PERSONALIZED QUANTIFICATION OF FACIAL NORMALITY USING ARTIFICIAL INTELLIGENCE

An Undergraduate Research Scholars Thesis

by

LAYAN AL-HUNEIDI[1], SALMA ABOELMAGD[2], SARA MOHAMED[3], AND KHALID AL-EMADI[4]

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisors: Dr. Erchin Serpedin
Dr. Mohammad Shaqfeh

May 2022

Major: Electrical and Computer Engineering[1,2,3,4]

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

We, Layan Al-Huneidi[1], Salma Aboelmagd[2], Sara Mohamed[3], and Khalid Al-Emadi[4], certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with our Research Faculty Advisors prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Personalized Quantification of Facial Normality using Artificial Intelligence

Layan Al-Huneidi[1], Salma Aboelmagd[2], Sara Mohamed[3], and Khalid Al-Emadi[4]
Department of Electrical and Computer Engineering[1,2,3,4]
Texas A&M University

Research Faculty Advisor: Dr. Erchin Serpedin
Department of Electrical and Computer Engineering
Texas A&M University

Research Faculty Advisor: Dr. Mohammad Shaqfeh
Department of Electrical and Computer Engineering
Texas A&M University

While congenital facial deformities are not rare, and surgeons typically perform operations to improve these deformities, currently the success of the surgical reconstruction operations can only be “measured” subjectively by surgeons and specialists. No efficient objective mechanisms of comparing the outcomes of plastic reconstruction surgeries or the progress of different surgery techniques exist presently. The aim of this research project is to develop an efficient software application that can be used by plastic surgeons as an objective measurement tool for the success of an operation. The long-term vision is to develop a software application that is user-friendly and can be downloaded on a regular laptop and used by doctors and patients to assess the progress of their surgical reconstruction procedures. The application would work by first scanning a face before and after an operation and providing the surgeon with a normality score of the face from 0 to 3 where 3 represents normal and 0 represents extreme abnormality. A score will be given when the face is

scanned before and after surgery. The difference between those scores is what we will call the delta. A high delta value would point to a high improvement in the normality of a face post-surgery, and a low delta value would indicate a small improvement. The first chapter of the thesis represents the introduction which describes the general aspects of the project. The second chapter presents the methodology employed for building the application and the existing solutions and proposed functional model structure. The results chapter presents the process behind collecting and labeling the image database and analyzes the scores produced by the program when fed with new images from the database. Finally, the last chapter of this thesis presents the conclusions. The list of references completes this work.

ACKNOWLEDGEMENTS

Contributors

We would like to thank our faculty advisors, Dr. Erchin Serpedin and Dr. Mohammad Shaqfeh, for their guidance and support throughout the course of this research.

Thanks also go to our friends and colleagues and the department faculty and staff for making our education at Texas A&M University a great experience.

Finally, our thanks go to Mr. Abdullah Hayajneh for his patience and continued immense help.

The Haar cascade analysis tool employed for Personalized Quantification of Facial Normality Using AI was provided by Dr. Mohammad Shaqfeh and Mr. Abdullah Hayajneh.

Thank you to Philip Wang, creator of thispersondoesnotexist.com which was utilized to create our database of training samples.

All other contributions conducted for the thesis were completed by the students independently.

Funding Sources

Undergraduate research was supported by the Electrical and Computer Engineering Department at Texas A&M University at Qatar.

1. INTRODUCTION

Human beings are highly social creatures. Communicating with other people plays a major role in the life of any human being. Facial abnormalities acquired from birth, also known as congenital facial disfigurements, tend to obscure certain facial expressions which results in significant psychological and social impact on such persons [1]. As a result, the majority of those who are impacted by facial abnormalities seek surgical methods to correct the facial anomalies.

1.1 Effects of Congenital Facial Deformity

Facial expressions act as a conduit to effective communication and to aid in recognizing the needs or emotions of others. Children with major facial deformities encounter social and psychological stresses, and while some remain resilient against these pressures as they grow older, more often than not, their quality of life is negatively impacted [2]. Furthermore, it also majorly affects their families. “Lansdown (1981) stated that the birth of a congenitally disfigured child is a shock to the family system...[and] that parents experience a variety of emotions, including ‘grief, anxiety, confusion, depression, disappointment, disbelief, frustration, guilt, hurt, inadequacy, rejection, resentment, shock, stigmatization, and withdrawal.” [3]. This leads the consideration of facial reconstruction treatment, though there lies a discrepancy between the way parents envision the outcome of the treatment and that of the treatment team. From an objective point of view, one difficulty lies in determining objectively how successful the proposed reconstruction operation is. Therefore, this project’s goal is to develop an objective tool to assess the degree of facial abnormality and to measure the degree of improvement offered by a surgical procedure. The current standard for determining the success of a surgery lies majorly in the perception offered by

surgeons, which may be biased due to their perspective of what a “normal” face looks like and what standard of beauty they believe to be the default.

1.2 Proposed Solution

The main purpose of the software application to be developed in this thesis is to collect a facial image and output a score of facial normality between 0 and 3, with 0 representing the abnormal case and 3 a normal case. If the face scores above a 3, it is still considered normal. The code goes through a series of steps to process the image provided. The image is first altered such that face characteristics such as the nose, eyes, and lips are visible at specified locations. The image is then processed, and the application generates a normality score for the user. This process is then repeated on the facial image post-surgery, and a delta value which represents the improvement in the normality score can be calculated. Since there are many techniques to conduct facial reconstructive surgery, the delta value (final abnormality score minus the initial abnormality score) can be used to measure the rate of success of a surgical operation. Such an objective metric will help to select the best techniques for facial reconstructive surgery. While a similar program was developed previously, the downside of that program is that it takes around 20 minutes to run on a supercomputer which is inefficient, not cost effective and not practical since it cannot be implemented as a user-friendly application. The program we plan on developing is anticipated to run within 20 seconds on a laptop which is a considerable improvement. Practically, medical doctors and patients can use the device in their surgical rooms or at home. Environmentally, running such an app would not require an excessive amount of energy and time. Monetarily, only a laptop will be necessary to run the app, which opens the possibility to make the app commercially available to users for low licensing fees. The development of this new technology into a clinically accessible, portable platform will usher in new opportunities for multi-centers collaboration and

objective comparison of outcomes between different conditions, techniques, operators, and institutions.

1.3 The Haar Cascade

The implementation of this project is done through using the Haar cascade classifier. It uses a machine learning approach where a cascade function is trained from positive and negative samples, then it is used to detect objects in other images. The classifier is essentially an object detection algorithm which is used to identify faces in images or real time videos through edge or line detection [4]. Our algorithm is utilizing sliding rectangles to scan the image. Once it detects a face, the algorithm outputs a "confidence" score which is what we utilized for this project. To train the Haar cascade algorithm, we have collected 140,000 images from an open-source website called "thispersondoesnotexist.com". We used this website to avoid any copyright concerns with existing datasets of images of children, and to avoid any privacy concerns. The images we had collected were a mix of both adults and children. Since our project is primarily focused on children, we had to find a way to filter out the images such that only images of children are in our dataset. For that, we have used a GPU-based TensorFlow algorithm that was trained on images of children.

1.4 Image Collection

To collect the images for our database, we have used an opensource website called "thispersondoesnotexist.com". Essentially, that website has an AI that generates random images of people's faces. Those people do not exist, hence the name "thispersondoesnotexist". The AI was built by using styleGANs, which was developed by NVIDIA. How it works? From the dataset, it selects an input facial image. From this input image, the "style-based generator" can learn and pinpoint the distribution of the face and apply those points onto a "novel synthesized image". It also utilizes "Stochastic variation", which is an important part of the process of generating the

artificial facial image. It is important because it allows GANs to “realize the randomization of detailed facial features…” [5].

To collect the images produced from this website, we have created an algorithm in Python that would refresh the page and take a screenshot every time the website was refreshed. While most of the images that were produced by the website seemed perfect, there were a few images that show the limitations of “thispersondoesnotexist”. Some of the images produced by the AI showed “glitches”. Some of those glitches include a second face in the background that is extremely distorted. Other glitches would show a mistake in the process of producing the facial image, which would make the website produce an image such as the image shown in Figure 1.1.

Figure 1.1: Example of a “glitched” facial image produced by “thispersondoesnotexist”

1.5 TensorFlow

The samples obtained to train the algorithm first need to be filtered out based on specific criteria: only images of children were required. Rather than manually filtering through hundreds of thousands of images, the collected image database was processed by an age detection algorithm which used TensorFlow for the detection task. TensorFlow is a Python library for quick numerical computing that supports traditional machine learning, and particularly focuses on training and

inferring neural networks. With high accuracy and quick speed, the TensorFlow algorithm had filtered out the 140,000 images collected from “thispersondoesnotexist”. The result was roughly 14,000 images of children and false positive adults were filtered out. After manually filtering out those 14,000 images to remove any false positive adults and any of the glitched images (as shown in Figure 1.1), we were left with roughly 11,000 images of children to work with.

1.6 Technical Standards, Constraints and Risks

Since the dataset is to be stored locally, the application does not need an Internet connection to run. Hence, this aspect helps to make the application more accessible in areas where the Internet is not be very stable as the speed of the algorithm is not affected by the low bandwidth. However, this means that the application requires a large amount of storage since all the data would be stored locally. Also, since the data is stored locally, the results cannot be shared seamlessly with other doctors.

There is also an advantage to security and privacy. Since the application is not cloud based, it is less susceptible to data leakage as the images can only be viewed by the user. However, the weakest link in any security feature is the human. Humans tend to make mistakes, and if the laptop was lost or left open and it was not password protected, then the scanned photos are prone to be leaked.

1.7 Performance Criteria

Since there are many different techniques for the same type of facial reconstructive surgery, this rating would allow surgeons to measure the success rate of each technique. It could also allow for collaboration between different hospitals to find better techniques that would increase the chances of a successful surgery which may add to the quality of public welfare.

In terms of cultural factors, we know that in the Middle East, privacy is a major concern for many. Therefore, we had to work around cultural restrictions. For example, it was exceptionally hard to collect photos of "abnormal" facial features in children or have people participate in any sort of survey unless assured of their anonymity. To maintain these cultural ideologies, we have only used photos that are in the public domain to conduct our surveys and those who answered the surveys will remain completely anonymous.

Since our aim is for our program to be implemented as a global standard for rating of facial reconstructive surgery, another issue we had to take into consideration is where in the world this project is going to be implemented. Different parts of the world require different training samples. To combat this, we collected a wide range of data across many different ethnicities. The program is then trained to rate faces based on average human norms in general rather than one specific region.

As for its environmental impact, since this program runs on a laptop rather than a supercomputer, there are minimal environmental effects. That is because laptops require much less power to run than a supercomputer.

1.8 Chapter Outline

This thesis is divided into four chapters: Chapter 2: Methods further discusses in detail the methods that this project implemented, how the training of the algorithm was done as well as the collection of its training dataset, a functional modelling diagram portraying the steps the project went through, and a comparison between our proposed solution and previously existing solutions. The third chapter, Results, focuses on our acquired dataset for training and its filtration to suit the requirements for training our algorithm, and the results we have received from training the Haar cascade algorithm. The fourth and final chapter summarizes the overall project as well as its findings and aims.

2. METHODS

This thesis goal is to identify a suitable algorithm for assessing the degree of facial abnormality and the improvement offered by a reconstruction surgery. The algorithm must be suitable for implementation on regular laptops to facilitate its adoption for clinical practice. The Haar cascade algorithm represents an approach that fits these requirements. A large dataset of non-copyrighted images is required for training the algorithm to make it as precise as possible. A solution to this challenge is getting facial images using the application website "thispersondoesnotexist.com". There is also the problem of getting many photos from that website, as doing it manually is extremely slow and inefficient. The solution to that is to write a Python script that would automatically refresh and screenshot the image produced by "thispersondoesnotexist".

The images produced by "thispersondoesnotexist" include both adults and children. Since we are only working on a project that is focused on children, we had to filter out the saved images to contain only children. To do so, we used a TensorFlow algorithm that is trained to detect children's faces. We made slight edits to that algorithm so that it suits our needs of automatically saving the images it detects as children into a separate file. We have done this to the 140,000 images we have collected and ended up with roughly 14,000 images.

2.1 Existing Solutions

There are other solutions that can implement the same functionality our project aims to achieve; however, our design is unique for its significant role in presenting the improvement from pre-surgery to post-surgery stage. Our project tends to measure the level of facial normality of any facial image and deliver an abnormality score pre- and post-reconstruction.

One recently implemented project's goal was to generate a normalized facial image for pre-reconstructive surgery. The project aimed to remove the abnormal part of a child's face and replace it with a generated facial feature that looks normal based on samples of human faces embedded within the application itself. The disadvantages of this project, however, were hard to ignore. Due to the lack of input databases, the features are limited in terms of variety and precision. It also raises the hopes of the patients' families by making them imagine or expect an unattainable result of their child's face.

There are also other solutions available in the market that can implement the same functionality; however, this requires doctors to manually use Photoshop and other morphing tools to construct the patient's face. These solutions were found inadequate due to their drawbacks. Most importantly, they harm the patient's confidentiality and privacy, as the patient's image is shared with a broader range of people to construct a normalized image of the patient's face [1]. Our project proposes to offer an objective assessment mechanism through which improvements can be measured instead of having an application that relies on providing ungranted or subjective results. A summary of the main distinguishing features between the proposed solution and existing approaches is presented in Table 2.1.

Table 2.1: Comparison between existing solutions and our facial scaling application.

Existing Solutions (facial generation application) [6]	**Existing Solutions (Photoshop and morphing tools) [6]**	**Our facial scaling application**
Generates a normalized automated face pre-surgery.	Constructs the patient's face.	Rates images based on how "normal" the faces are (with 0 denoting the abnormal face and 3 representing the normal face).
The application ensures privacy.	Less privacy. As this process requires a morphing tool expert.	The application ensures absolute privacy, since only the surgeon gets to see the image.
Precise but not convenient as it sets expectations that doctors aren't necessarily going to meet.	Subjective as it relies on the personal bias and beauty standards of the surgeon.	Convenient and precise as it provides a scale across which improvements will be measured.

2.2 Proposed Solution Analysis

Upper-Level Functional Model

Figure 2.1: Top level one-block functional model

For the upper-level functional model, the main program is represented by the NEKSUS software as depicted in Figure 2.1. The user would input a facial image into the software, then the software will output a normality score between 0 and 3, where 0 represents an abnormal face and 3 represents a normal face. The input to the program would simply be a facial image, and the output would be a score (number).

The surgeon will have the choice to pick between two options: to either pick the camera so that they could take a photo of the patient through a camera connected to the computer, or to browse through their computer and pick an image that was already taken before. Either way, the image would be scanned regardless of how the image was taken. The format of the image and the size (resolution) of the image do not matter and do not affect the accuracy of the algorithm, as long as a complete face can be seen in the image. Also, it is important that only one face (the face to be scanned) is visible in the image.

2.3 Collecting the Dataset

First to train our algorithm, we had to collect a large number of various normal faces falling within the age range of children since this is the focus of our scale. Moreover, to avoid concerns of breaching privacy and copyright, the images were acquired from thispersondoesnotexist.com website which utilizes artificial intelligence tools to generate random human faces. The website was refreshed and screenshotted by an algorithm that we have developed. This algorithm can be found in Figure 2.2. The algorithm shown in this figure was developed in Python. The main library used is called “selenium”. This library allows Python to interact with a web browser as if it is a user. The browser we used for the image collection was Google Chrome. In the third line, we defined the path where selenium can find the browser so that it could launch it. Since we have used a laptop that has a 4k resolution display, we set up the window size as shown in the fifth line of the code in Figure 2.2. Essentially, the code screenshots an image, renames it as “image” and adds the number of screenshots it is based on in the while loop and the increment value of “x”. This repeats 10,000 times in each folder. Therefore, we have 14 folders, and each folder contains 10,000 images each, which equates to collecting 140,000 images of adults and children.

Since the primary focus of this project is on children, the images we collected had to be filtered to contain only images of children. To do so, we used another machine learning algorithm that extracts images of faces that follow a certain criterion. This machine learning algorithm utilizes TensorFlow. TensorFlow is a deep learning platform that can utilize both the CPU and the GPU and is typically used in machine learning applications. Some projects that are very accurate and fast can be setup in TensorFlow. An example of such project is called YOLO.

Setting up TensorFlow required installing Microsoft Studio C++, Nvidia’s CUDA and cuDNN software. CUDA and cuDNN allows TensorFlow to access the GPU, so it is important to

install this software. The TensorFlow algorithm was trained to recognize images of children. We made slight additions to the code such that it would save the images it filters out into another file. We have filtered files 1 to 14, and we have ended up with roughly 14,000 images. These 14,000 images are not purely children though, as there were instances of false positives.

The TensorFlow algorithm was very accurate, as most of the images it filtered out were images of children. However, there were some images of adults that flagged a false positive. To deal with this, we had to remove the false positives from the 14,000 images that were filtered out by the TensorFlow algorithm. Even some of the positive images were not suitable to be used. Since "thispersondoesnotexist" is an AI website that generates images, there are instances where the photo shows some "glitches". We also had to look through the images to see if we could spot any of these glitches on the faces that were produced by the website. Also, it was important to check if the images do not contain another face in the background as it could affect the training of the Haar cascade algorithm we developed. Just like when the images were first collected, the images that were filtered out were each saved across 14 different files. However, this time, they are not equal in numbers. The range was between 1,100 images to 1,400 images across each of the files. Manually removing the false positives took a lot of time. However, this process was made faster by an extraordinary amount because of the TensorFlow algorithm, as we did not need to look through the original 140,000 images. We ended up with around 11,000 images to use after manual searching.

```
from selenium import webdriver
import time
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver= webdriver.Chrome(PATH)
driver.set_window_size(16000, 12000)
driver.get('https://thispersondoesnotexist.com/')
x = 1
while x<10001:
    driver.save_screenshot('C:/Users/ECEN/Desktop/Website_Refresh_Test/14/image'+str(x)+'.jpg')
    x+=1
    time.sleep(0.5)
    driver.refresh()
```

Figure 2.2: The algorithm developed for the collection of images

2.4 Training the Haar Cascade

To train the Haar cascade algorithm we had to acquire two types of datasets, a positive and a negative dataset. The positive dataset is the dataset of children's facial photos we have acquired from "thispersondoesnotexist". The negative dataset is composed of any kind of image that does not contain any trace of faces. Training the algorithm with a negative dataset is a technique used to train machine learning models to help assess the specificity of the model. The algorithm is trained with a scenery of images that contain no faces. In this way the algorithm will differentiate a facial image from a non-facial image. The goal is to make the algorithm overcome the limits of our instinctive knowledge of faces.

To start off, we had to download OpenCV version 3.4.11, since the versions before OpenCV 4.X contain the executable files required to train the cascade. We placed the OpenCV file into a folder. Then, we had to create two separate folders, one folder being for the positive images and one for the negative images. We placed those two files in the same folder as the downloaded OpenCV file to make things easier. There are some important executable files that we used from the OpenCV file to train the cascade.

Firstly, we had to create a "txt" file that contained the names of the negative images and their directories (where they can be found). To do so, we created a four-line Python script that would open a txt file, then open the file we called "negative" that had the negative images, and write "negative/(image name)". The script we have written for that is in Figure 2.3. Afterwards, we had to create a "txt" file for the positive images. However, the positive images had other sets of requirements that did not make it as easy to obtain as the negative "txt" file. To start off with the positive samples, we had to use an executable file created by OpenCV called "opencv_annotation.exe". On the Python terminal, we had to tell Python to run this executable

file and created the “txt” file by using the “annotations” parameter. We have also informed the executable file where to find the positive images folder from the “images” parameter. Running this executable file in the terminal resulted in a window popping up, showing one of the images that we have saved into the positives file. A prompt on terminal window was printed, giving us instructions to highlight those that we want to be considered positive, and what certain keys do when pressed. Essentially, we had to manually insert a rectangle on the faces of around 11,000 images of children. However, doing so would have required an immense amount of time and effort. Therefore, we employed another algorithm that was already trained on faces to automatically write the directory of each image and the coordinates of the rectangle (number of rectangles, x, y, width, height).

```
1  import os
2  with open('neg.txt', 'a') as f:
3      for filename in os.listdir('negative'):
4          f.write('negative/' + filename + '\n')
```

Figure 2.3: The script written to generate the negative “txt” file

The algorithm that we used to automatically generate the positive images annotations is called “Dlib 68 points Face landmark Detection” [7]. We downloaded that algorithm and the required trained dataset. We had to make a few changes to suit our need, as the algorithm sets a coordinate for the 68 points it generates on a face. What we wanted it to do is to write the coordinate of the top left corner of the rectangle it drew on the face, along with its length and width. We also wanted it to write the directory and the name of the image. All of this text had to be in a uniform fashion, such that the executable file in OpenCV would be able to understand the “txt” file. We modified the algorithm such that it scans every image in the file, writes down the name of the image and its directory, and next to the directory and the name it would write the

number of faces it detected. Then, after the number of faces is detected, it would write the x coordinate and the y coordinate of the upper left corner of the rectangle it drew on the face it detected. Finally, it would write the width and the height of the rectangle it drew on the face and enter a new line for the process to repeat. An example of the resulting "txt" file can be seen in Figure 2.4.

```
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image0.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image10.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image100.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1000.jpg 1 197 238 372 372
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1001.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1002.jpg 1 138 187 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1003.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1004.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1005.jpg 1 138 237 446 446
C:/Users/ECEN/Desktop/DATASET_720x720/CHILD/image1006.jpg 1 138 187 446 446
```

Figure 2.4: Snippet of what the "Dlib 68 points Face landmark Detection" algorithm with our modification outputs

After obtaining the "txt" file from the modifications we have made to the "Dlib 68 points Face landmark Detection" algorithm, the next step would be to create the samples. Using the terminal on Python, we ran the executable file "opencv_createsamples.exe". This executable file would convert the positive images into a size and format that the training executable file can understand. The "createsamples" executable file has parameters "w" and "h", which are the width and the height it would convert the images to. The higher the "w" and "h", the larger the image, and therefore the more detailed it is. Typically, the more popular choice would be to go for a width and height of 24. We instead chose to go with a width and height of 32, as it is more detailed than 24 but not as slow as a very detailed width and height of 48. The executable file outputted a "vec" file that the training executable file will use for the training of the Haar cascade.

Training the Haar cascade algorithm required us to use an executable file from OpenCV. The executable file is called “opencv_traincascade.exe”. To start training, we had to set several parameters that would affect the resulting (produced) “xml” file (the dataset). Of the parameters that we had to set were the number of positive samples and the number of negative samples. The ratio between the positive samples and the negative samples plays a role in the quality of the dataset that gets produced by the executable file. Also, there was another parameter called “maxFalseAlarmRate” that also contributed to the quality and accuracy of the produced “xml” file. To test the parameters and see what ratios and numbers yield the best results, we created six different datasets with three different ratios and two different “maxFalseAlarmRate”. The number of positive samples during the testing phase stayed 400, while the negative samples were changed such that the ratio between positive and negative samples is changed. The ratios that we tested were 1 positive sample to 1 negative sample, 1:2 and 2:1. The two values of “maxFalseAlarmRate” that we tested were 0.4 and 0.5. The dataset that showed the best and most consistent results was the ratio of 2 positive samples to 1 negative sample, and a “maxFalseAlarmRate” of 0.5.

After the testing phase, we set up the executable file to train for a much bigger number of samples. We have decided to use 8000 positive samples, and 4000 negative samples with “maxFalseAlarmRate” of both 0.4 and 0.5 to check for any differences in accuracy. Setting the right number of stages in the “numStages” parameter of the executable file is a major factor to the accuracy of the produced “xml” file. Too many stages would mean that your “xml” file is overtrained and would only recognize the images of the positive samples you used to train it. Too few stages means that the Haar cascade would output many false positives. An advantage of Haar cascade is that when you train it, it saves every stage directly when it completes the stage.

That means that when you stop the training, you can continue the training where the Haar cascade's last stage saved. This also means that we could overtrain the Haar cascade, but then run the training executable file again with less stages and the xml file with the lesser stages would be instantly produced. We have started off the training with 8000 positive samples, 4000 negative samples, mFAR ("maxFalseAlarmRate") of 0.5 and 18 stages. This training took around two days to complete. Next, we changed the mFAR to 0.4, and the number of stages to 16. With these parameters, the training took three days to complete.

To ensure that the trained algorithm exhibits a more human-like feel for what is considered as a normal or an abnormal face, several surveys were conducted where people were given different images and were asked to rate them in the same range from 0 to 3 as the code adopted. Forty images of children with different degrees of facial abnormalities across eight different survey links were distributed amongst random individuals. Two hundred responses were collected, and the results from those surveys were averaged. This step is important, because we will use the human judgement of the abnormalities to test the accuracy of our Haar cascade algorithm. If the algorithm produced a score that was far-off from the averaged survey results, a deeper level of training with more positive and negative samples is planned to be executed.

2.5 Setting up the Python Script

After the training was complete and we obtained the "xml" file, we wrote a script that would utilize this dataset and allow the Haar cascade to output a confidence score. The design of the algorithm was setup with the user in mind. We wanted to create the application such that the doctor or surgeon could easily input a photo into the algorithm and get an output back. To do so, we had to use a Python library called "tkinter" that would pop-up a window upon running the code. We have designed the code such that a window would pop-up asking the user to select how

they would like to input the image. We created two buttons. The first button is called “Camera” where the user can use the camera on the laptop or any connected camera and take a photo from the camera. The second button is called “Browse Image” where the user can select an image from anywhere on the laptop. The window that pops up is shown in Figure 2.5.

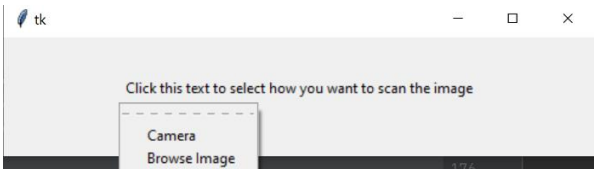

Figure 2.5: Screenshot of window that pops up when algorithm is first run

To allow the code to use the camera, we had to use OpenCV. We programmed the camera part as its own definition so that the camera part of the code and the image browser part of the code can work separately. The part of the script that is responsible for the camera part is shown in Figure 2.6. We set the dimensions of the camera window that pops up as 1280x720 as some laptops still have 720p webcams, so we want to include those laptops too. We also mirrored the camera again because OpenCV mirrors the original webcam such that if a person raises their right hand, the window will show the person is lifting their right hand instead of their left hand like a mirror would do. We mirrored the camera feedback again to make it easier for the user to position the face of their patient. We also highlighted in the camera window as a red box where the face of the child should be for the most accurate results. When the spacebar is pressed, a screenshot of the camera window is taken. Then, the image is cropped such that the image becomes a square, and the area of the red square (where the face must be) will be centered in the image, which would result in the image of resolution 720x720. The red square that highlights where the face should be in the camera window also shows up in the screenshot that the user takes when the spacebar is pressed. However, this does not show any signs of tampering

with the confidence score the Haar cascade gives. Once the user takes the photo with the spacebar, the user must press the “ESC” button on the keyboard for the camera window to close and for the algorithm to continue running to output the scores. These instructions are shown on the window title.

```
def camera():
    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
    cv2.namedWindow("Press the spacebar to capture an image. Press ESC to exit.")
    while True:
        p, vid = cam.read()
        vid = cv2.flip(vid, 1)
        cv2.rectangle(vid, (84, 84), (644, 640), (0, 0, 255), 2)
        if not p:
            break
        cv2.imshow("Press the spacebar to capture an image. Press ESC to exit.", vid)

        k = cv2.waitKey(1)
        if k == 27:
            break
        elif k == 32:
            image_name = "abn_test.png"
            cv2.imwrite(image_name, vid)

    img = cv2.imread("abn_test.png")
    test_input = img.copy()
    test_input = test_input[0:720, 0:720, :]
    img = img[0:720, 0:720, :]
    # dim = (224, 224)
    # img2 = cv2.resize(test_input, dim, interpolation=cv2.INTER_AREA)
    cv2.imwrite("abn_test.png", img)
    cam.release()

    cv2.destroyAllWindows()
```

Figure 2.6: The code responsible for the camera part

For the image browser part, we had to use a Python library called “pillow” so that Python could process the images that are selected by the user from the image browser. The image file types that can be viewed by the user are “.jpg”, “.png” and “.jpeg”. The user can add to this list if they have an image type that was not mentioned previously. The function of this part of the code is that it would copy the image the user has selected, save it to the directory of the Python algorithm, rename it to “abn_test.png” and then produce the confidence score on that image. The part of the code that is responsible for this process is shown in Figure 2.7.

```
def file():
    path = filedialog.askopenfilename(filetypes=(("Image File", "*.jpg"), ("Image File", "*.png"), ("Image File", "*.jpeg"),))
    shutil.copy(path, "abn_test.png")
    shutil.move(path, path)
    im = Image.open(path)
    image2 = ImageTk.PhotoImage(im)
    var = Label(root, image=image2)
    var.image = image2
    var.pack()
```

Figure 2.7: The code responsible for the image browser part

3. RESULTS

3.1 Image Database Collection

The development of the facial scoring program relies heavily on an extensive training period for the machine learning algorithm. In order to ensure that the training process is performed efficiently and correctly, a very large database of images is required to be collected in order to train the algorithm. This was achieved by dividing the training process into two sub-processes: first, saving the images into a file which constitutes the database and then filtering out any erroneous images which may contain glitching which could possibly hinder the progress of the training. To collect the database required for the machine learning application, an algorithm for saving images was first developed over multiple stages. Firstly, an open-source website was used to collect the facial images that were used to build up the database that would later be used to train the algorithm. The website that was used to collect the images is "thispersondoesnotexist.com", and it is, essentially, a machine learning algorithm in and of itself that outputs random facial images of people who do not exist by using features from a large pool of options stored in the website's database and practically "building" a facial image from features that could constitute a realistic-looking image. The database that the website uses to extract the features is composed of images of real children and adults of varying ages, ethnicities, and genders which helps the training process by exposing the machine learning algorithm to various characteristics and identities which can be classified as normal. By collecting images from "thispersondoesnotexist.com", privacy and copyright concerns due to nonconsensual use of personal images are avoided. Figure 3.1 shows an example of a facial image that the website produced. The images built by "thispersondoesnotexist.com" present a solid sense of reality, and

concerns of unrealistic looking images are debunked. We collected 140,000 images in ".jpg" format by running the image-saving algorithm that was developed. The algorithm continuously refreshes the website, and then it takes screenshots of a specific size of each image displayed. It is important to emphasize the fact that the images collected using this method are composed of both adults and children of varying ethnicities.

Figure 3.1: Example image of a child who does not exist - produced by "thispersondoesnotexist.com"

The image shown in Figure 3.1 is an example of the images that were screenshotted using the algorithm we developed. However, as seen in the figure, the images have black bars to the left and to the right of the actual image. Therefore, to remove these black bars, we developed an algorithm that would crop the photos we collected. The resulting images would turn out to be a square of 1800x1800 resolution. This was deemed to be too large, as we found that OpenCV does not open images greater than 1920x1080 resolution in a proper way (the image would be too big for the screen). Therefore, we created a script that would simply reduce the size of the images down to 720x720. An example of such image can be seen in Figure 3.2.

Figure 3.2: Example image of a child who does not exist – produced by "thispersondoesnotexist.com"– in 720x720

Since the facial normality application was developed mainly for use by medical surgeons who will be working on reconstructive surgeries for children, the database of the application focuses on facial images for children. This represents a problem to be solved, since there are very limited methods of directly filtering out images of adults directly from the website.

In essence, this method relied on first collecting a database of images, regardless of age, and then filtering out the desired images from the saved database. To do so, a filtering technique was necessary to filter out a subset database which is made up of facial images of children. This was achieved by implementing a TensorFlow algorithm which runs through the 140,000 images that were previously collected, and it detects images of children through its own built-in age detection technique. The TensorFlow algorithm was successful in filtering out a total of 14,000 images. However, of those 14,000 images there were a few false positives and images that contain "glitches". After manually removing those images, we were left with roughly 11,000 images of children to use for the dataset.

3.2 Haar Cascade Algorithm Test

As mentioned previously, the main algorithm that was implemented in the facial scoring program utilizes the Haar cascade to produce an abnormality score of an image of a face ranging

from 0 which represents an extreme abnormality to 3 which denotes a normal face. When the "camera" input option is selected, the computer's webcam turns on and a red square appears. This square is where the face must be located for the most optimal facial detection. The face is adjusted in a certain way so that facial features like nose, eyes and lips are present at specific points. Following this approach, the face will match the x-y coordinates of the images collected from "thispersondoesnotexist" website using a grid approach (adding an invisible grid to the image whereby the exact coordinates of a pixel can be detected). Figure 3.3 shows the camera window that pops up when the "Camera" option is selected from the pop-up window. The camera is not showing an image because it is covered by a sticker.

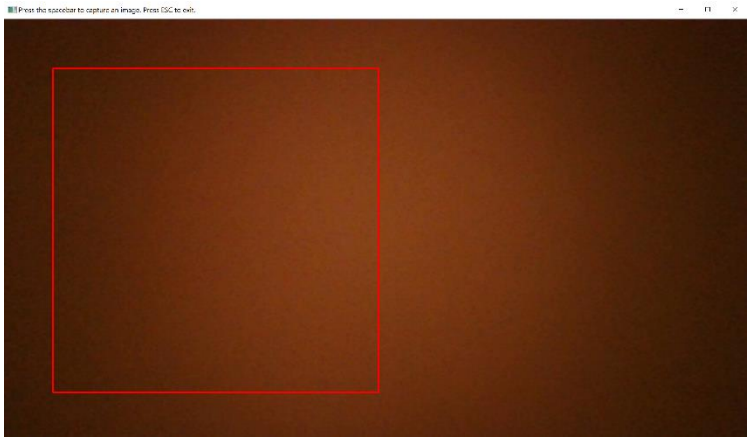

Figure 3.3: Camera window pop-up

The current version of the scoring algorithm was tested, and the image used for preliminary testing can be seen in Figure 3.4. It represents a case of a cleft lip- a very common case of abnormality that medical professionals in the reconstructive surgery field tend to work with. Figure 3.5 shows the output score that the algorithm produced. As seen in Figure 3.5, a normality score for the face, mouth, and eyes up to 16 decimal places of precision was produced. Our main focus is on the face, as the scores for the mouth and eyes are there to test whether a

face was truly detected or not. The xml file used for the mouth and the eyes scores are xml files created by OpenCV. However, the xml file used for the face is the xml file we have trained and created.

Figure 3.4: Image of child with cleft lip used to test the algorithm [8]

```
Face Score:  1.6796487867832184
Eyes Score:  2.7024488486349583
Mouth Score:  2.396147096529603
```

Figure 3.5: The facial scoring algorithm's output of the abnormality score of the child in Figure 3.4

While the normality score of the mouth is close to the anticipated normality score from a human-like perspective, the score for the eyes may not be very accurate when judged from a surgeon's viewpoint. The reason why the score given to the child's eyes was low may have been because the child was not looking directly at the camera. This issue was not considered a substantial obstacle in the success of the program, as it is a very minor issue that can be resolved by simple adjustment of the subject and/or camera angle and positioning.

To further test out the algorithm, we have used 60 images of various “abnormality” degrees to see whether the algorithm would flag a normal image as abnormal or an abnormal image as normal. Amongst those 60 images, there were two images that were before and after surgery of a cleft lip on a child. The image that was before surgery scored a score of 2.4063 which indicates abnormality, while the image after the surgery scored 3.1167 which is a normal score. Since we have the scores for the face before and after surgery, we can find the delta value. The delta value can be found by subtracting the score of the image after surgery to the score of the image before the surgery. Therefore, the delta value in the case of the two aforementioned images is 0.7104. Since our scale (0-3) is small, an improvement of 0.7104 can be concluded to be a “good” improvement. It cannot be denied that a high delta value is an indicator of a highly successful surgery.

From the 60 images that were tested, only two of the abnormal images were falsely scored as “normal”. However, a slight cropping to the two photos fixed the scoring to an abnormal score. Out of those 60 images as well, only one normal photo scored a 2.6844. Even with those minimal false positives, the rest of the images scored as expected and we were satisfied with the results our trained xml file produced.

The tests mentioned in the previous paragraphs were done by using the 0.5 mFAR and 16 stages xml file. However, we have done the same tests with 7 other xml files to see which of them best follows our criteria. Those xml files each had different numbers of stages and different numbers of mFAR (“maxFalseAlarmRate”). For the 0.4 mFAR, we tested for stages 13, 14, 15 and 16, while for the 0.5 mFAR, we tested for stages 15, 16, 17, 18. The most accurate of the eight tests was the 0.5mFAR at 16 stages, and that was the xml file used to detect the image in Figure 3.4 and output the score in Figure 3.5.

4. CONCLUSION

4.1 Overall Summary

The ability to express oneself through facial expressions is vital to many people as it provides cues for social engagement, and a side goal of this project is in assisting in implementation of better surgical techniques for facial reconstruction surgery.

The aim of this research project is to create a machine learning program with which surgeons can obtain a normality score for a face pre and post facial reconstructive surgery. Subtracting the post-surgery facial image score to the pre-surgery facial image score, they can obtain the delta value, which indicates the degree of success of the surgery. This paper presents the background which motivated this project. It also presents the procedure followed to collect the database by exploiting "thispersondoesnotexist.com". This website was used for generating a substantial number of images while maintaining personal privacy since the resulting images are not of real people. They are images that are produced by an artificial intelligence tool, called generative adversarial network (GAN). We were able to collect 140,000 images from that website and we had to filter those images to include only children. After the process of filtration, we ended up with roughly 11,000 images of children to work with. By using OpenCV's executable files, we were able to train the Haar cascade using 8000 images of children and 4000 images that does not contain a face. After a series of testing and trials, we have come up with the conclusion that an xml file trained to 16 stages at a "maxFalseAlarmRate" of 0.5 was the best xml file. This xml file would give a score of 3 and above to facial images with no abnormalities, to a score of 0 that represents the most severe case of facial abnormalities.

REFERENCES

[1] Q. Xu, Y. Yang, Q. Tan, and L. Zhang, "*Facial Expressions in Context: Electrophysiological Correlates of the Emotional Congruency of Facial Expressions and Background Scenes*," Frontiers in Psychology, vol. 8, 2017.

[2] T. Pruzinsky, "*Social and psychological effects of major craniofacial deformity*," The Cleft Palate-Craniofacial Journal, vol. 29, no. 6, pp. 578–584, 1992.

[3] R. Bull and N. Rumsey, in *The Social Psychology of Facial Appearance*, New York, NY: Springer-Verlag, 1988, pp. 180.

[4] G. S. Behera, "*Face Detection with Haar Cascade*," Medium, 29-Dec-2020. [Online]. Available: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08. [Accessed: 03-Mar-2022].

[5] T. Peng, "*Gan 2.0: Nvidia's Hyperrealistic Face Generator*," Synced, 14-Dec-2018. [Online]. Available: https://syncedreview.com/2018/12/14/gan-2-0-nvidias-hyperrealistic-face-generator/. [Accessed: 11-Mar-2022].

[6] O. Boyaci, E. Serpedin, and M. A. Stotland, "*Personalized quantification of facial normality: A machine learning approach*," Scientific Reports, vol. 10, no. 1, Dec. 2020.

[7] Shekharpandey, "*Dlib 68 points face landmark detection with opencv and python*," Studytonight, 12-Aug-2021. [Online]. Available: https://www.studytonight.com/post/dlib-68-points-face-landmark-detection-with-opencv-and-python. [Accessed: 22-Mar-2022].

[8] R. Tharu, "*Multifactorial Birth defects*," Medindia, 05-Apr-2014. [Online]. Available: https://www.medindia.net/patients/patientinfo/multifactorialbirthdefects.htm. [Accessed: 27-Mar-2022].