COMSW 4735 Project – *Color Analysis* Report

Aaron Ashery (ala2201)
Alexandra Long (ael2203)
Tiffany McBrayer (ttm2126)

May 2, 2023

Contents

1 Introduction

In COMSW 4735: Visual Interfaces, our group consisting of Aaron Ashery, Alexandra Long, and Tiffany McBrayer developed a program to determine a personal color palette based on an individual's features. Multiple social media platforms have seen a rise in popularity of this service in which people receive color analysis to better dress themselves based on colors that suit them. Usually an individual would go to a professional color analyst or consult online articles to find the right colors based on their eyes, hair, and skin undertone. By allowing the user to access this assessment online, we make the process more efficient and objective.

1.1 History

Color analysis has been prevalent in American fashion since the 1980's with numerous resurgences throughout the years since.[1] Artists, chemists, and mathematicians all separately found certain colors work well together, and more importantly that combining the wrong colors can clash and emphasize flaws.

In 1981 Carole Jackson published '*Color Me Beautiful: Discover Your Natural Beauty Through The Colors That Make You Look Great & Feel Fabulous!*.' Jackson describes the attributes based on skin tone, eye color, and hair color to categorize a person into a color season and provides color palettes for each of the four seasons. The descriptions are accompanied by images of different women showing colors that match and do not match their overall attributes. Although there are many examples of women in each season, there are no real thresholds for who is a Winter, Summer, Fall, or Spring.

Color Me Beautiful has faced criticism for its lack of inclusivity towards people of color. Many have argued that the system is designed for a white audience, describing white people accurately but failing to acknowledge the diversity present across all individuals. Unfortunately, this has been true in many cases in which websites base their information off of *Color Me Beautiful*, where people with deeper complexions or brown hair and brown eyes have been automatically typed as Winter without proper consideration of their coloring. To address this issue, it is important to provide examples of people of color for all seasons.

Our research aims to develop a system that not only incorporates Jackson's ideas but also accurately categorizes people of all diversities.

[1] https://www.kettlewellcolours.co.uk/blog/jo/a-brief-history-of-colour-analysis

1.2 Competitor Analysis

During our competitor analysis and research, we tested various websites that claim to follow Carole Jackson's suggestions. In doing so, we found that each website chooses different aspects of the book to use as their method. We identified the aspects that we found to be helpful and those that needed improvement. Through this process, we also identified areas where we could enhance our own system to provide a more comprehensive and accurate color analysis and season categorization for our users.

As part of our competitor analysis, we reviewed a website[2] that asks users to upload an image and use color-picking to identify their skin, eyes, and hair color. The website provides instructions for users to "select a clear, high-resolution head shot in good lighting and only choose the most prominent tones for each attribute to ensure accurate results." There is a limitation here when relying on a single color sample for each attribute, which may not capture the full color range of an individual's characteristics. Additionally, when we tested this website with women of color, we found that the majority of them were categorized as Winter, regardless of their actual undertone. This suggests that the system may not accurately account for the full variety of skin tones and may not be as inclusive as advertised.

We reviewed another website[3] that attempts to provide color palettes and season categorization for individuals by allowing them to choose pictures of hair, eyes, and skin that most accurately represent themselves. The website suggests having a friend or family member assist in selecting the closest options to achieve the most precise results. Though that method is also subjective, one's personal preferences can significantly impact the categorization outcome. This method has its limitations as it does not involve analyzing a user's photo and rather relies on a potentially inaccurate representation of the user.

Lastly, we reviewed many websites 8 that attempt to categorize a user by providing methods to determine skin undertone. For instance, when determining skin undertones, certain websites suggest holding up silver and gold jewelry against your skin to decide if you have a cool or warm undertone based on which metal looks better on you. However, this method is also highly subjective and can be influenced by personal preferences in jewelry, leading to potential bias. Another method advises users to determine their skin undertone based on the color of their veins, which can exhibit either a blue-purple or green hue.

[2] https://colorwise.me/self-analysis
[3] https://www.colorenalysis.com/

2 Specification

2.1 What Is Color Analysis

For color analyses performed by professionals, typically, an individual would be consulted through multiple stages of color swatching. The consumer starts off with the identification of their eye, hair, and then their undertone colors. With that information, there are multiple methods that can be used for color analysis, for example, categorizing people into certain seasons (Winter, Summer, Fall, and Spring) which correspond to colors that will suit them fashionably. There are a lot of online applications as mentioned in section 1.2 that also try and determine your season and color palette.

The challenge for us lies in creating a system that accurately chooses regions, values of color, and values of lightness for the skin, eyes, and hair. Then, leverage this information to determine the user's season and other metrics, such as a personalized color palette.

We compiled all the instances from *Color Me Beautiful*, and here are the findings:

- Winter
  - Skin: Cool undertone
  - Eyes: Deep black, brown, gray-blue, dark blue, or rosy brown (some have clear green, gray-green, or hazel eyes with a gray tone)
  - Hair: Black, dark brown, blue highlights, ash brown, or silver gray, NO red highlights
- Summer
  - Skin: Cool undertone
  - Eyes: Blue, aqua blue-green, deep blue, hazel, or rosy brown
  - Hair: Blonde, light brown, or dark brown
- Fall
  - Skin: Warm undertone
  - Eyes: Dark golden brown, golden green, green, or hazel
  - Hair: Golden brown, golden blonde, gold, or red highlights
- Spring
  - Skin: Warm undertone
  - Eyes: Blue, green, pale blue, deep blue eyes, hazel, or amber
  - Hair: Golden blonde, vivid red, or auburn

As you can see from the categorization outlined above, there is some overlap between the different seasons, leaving room for interpretation.

3 Algorithms/Data Structures

3.1 Initial Color Correction

Once all the images have been read using OpenCV's imread function, we color corrected all images using an algorithm found on stack overflow[4]. Our color correction removes any color casts on the image that are unwanted. This is done by first making a copy of the image, converting it into HSV color space and adding 90 degrees to the hue channel then modulus 180. This gives one color that when blended with the original, corrects the white balance which makes the photos appear to be more neutral.

Figure 1: Color Correction Example (Before and After)

3.2 Final Color Correction

After taking our own photos we saw that they were different lighting wise from the original database photos. This meant they would produce different ranges of values when getting the color swatches. We needed to account for the differences in the database vs. the photos we would capture, so we implemented a second color correction algorithm. This was done by converting the photos to HSV color space, then splitting the hue and saturation channels. From these we obtained the differences in values of the background in the database photos and the background of our photos. We then accounted for these values by adding the difference and merging back the channels to produce the corrected image.

[4]https://stackoverflow.com/questions/70876252/how-to-do-color-cast-removal-or-color-coverage-in-python

Figure 2: Color Correction on Our Visual Data (Before and After)

3.3 Feature Detection

To begin our color analysis system, we needed to detect the features of the skin, eyes, and hair. We use Dlib's[5] pre-trained model file: shape_predictor_68_face_landmarks.dat for facial landmark detection, which is a process of identifying key points on a face. For each image, the key points always lie in the same place so you can accurately use these points to identify areas of interest. These points were used to create masks for the skin swatches and eyes. All of the functions are explained further in the files named facial_features.py and

Figure 3: Dlib Example

color_correct.py.

[5]http://dlib.net/

The function detect_facial_landmarks detects facial landmark points in an input image using the Dlib library. The facial landmark points are used to identify the features: left eye, right eye, left cheek, right cheek, and forehead. With the list of points for each feature, we then create a mask for each. Finally, the function returns a list of images representing the facial features.

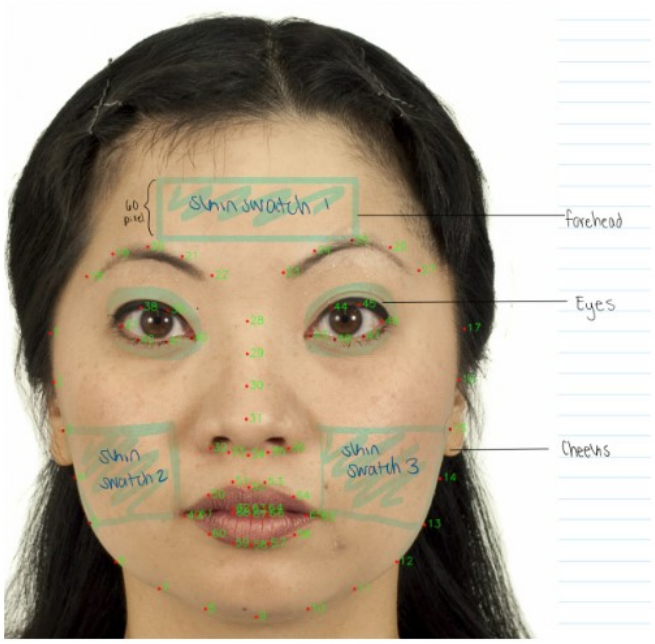


Figure 4: Facial Landmarks Drawn

Then the function calls create_mask as a helper function used in detect_facial_landmarks to create a mask based on the specific landmark points provided for the feature. The left cheek, right cheek, and forehead need extra points to be added to create their masks. The left cheek adds a point at the x position of landmark 49 and the y position of landmark 3. The right cheek adds a point at the x position of landmark 55 and the y position of landmark 15. The forehead needs two extra points to complete the rectangle. The top left corner of the rectangle would be the x position of landmark 20, and the top right corner would be the x position of landmark 25. The y position for the top corners is created by subtracting 60 pixels from the minimum y position of landmarks 20 and 25. Once all the necessary points are added for the mask, the function returns a cropped image of the original with the mask applied.

3.4 Skin

After obtaining the three skin images of the forehead and cheeks, we can determine the skin undertone by following the widely-used criteria from *Color Me Beautiful* and what is used as the industry standard in cosmetics. The skin undertone is typically categorized into two categories: warm and cool. Cool skin tones usually have difficulty tanning and are characterized by pinkish hues (not necessarily red skin from rosacea) with a bluish tint. On the other hand, warm undertones have more yellow or green in their skin and tan easily. With this information in mind, we decided to convert the skin images to Lab color space. Lab colors use three values: L for luminosity, A for green to red, and B for yellow to blue.

The approach we employed to calculate the Lab averaged value for the 3 skin swatches is implemented in the total_under_tone function. The function uses get_lab_color_space to determine the undertone of an image and then calculates the average Lab color values of each swatch. Finally, it returns a tuple containing the average Lab values.

The following is an example of the 3 skin swatches and the corresponding averaged Lab color value.

Figure 5: Lab Values Example

The following is an example of the skin swatches and Lab values ordered by “a” values singling out the swatches closer to the max and min.

Figure 6: A Values Example

The following is an example of the skin swatches and Lab values ordered by "b" values singling out the swatches closer to the max and min.

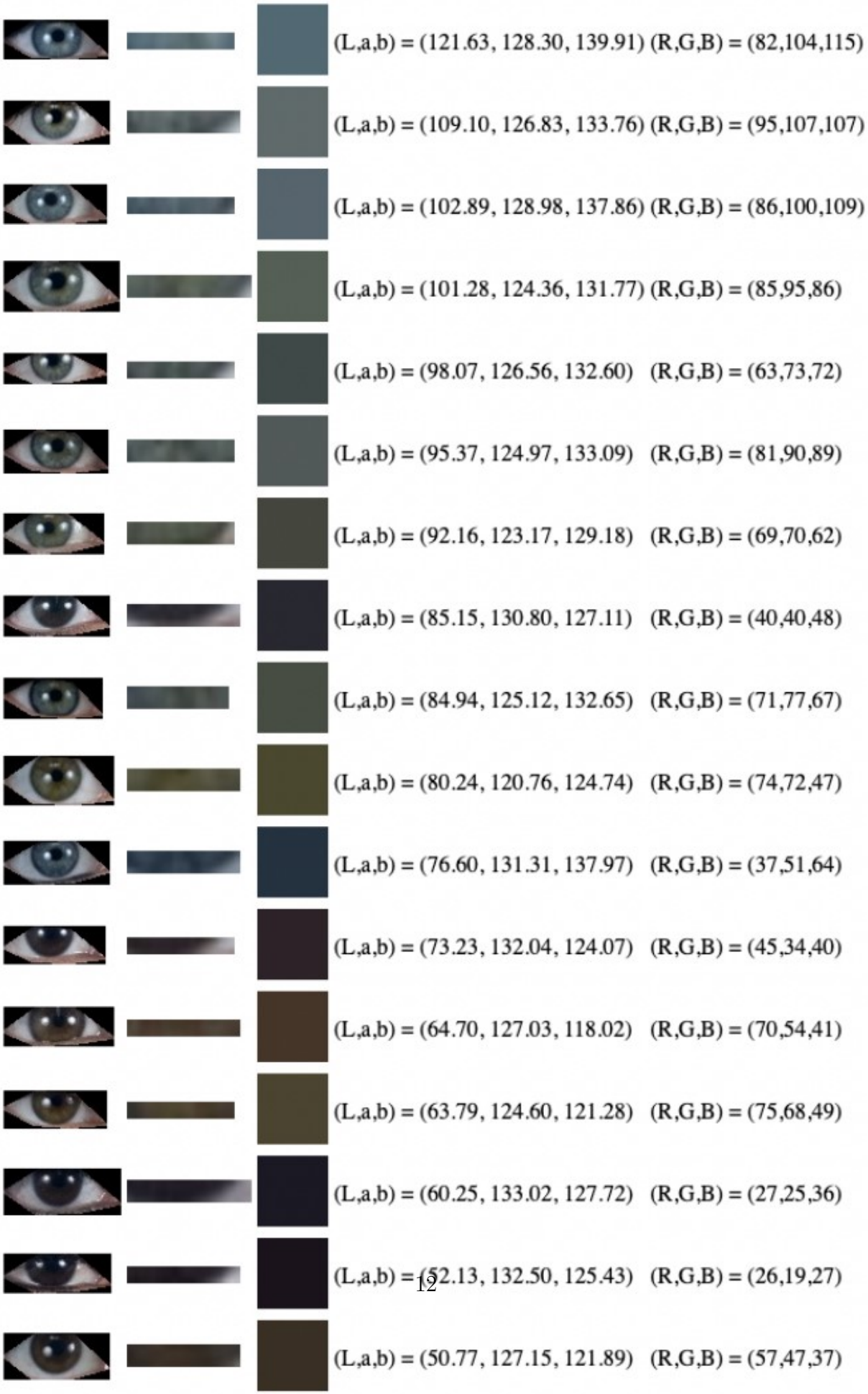
Figure 7: B Values Example

The L, a, and b values will be outputted in a range that is compatible with OpenCV. L values ranging from 0 to 100 will be converted to a range of 1 to 255. For a and b values ranging from -127 to 127, they will be shifted by 128 to a range of 1 to 255.

3.5 Eyes

We can use the images of the eyes collected from the detect_facial_landmarks function to determine the Lab and RGB values of the iris. For this, we can use the find_iris function, which detects a smaller bounding box of the iris beneath the pupil using OpenCV's boundingRect function. A color histogram is created based on the bounding box to calculate the most prominent RGB color. Additionally, the Lab color space of the mask is computed using the get_lab_color_space function. Finally, find_iris returns the most prominent RGB color, the average Lab color, and a new image of the iris.

The following is an example of the iris boxes, RGB, and Lab values ordered by L (brightness) values.


(L,a,b) = (121.63, 128.30, 139.91) (R,G,B) = (82,104,115)
(L,a,b) = (109.10, 126.83, 133.76) (R,G,B) = (95,107,107)
(L,a,b) = (102.89, 128.98, 137.86) (R,G,B) = (86,100,109)
(L,a,b) = (101.28, 124.36, 131.77) (R,G,B) = (85,95,86)
(L,a,b) = (98.07, 126.56, 132.60) (R,G,B) = (63,73,72)
(L,a,b) = (95.37, 124.97, 133.09) (R,G,B) = (81,90,89)
(L,a,b) = (92.16, 123.17, 129.18) (R,G,B) = (69,70,62)
(L,a,b) = (85.15, 130.80, 127.11) (R,G,B) = (40,40,48)
(L,a,b) = (84.94, 125.12, 132.65) (R,G,B) = (71,77,67)
(L,a,b) = (80.24, 120.76, 124.74) (R,G,B) = (74,72,47)
(L,a,b) = (76.60, 131.31, 137.97) (R,G,B) = (37,51,64)
(L,a,b) = (73.23, 132.04, 124.07) (R,G,B) = (45,34,40)
(L,a,b) = (64.70, 127.03, 118.02) (R,G,B) = (70,54,41)
(L,a,b) = (63.79, 124.60, 121.28) (R,G,B) = (75,68,49)
(L,a,b) = (60.25, 133.02, 127.72) (R,G,B) = (27,25,36)
(L,a,b) = (52.13, 132.50, 125.43) (R,G,B) = (26,19,27)
(L,a,b) = (50.77, 127.15, 121.89) (R,G,B) = (57,47,37)

3.6 Hair

The process of determining the hair mask was challenging due to the limitations of the current Dlib model, which doesn't detect points above eyebrows. Initially, we converted the image to grayscale, applied a median blur to reduce noise, and used a flood fill function to fill the hair region with a specific seed point value located at the top middle of the image. This filled hair region was then used to create a hair mask, which was applied to the original image to extract the hair. Finally, the function returns the average pixel value of the detected hair regions in the color image and the Lab color space value of the masked image. Although this method was able to get an accurate result for the average hair RGB value, we wanted to improve on the mask to be able to get more of a range of hair color and Lab value.

We were able to find an enhances facial pre-trained model file: shape_predictor_81_face_landmarks.dat, which identifies an "additional 13 landmark points to cover the forehead area"[6]

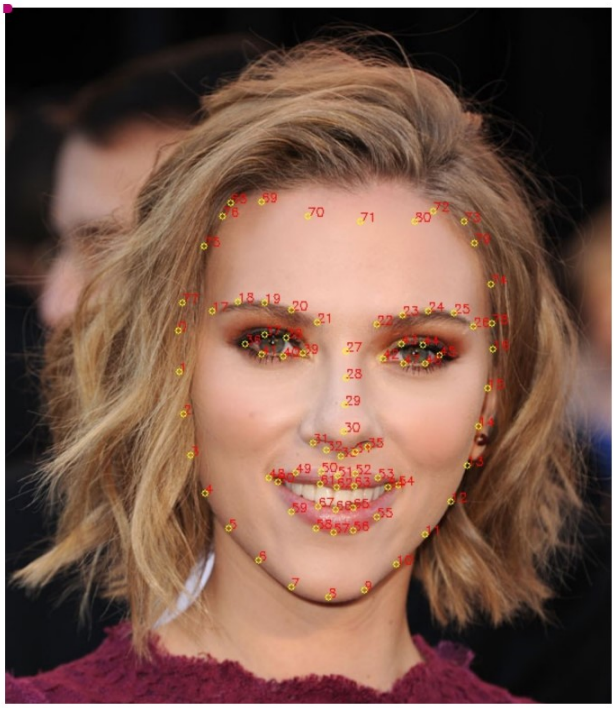


Figure 9: Shape Predictor 81 Face Landmarks Example

The newly added landmark points can now be utilized to create an initial bounding box starting from the top of the forehead going upwards. The get_hair_mask function utilizes landmark points to generate a mask of the hair region. Specifically, it uses points 75, 76, 68, and 69 on the left side and points 72, 73, 79, and 74 on the right side. However, the current points 70 and 71 are situated too low on the forehead. To address this, additional points are created to form a top center point on the forehead. This is achieved by computing

[6]https://github.com/codeniko/shape_predictor_81_face_landmarks

the intersection point between the line formed by points 68 and 69 and the line formed by points 72 and 73. To create a mask similar to the one below, we add extra points for the top-left and top-right corners of the image, as well as for the left and right edges aligned with landmark points 75 and 74.

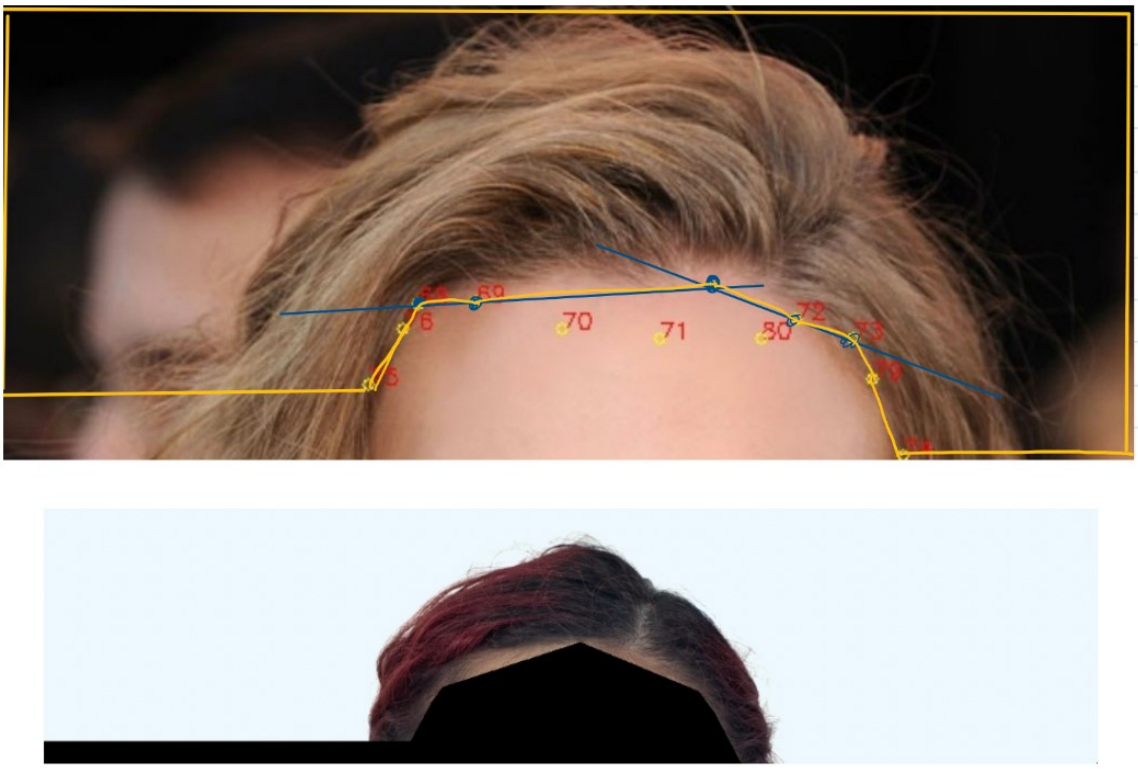


Figure 10: New Landmark Positions

Next in the function, the image is cropped to fit the hair mask created, the image is converted to grayscale and a medium filter is applied to remove noise. Then, a threshold value is applied to create a binary image to create a mask of just the hair without the background or left over parts of the forehead. With the binary mask we find the contours of the hair in the image using the findContours function from OpenCV. The contours are then drawn on a black mask with the same size as the image using the drawContours function. The resulting mask is inverted to select only the background outside of the hair region. Finally, the original image is combined with the mask using the bitwise_or function to produce the final result.

The current threshold value of 70 performs well in generating accurate hair masks for darker hair, but fails to distinguish lighter hair from the background. Our function get_hair_values takes an image and returns a tuple containing the top 3 RGB values, Lab values, and hair

Figure 11: Final Hair Mask

image created by get_hair_mask. If the hair's L value is greater than 50 (indicating high brightness), the threshold value is increased to 120 and the hair mask and Lab values are recalculated before computing the top 3 RGB values. The results obtained using thresholds of 70 and 120 are presented below.

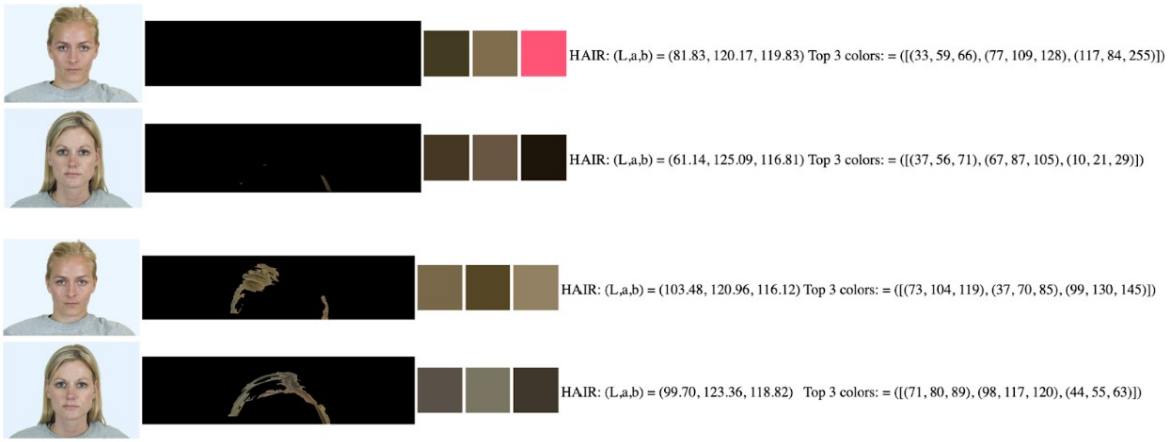


Figure 12: Hair Mask on Blonde Hair

In our implementation, we have chosen to retrieve the top 3 colors and use a threshold of difference in colors to be 25 in the function get_top_color. The function starts by computing the color histogram of the hair mask and excludes the black background color. Then, it sorts the histogram indices in descending order and retrieves the top 3 colors with a value difference greater than the threshold, which in our case is 25. Despite the absence of a complete hair mask in the image above, the colors of the hair are accurately identified, indicating that the system is able to successfully detect and differentiate hair colors even in cases where the full hair mask may not be present.

The following are some examples of the hair mask on multiple images:

Figure 13: Blonde Hair Masks

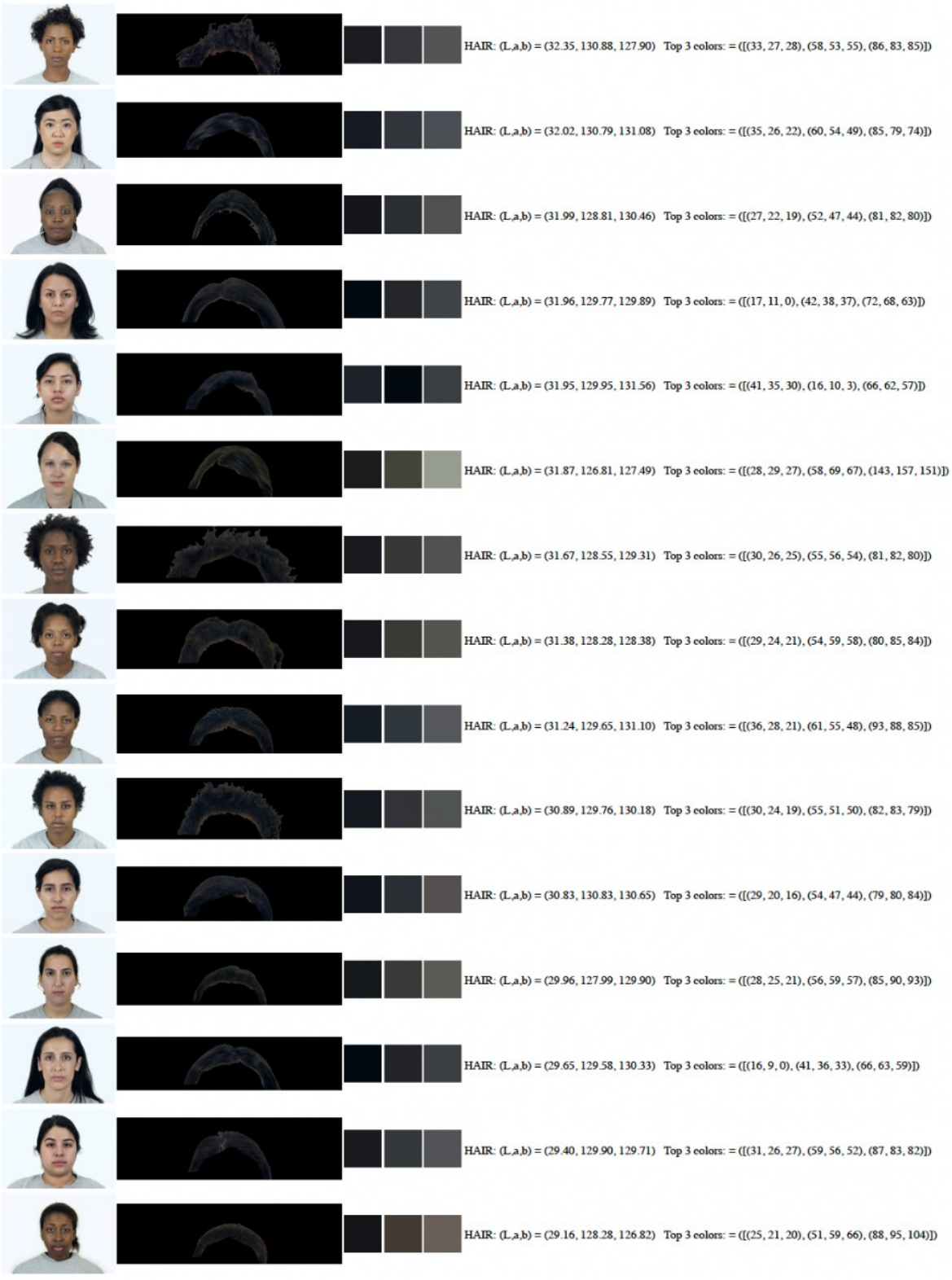


Figure 14: Brown Hair Masks

4 Data Collection

For the start of the system, we used a large dataset of photos from the Chicago Face Database[7] to get data for eye color, hair color, and skin undertone. We found a good standardized dataset of photos that are well lit and high resolution. These photos also include people of all different backgrounds which will be necessary for ensuring diversity inclusion. Due to limits in time, we solely used women from this database to train the program because they do not have facial hair which would cause problems in the future.

Figure 15: Example of Photo from Dataset

Since we trained our program on these photos, we collected our own photos later attempting to match the database domain. In order to make our photos mimic those in the dataset, we used a high quality camera, a Sony a7r II Full-Frame Mirrorless Camera with a Sony FE 100-400mm f/4.5-5.6 GM OSS lens. To portray the bright surrounding lighting we utilized a ring light. The dataset also features a white background which we attempted to copy by having our subjects stand in front of a solid white background.

As shown in fig 16, we captured photos from the shoulders up of individuals who will use our system. As mentioned in section 2.1, we needed photos of the face to capture the eyes,

[7] https://www.chicagofaces.org/

hair, and skin undertone.

Figure 16: Example of Our Visual Data

The photos we collected, though high resolution, did not have a white enough background to be exact to the database photos. We were able to account for differences in these photos by applying a second color correcting algorithm mentioned in section 3.2.

5 Neural Network

5.1 Initial Attempts

With all of the features detected and values derived from them, we could now categorize people into their season. The first thought was a simple decision tree composed of many "if-else" statements. In order to do that, we would need to come up with thresholds for the values of features that would essentially define a characteristic. Here is an example with rough thresholds: Someone with a high a-value for skin and dark eyes is a winter. A high skin a-value could be above 128 and dark eyes could be an RGB triple with each value below 55. It turns out that creating thresholds like this to create a decision tree is not so clear cut.

There is a lot of crossover between the seasons and there are a lot of different features to consider.

To make a more complex decision structure we turned to machine learning. We created a neural network that takes in the values of the detected features and outputs a season. Neural networks can sometimes appear to be a black box where it is hard to interpret why it makes the decisions it does. Our goal was to control as much of the network as possible in order to analyze why it made the decisions it did so it could continuously be improved.

The first step to a neural network is to prepare the data. As a team we went through all 304 photos in the Chicago Face Database and based on our criteria in 2.1, placed each person into a season. The hardest part about doing this was determining skin undertone. Most of the time it was too difficult to choose warm or cool just looking at someone's skin tone. To help we looked at each person's skin a and b values from the Lab color space. By comparing the group's values to each other we got a better sense of what warm and cool undertones looked like. With the help of those values we were successfully able to determine each person's color season. The last step of preparing the target data was to transform each target into a one hot vector. For example summer is encoded as 0 which as a one hot vector is [1, 0, 0 ,0]. This is useful for neural networks where the goal is to put the input pattern into a category, which is exactly what we are doing.

Once we had the data labels we needed to decide what the data patterns would be. To start off we used every piece of data that we had for a person. This turned out to be 21 features: The Lab values for skin, eyes, and hair, and the RGB values for eyes and the top 3 hair colors. With the data features and the data labels set we were able to begin training the network. 80% of the data was used for training, while 20% was used for testing, and 20% of the testing data was used for validation.

The network itself was composed of an input layer that took in the 21 features, followed by two fully connected dense layers of 30 and 16 nodes respectively. The two dense layers used ReLU as an activation function. Lastly was the output layer which was four output nodes each corresponding to a season. The activation function for the output layer is softmax, a commonly used activation for multi-categorical classification. The network was compiled with categorical cross entropy to measure the loss function and used the Adam optimizer[8].

Early stopping was also implemented so that if the validation data stopped improving, the network stopped training. The last thing to do was prepare our data for testing to see how well the neural network does on non database photos. Once again, we decided on the ground truth and prepared the data to be tested on.

[8]https://keras.io/api/optimizers/adam/

The first trial with this data and neural network setup did not go as well as we hoped. The training data only got around 60% accuracy with the validation data well below. The testing data achieved only 37% and only $\frac{3}{13}$ of our own photos were predicted correctly. It is to be expected that the first trials are going to have much worse performance than the eventual output, however these were only slightly better than random.

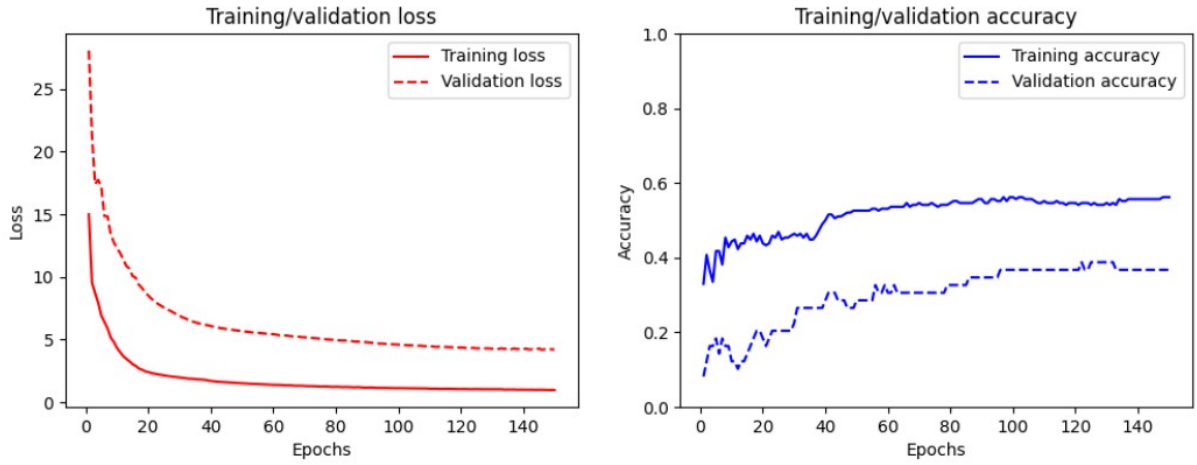


Figure 17

5.2 Improvements

The first major improvement to the network was adding shuffling. The photos in the dataset were organized by race so using the last 20% for testing meant that the testing data was significantly different than the training data. Shuffling was also added to the batches between each training epoch in the neural network. The training accuracy and the validation accuracy were more jointly connected which was a promising sign that the network was truly improving. The testing data success rate rose to 50%, but still only $\frac{4}{13}$ of our photos were correctly categorized.

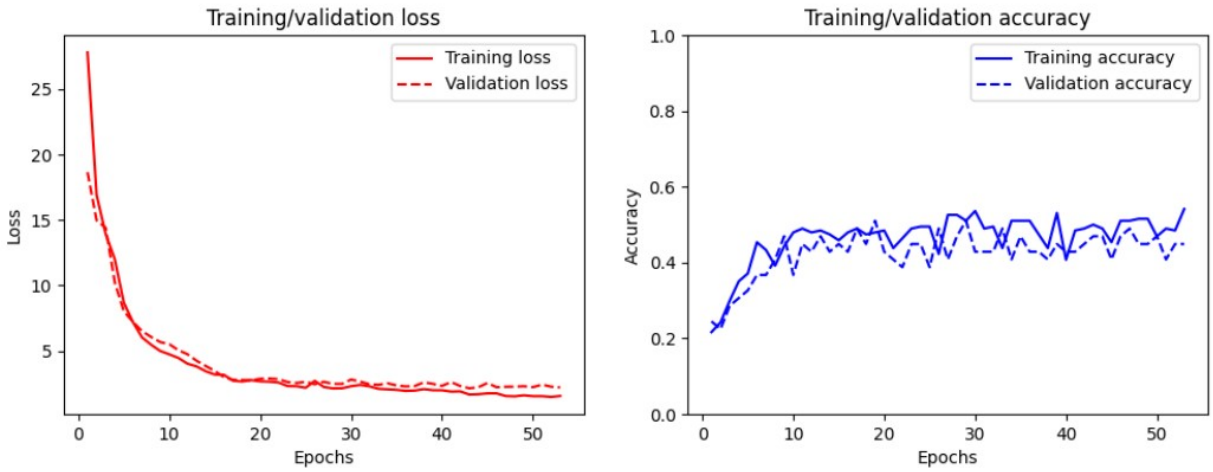


Figure 18

Upon analyzing these results it was clear the network was being trained to choose winter around 75% of the time and summer 25% of the time while never picking spring or fall. One reason for this was that our data was skewed. We classified around 44% of the database photos as winter along with 17% summer, 9% fall, and 30% spring.

The next steps all were implemented together and produced large increases in accuracy giving us the neural network we currently use. The first major improvement was data augmentation. We needed more data to train on which is the major benefit of data augmentation. Before each training sample got added to the training set it was copied three times. Then each feature in the samples got a random number between -3 and 3 added to it. This introduced three similar patterns to the data set but with a bit of variation which helps introduce noise and more overall patterns. The issue of skewed data towards winters still persisted. To solve that we used a library to produce more samples of underrepresented training data resulting in an equal amount of labels for each season. We also switched the training data to include every photo in the database and the testing data was our 13 photos. At this point we now had 1596 training patterns, over five times our original amount. The last thing we did was normalize the data. Normalizing the training data takes away the differences in sheer magnitude of numbers and makes the network look at each feature more equally. Each feature in a pattern was normalized to a value between 0 and 1 based on the minimum and maximum values that a feature could be. Two other small things were added to help tune the network even more. We decided to give the network less inputs. Instead of 21, it now takes in 10 and they are the a and b values of the skin undertone, the rgb values of the eyes and the dominant hair color, and the L-values of the eyes and the dominant hair color. The L-values of eyes and hair help differentiate light and dark in addition to the actual colors of them.

After implementing these changes the network succeeded at predicting all 13 images in the correct season. While 13 images is not a lot of data, it is very impressive it was able to do this. The testing images were taken under different lighting, with a different camera and much less standardized than the database. Even within the 13 images, they were taken under different environments. To be able to categorize all of them is a great success. Below are the final graphs for training/validation loss and accuracy.

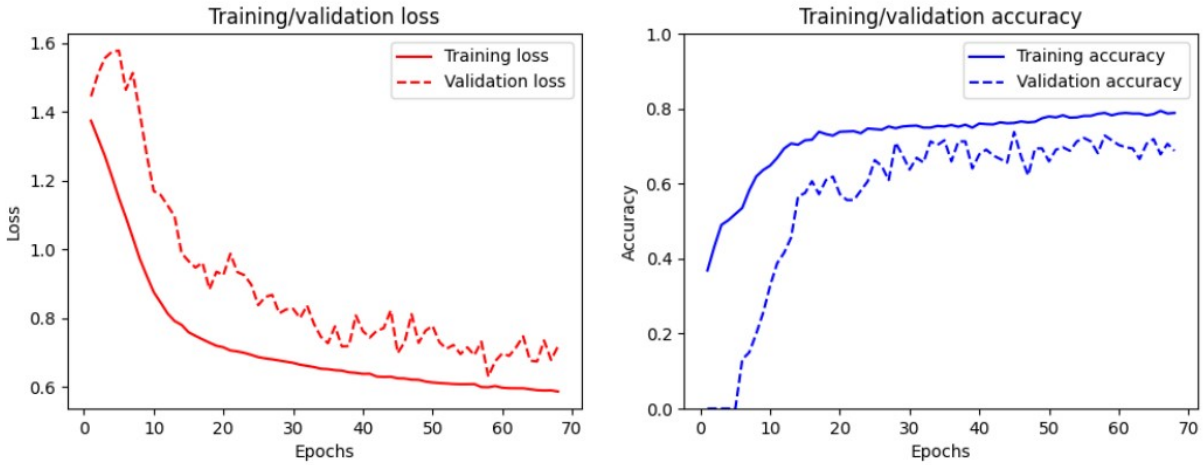


Figure 19

6 Website

We created a website that can be launched locally from the program. The website is simple, it first takes the user to the homepage and explains the project. The user can then move on to try out our color analyzer by pressing the "Try it out!" button.

Figure 20: Front Page of Website

That will take them to a page that explains what type of photo to take and in which format.

After uploading this photo, the page will load and display the original photo uploaded as well as all the masks received from the photo.

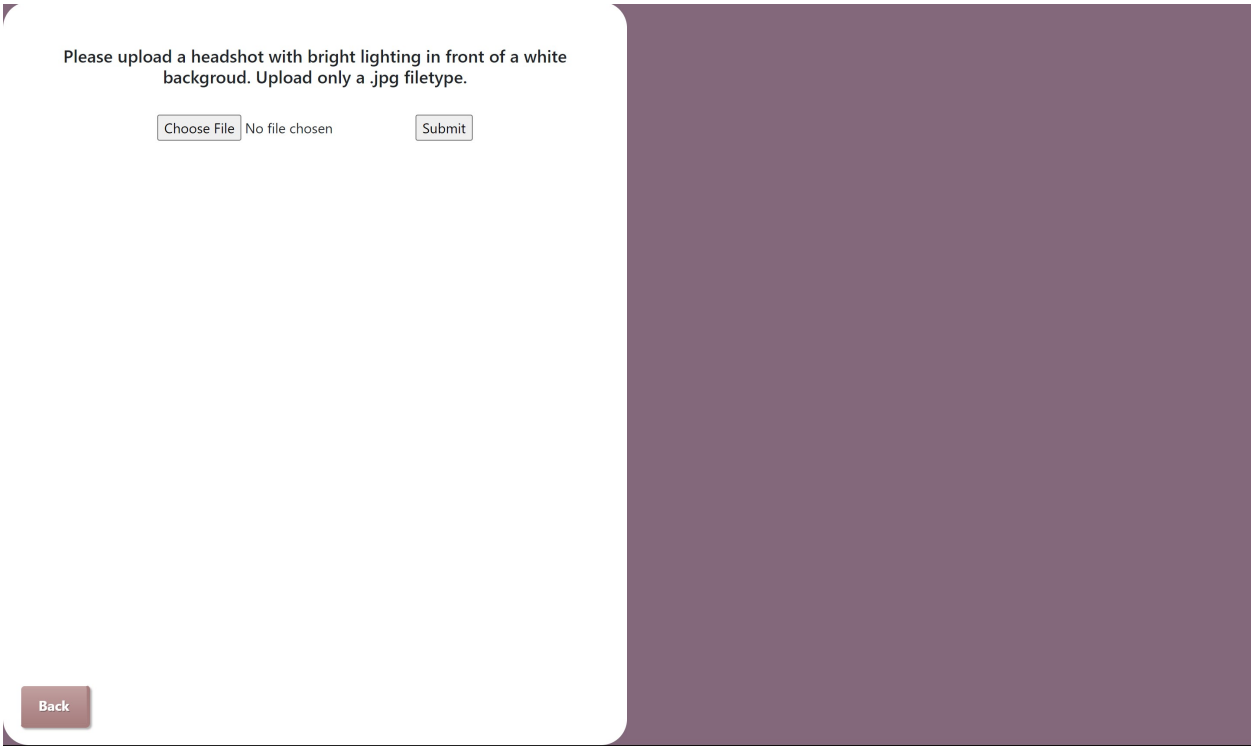


Figure 21: Upload Photo Page

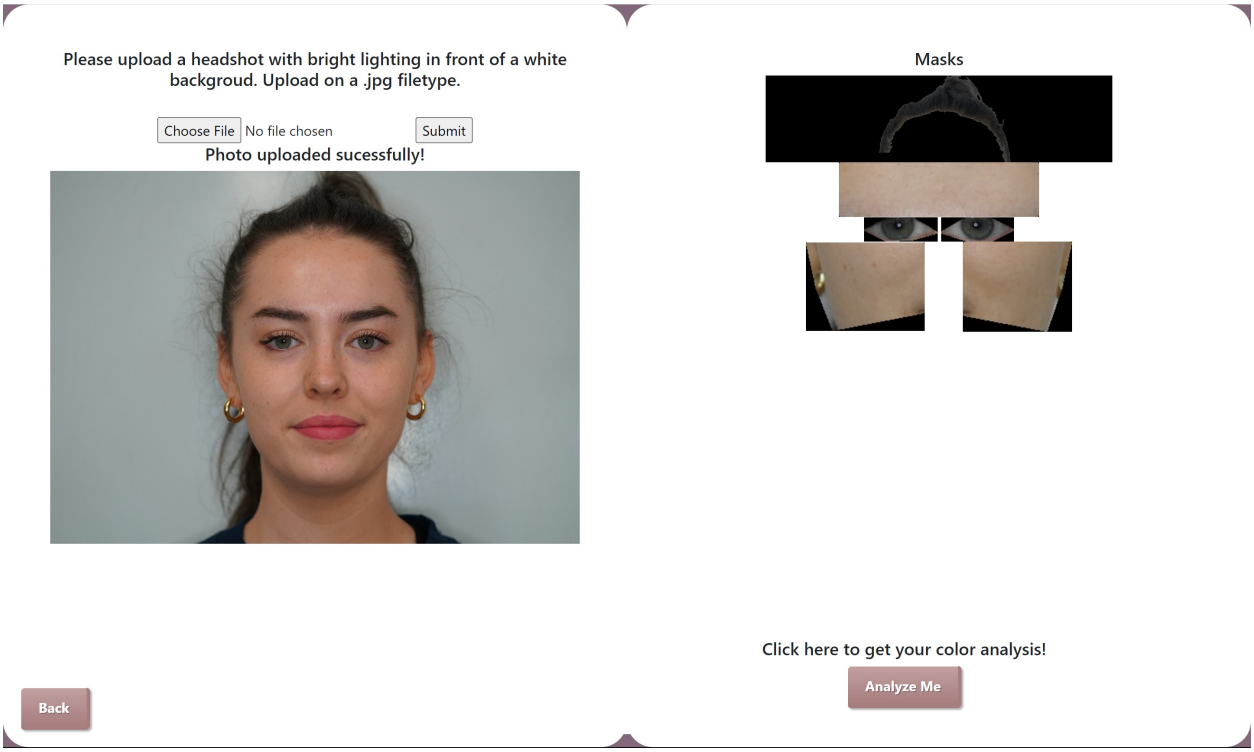


Figure 22: Uploaded Photo with Masks

Lastly the user can press "Analyze me!" and you'll be directed to a new page which shows what season they are and which colors will complement them. These colors are taken from *Color Me Beautiful* and formatted as a color palette.

Figure 23: Final Color Analysis with Color Palette

7 Conclusion

7.1 Evaluation

Overall we believe our system is very successful. We predict all 13 of our own visual data photos correctly (not seen by the network), and 79.6% of the 304 database photos. There are a few interesting parts of the system to discuss. The first major discussion point is the lack of testing data. Our testing data consists of 5 winters, 2 summers, 3 falls and 2 springs. While it is impressive the system got all of them correct, there is not enough data to make the claim it has 100% accuracy, but we can say that it looks like it will still have a very high accuracy. Another interesting point is how we determined what season our database subjects and our testing subjects were. As mentioned before we looked at the a-values and b-values that our system detected to help determine each person's skin undertone. These values are the exact same ones that the machine uses to place each person in a category so it is not surprising that it matched our conclusions on if somebody had warm or cool undertones. We believe this is ok, undertones are hard to detect from just a picture of somebody's face

especially in different lighting environments. Part of the job of a computer system is to aid humans, and that is exactly what it does. The hair and eye colors we used to place people into seasons were determined based on the original images not the machine values.

7.2 Future Improvements

The three of us intend on continuing this project past the time we submit it. In the future we would like to gather more data. One possible route would be to gather data that has been labeled by a professional color analyst and see if the system agrees with their analysis. In our project proposal we discussed the possibility of going further into color analysis giving back a subseason. There are 12 subseasons (3 for each season). This could be a possible feature addition now that the system is in place and working for the four seasons. Another area that can be improved with a bit more time is system robustness. Currently there is very little error checking done. We have made assumptions that the inputted photos will be a woman with hair, centered in the frame, in a well lit room with a solid white background. In the case that these conditions are not met the system will try its best but most likely output a random season. Along with that, we'd like to improve the system's color correcting algorithm. Right now it works best with our own visual data but, we want the program to be able to detect different environments and correct for them based on the original dataset.

7.3 Contributions

This project was completed as a team. We all helped in each area of the system, but we did have focuses. Tiffany McBrayer retrieved the facial features and evaluated the skin undertones and hair color. Alexandra Long color corrected the images, evaluated the eye colors, and created the interactive website. Aaron Ashery helped evaluate the hair color and created the neural network that outputs the final season. Together we collected our own visual data of individuals in our classes. These are the major parts of our system that we each worked on, but once again we all worked together, helped each other, and each did many smaller parts that contributed to the success of the overall system.

We'd like to thank John Kender and Srishti Srivastava for their feedback on the project. Furthermore, the code for this project is available on our GitHub repository, which can be accessed at https://github.com/tiffanymcbrayer/ColorSeasonAnalyzer.

8 References

1 Nicholson, Melissa. "Shop Women's Clothing by Color: Kettlewell Colors USA." Shop Women's Clothing By Color — Kettlewell Colors USA, https://www.kettlewellcolours.co.uk/blog/jo/a-brief-history-of-colour-analysis.

2 "Color-Analyze Yourself like a Pro." Colorwise.me, https://colorwise.me/self-analysis.

3 "Color Analysis - Free Online Personal Color Analysis Test." Color Enalysis, https://www.colorenalysis.com/.

4 Stordahl, Desiree. "How to Determine Your Skin's Undertone: Paula's Choice." Shop Paula's Choice, Paula's Choice, 14 June 2022, https://www.paulaschoice.com/skin-care-advice/skin-care-how-tos/how-to-determine-your-skin-tone-and-undertone?cjdata=MXxOfDB8WXww&;cjevent=66ad3c6ac8ae11ed816591c40a82b82d&;utm_campaign=Skimlinks&;utm_medium=affiliate&;utm_source=cj.

5 Lenahan, By: Brad, and Brad Lenahan. "What Is My Skin Tone? A Guide to Finding Your Undertone." Colorescience, 16 Mar. 2018, https://www.colorescience.com/blogs/blog/how-to-determine-your-skin-tone-before-buying-face-products2.

6 Titus, Missy. "Determine Your Undertone &; Overtone (Once and for All!): Simplified Wardrobe." Missy Titus — Sacramento Personal Stylist &; Capsule Wardrobes, Missy Titus — Sacramento Personal Stylist &; Capsule Wardrobes, 14 July 2020, https://www.simplifiedwardrobe.com/blog/how-to-do-a-color-self-analysis.

7 "How to Determine Your Skin Tone and Undertone: A Guide." PROVEN Skincare, https://www.provenskincare.com/skin-tones/.

8 "Let's Get Started." Start Your Color Analysis Now!, https://www.colorenalysis.com/hair_color.html.

9 fmw42. "How to Do 'Color Cast Removal' or 'Color Coverage' in Python." Stack Overflow, 27 Jan. 2022, https://stackoverflow.com/questions/70876252/how-to-do-color-cast-removal-or-color-coverage-in-python.

10 "Dlib C++ Library." Dlib C++ Library, 2022, http://dlib.net/.

11 Codeniko. "Codeniko/shape_predictor_81_face_landmarks: Custom Shape Predictor Model Trained to Find 81 Facial Feature Landmarks given Any Image." GitHub, https://github.com/codeniko/shape_predictor_81_face_landmarks.

12 CFD, 2015, https://www.chicagofaces.org/.

13 Team, Keras. "Keras Documentation: Adam." Keras, https://keras.io/api/optimizers/adam/.