Hair Type Identifying Model and Product Recommender System

Talia Zalesne, Jordyn Gerstle-Goodman, Joey Athanasiou, Marcos Fontes, Marie Axalan
Leeds School of Business University of Colorado, Boulder
talia.zalesne@colorado.edu jordyn.gerstlegoodman@colorado.edu
joseph.athanasiou@colorado.edu marcos.fontes@colorado.edu
marie.axalan@colorado.edu

Abstract

Hair detection is a newly budding area of AI. It is useful for many applications including digital recognition, surveillance, and hair modeling. However, there is a much more personalized use for this type of technology: product recommendation. Though hair type may seem trivial to some, it is certainly not obvious to everyone. Using a Convolutional Neural Network with hand-labeled training data pulled from Figaro (an image database for hair detection), kaggle, and google image searches, we were able to achieve over 90% accuracy classifying images into four hair types: straight, wavy, curly, and tightly coiled. Based on these classifications and further personalized questions about hair needs, we were able to build a product recommender system that can be adapted by any hair care brand.

1. Introduction

Hair care is often associated as one of the main indicators of a person's hygiene and overall health. Hair is an ever-changing aspect of personal image where hormones, stress, and many other factors can change textures and curl patterns as we age. In fact, 88% of women attribute their hair to their overall self confidence [1]. With an overwhelmingly abundant market, hair care can often be intimidating. One of the most daunting aspects of hair care, however, is often the most overlooked: hair type is the baseline level to entry for many haircare regimens and with so many different types of hair there is no one solution for all.

To combat this difficulty, we have developed an AI application that can identify hair types with 91.25% accuracy and subsequently recommend products based on specific consumer wants and needs. The use of this application will help its users to understand the comprehension of their hair and how they can properly and effectively maintain it. Benefits extend from pure aesthetics of an individual to overall health implications. Further, the application is fully customizable and can be adapted by any brand, from luxury to casual hair care, allowing consumers of all levels of hair knowledge to receive personalized recommendations for their own needs in a very cost efficient manner.

1.1. Related Work

We were aware going into this project that there are a few applications already attempting to accomplish similar goals. We felt, however, that our use case of a personalized product recommendation system sufficiently set us apart from pre-existing hair recognition programs.

Before we began modeling and searching for data, we did some research into similar projects. One of the main resources we uncovered was a publication from Image and Vision Computing, Volume 71 titled *Hair Detection, Segmentation, and Hairstyle Slassification in the Wild* by Muhammad and colleagues. In the abstract, they highlighted the many applications of hair detection stemming from how much human appearance is characterized by hair type. Their approach to the problem was as follows: "We first build a hair probability map by classifying overlapping patches described by features extracted from a CNN, using Random Forest. Then modeling hair from high probability regions, we segment at pixel level uncertain areas by using LTP features and SVM." For their experimentation, they utilized Figaro1K, a dataset they had composed as an extension of Figaro of more than 1,000 unconstrained images, with 7 eventual classifications: straight, wavy, curly, kinky, braids, dreadlocks, and short-men's hair [2].

The CNN built by Muhammad and colleagues was based on CaffeNet, a variant of AlexNet. CaffeNet is a popular CNN for classification which is utilized for large scale visual recognition. It specifically thrives because of its expressive architecture and speed, meaning a lot of the

optimization is hard coded and able to process over 60 million images per day.

This research into related work informed our project in a few ways. First, we also chose to employ the Figaro1K dataset as a jumping off point in our data search. Further, we were able to observe and experiment with the logic from this modeling but chose to use VGG instead of AlexNet in our final model. All of these choices are explained in more detail in subsequent sections.

2. Data

The data gathered for our classification model included a total of 1,504 images of women and men with longer hair. We gathered the data from two key places: Figaro 1K and Kaggle. Figaro 1K is a database containing 1,050 images belonging to seven different hairstyles (Straight, Wavy, Curly, Kinky - or Tightly Coiled as we labeled it, Braids, Dreadlocks, and Short). Since we are interested in only the four main hair types, we scrapped the latter portion of the data leaving us with a total of 600 images. This database contains only unconstrained view images, meaning it is made up of professionally taken photographs meant to highlight the hair specifically. Most of the photos were taken from the back and there are no selfies or lower quality photos included in the database [3].

The data gathered from Kaggle amounted to 904 images pre-classified into three folders: Straight, Wavy, and Curly. Since this data did not make a distinction between Curly and Tightly Coiled hair, we entered into the folder containing curly data and split it by hand. The images from Kaggle were scrapped from a google image search and subsequently sorted out by the author [4]. This data added an element of less intentional photographs to the more professional pictures we had gathered from Figaro 1K. Since we foresee our model being used by regular consumers who would likely be snapping their own pictures to upload, we felt this element of chaos in the training was very important.

Figure 1: Sample Image Inputs
Left to Right: Straight, Wavy, Curly, Tightly Coiled

2.1. Cleaning

For the most part, the data we gathered came pre-labeled. Thinking we would be good to go, we set aside some data for testing and got right into modeling. After attaining generally poor accuracy results we realized we would need to re-examine the data. It was shocking to see how much of the data was labeled inconsistently or flat out incorrectly. Examples of these disparities included multiple images appearing under two or more of our labels or very similar photos being labeled in differing ways. The majority of our time at this point was spent re-labeling all of our images as this seemed like the most effective way to generate accurate modeling results. Through this process, we threw away 480 images - either because they were repetitive or too ambiguous.

2.2. Final Dataset

Our finalized dataset for training consists of a total of 1,002 images split into training, validation, and testing. The training data contains 301, 227, 221, and 151 images labeled Straight, Wavy, Curly, and Tightly Coiled respectively. Further, each label has an additional 36 images set aside for validation and 20 set aside for testing. The images were all shuffled together before being split into train/validation/split to ensure there would be equal representation from our two data sources.

2.3. Preprocessing

In order to pass the images we gathered into our Convolutional Neural Network, they first had to be preprocessed. We accomplished this using the ImageDataGenerator class from the Keras package. ImageDataGenerator applies image augmentation to the dataset, adding transformations to the photos to expand the training data and help avoid overfitting. The generator has many different features, such as *rotation range, width shift range* and *height shift range, brightness range, shear range, zoom range, horizontal flip* and *vertical flip,* and many more. All of these provide different transformation methods to ensure the model receives a variety of training images.

After playing with different variations of the aforementioned settings, we found that our model performed best with two simple settings. We first employed *rescale = 1.0/255* which essentially rescales the pixel values between 0 and 1 to allow our model to easier process the numbers. Second, we utilized *data_format = 'channels_last'* to tell the model the ordering of the dimensions of the inputs, in our case: (batch_size, height, width, channels). Finally, we read our images into the notebook in three steps: training, validation, and testing. For all three, we used a batch size of 8 and a target size of (224,224). The batch size sets the number of samples that

will be propagated through the network during each round of training. A lower batch size has the advantage of using less memory but can result in less accurate gradients. The target size is used to resize all of the images to match the input image shape the final model is expecting - in our case, the VGG16 model expects 224x224 images by default.

2.4. Recommender System Dataset

There are a few different types of recommender systems but the one that fit our needs best was a knowledge based recommender system - which makes recommendations based on functional knowledge about the items within the system. Our system specifically allows users to submit their needs and preferences which can then be run through our knowledge database and output the appropriate products based on their hair classifications and personal product preferences.

In order to successfully build our recommender system, we first had to gather a dataset. Since this model is meant to provide a demonstration of how our program can be mounted on any company's website, we decided to use a well-known brand that has a wide variety of products readily available: Pantene. We began by scraping the Pantene website and pulling a list of their products and ratings - 54 products in total [5]. The products were not marked with hair type information so we had to conduct our own research into each hair type's specific needs in order to create our own labeling. We accomplished this using binary encoding for each product - 1 if the product was compatible with the hair type and 0 otherwise.

Once fully compiled, the structure of the dataset was 54 rows (one per product) and 8 columns - Product Class, Product Name, Product Purpose, Rating, and the four hair type columns. The dataset contains 7 unique classes of products: 8 conditioners, 2 dry shampoos, 4 serums, 12 shampoos, 7 shampoo/conditioners(co wash), 14 styling products, and 7 hair treatments. In addition, the data contains 11 unique product purposes to provide users with further customization options - hydrate, style, color care, repair, volume, and more. We formatted this data intentionally to be easily accessible to our recommendation system. As a further step, we included consumer ratings since some combinations of consumer needs could return multiple product options. For example, someone with wavy hair in search of hydrating shampoo would be recommended *Miracle Moisture Boost Shampoo with Rose Water* and *Hydrating Glow Shampoo with Baobab Essence.* However, to further help inform their decision, they would also see that the latter has a rating of 4.9 while the former only has one of 4.2.

3. Methods

3.1. Convolutional Neural Network

Our final model is a Convolutional Neural Network created with the Keras package in Python. A Convolutional Neural Network (CNN) is a deep learning method that specializes in processing data with a grid like topology (such as an image). Computers process images as a matrix where each entry represents the value of brightness and color for each pixel (where 0 represents black and 255 represents white). The CNN then uses kernels (a matrix of the set of learnable parameters) and computes the dot product with a portion of the image matrix, moving through the whole image. This allows the algorithm to learn patterns within the images and, with deeper and deeper CNN's, patterns of those patterns [6].

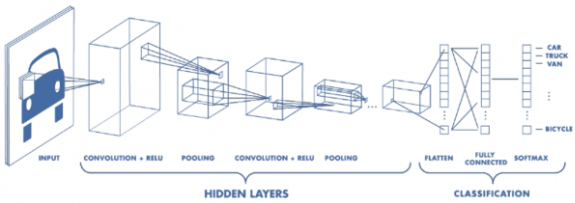

Figure 2: Typical Architecture of a Convolutional Neural Network

3.2. VGG16 Model

In order to create the most effective model we could, we employed transfer learning. Transfer learning is a method in which we start with a powerful pre-trained deep learning model and essentially repurpose it to classify our task.

The VGG16 Model is a CNN that won ILSVR (Imagenet) in 2014 with its powerful machine vision capabilities. The architecture of VGG16 is distinct from other CNNs because it is consistently arranged throughout - convolution layers with a 3x3 filter, stride of 1, and same padding followed by a max-pooling layer with a 2x2 filter and stride of 2 [7]. Overall, VGG16 is composed of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers making the number of layers having tunable parameters 16 (hence the name). The number of filters in the first block is 64, then this number is doubled in the later blocks until it reaches 512. This model is finished by two fully connected hidden layers and one output layer. The two fully connected layers both have 4096 neurons and the output layer consists of 1000 neurons corresponding to the number of categories of the Imagenet dataset [8].

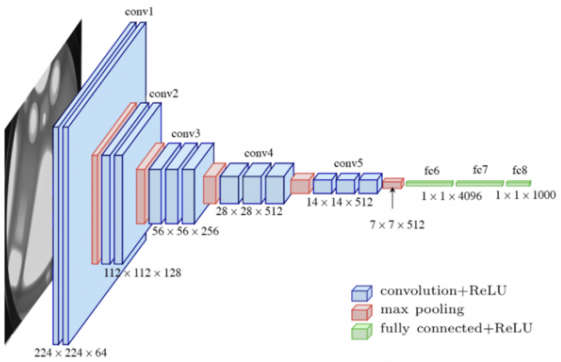

Figure 3: Architecture of VGG16 CNN

When employing transfer learning in Keras, it is common to set the *include top parameter* to False which means the output layer described above is not loaded with the rest of the model. Instead, we add our own output layer to the top corresponding to the predictions we aim to make. In our case, a fully connected layer with 4 neurons for the four hair types.

With transfer learning, it is typical to freeze the weights in the loaded model and only learn weights for the layers added on top. We employed this tactic on our model for the first run and then did another run through our model with the weights unfrozen, but the learning rate was set very low in order to employ the fine tuning in accordance with other specifications that were previously set.

3.3. Final Model and Experimentation

In trying to tune our model we played with a few things. First, we tried different models for our transfer learning including VGG19 and ResNet50, but found overall that VGG16 was producing the best results. We then played with how many layers to add on top of the VGG16 model and the different hyperparameters to use when compiling and fitting the model. Our final model uses the VGG16 model with three dense layers added on top of the flattened layer. The two dense layers both have 4,096 neurons and use the *'relu'* activation (The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero [9]). The last dense layer is our output layer. It has 4 neurons - 1 for each hair type - and uses the *'softmax'* activation function (Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector [10]). We also added dropout here to remove 50% of our neurons and help avoid overfitting.

For our first round of training, we froze the weights in the VGG16 layers and compiled the model with the Adam optimizer (which is an extension of Stochastic Gradient Descent) using a learning rate of 0.001 and categorical cross-entropy for our loss function. We ran the model for 20 epochs, taking 15 steps per epoch, and employed a callback function to ensure the model didn't overfit with too many epochs. For the final round of training, we unfroze the VGG16 weights and reset the learning rate to 0.00001, then ran another 20 epochs with 15 steps per epoch. After loading the weights from our best saved model, we were able to achieve 91.25% accuracy and 0.34 loss.

```
model.evaluate(test_generator)

10/10 [==============================] -
[0.3395267426967621, 0.9125000238418579]
```

Figure 4: Final Model Evaluation on Testing Set
Read From Left to Right: [Loss, Accuracy]

4. Empirical Applications, Experiments, and Results

4.1. Empirical Application

In order to test our application, we tested the whole process using images of our group members' hair, where we have already self-identified what classifications to expect. As explained in more detail below, we did not utilize photos of Joey or Marcos for this step because our classifier is not set up for the majority of men - instead, we substituted a friend whose hair is wavy. As we hoped, the CNN was able to correctly identify all four of our hair types.

In the next step, our application prints the photo uploaded by the user and their hair type and then asks them what sort of product they're interested in (shampoo, conditioner, etc.). At this step, the program also prints a list of products available for the specific hair type since it is not uniform among each option. For example, straight hair naturally allows for oils from their scalp to moisturize, while the curlier your hair, the harder it is for the oils to reach the ends of the hair. This would affect the recommendations in that, if you have curly or tightly curled hair, you won't be suggested products with heavier oil base. After the user enters their choice and the program has confirmed that their input is valid, it moves on to ask what sort of features the user desires (hydration, repair, etc.), again first printing a list of options in our database. Finally, based on the user input from the questions, our program prints all of the products in our database meeting the requirements along with their user ratings.

4.2. Results

Here we will show some examples of what the output of our model looks like. First, we show the correctly classified photos of our teammates and friend we used to test our model.

Figure 5: Classification Identified by Model
Left to Right: Straight, Wavy, Curly, Tightly Coiled

Further, below is a screenshot of our final model output. This testing case used a different photo of Talia, which was also correctly classified as tightly curly. After completing the questions asked by the program and indicating interest in a "treatment" for "protection", it recommended "Split Ends Treatment" which has a consumer rating of 4.6. It can also be seen how our program handled the misspelling of "Treatment" and asked the user to double check their spelling before moving on. Keep in mind that we have not deployed this model to a formal website. Our testing is still being done out of our Google Colab Notebook so the results do not have a very clean look.

Figure 6: Recommender System Demonstration
Recommendation for an Individual with Tightly Coiled Hair Interested in a Protective Treatment Product: Split Ends Treatment (4.6 Rating)

4.3. Experiments - LIME

In order to understand what our model was looking at to make predictions, we enlisted the help of a local surrogate model. LIME (Local Interpretable Model-Agnostic Explanations) is an interpretable model that can be used to explain individual predictions of a not very interpretable deep learning model. Surrogate models are trained to help approximate the predictions of a black box model. This can be done either globally by attempting to explain how the model as a whole is working or, in the case of LIME, locally by looking at individual predictions and determining how the model came to its conclusion. LIME works by generating a new dataset consisting of perturbed samples of an instance and the corresponding predictions from the model. It then trains an interpretable model on this new dataset which is weighted by the proximity of the sampled instances to the instance of interest [11].

For a computer vision application, the output of LIME essentially shows us which part of the image was important for the resulting prediction. This is important to ensure that our Neural Network is in fact looking at the hair and not picking up on any other pattern that happened to be present within the training data. In our endeavor to use LIME for testing, we conducted some research and enlisted the help of available open source code [12]. In the image below, we show an output example of one of our photos. The area highlighted in green represents the part of the photo that the model used to determine Talia has tightly coiled hair. It is clearly highlighting Talia's hair and ignoring other areas of the photo such as her t-shirt.

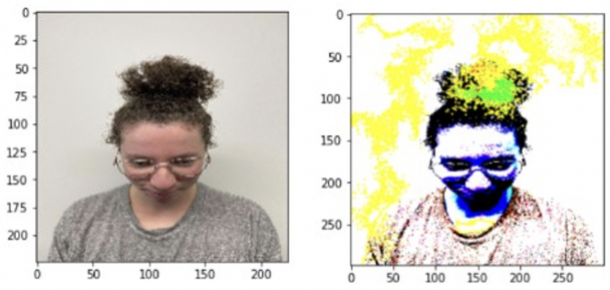

Figure 7: LIME Output
The Highlighted Green Area Represents the Model's Focus

Overall, we were very pleased with the results of our LIME experimentation but did notice a few instances where the program may have been having a harder time with darker hair colors. Though they were still correctly identified, we theorized that lightening the hair to make it more distinguishable may provide better results. Further, we may ask our users to aim for a light background in the photos they upload to ensure better accuracy. Some examples of this follow:

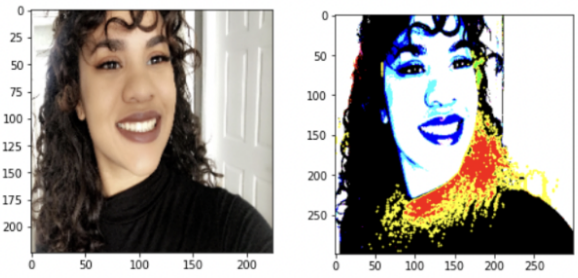

Figure 8: LIME Output
Though The Model Sees Jordyn's Hair, It Also Sees Her Black Shirt

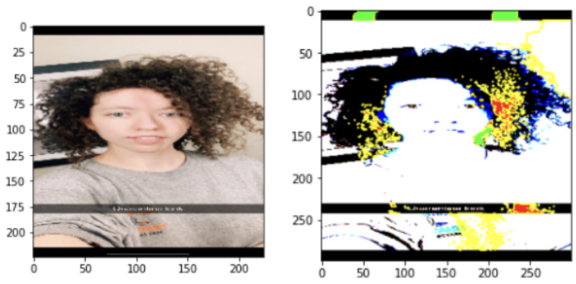

Figure 9: LIME Output
Though The Model Sees Talia's Hair, It Also Sees The Black Caption and Space at the Top of the Photo

5. Conclusions and Recommendations

Overall, the final result that we achieved is something to be proud of. Our model is able to discern with 91.25% accuracy if a person has straight, wavy, curly, or tightly coiled hair and is then able to make recommendations for products based on the wants and needs of that individual. Further, after adapting LIME, we were able to see that our model is looking at the hair of each individual to make its decisions. In working through this project, we learned a lot about applying different Artificial Intelligence models. It was amazing to see how quickly the transfer learning provided meaningful results but we were surprised that adding additional layers did not seem to help. Further, we noticed ourselves applying everything we have talked about in the past week of class. Some examples of this include taking a more data-focused approach with our improvements, implementing LIME to test our model, and thinking about the potential social implications of our model in trying to ensure we do not produce racist results (more on this below). To conclude, we will lay out some of our model's limitations, business implications, and next steps.

5.1. Model Limitations

Though we were able to attain very high accuracy with our model, we wanted to touch on some of the limitations we ran into while building and tuning it. Mainly, limitations relating to our choice of using data-centric AI methodology and ethical modeling.

5.1.1 Data Centric Artificial Intelligence

Our first limitation surrounds the data we had available. We made sure to set aside the same amount of validation and testing data for each of the four class labels but were left with progressively less data in each of our training sets. One of the reasons for this discrepancy goes hand in hand with the simple fact that we are not hair experts ourselves. Our biggest focus in model tuning was refining our data which led us to throw a lot of it out simply for the reason of it being ambiguous. With the help of a professional to classify our training data, we would have been able to keep more of the original photos we had and in turn improve accuracy on a larger variety of photo types. An example of what our model is unable to classify well is photographs of men. With our lower level of experience, these photos were too ambiguous for us to classify so we ended up throwing them out to ensure we could maintain consistency within our training data. We believe someone with more experience would have been able to help determine the hair type of the men in our training photos.

5.1.2 Ethical Modeling

Another consideration with our model was the potential problem of racism. Almost all of the photos we have of people with tightly coiled hair are represented by people of color. Though this may be relatively realistic, we wanted to ensure that our model was actually looking at hair and not just classifying any African American as having tightly coiled hair - which is not realistic. Our experimentation with LIME as well as the model accurately classifying Talia's photo as tightly coiled gives us hope that we were able to overcome this implication but further data would have also helped us to create a model that can classify a diverse group of people.

5.2. Business Implications

This project goes beyond the basic computer vision task, it also provides the framework to make targeted recommendations for consumers that are motivated both by their needs and what's popular. The idea is to supply hair care companies with an interactive method for recommending their products to online consumers. For our purposes, we chose to use Pantene products because they carry a very wide range of products for all different demographics. However, our long term vision for this system is to open it to more luxury and exclusive brands to allow them to make real time recommendations to their customers - who may not all be experienced with hair needs. This will allow less informed consumers to receive personalized recommendations for more complex and luxury brands without the need for an interaction with an

expert stylist or the extra step and cost of a salon appointment.

5.3. Next Steps

We understand that this sort of project can seem trivial to many people: why do we need a deep learning model to tell them if they have straight, wavy, curly, or tightly coiled hair? Though this may be true for some, these four types of hair are not the limit of what this model could help identify. There are actually 12 distinct types of hair, explained in the chart below [13]:

Type	1 (Straight)	2 (Wavy)	3 (Curly)	4 (Coiled/Kinky)
A	Fine, Thin hair, Prone to oil	Fine (Has S-shape)	Fine, Loose curls	Tight, Springy coils
B	Medium, Some volume	Medium (Has S-shape with some frizz)	Medium or Tight curls	Z-coils
C	Coarse, Thick, Won't hold curls	Coarse (Has S-shape & prone to frizz)	Tight, Thick curls	Very tight, Coarse coils

Figure 10: The 12 Hair Types

Classifying one's own hair to this level of detail then becomes much less trivial for the average person. With more time, we would gather a sufficient amount of data and enlist the help of a professional to classify into each of these 12 categories and, therefore, be able to provide consumers with even more specific results and targeted product recommendations. Further, we would like to be able to generalize our model to help make hair health recommendations specific to each person. With more zoomed in and higher resolution photos, we would teach our model to identify split ends and different textures– showing damage to hair follicles. These improvements would also provide us the tools to make stronger and more targeted determinations of how to care for the hair and which products will be complementary to that.

References

[1] *Why Healthy Hair Is Important To Your Overall Well-Being*. (2020, June 20). Grow Knoxville. Retrieved April 25, 2022, from https://growknoxville.com/why-healthy-hair-is-important-to-your-overall-well-being/

[2] Muhammad, U. R., Svanera, M., Leonardi, R., & Benini, S. (2018). Hair detection, segmentation, and hairstyle classification in the wild. Image and Vision Computing, 71, 25-37.

[3] Riaz, M. U., Svanera, M., & Benini, S. (2017). *Figaro 1K*. http://projects.i-ctm.eu/it/progetto/figaro-1k

[4] Bhatia, V. (2020). *The Tree Hair Types*. Kaggle. https://www.kaggle.com/datasets/vyombhatia/the-three-hair-types

[5] *Pantene*. (n.d.). Pantene: Hair Products For All Hair Types. Retrieved April 25, 2022, from https://pantene.com/en-us

[6] Mishra, M. (2020, August 26). *Convolutional Neural Networks, Explained | by Mayank Mishra*. Towards Data Science. Retrieved April 25, 2022, from https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[7] Thakur, R. (2019, August 6). *Step by step VGG16 implementation in Keras for beginners*. Towards Data Science. Retrieved April 25, 2022, from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[8] Le, K. (2021, March 25). *An overview of VGG16 and NiN models | by Khuyen Le | MLearning.ai*. Medium. Retrieved April 25, 2022, from https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models-96e4bf398484

[9] Brownlee, J. (2019, January 9). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Machine Learning Mastery. Retrieved April 25, 2022, from https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[10] Brownlee, J. (2020, October 19). *Softmax Activation Function with Python*. Machine Learning Mastery. Retrieved April 25, 2022, from https://machinelearningmastery.com/softmax-activation-function-with-python/

[11] Molnar, C. (2022, March 29). 9.2 Local Surrogate (LIME) | Interpretable Machine Learning. Retrieved April 25, 2022, from https://christophm.github.io/interpretable-ml-book/lime.html

[12] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier." In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining,* pp. 1135-1144. 2016. <https://github.com/marcotcr/lime>

[13] *How to determine your hair type - A complete guide*. (2020, June 23). Papilla Haircare. Retrieved April 25, 2022, from https://www.papillahaircare.com/research/a-complete-hair-type-guide