specifications of the test environment

Parameters	**Specifications**
Workstation	Intel ® core™ 15-8250U, 1.8 GHz (4 cores)
Development environment (IDE)	Jupyter Notebook, Spyder
Programming language	Python
Image processing libraries	OpenCV 4.0, Scikit-image
Machine learning libraries	Scikit-learn

We used 30 images per person in various expressions such as smiling, eyes open, eyes closed, talking, etc. in the frontal pose for training. We have tested for 13 poses, i.e. -90°, -75°, -60°, -45°, -30°, -15°, 0°, 15°, 30°, 45°, 60°, 75°, and 90° yaw angles. For law enforcement applications, the images in the database are usually frontal and they have to match a non-frontal profile photo with a frontal image. The same scenario was replicated here by training on frontal images only and testing on images from -90° to 90° yaw angle in 15° increments. We have considered 3 classes such as 'FaceID1', 'FaceID2', and 'Unknown'. 'FaceID1' and 'FaceID2' images are captured as part of image acquisition. For the 'Unknown' class, we took images of 30 different persons from the web for training and 12 images of different people for testing. The important thing to note here is that images in the 'Unknown' class are always frontal even for yaw angles from -90 degrees to 90 degrees. The images were captured during daytime with the

presence of daylight and daylight led bulbs with no flash. The specifications of the camera used to capture the images in the dataset are presented in Table 2.

Table 2. Specifications of the camera

Parameters	**Specifications**
Camera	Google Pixel 3
Resolution	A front camera of 12.2 MP
Video	1920x1080 pixels at 30 fps
Aspect ratio	16:9
Flash	No flash

Comparison of Face Detection Models on Non-frontal Poses

Face detection was performed with the OpenCV Haar cascade classifier, MTCNN, and OpenCV DNN. The Viola-Jones algorithm-based Haar cascade classifier could only detect faces from -60 degrees to +60 degrees yaw angle. The main drawback of the Haar cascade classifier was that it could not detect extreme non-frontal poses from ∓60° to ∓90°. We also conducted face detection using the MTCNN face detector and found that the deep learning-based face detector could detect faces from -75° to +75° yaw angle. The MTCNN face detector could not detect extreme non-frontal poses at ∓90°. We aim to test the face recognition model for yaw angles from -90° to +90°. Therefore, we used a deep learning-based face detector OpenCV DNN. OpenCV DNN was able to detect all the poses from -90° to +90° yaw angle. Hence, the OpenCV DNN face detector was further used for face recognition from -90° to +90°. The results of face recognition with OpenCV DNN will be explained later in this chapter.

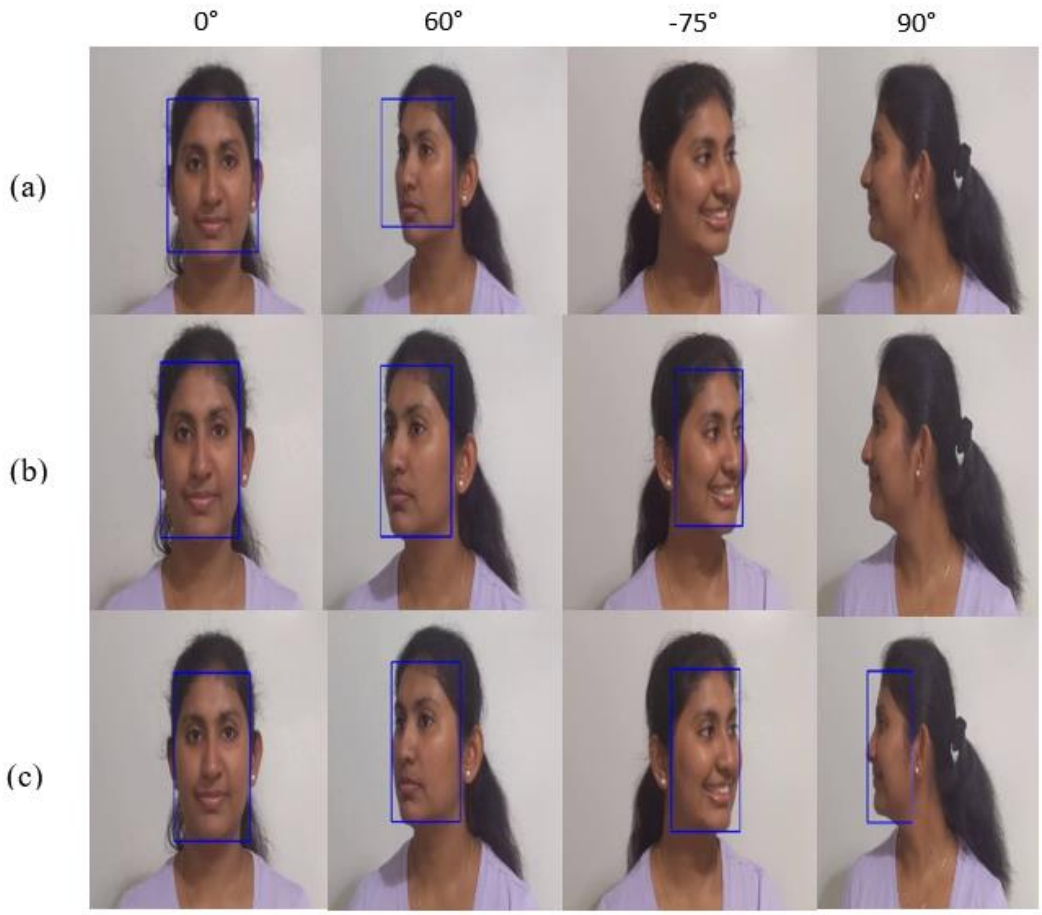


Figure 30. Comparison of performance of face detection models on non-frontal poses (a) Haar Cascade classifier (b) MTCNN (c) OpenCV DNN

The performance of Haar cascade classifier, MTCNN, and OpenCV DNN for non-frontal poses is illustrated in Figure 30. The blue bounding box indicates the face detected in the image. If the image does not have a bounding box, that means that the face is not detected. From Fig. 28 we can understand that OpenCV DNN was able to detect faces at 0°, 60°, -75°, 90° yaw angles. However, MTCNN could not detect faces at 90°and Haar Cascade classifier could not detect faces with yaw angles 75° and 90°.

PCA Feature Extraction

After the face detection, PCA features are extracted from the training images. Choosing the appropriate number of PCA components played an important role in the performance of face recognition. In our case, we found that 20 numbers of PCA components give us the highest performance. Example eigenfaces extracted from the training dataset is shown in Figure 31. The first four Eigenfaces contributed to 80 percent of the total variance in the training dataset.

Figure 31. The top 8 Eigenfaces

Face Recognition Model

The face recognition model was developed as shown in Figure 13. We used various classifiers such as LinearSVC and KNN classifier for creating the model. From our results, we found that LinearSVC is the best classifier for face recognition. The results of comparing various classifiers are shown in Table 3. We have trained our model with frontal images and tested the model for 13 poses. OpenCV DNN face detector was used for face detection. Hyperparameter tuning was performed to find the optimized model. The optimal parameters that produced the best validation accuracy are shown in Table 4. The initial results of the face recognition model for various poses are displayed in Table 5.

Table 3. Comparison of classifiers

Classifier	**Training accuracy (%)**	**Testing accuracy (%)**
Linear SVC	100	94.4
K-Nearest Neighbors (KNN)	95.5	97.2

Table 4. Optimal hyperparameters

Parameter	**Optimal value**
Number of PCA components	n_components=20
Linear SVC ‘C’	C=0.0001

Table 5. Comparison of performance-pose

Pose (Degrees)	**Initial (%)**	**Skin segmentation (%)**	**StyleGAN based pose correction (%)**
-90	61.1	74.3	-
-75	86.1	94.2	94.4
-60	89	94.2	94.4
-45	94.4	94.2	94.4
-30	94.4	94.2	94.4
-15	94.4	94.2	94.4
0	94.4	94.2	94.4
15	94.4	94.2	94.4
30	94.4	94.2	94.4
45	86.1	94.2	94.4
60	61.1	94.2	94.4
75	61.1	94.2	94.4
90	61.1	83	-

From the initial results presented in Table 5, we can tell that face recognition

accuracy decreases significantly from 45 degrees to 90 degrees and -60 degrees to -90 degrees. From -45 to 30 degrees, the face recognition accuracy is almost the same. The face recognition accuracy for extreme non-frontal poses from -60 degrees to -90 degrees and +45 to +90 degrees was improved by skin segmentation. The hyperparameters that gave the best accuracy for non-frontal poses are listed in Table 6.

Table 6. Optimal hyperparameters for skin segmentation-based pose improvement

Parameter	**Optimal value**
Number of PCA components	n_components=20
Linear SVC 'C'	C=0.0001
HSV threshold	Lower threshold: [0,30,0] Upper threshold: [55,255,255]
Number of erosions	1
Number of dilations	2

Table 7. Performance improvement for extreme non-frontal poses

Pose (Degrees)	**Skin segmentation (%)**	**StyleGAN based pose correction (%)**
-90	21	-
-75	9	9.6
-60	5	6
45	9	9.6
60	54.1	54.5
75	54.1	54.5
90	35.8	-

The results in Table 5 shows that performing skin segmentation in HSV color

space before face detection has improved the face recognition accuracy at poses such as -90, -75, -60, 45, 60, 75, and 90 degrees considerably. From Table 7, we can see that skin segmentation has increased the face recognition accuracy for faces at 60 degrees and 75 degrees by 54%.

The improvement in face recognition accuracy for non-frontal poses by StyleGAN based pose correction is presented in Table 5. In this method, we have created a frontal view from a non-frontal image. From our experimentation, we found that this technique only works for poses from -75 degrees to +75 degrees yaw angle. StyleGAN based pose correction does not work for -90 degrees and 90 degrees yaw angles. Firstly, a ResNet encoder determined an estimate of the latent vector from the actual image. The obtained latent vector was fed to the StyleGAN to generate images. The L2 loss in the VGG space was optimized by stochastic gradient descent over several iterations until the generated image looked like the actual image. From Figure 32, we can observe that the StyleGAN generated image and the actual image looks the same. However, we aim to create a frontal view of the StyleGAN generated image. The output of the StyleGAN generator is the optimal latent vector, that generated an image that looks exactly like the actual image. The obtained latent vector was then adjusted in the direction of the head pose yaw direction. The new head pose yaw direction adjusted latent vector was fed to the StyleGAN to create a frontal pose from a non-frontal pose. From our experimentation, we found that the coefficient value controls the direction of the head pose yaw. The StyleGAN generated images for various coefficients of head pose direction is shown in Figure 33. The image shown in Figure 33 is an image at 75 degrees yaw angle. To convert a non-frontal image to a frontal image, we used a coefficient of -4.

The coefficient -4 generated image is the most frontal, and this frontal image will be used for testing.

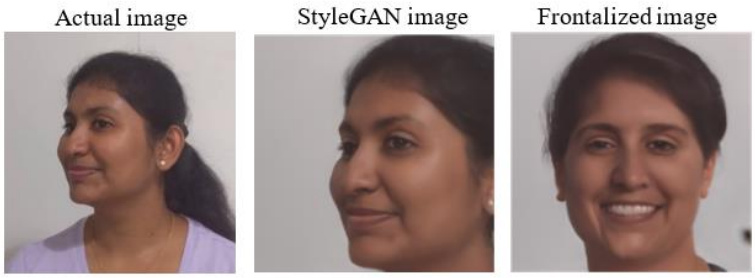


Figure 32. StyleGAN based pose correction

Figure 33. Variation of the pose with coefficients

Table 8. Specifications of StyleGAN based pose correction

Parameters	**Specifications**
GPU	Nvidia Tesla T4, 8.1 TFLOPS
Learning rate	0.02
L1_penalty	0.3
Number of iterations	400

The model was trained on real frontal images and tested on synthetic pose corrected images. From the initial results, it was found that face recognition accuracy at -15 degrees and +15 degrees yaw angle is the same. Pose correction was only performed for poses such as 30, -30, 45, -45, 60, -60, 75, and -75 degrees yaw angle. The hyperparameters of the StyleGAN ResNet encoder used to generate StyleGAN image are

shown in Table 8. The StyleGAN based pose correction required a GPU to perform computations. Even though we used a pre-trained StyleGAN model, generating a latent vector with the Resnet encoder, and optimizing the loss through stochastic gradient descent required almost 400 iterations. We used Google Colaboratory (Google Colab) GPU to perform the computations. The specifications of the GPU used are listed in Table 8. It took 13 minutes to generate 4 images through the StyleGAN even with the GPU. Therefore, generating a synthetic image that looks exactly like the actual image and converting the non-frontal pose to a frontal pose is a compute-intensive process.

Comparison of the Proposed Pose Improved Face Recognition for Non-frontal Poses

When comparing skin segmentation-based pose improvement and StyleGAN based pose correction, pose improvement with skin segmentation has more merits when compared to StyleGAN-based pose correction. In pose improvement with skin segmentation, we improved poses from -60 degrees to -90 degrees and 45 degrees to 90 degrees yaw angle as shown in Figure 34 and Figure 35 whereas, in StyleGAN-based pose correction, we improved face recognition accuracy for poses from -60 degrees to -75 degrees and 45 degrees to 75 degrees. The StyleGAN-based correction was not able to generate images for ± 90 degrees yaw angle. A comparison of improvement in face recognition accuracy for extreme non-frontal faces is shown in Figure 36.

Another advantage of skin segmentation-based pose improvement is that it is not compute-intensive. A GPU is not required to perform skin segmentation. On the other hand, StyleGAN based pose correction is compute-intensive and takes around 13 minutes for generating 4 images. A demerit of skin segmentation-based pose improvement is that skin segmentation only works well in images captured under controlled conditions. When

there is a lot of background or a background similar to skin color, skin segmentation does not work properly.

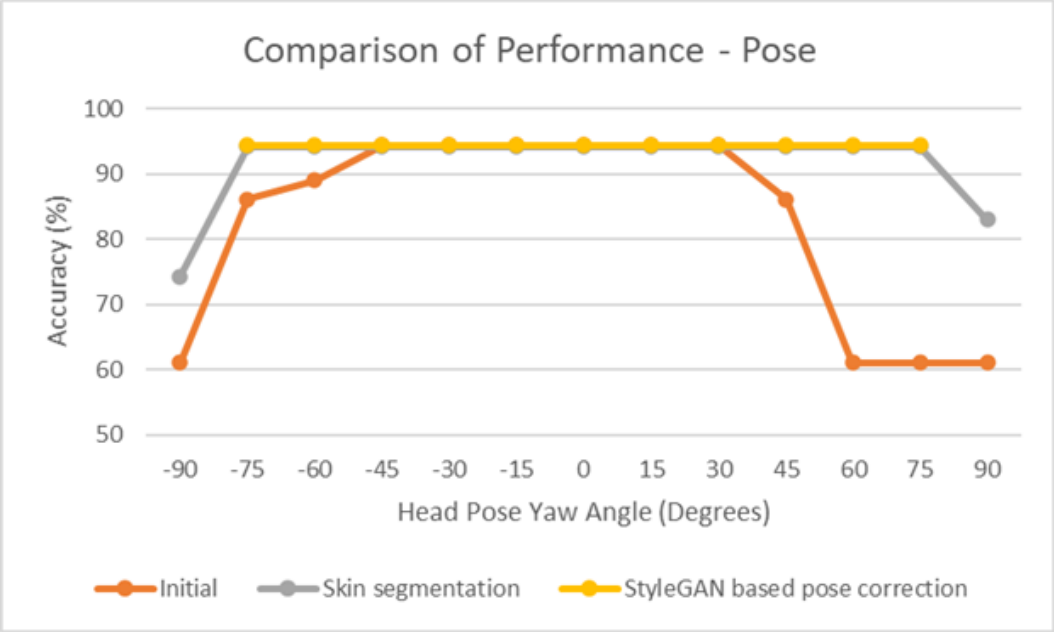


Figure 34. Comparison of performance-pose

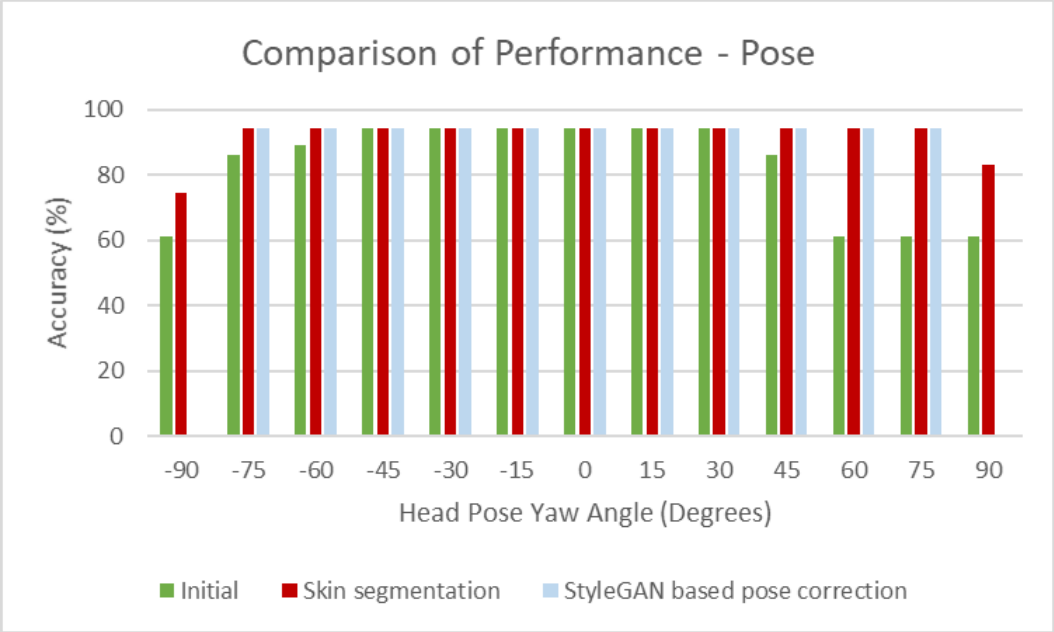


Figure 35. Performance comparison-pose

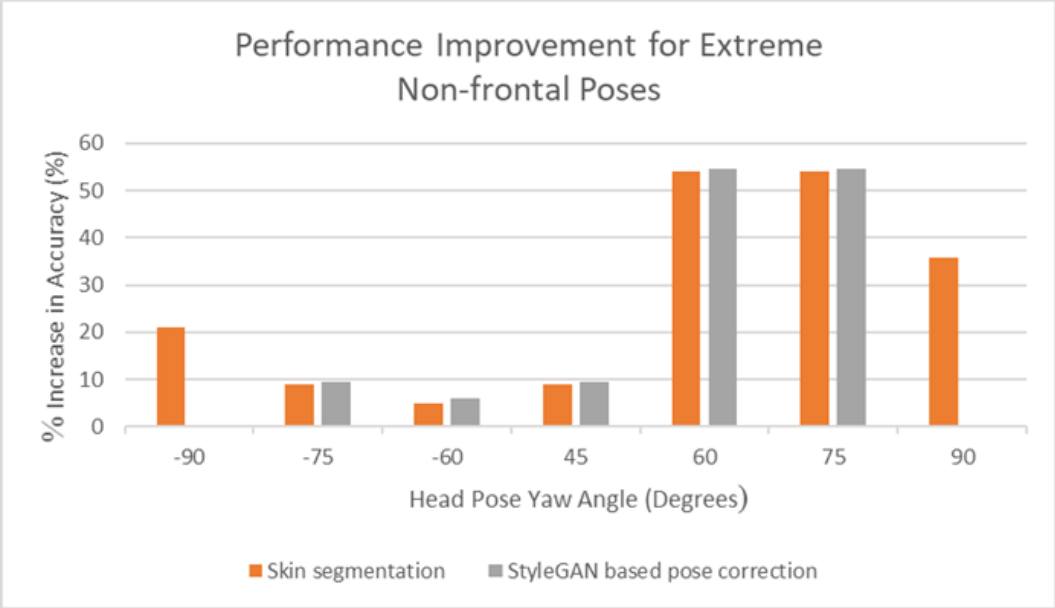


Figure 36. Performance improvement for extreme non-frontal poses

VIII. RESULTS AND DISCUSSION-SKIN

This section presents the improvement of face recognition accuracy for dark-skinned faces. Firstly, we present the initial results obtained after testing the Celeb-Skin dataset. Secondly, we present the improvement of face recognition accuracy of dark skin faces by creating different training datasets specific to the skin color. Finally, we compare the initial results with the results skin invariant face recognition system. We also compare the results of our proposed methodology with the method discussed in [2]. Previous studies on skin color bias on face recognition technology suggested that creating a diverse training dataset and creating a dataset specific to a particular race would improve the face recognition accuracy for dark skin faces. Table 9 presents the details of the test environment.

Table 9. Test environment specifications for skin

Parameters	**Specifications**
Workstation	Intel ® core™ 15-8250U, 1.8 GHz (4 cores)
Development environment (IDE)	Jupyter Notebook, Spyder
Programming language	Python
Image processing libraries	OpenCV 4.0
Machine learning libraries	Scikit-learn

Initial Results on the Effect of Skin Tone on Face Recognition

The Celeb-Skin dataset consists of 480 frontal images of male celebrities of races such as White, African American, and Asian. The dataset is diverse and is uniformly distributed with an equal number of images for every class. There are 4 white, 4 African American, and 4 Asian classes. Each class consists of 20 training images and 20 testing

images. Therefore, our initial face recognition model was trained on 240 images of 12 classes and tested on 240 images of 12 classes. The images were collected from the web and there is variation in illumination, scale, and resolution among images.

The first step of face recognition is face detection and we used a Haar cascade classifier in OpenCV for face detection because the Celeb-Skin dataset only contains images in frontal poses. The parameters of the Haar Cascade classifier played an important role in face detection. Since the images were of different resolutions and various scales, we used a scale factor of 1.1. The optimal hyperparameters of the Haar cascade classifier that detects most faces are shown in Table 10.

Table 10. Specifications of the Haar Cascade classifier

Parameter	**Optimal value**
Color of image	gray
Scale factor	‘scaleFactor’ =1.1
Minimum number of neighbors	‘minNeighbors’=5
Minimum size of face detected	‘minSize’=(150,150)

After face detection, the most important features were extracted through PCA. In order to extract features with PCA, all faces should be of equal size, Therefore, the detected faces were resized to 500x500 pixels. Before applying PCA, the pixel intensities were standardized to optimize the performance of PCA and face recognition algorithm. The number of PCA components plays an important role in face recognition accuracy. Therefore, tuning the number of PCA components for maximum accuracy is essential. For the Celeb-Skin dataset, we found that 175 PCA components gave the highest accuracy for the model. Table 11 illustrates the performance of the face recognition

model for various classifiers such as LinearSVC, K-Nearest neighbors (KNN), and Support Vector Machine (SVM) with a linear kernel. Linear SVC has given the maximum accuracy compared to other classification algorithms. Hence LinearSVC was selected as the best classifier for Face recognition. Both LinearSVC and SVM are support vector machines. The only difference is that the LinearSVC model in Scikit-learn uses a 'liblinear' solver whereas the SVM model uses 'libsvm' solver. 'Liblinear' has more control over losses and penalties when compared to 'libsvm'. The optimal hyperparameters of LinearSVC that gave the highest accuracy are presented in Table 12.

Table 11. Comparison of classification algorithms

Classifier	**Training accuracy (%)**	**Testing accuracy (%)**
Linear SVC	100	94.5
K-Nearest Neighbors (KNN)	72.5	50.4
SVM-Linear kernel	100	87.5

Table 12. Optimal hyperparameters of face recognition model-skin

Parameter	**Optimal value**
Number of PCA components	'n_components'=175
Linear SVC 'C'	'C'=0.1
Linear SVC 'dual'	'dual' =False

The LinearSVC model gave a face recognition accuracy of 94.5%. The confusion matrix indicating the true positives, true negatives are shown in Figure 37. From the confusion matrix, the face recognition accuracy for White, African American, and Asian faces are calculated. The classes 'Allan', 'Aziz', 'Dhanush' and 'Sendhil' represent Asian

faces, ‘Chadwick’,’ Doncheadle’,’ KevinHart’ and ‘JamieFox’ represents African American faces, and ‘BradPitt’, ‘MattDamon’, ‘TomCruise’ and ‘TomHolland’ represents White faces. The accuracy for races such as White, African American, and Asian are presented in Table 13. From Table 13, we can infer that face recognition accuracy for white faces is greater than the face recognition accuracy for African American and Asian races. Asian race was the most misclassified race. Therefore, we aim to improve the accuracy of the Asian and African American race. The accuracy of dark-skinned faces in Asian and African American races was improved by creating a specific training dataset according to the skin tone of a person.

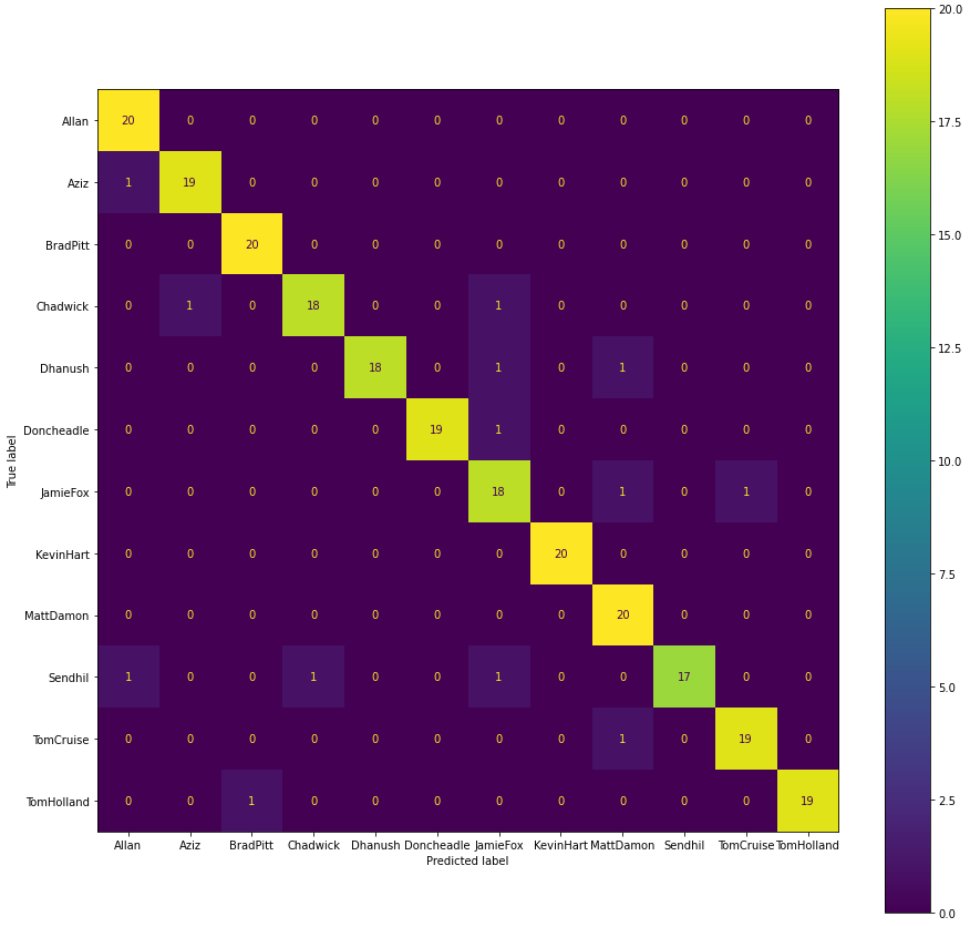


Figure 37. Confusion matrix

Table 13. Face recognition accuracy for various races

Race	**Recognition accuracy (%)**
White	97.5
African American	93.75
Asian	92.5

Improvement of Face Recognition Accuracy for Dark-skinned Faces

The face recognition accuracy of dark-skinned faces was improved by creating a dataset based on the skin tone of a person. The skin tone of a face image was obtained by extracting the most dominant skin tone. The skin color of a person is dependent on illumination conditions and because of that, each person will have different RGB values for different images. To reduce the variance of RGB values, a Contrast Limited Adaptive Histogram Equalization (CLAHE) was applied to each image as a preprocessing step. CLAHE was performed on HSV color space rather than in RGB color space. Histogram equalization is performed only on the value component in HSV color space. Skin segmentation was performed on the contrast limited histogram equalized image to separate skin and non-skin pixels. Figure 38 illustrates an example of a contrast limited adaptive histogram equalized image and skin segmented image.

Figure 38. Applying CLAHE and skin segmentation on an image

Color Information

```
{'cluster_index': 2,
 'color': [103.14376873990868, 71.23195202583237, 56.095679249634685],
 'color_percentage': 0.416352}

{'cluster_index': 1,
 'color': [28.267928649437337, 20.536140436032078, 16.96456740686684],
 'color_percentage': 0.394512}

{'cluster_index': 0,
 'color': [181.86372524016036, 142.1191872821567, 120.25444189407413],
 'color_percentage': 0.189136}
```

Color Bar

Figure 39. Extraction of dominant skin tone

The skin segmented image was given as input to a K-means clustering algorithm. The number of clusters was chosen as 2. Black will also be a dominant skin tone since the image is a skin segmented image,. Therefore, we have 3 skin clusters. The K-means clustering algorithm cluster the skin tones into 3 clusters and returns the mean pixel values of each skin tone. An example of the output of the K-means clustering algorithm for the image shown in Figure 38 is presented in Figure 39. From Figure 39, we can understand that there are 3 skin clusters and the centroid of each cluster represents the RGB values of each cluster. The most dominant skin tone was found by calculating the percentage of occurrence of each skin tone among three skin tones. The skin tone that has the highest occurrence percentage is the most dominant skin tone. From Figure 39, ‘cluster_index 2’ is the most dominant skin tone, containing 41.6% of total pixels and the mean RGB values of the ‘cluster_index 2’ are 103.143,71.23,56.09. Figure 38 is an example image of class’ ‘Doncheadle’. The RGB values of all the images in the training dataset of class ‘Doncheadle’ are calculated. The RGB values are not constant because

RGB values depend on the lighting conditions in which the image was captured. Hence the range of RGB for class ‘Doncheadle’ was calculated and we found the range of RGB values for ‘Doncheadle’ as R: 95-120, G: 60-90, B: 40-80. Similarly, we have obtained a range of RGB values for all classes in the training dataset. The range of RGB values for each class is shown in Table 14.

Table 14. RGB values for each class

Class	**Race**	**R**	**G**	**B**
BradPitt	White	190-230	140-180	120-160
MattDamon	White	190-220	140-170	120-160
TomCruise	White	190-230	140-180	120-170
TomHolland	White	190-230	140-180	120-170
Chadwick	African American	105-130	70-100	40-70
Doncheadle	African American	95-120	60-90	40-80
JamieFox	African American	115-140	70-100	45-75
KevinHart	African American	95-120	60-90	40-70
Allan	Asian	105-115	60-80	40-60
Aziz	Asian	105-130	60-90	40-70
Dhanush	Asian	115-140	80-110	45-90
Sendhil	Asian	115-140	70-100	50-80

Based on the RGB values, the classes whose RGB values fall in the same range were considered as one group. After grouping classes based on the RGB values, we divided the training dataset into 4 training datasets such as White, Brown, Dark Brown, and Black as shown in Table 15. After creating 4 different training datasets based on skin

tone such as white, brown, dark brown, and black, 4 machine learning models were created for each training datasets. The hyperparameters of the machine learning model for different skin tone is presented in Table 16. The improved face recognition accuracy for dark skin tones is illustrated in Table 17.

Table 15. Creating skin tone specific training dataset

Class	**Skin tone**	**R**	**G**	**B**
BradPitt	White	190-230	140-180	120-170
MattDamon				
TomCruise				
TomHolland				
JamieFox	Brown	115-140	70-100	45-90
Sendhil				
Dhanush				
Chadwick	Dark Brown	105-130	60-100	40-70
Allan				
Aziz				
Doncheadle	Black	95-120	60-90	40-80
KevinHart				

Table 16. Optimal hyperparameters for skin tone specific datasets

Dataset	**PCA components**	**Linear SVC 'C'**
White	65	0.1
Brown	55	0.1
Dark Brown	11	0.1
Black	11	0.1

Table 17. Face recognition accuracy for the proposed methodology

Race	**Recognition accuracy (%)**
White	97.5
Brown	100
Dark Brown	100
Black	100

Comparison of the Proposed Methodology with Previous Research

Previous research suggested that face recognition accuracy for a particular race can be increased by specifically training on images of that race i.e. accuracy of African American faces can be increased by training on African American faces alone. The research [2] suggested that the accuracy of African American faces has increased 2 percent by training specifically on African American faces. The reason the researchers find the improvement in accuracy while training specifically on a race is based on the idea that people in the same race can recognize better than from another race. For example, a white person can recognize white faces better than Asian faces and an Asian person can recognize Asian faces better than white faces.

In this section, we compare our proposed methodology i.e. creating a training dataset specific to skin tone with creating a training dataset specific to a race. The results of the face recognition model while training specifically on race are shown in Table 18. The hyperparameters of the face recognition model that gave the best performance are tabulated in Table 19. Table 20 illustrates the improvement of face recognition accuracy for Asian and African American faces by specifically training on that particular race. Specifically training on race means that there will be separate training datasets for each

race rather than training on a diverse dataset. Our initial results for skin were captured with diverse data, yet the accuracy for dark-skinned faces was low compared to white faces. Researchers claim that a diverse dataset could not solve the dark skin bias in face recognition technology. By specifically testing African American faces on an algorithm trained on African American faces, the recognition accuracy was improved by 6.66 percent. Similarly, testing an Asian face on an algorithm trained on Asian faces improved the accuracy of Asian faces by 5.4 percent.

Table 18. Face recognition accuracy for training based on race

Race	**Recognition accuracy (%)**
White	97.5
African American	100
Asian	97.5

Table 19. Optimal hyperparameters for training based on race

Race	**PCA components**	**Linear SVC 'C'**
White	65	0.1
African American	60	0.1
Asian	60	0.1

Table 20. Improved face recognition with training based on race

Race	**Initial (%)**	**Race-specific training dataset (%)**	**Improvement (%)**
White	97.5	97.5	0
African American	93.75	100	6.66
Asian	92.5	97.5	5.4

To compare our methodology with the method of training based on race, we have considered Asian and African American faces as 'Dark' skinned faces by calculating the average of face recognition accuracy for Asian and African American faces. Figure 40 presents a graphical representation of a comparison of our methodology with previous research. The comparison results of the proposed methodology and method of creating datasets specific to ethnicity are illustrated in Table 21. From Figure 40, we can understand that specifically training on a white skin tone or specifically training on a white race did not improve the accuracy of white faces.

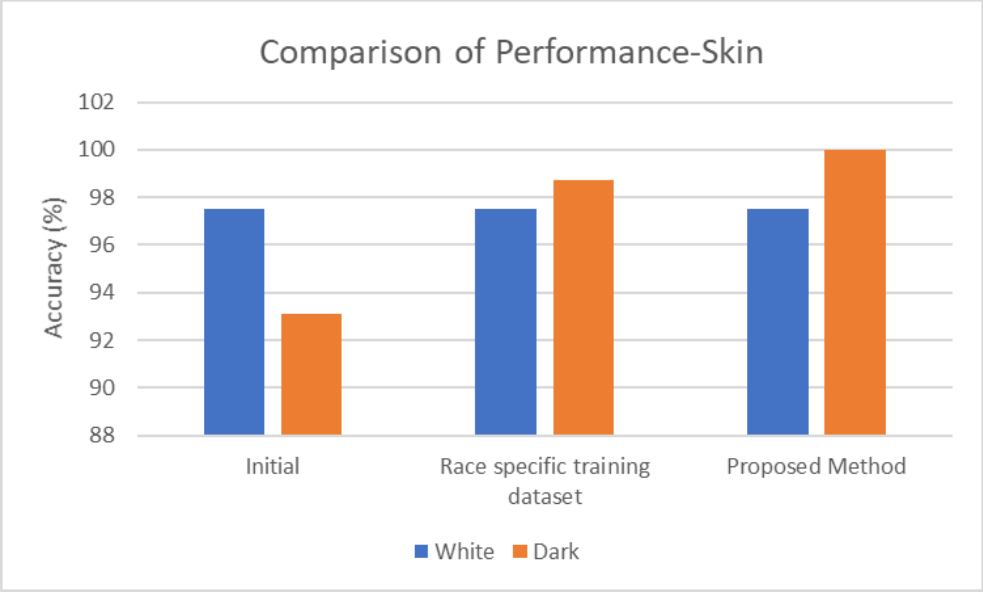


Figure 40. Comparison of performance-skin

Table 21. Comparison of performance-skin

Skin tone	**Initial (%)**	**Race-specific training dataset (%)**	**Proposed method (%)**
White	97.5	97.5	97.5
Dark	93.125	98.75	100

Table 22. Comparison of improvement in performance-skin

Skin tone	**Race-specific training dataset improvement (%)**	**Proposed method improvement (%)**
White	0	0
Dark	6.04	7.38

The comparison of improvement in face recognition accuracy of dark-skinned faces for the proposed methodology and the method based on training specifically on ethnicity is presented in Table 22. Table 22 shows that the proposed methodology improved the accuracy of dark-skinned faces by 7.38 percent whereas the method based on training on a specific race enhanced the accuracy of dark-skinned faces by 6.04 percent. Comparing both methods, our methodology offers higher performance on dark-skinned faces compared to the previous method by 1.34 percent.

Merits of the proposed method are that it offers a higher face recognition rate on dark-skinned faces compared to the method of specifically training on a particular race. However, skin color is not a reliable parameter, and the skin tone of a person varies according to the lighting conditions. Determining the appropriate training dataset based on extracted RGB values might be tricky. For instance, consider that RGB values are extracted from an image and the RGB values fall in the range of two skin tones, for example, brown and dark brown. To find out where the actual image does belong, we have to test the image on both brown and dark brown training datasets and determine the best dataset for testing the image. This problem of finding the best dataset based on skin tone can be solved if the image is captured under controlled conditions, similar to images captured for national security purposes, e.g. Driver's license photo, then the range of

RGB values of the same person would be smaller compared to images with varying illumination. The images in the Celeb-Skin dataset are mostly images of male celebrities and the images were captured with high-resolution cameras and had variations in illumination, scale, makeup, and hairstyles. This is the reason why the images of the same class in the Celeb-Skin dataset had a variation of 25-pixel intensities i.e. R: 115-140. The image captured in controlled conditions had only a small variation of 5-pixel intensities i.e. R: 105-110 over 20 images. Therefore, using images captured in controlled conditions and training specifically on the skin tone improves the face recognition accuracy for dark-skinned faces.

IX. CONCLUSION

Pose and skin color are the two major factors that affect the performance of a face recognition system. Traditional face recognition algorithms like Eigenfaces has lower accuracy for non-frontal poses. Current face recognition technology is biased to dark skin tone and puts the people of color the most vulnerable when employed for law enforcement applications. In this research, we developed a pose invariant and skin tone invariant face recognition system for surveillance and law enforcement applications. This research studied the effect of pose and skin color on face recognition separately.

From the initial results conducted to study the effect of pose on face recognition, it was found that face recognition accuracy drops significantly from 45 degrees yaw angle to 90 degrees yaw angle and -60 degrees yaw angle to -90 degrees yaw angle. The face recognition accuracy for non-frontal poses from 45 degrees to 90 degrees yaw angle and -60 degrees to -90 degrees yaw angle are improved by performing skin segmentation in HSV color space, as an image preprocessing step before applying face detection. We also improved the face recognition accuracy from 45 degrees yaw angle to 75 degrees yaw angle and -60 degrees yaw angle to -75 degrees yaw angle through StyleGAN based pose correction. In StyleGAN based pose correction, a frontal image was created from a non-frontal image through StyleGAN based pose correction. The frontalized images were used for testing while the model was trained on frontal images. The comparison of the performance of face recognition accuracies for both methods was performed and pose improvement with skin segmentation was determined as the best method to improve the face recognition accuracy for non-frontal poses.

This research also conducted a study on the effect of skin color on face

recognition. For this study, we created the Celeb-Skin dataset that contains 480 images with 20 training images per person and 20 testing images per person. The Celeb-skin dataset only contains frontal images and is a diverse dataset that mostly contains images of male celebrities with races such as White, African American, and Asian. From our initial study, we found that face recognition accuracy of the White race was greater than that of African American and Asian races. Asian race had the least recognition rate. The face recognition accuracy for dark-skinned faces was improved by creating a training dataset specific to skin tone. The dominant skin tone of each image was obtained and RGB values were extracted for each person. Based on RGB values extracted, the range of RGB values for each class was determined, and classes whose RGB values fall in the same range are grouped as one skin tone. As a result, the Celeb-skin dataset was divided into 4 training datasets, based on skin colors, such as White, Brown, Dark Brown, and Black. The face recognition for dark-skinned faces was increased by 7.38 percent after testing images on algorithms specifically trained on the same skin tone. We also compared our method of creating a training dataset specific to skin tone with the method of creating a training dataset specific to race and found that our method improved face recognition accuracy by 1.34 percent compared to that of creating race-specific training datasets.

One of the main concerns with the results accomplished from pose and skin color improved face recognition is the limited data. Therefore, the most significant future work of the research is scaling the machine learning model to include more data. In addition to training the model on more data, another area where research is required is to include male faces with facial hair and female faces for training the machine learning model. In

the future, while training on larger datasets, it is also better to use a deep learning neural network classifier that would give a robust performance for variation in poses and skin color. Apart from the future work discussed, automatic pose estimation of image and creating frontal faces when needed can be considered as another future work in the area of pose improved face recognition. For improving the performance of face recognition of dark-skinned faces, we created a Celeb-skin dataset by collecting images from the web and it was difficult to determine the correct skin tone from the images. On the other hand, creating a dataset with images captured under controlled conditions would help us to predict the skin tone of a person without difficulty. This would help in creating a better model that provides a higher accuracy on dark-skinned faces.

REFERENCES

[1] J. Buolamwini and T. Gebru, “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,” p. 15.

[2] 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 Trans. Inf. Forensics Secur.*, vol. 7, no. 6, pp. 1789–1801, Dec. 2012, doi: 10.1109/TIFS.2012.2214212.

[3] A. Elgammal, C. Muang, and D. Hu, “Skin Detection - a Short Tutorial,” p. 10.

[4] V. P. Kshirsagar, M. R. Baviskar, and M. E. Gaikwad, “Face recognition using Eigenfaces,” in *2011 3rd International Conference on Computer Research and Development*, Mar. 2011, vol. 2, pp. 302–306, doi: 10.1109/ICCRD.2011.5764137.

[5] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.

[6] X. Chai, S. Shan, X. Chen, and W. Gao, “Locally Linear Regression for Pose-Invariant Face Recognition,” *IEEE Trans. Image Process.*, vol. 16, no. 7, pp. 1716–1725, Jul. 2007, doi: 10.1109/TIP.2007.899195.

[7] R. Gross, I. Matthews, and S. Baker, “Appearance-based face recognition and light-fields,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 449–465, Apr. 2004, doi: 10.1109/TPAMI.2004.1265861.

[8] V. Blanz and T. Vetter, “Face recognition based on fitting a 3D morphable model,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1063–1074, Sep. 2003, doi: 10.1109/TPAMI.2003.1227983.

[9] A. Asthana, T. K. Marks, M. J. Jones, K. H. Tieu, and M. Rohith, “Fully automatic pose-invariant face recognition via 3D pose normalization,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 937–944, doi: 10.1109/ICCV.2011.6126336.

[10] C. Ding, C. Xu, and D. Tao, “Multi-Task Pose-Invariant Face Recognition,” *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 980–993, Mar. 2015, doi: 10.1109/TIP.2015.2390959.

[11] D. Pal, C. Bhagavatula, Y. Zheng, R. Tao, and M. Savvides, “Is Pose Really Solved? A Frontalization Study On Off-Angle Face Matching,” in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2019, pp. 2058–2067, doi: 10.1109/WACV.2019.00223.

[12] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, “Towards Large-Pose Face Frontalization in the Wild,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 4010–4019, doi: 10.1109/ICCV.2017.430.

[13] S. Zhang *et al.*, “Pose-Weighted Gan for Photorealistic Face Frontalization,” in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 2384–2388, doi: 10.1109/ICIP.2019.8803362.

[14] R. Huang, S. Zhang, T. Li, and R. He, “Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2458–2467, doi: 10.1109/ICCV.2017.267.

[15] Y. Yin, S. Jiang, J. P. Robinson, and Y. Fu, “Dual-Attention GAN for Large-Pose Face Frontalization,” *ArXiv200207227 Cs*, Feb. 2020, Accessed: Oct. 12, 2020. [Online]. Available: http://arxiv.org/abs/2002.07227.

[16] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 4396–4405, doi: 10.1109/CVPR.2019.00453.

[17] O.-O. D. Science, “The Impact of Racial Bias in Facial Recognition Software,” *Medium*, Oct. 15, 2018. https://medium.com/@ODSC/the-impact-of-racial-bias-in-facial-recognition-software-36f37113604c.

[18] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Dec. 2001, vol. 1, p. I–I, doi: 10.1109/CVPR.2001.990517.

[19] D. Hefenbrock, J. Oberg, N. T. N. Thanh, R. Kastner, and S. B. Baden, “Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs,” in *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2010, pp. 11–18, doi: 10.1109/FCCM.2010.12.

[20] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[21] J. Brownlee, “How to Perform Face Detection with Deep Learning,” *Machine Learning Mastery*, Jun. 02, 2019. https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/.

[22] "Face detection with OpenCV and deep learning - PyImageSearch." https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/.

[23] "sklearn.preprocessing.StandardScaler — scikit-learn 0.23.2 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html .

[24] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition*. Packt Publishing, 2019.

[25] Fast, optimized "for" pixel loops with OpenCV, and P.-P. says, "Local Binary Patterns with Python & OpenCV," *PyImageSearch*, Dec. 07, 2015. https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[26] K. S. do Prado, "Face Recognition: Understanding LBPH Algorithm," *Medium*, Feb. 03, 2018. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b.

[27] A. Tofighi and S. A. Monadjemi, "Face Detection and Recognition Using Skin Color and AdaBoost Algorithm Combined with Gabor Features and SVM Classifier," in *2011 International Conference on Multimedia and Signal Processing*, May 2011, vol. 1, pp. 141–145, doi: 10.1109/CMSP.2011.35.

[28] G. Xu, Y. Xiao, S. Xie, and S. Zhu, "Face detection based on skin color segmentation and AdaBoost algorithm," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Mar. 2017, pp. 1756–1760, doi: 10.1109/IAEAC.2017.8054314.

[29] "HSL and HSV," *Wikipedia*. Oct. 02, 2020, Accessed: Oct. 13, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=981510049.

[30] H. to B. a K.-A. M. D. S. in J. 5 M.-P. says, "Tutorial: Skin Detection Example using Python and OpenCV," *PyImageSearch*, Aug. 18, 2014. https://www.pyimagesearch.com/2014/08/18/skin-detection-step-step-example-using-python-opencv/.

[31] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia, "Human Skin Detection Using RGB, HSV and YCbCr Color Models," presented at the International Conference on Communication and Signal Processing 2016 (ICCASP 2016), Lonere, India, 2017, doi: 10.2991/iccasp-16.2017.51.

[32] J. Brownlee, “A Gentle Introduction to Generative Adversarial Networks (GANs),” *Machine Learning Mastery*, Jun. 16, 2019. https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

[33] J. Brownlee, “A Gentle Introduction to StyleGAN the Style Generative Adversarial Network,” *Machine Learning Mastery*, Aug. 18, 2019. https://machinelearningmastery.com/introduction-to-style-generative-adversarial-network-stylegan/.

[34] *NVlabs/stylegan*. NVIDIA Research Projects, 2020.

[35] spiorf, *spiorf/stylegan-encoder*. 2020.

[36] “Matt Damon,” *IMDb*. http://www.imdb.com/name/nm0000354/.

[37] W. to Outsons, “How to Get Brad Pitt’s Hairstyle - Every Major Haircut Included!,” *Outsons | Men’s Fashion Tips And Style Guide For 2020*. https://outsons.com/get-brad-pitts-hair-style/.

[38] “Tom Cruise,” *Biography*. https://www.biography.com/actor/tom-cruise.

[39] K. Fasanella, “Tom Holland Just Cut His Brother Harry’s Hair at Home,” *Allure*. https://www.allure.com/story/tom-holland-cuts-brothers-hair-at-home.

[40] J. B. Young, “Aziz Ansari plays the Dr. Phillips Center in the wake of this year’s #MeToo controversy,” *Orlando Weekly*. https://www.orlandoweekly.com/Blogs/archives/2018/10/19/aziz-ansari-plays-the-dr-phillips-center-in-the-wake-of-this-years-metoo-controversy.

[41] “Dhanush image | Actors images, Cute actors, Actor photo,” *Pinterest*. https://www.pinterest.com/pin/776800635706396516/.

[42] N. Andreeva and N. Andreeva, “Sendhil Ramamurthy Joins NBC’s ‘New Amsterdam’ In Recurring Role,” *Deadline*, Nov. 06, 2018. https://deadline.com/2018/11/sendhil-ramamurthy-cast-new-amsterdam-recurring-role-nbc-1202496881/.

[43] “Chadwick Boseman Told Friends to ‘Take Advantage of Every Moment’ Over Text,” *Toofab*. https://toofab.com/2020/08/29/chadwick-boseman-told-friends-take-advantage-every-moment/.

[44] “Jamie Foxx Net Worth,” *Celebrity Net Worth*, Feb. 02, 2020. https://www.celebritynetworth.com/richest-celebrities/actors/jamie-foxx-net-worth/.

[45] "Don Cheadle on Avengers: End Game Passing $1B," *American Urban Radio Networks*, Apr. 29, 2019. https://aurn.com/don-cheadle-on-avengers-end-game-passing-1b/.

[46] E. Kirkpatrick, "Kevin Hart Didn't Tell Anyone He Had Coronavirus Because of Tom Hanks," *Vanity Fair*. https://www.vanityfair.com/style/2020/08/kevin-hart-coronavirus-tom-hanks-dave-chappelle.