SFL SCIENTIFIC
YOUR DATA. OUR SCIENCE. DISCOVER MORE.

Deep Feature Interpolation for Facial Hair Transformation

Issued June 21, 2019

SFL Scientific LLC
3 Batterymarch Park
Suite 405
Quincy, MA 02169

Prepared By:
Michael Luk
Chief Technology Officer
mluk@sflscientific.com

Ryan Hedges
Data Scientist
RHedges14@gmail.com

Table of Contents

Executive Summary

Deep Feature Interpolation (DFI) is a method that leverages feature representations to apply image transformations via an interpolation vector.

For this project, a DFI algorithm was implemented to transform facial hair. In particular, the algorithm is trained to add and remove attributes such as beards and mustaches.

A Fast DFI algorithm was also created; this architecture was based on work in Fast Style Transfer where a network is trained with the ability to perform a single forward pass on an image to generate a facial transformation mask.

Various nuances and additional enhancements to further improve the algorithm are also discussed.

Introduction and Background

The study of the human face using computer vision has been a widely explored topic over the last decade. Tasks such as facial detection, recognition, landmark detection, and expression detection have seen tremendous progress.

In particular, facial attribute transformations have been a common topic of interest over the past few years. Variational Autoencoders and other techniques have been relatively successful at this task [1, 2].

In June of 2017, researchers from Cornell University introduced Deep Feature Interpolation where facial transformation results outperformed all prior techniques, while simultaneously avoiding the use of heavily sophisticated networks.

Objective

The objective of this project is to leverage Deep Feature Interpolation (DFI) techniques to apply facial hair transformations to facial images. Multiple implementations will be explored. An approach is first constructed that involves an individual training optimization phase for each image before extending the project to an enhanced implementation where a single neural network will be trained per transformation type that has the ability to generalize to any input image, thus providing for a functionality where the transformation can render in a matter of seconds.

Dataset and Preprocessing

Dataset Description

The **CelebA** [3] dataset is used for this project, consisting of 203K images of roughly 10K subjects, all of which are celebrities across a variety of professions. The images are accompanied by 40 binary attribute annotations covering various facial characteristics such as expression, hair color, and gender - a full list of attributes can be found here [3].

Figure 1

Exploratory Data Analysis and Preprocessing

The first step in the project is to learn about the data as it may relate to the objective. First of all, the resolution of the images is relatively weak (218 by 178), which may contribute to transformations of weaker clarity later on.

58% of the images are female, resulting in 118K female images and 88K male images. Since the primary objective is to manipulate facial hair, the female images are discarded for now. However, female images could be useful later on, as future enhancements could expand the attribute transformation options beyond facial hair.

Of the 88K male images, 38% have a beard, equating to 33K bearded individuals. Additionally, among the bearded men, 13K have a goatee, 22.5K have 5 o'clock shadow, and 8K have a mustache. (Figure 2)

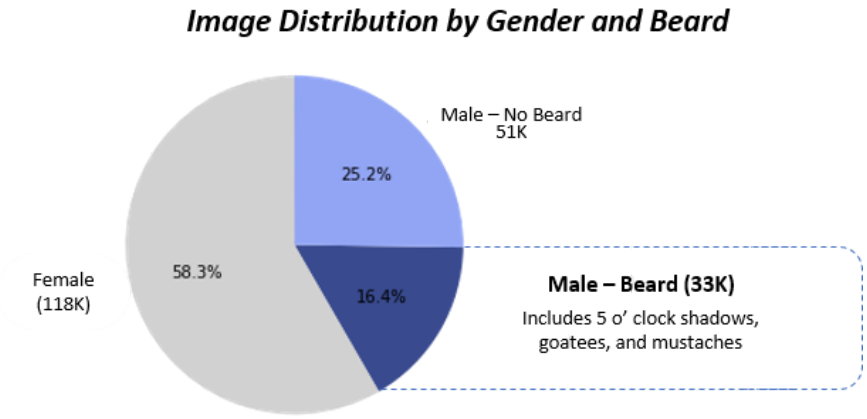

Figure 2

Prior to conducting further EDA, a few preprocessing steps are employed to extract important features from the images.

- The first step in processing the data is to identify the faces within the images. ***dlib's get_frontal_face_detector*** model (HOG + SVM) is used to identify faces in an image and create a bounding box around the face (Figure 3). Note that the model is unable to locate faces in 6K images. Upon inspection, this appears to be due to objects obscuring faces (ex. tennis racket) or faces that are oriented significantly off-center (Figure 4). These images are discarded for this analysis. The large size of the full dataset provides for the convenience to comfortably discard potentially problematic images. Further studies could look into examining how the inclusion of these obscured images would impact results.
- ***dlib's 68_face_landmarks_predictor*** (ensemble of regression trees) is applied to locate and denote 68 facial landmarks outlining the face, mouth, nose, and eyes, which will be useful for the analysis (Figure 3).

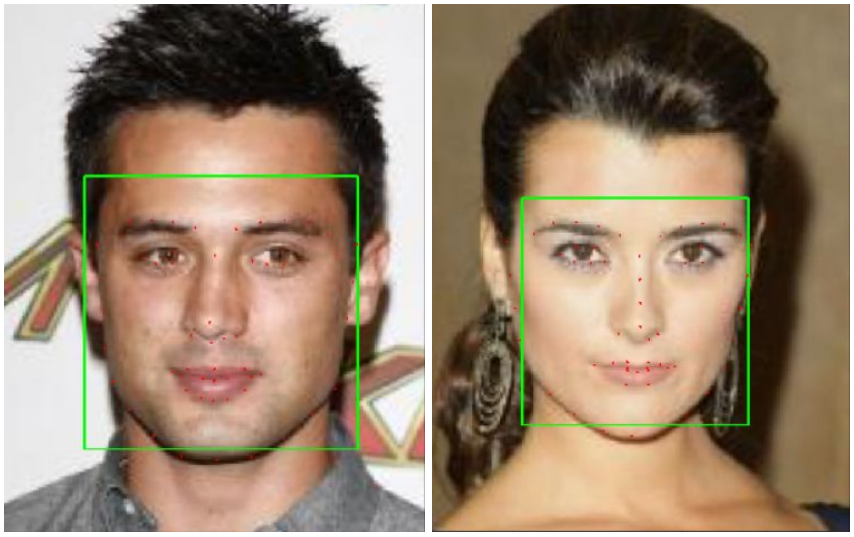

Figure 3

Figure 4

An unsupervised learning method tSNE on the facial landmark data is employed to better understand the shape of the data and identify any nuances to be aware of. tSNE results visualized using a Kernel Density Plot highlight the large variety in facial orientation within the dataset (Figure 5).

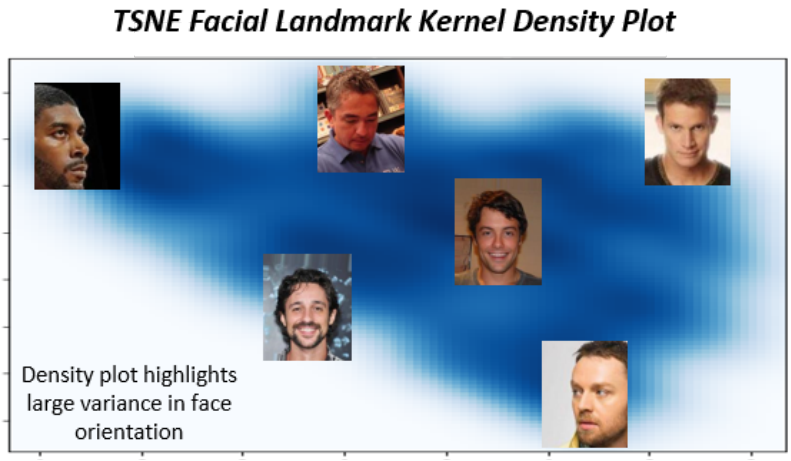

Figure 5

Figure 5 highlights an important broader characteristic and limitation of the CelebA dataset: the inconsistency of the images.

Dataset Limitations

Many images have faces turned to the side, contain obscuring objects blocking parts of the face, or include inconsistent lighting and shadows. Consistency in facial orientation, lighting, and

background allows for the deep feature representations to capture similarities and differences that will be most useful for the interpolation phase. Therefore, alignment in the locations of facial landmarks, the orientation of faces, and facial characteristics such as hair color, face shape, and skin tone will simplify the required model complexity and data size requirements. The following section outlines steps to address these limitations.

Preprocessing steps

After dropping the poor quality images described above, the following preprocessing steps are implemented with the aim of improving the consistency of the remaining images:

- Rotate and scale faces to align to a unified template using dlib's detected landmarks. This is performed by using the angle of the vector that connects the two eyes of a face, along with the distance between the two eyes; an affine transformation is applied to rotate the face similar to the movement of a clock, and scale the faces to meet a more unified template (Figure 6).
- Intelligently crop and downsample images around face by leveraging eye and chin landmarks to promote faster processing time (Figure 6)

Figure 6

- Some facial orientations cannot be aligned using a 2D affine transformation, such as heads looking/turning left, right, up, or down. However, the magnitude of the angles is captured by using facial landmarks and is leveraged as a feature. Images of centered orientation are used for this project, however future enhancements involve generalizing the process to all facial orientations (Figure 7).

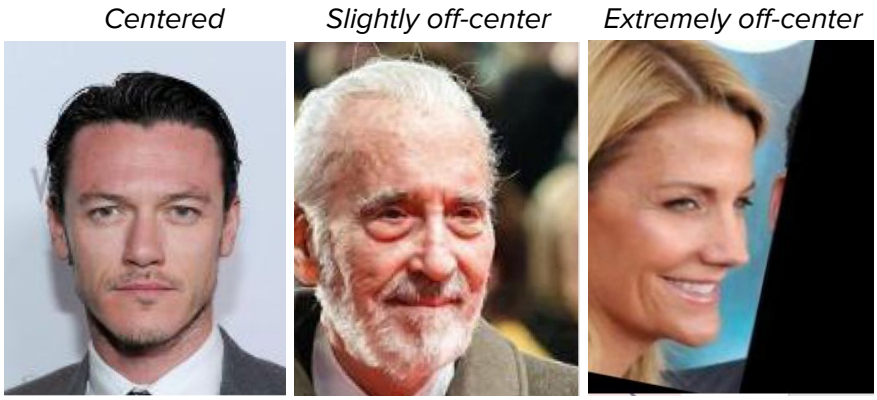

Figure 7

- Create a feature by computing the average pixel intensity of each subject's cheeks, capturing information related to lighting, shadows, objects obscuring the face (sunglasses), and skin tone. Images of average pixel intensity are used for this project, however future enhancements involve generalizing the process to other groupings.
- Group and filter images by hair color; brown and black hair are used for this project, however future enhancements involve generalizing the process to all hair colors.
- Image size ratio is slightly tweaked from 218 by 178 to 220 by 176..
- Normalize pixel values between 0 and 1.

Following the aforementioned steps to improve the consistency of the images, the final dataset to be used for the modeling includes 9,076 images. While this subset is significantly smaller than the original size, it should still be large enough to achieve the objective, especially given the improved consistency following the preprocessing steps.

Modeling

Deep Feature Attribute Vector

Deep Feature Interpolation leverages the idea that images can be projected into ***deep feature representations.*** This is achieved by processing images through a pre-trained Convolutional Neural Net (in this case VGG19) and concatenating the outputs at various convolutional layers into a single feature vector.

The idea is that the deep feature representations capture important information and details related to the content of the images. DFI leverages these abstract representations in constructing deep feature ***attribute vectors (w)*** that can be used to direct movement across latent space in order to transform specific image characteristics.

The algorithm proceeds as follows:

- Using the dataset, create two groups of images, one of which has a unique attribute and one of which that does not (ex. bearded vs. non-bearded faces).
- All images within each group are projected into deep feature space using VGG19. The resulting vectors form a geometric representation of the images in deep vectorspace while capturing all of their important information (Figure 8).

  Note that the architecture of VGG19 consists of 16 convolutional layers with increasing channel sizes of 64, 128, 256, and 512. Five maxpooling downsampling layers are interspersed throughout.

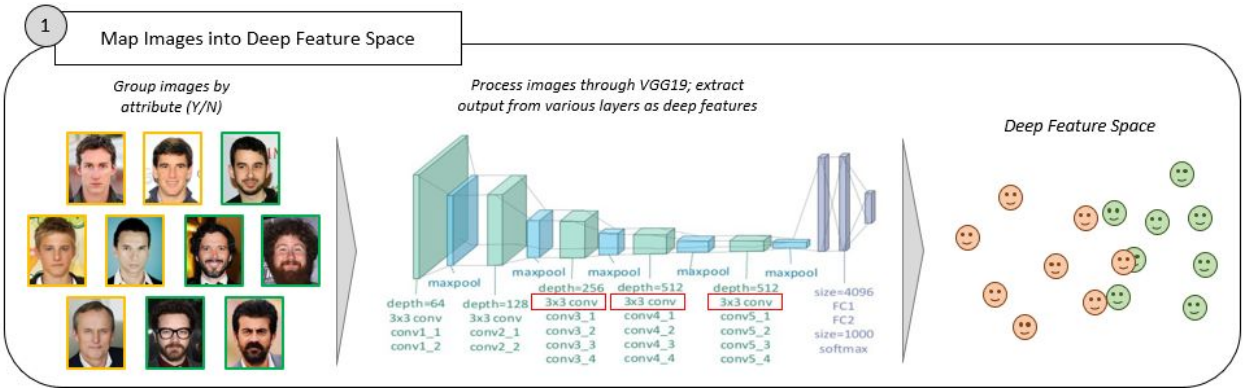

Figure 8

- A centroid point in space is then calculated for each of the two groups by computing the mean value per feature dimension. Each group is now represented by a centroid vector (Figure 9):

$$\phi(X_{YES\_attribute}) \text{ and } \phi(X_{NO\_attribute})$$

- The delta between these two centroids represents the deep feature interpolation vector, $\boldsymbol{\omega}$ (Figure 9):

$$\omega = \phi(X_{YES\_attribute}) - \phi(X_{NO\_attribute})$$

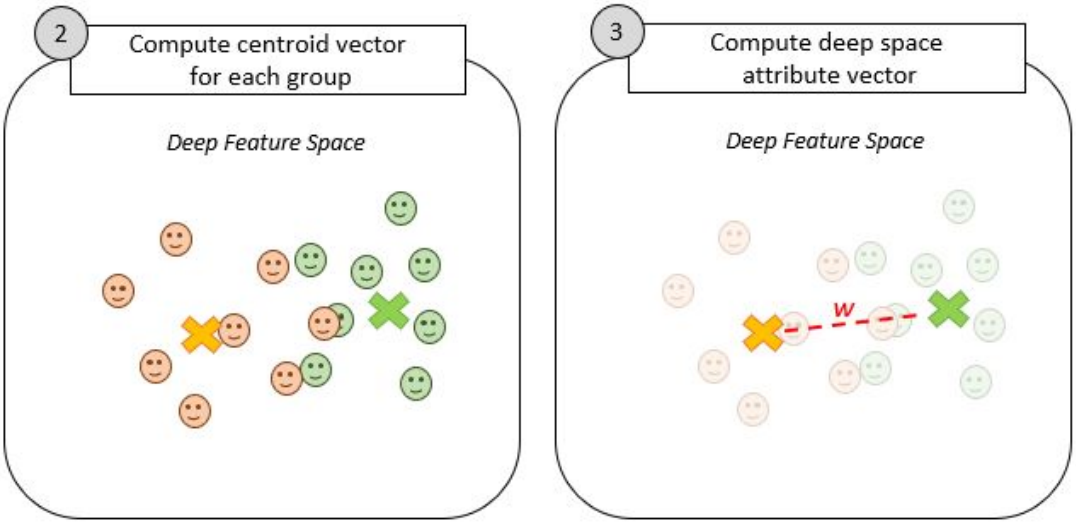

Figure 9

Ideally, the only information that this interpolation vector contains is that of the attribute of interest. Interestingly, this assumption is fundamental to the DFI approach, and it is noted that nowhere is this linear relationship necessarily enforced by the neural network.

Note that it is possible for systemic bias to exist between the attribute groups. The attribute vector could capture unintended information if correlations exist to other facial characteristics. One example is age, where older people may be more likely to have a beard - thus, the interpolation vector used for applying/removing a beard will contain an “aging” element as well. This challenge is addressed by aiming to have groups that are as consistent, diverse, and robust as possible, where the variation across non-attribute specific facial characteristics and backgrounds is cancelled out.

Residual Deep Feature Interpolation (RDFI)

Once the attribute vector is known, Residual Deep Feature Interpolation (RDFI) is ready to be performed. The algorithm proceeds as follows (Figure 10):

- Use VGG19 to project the image that is intended to be transformed into deep feature space.

$$\phi(x)$$

- Add the previously calculated attribute vector to the deep feature representation of the image. The resulting point in space is the target destination in feature space.

$$\phi(x) + \omega * \alpha$$

- Using backpropagation, train a **mask (r)** that is to be applied to the **input image (x)**, so that the deep feature representation of the **masked input (x + r)** learns its way through latent space in the direction of ω (the target destination). Note that the purpose of training a mask, as opposed to manipulating the original input image altogether, is to allow for post-processing work to be applied directly to the mask, lending to an improved final product.
- The loss function captures the distance in space between the deep feature representation of the masked input (x + r) and that of the target destination that was previously computed using ω.

$$|\phi(x+r) - (\phi(x) + \omega * \alpha)|$$

- The parameter ***alpha (a)*** controls the magnitude of the attribute vector. Larger alphas result in more pronounced transformations. However, this can cause the image to become more vulnerable to other forms of distortion. A more in-depth discussion of alpha is to come.

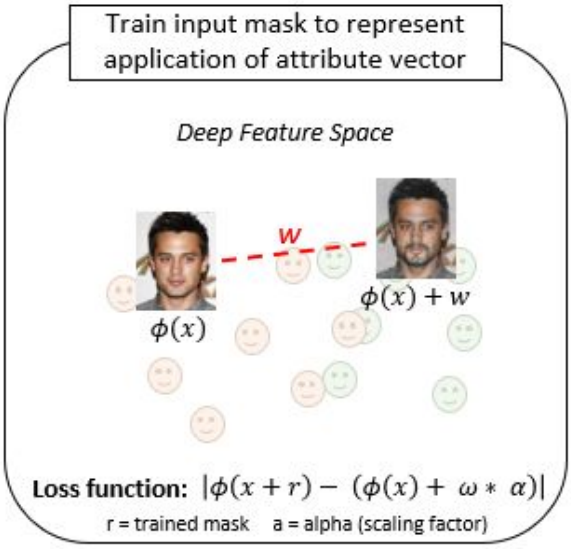

Figure 10

Fast Residual Deep Feature Interpolation (FRDFI)

The second algorithm, Fast RDFI (FRDFI), builds on the ideas from RDFI. For RDFI, individual masks are trained and optimized for specific images. As an enhancement to RDFI, FRDFI trains a single neural net that can generalize the transformation to any input image. The idea is that only a single forward-pass of an image will be required during inference, resulting in a transformation that takes just a few seconds to process.

To achieve this objective, a network is constructed that can be visualized as the concatenation of two networks. The top network in Figure 11 contains weights that are to be learned and optimized in order to generate the attribute mask. The bottom network is the pre-trained VGG19 net, which is used to calculate the loss function in similar fashion to RDFI.

The architecture of the top network is as follows:

- Reflection padding layer of dimensions 4x4
- Three convolutional downsampling layers with the following characteristics:
    - Channel sizes of 32, 64, and 128
    - Stencil dimensions of 9x9, 3x3, and 3x3
    - Stride lengths of 1, 2 and 2.
    - Each layer is followed by a Batch Normalization layer and a relu activation
- Two upsampling convolutional transpose layers with the following characteristics
    - Channel sizes of 64 and 32
    - Both stencil dimensions are 3x3
    - Both stride lengths are 2
    - Each layer is followed by a Batch Normalization layer and a relu activation
- Reflection padding layer of dimensions 4x4
- Convolutional layer of 3 channels, 9x9 stencil size, stride of 1, and a tanh activation

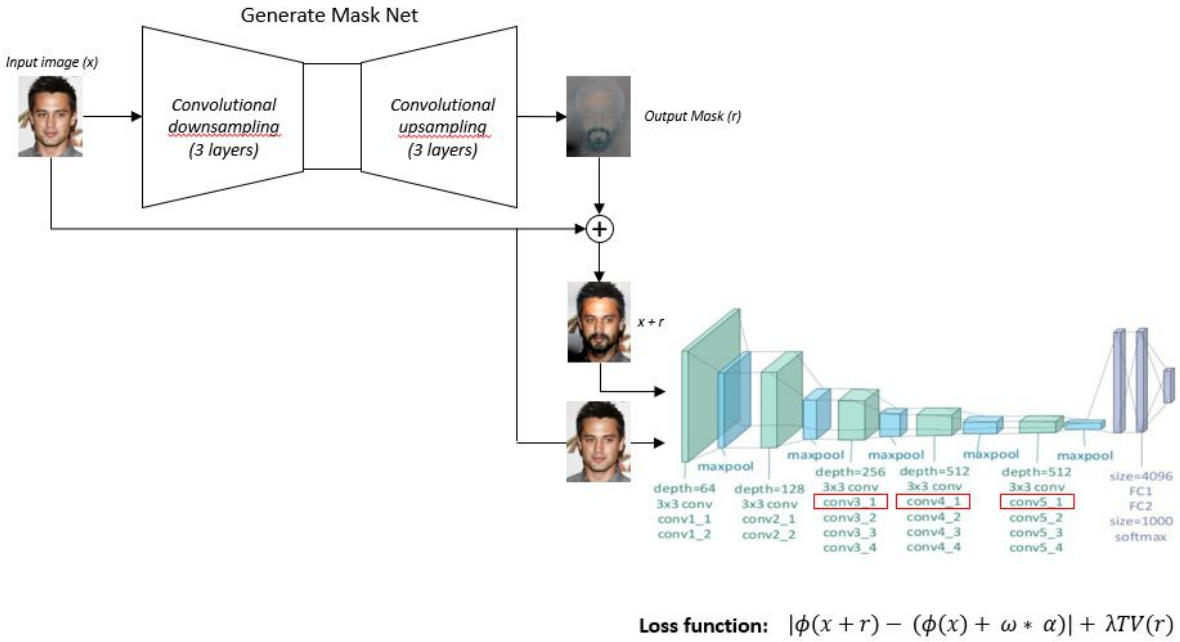

Figure 11

The learning process operates as follows:

- An input image is processed by the top network to produce an output mask.

$$r$$

- The output mask (r) is then added to the original input image (x), resulting in a masked input.

$$x + r$$

- Both the masked input and the original input are projected into deep feature space using the VGG19 network.

$$\phi(x + r) \text{ and } \phi(x)$$

- The attribute vector (w) is added to the deep feature representation of the original input, resulting in the target destination.

$$\phi(x) + \omega * \alpha$$

- These two deep feature representations are then used to calculate the loss, similar to how it was computed in RDFI. Note that an additional expression is included in the loss function, Total Variation Loss, which will be discussed in a later section.

$$|\phi(x + r) - \phi(x) + \omega * \alpha| + \lambda TV(r)$$

- Backpropagation is used to calculate the derivatives of the weights in the top network, which are used to update and optimize the network.

Post-Processing

The post-processing stage is key to the success of the transformation. The interpolation may often trigger non-beard related impacts, such as altering the appearance of the eyes, skin, and image color complexion.

A straightforward method to address these impacts is to intelligently crop the mask around the area of interest prior to applying the mask to the original image. This is completed using the following steps (Figure 12):

1. Leverage ***dlib's 68_face_landmarks_predictor*** to outline the jaw and mark the nose on the original input image. These landmarks are used to carve out the region of the face where a beard would be present.
2. Binarize the carved out facial region. Apply a function to phase out the top region of pixels to promote a smooth transition from the masked image to the original image.
3. Apply element-wise multiplication of the learned mask by the binarized cropped region. This operation carves out the bearded area from the learned mask.
4. Add the carved out beard mask to the original input image to produce the final output image.

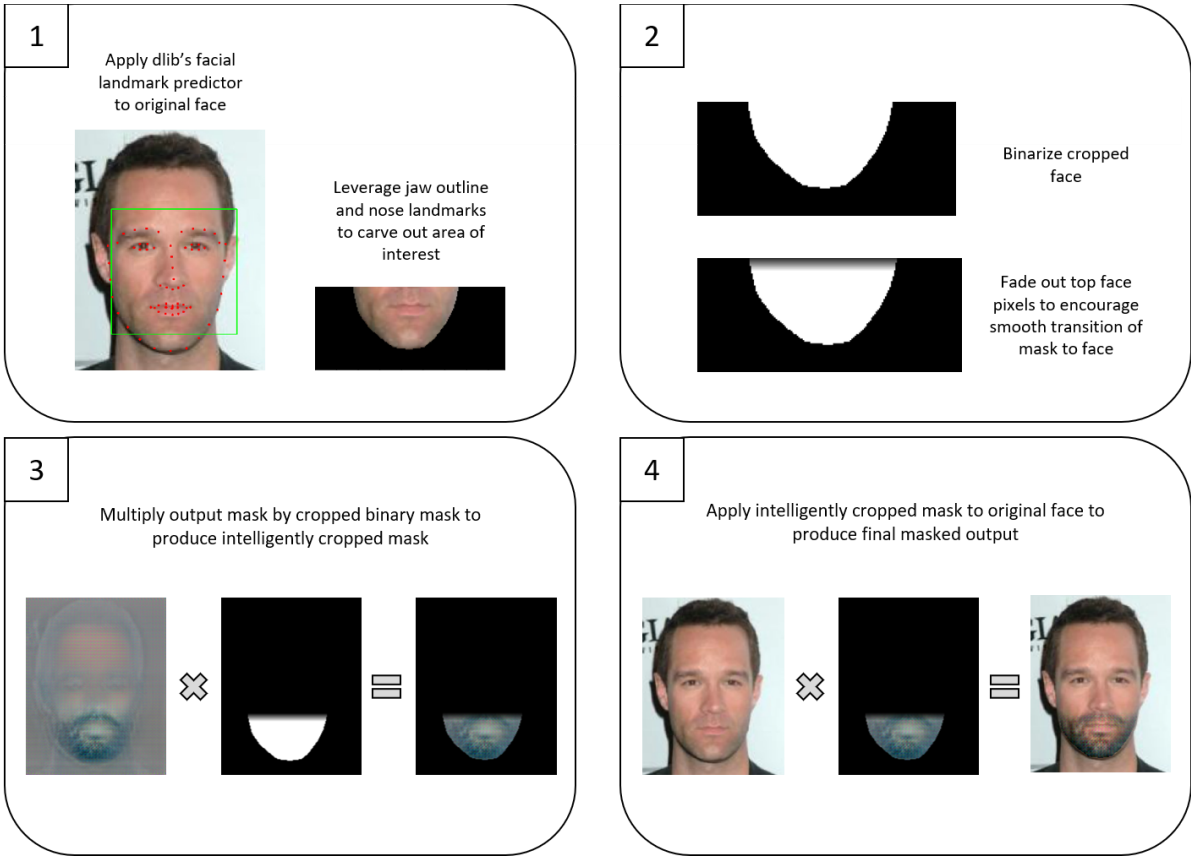
1
Apply dlib's facial landmark predictor to original face
Leverage jaw outline and nose landmarks to carve out area of interest
2
Binarize cropped face
Fade out top face pixels to encourage smooth transition of mask to face
3
Multiply output mask by cropped binary mask to produce intelligently cropped mask
4
Apply intelligently cropped mask to original face to produce final masked output

Figure 12

Results and Discussion

RDFI - Clean-to-Beard

The implementation of RDFI produced good results overall (Figure 13). There is variation in the perceptual quality of the beard, but most results look reasonably natural. Additionally, it is important to remember that the input image is of 218x178 resolution, which can contribute to a bearded region that appears more pixelated.

1. Includes functions to tune image saturation, brightness, and contrast

Figure 13

The color of the beard tends to follow that of the hair color; darker hair results in darker beards, while lighter hair results in brown or more blonde beards. This exhibits how deep feature representations capture hair color, and that the color of the beard that results from applying the attribute vector depends on the image's starting location in deep feature space.

The output masks are also noteworthy. If the non-beard characteristics between the two groups perfectly cancelled each other out, the only visible object in the mask should be the beard. This is

not the case, however, as a silhouette of the face is present. This is an example of how the cumulative characteristics between the two groups do not cancel each other out.

In particular, artifacts near the eyes are notable. When comparing the eyes of the original images to the masked input images, the eyes appear to slightly open up, and sometimes develop bags under them. This could be indicative of an ageing element to the interpolation process.

Another undesired impact that is exhibited is the general graying of the image. This is likely driven by how the movement through deep feature space captures a degree of randomness, in addition to the intended attribute transformation.

For both the impacts to the eyes and color drift, the solution is to apply the post-processing task described in the previous section. Note however, that for this RDFI a simpler post-processing function is applied that tunes for image brightness, saturation, and contrast.

While a more in depth discussion of the impact of alpha is to follow, it is relevant now to think about how alpha impacts these masks. By increasing alpha beyond 1, amplified assumptions are made about the linearity of the relationship between beard strength and deep feature space. In reality, alphas of greater magnitude push the interpolation vector deeper into a space of unknown impacts and relationships between features. Since this analysis uses an alpha of 4, the susceptibility to these unknown impacts and distortions is present.

FRDFI - Clean-to-Beard

The training of the Fast RDFI network faced initial challenges. The learning process struggled to maneuver around a local minima, where the results produced darker gridlike outputs, and the beard was not yet present (Figure 14).

Note that the patterned gridlike masks are caused by the interaction between the random weight initialization and the convolutional layers. While a more in-depth discussion of the solution is to follow, this challenge is addressed by penalizing the differences in intensity of neighboring pixels learned during the training process.

Figure 14

The solution to the inability for the initial training to surpass a local minima is to first train the network on a single image. This process allows the network to more efficiently navigate closer to the global minima. The resulting network and weights can then be transferred and retrained on the entire dataset in order to generalize to a broader set of inputs.

Figure 15 illustrates how the mask initially blots out the entire face, which is due to the random weight initialization. A major component of the initial optimization is to learn which weights are involved in generating the desired attribute transformation, and to suppress all weights that impact other areas such as backgrounds and non-beard facial attributes.

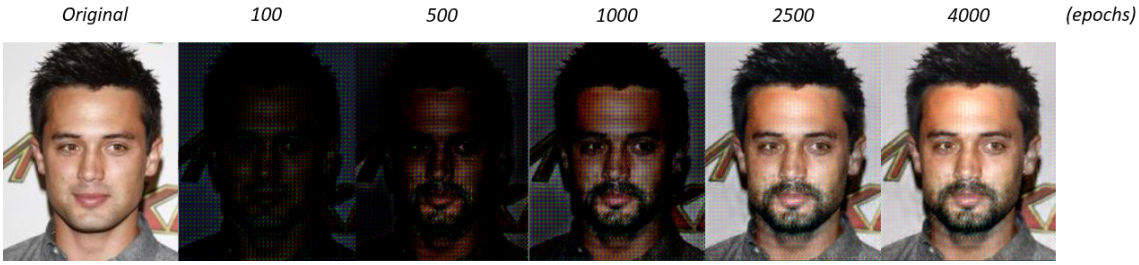

Figure 15

The next step is to then refit the network on the entire training data. The weights are initialized with the weights that were learned from the network fitted previously to the individual image.

Figure 16 illustrates the progression of results by training epoch, where epoch 0 represents the weights initialized from the network previously fit to the individual image. It is interesting to compare the epoch 0 predictions of the new images, as varying impacts to the eyes and lower face take place. From epoch 0 to epoch 3000, a transformation takes place characterized by first removing the artifacts that are initially present at epoch 0, filling in the bearded region with darker pixels, and then finally tweaking the bearded region to make it appear more realistic by epoch 3000.

Finally, the previously outlined post-processing steps intelligently crop the mask and apply it to the original image, thus preventing any non-beard related impacts.

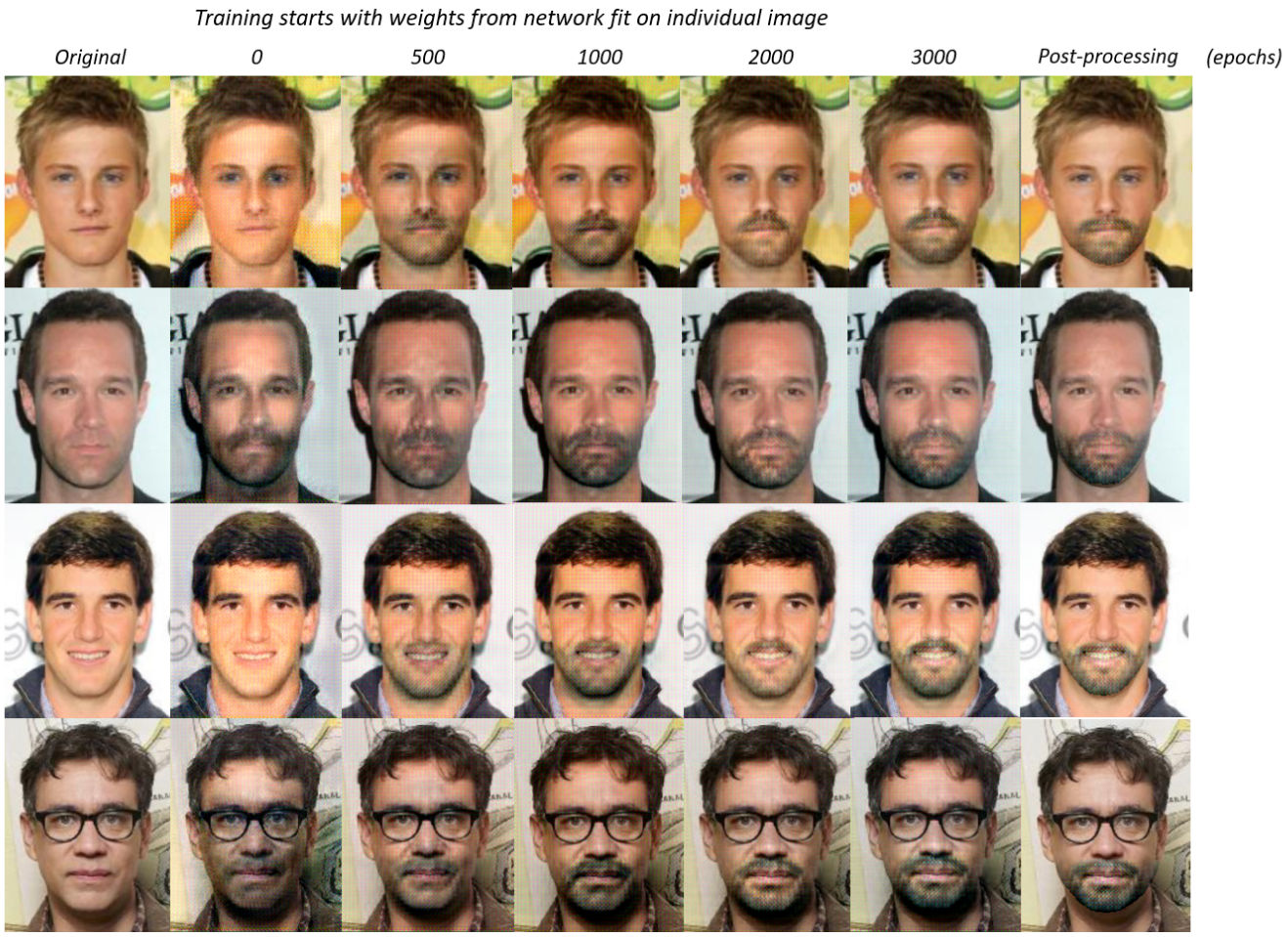

Figure 16

Figure 17 examines more closely the impacts of the post-processsing function. A few of the most noticeable improvements are:

- Overall image is clearer; clarity is in line with the original image
- Beard no longer drifts below chin onto neck
- Eyes and skin are no longer distorted and are identical to the original image

Figure 17

A final look at a larger output highlights some opportunity for improvement in the clarity of the beard (Figure 18).

Original *Final Output*

Figure 18

Beard-to-Clean

The beard-to-clean transformation proved to be more challenging than applying a beard. Note that due to time constraints only the FRDFI and not the RDFI algorithm is applied for the beard-to-clean transformation. Further analysis could explore the differences in results between the two algorithms.

The network is trained similarly to the clean-to-beard network, where the weights are first optimized on an individual image before retraining on the full dataset (Figure 19 and 20).

Figure 19

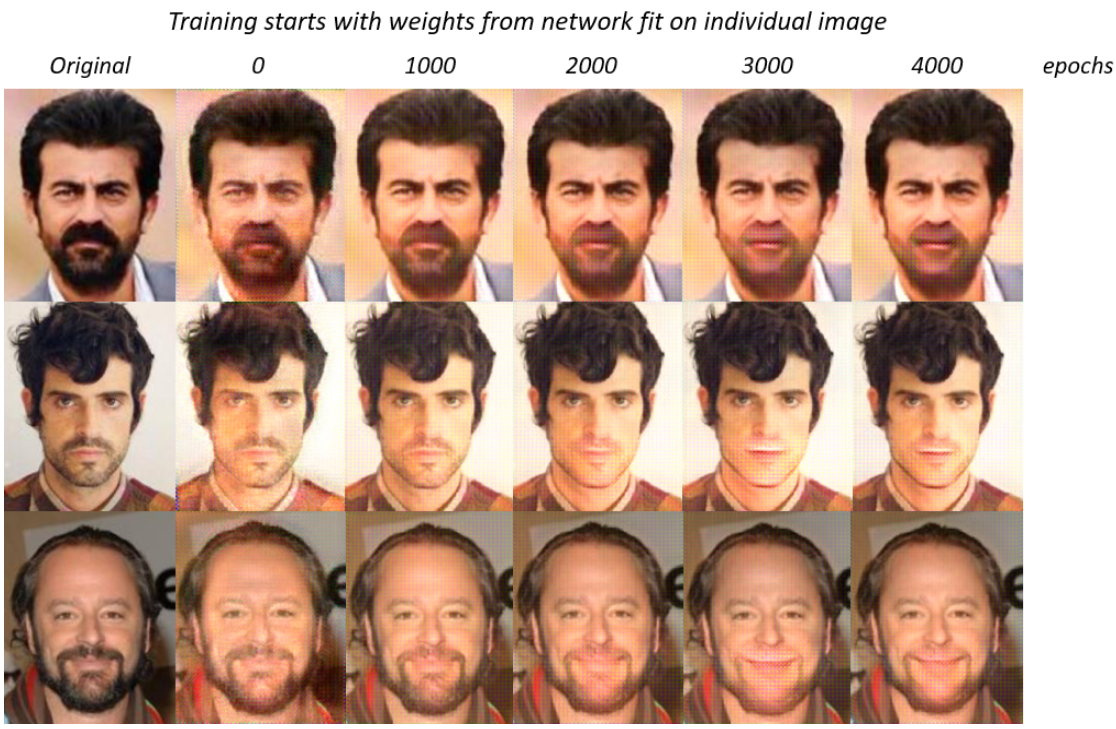

Figure 20

In general, the beard removal is not clean. Oftentimes the entirety of the beard is not removed. Unexpected artifacts also result, such as the white strip on top of the second subject's lip in Figure 20. Additionally, the resulting color scheme and skin complexion is notably brighter overall.

A primary issue is that the network seems to struggle with different shapes and sizes of beards. This makes sense, as a single attribute vector is being leveraged to operate on all beard styles. The attribute vector was calculated using a deep feature representation of the ***average*** beard, therefore applying it to remove larger or uniquely styled beards involves implausible assumptions regarding the linearity of deep feature space. A solution is to employ a grouping method, where different attribute vectors are calculated and leveraged based on more defined beard characteristics of the subjects.

Additionally, it is a more complex task to remove an object and unearth an unknown image beneath what is being removed, compared to stacking a bearded mask on top of a clean image. This is a task that would benefit from exploring additional cutting-edge techniques, such as GANs.

Figure 21 exhibits the results after post-processing. This figure highlights the challenges of recreating the subject's skin complexion following beard removal and displays the lack of clarity in the transformed region. Finally, the dark strip that remains underneath the chin indicates how post-processing for beard removal involves the added challenge of generalizing the operation to beards of various sizes and styles.

Original *Final*

Figure 21

Parameters and Hyperparameters

Alpha

The hyperparameter ***alpha*** was the most impactful hyperparameter to tune during the modeling process. Alpha is the scaling factor of the deep feature attribute vector. Alphas of increasing value will cause the network to learn transformations of greater magnitude. This can be useful in generating a more pronounced beard. However, greater alphas also introduce vulnerability where the learning process can trigger unexpected changes.

As previously mentioned, as the interpolation vector stretches deeper into the vectorspace beyond that of the original length of the attribute vector, the relationships between facial characteristics and the interpolation vector become increasingly non-linear and ambiguous.

The examples in Figure 22 exhibit how using an alpha equal to 1 results in minimal beard growth, while an alpha of 8 results in heavy beard growth accompanied by image distortions around the neck and cheeks as well as significant discoloration. The darker/grayer results for when alpha is equal to 8 could be attributed to the idea that the attribute vector is stretching to a point in vectorspace where few images exist and therefore the point in space has less and less meaning, resulting in a mask characterized by randomness, averaging out to grayer pixels.

An alpha of 4 was arrived upon as a happy medium. Note that the discoloration that still does take place when alpha is equal to 4 is addressed by post-processing operations previously discussed.

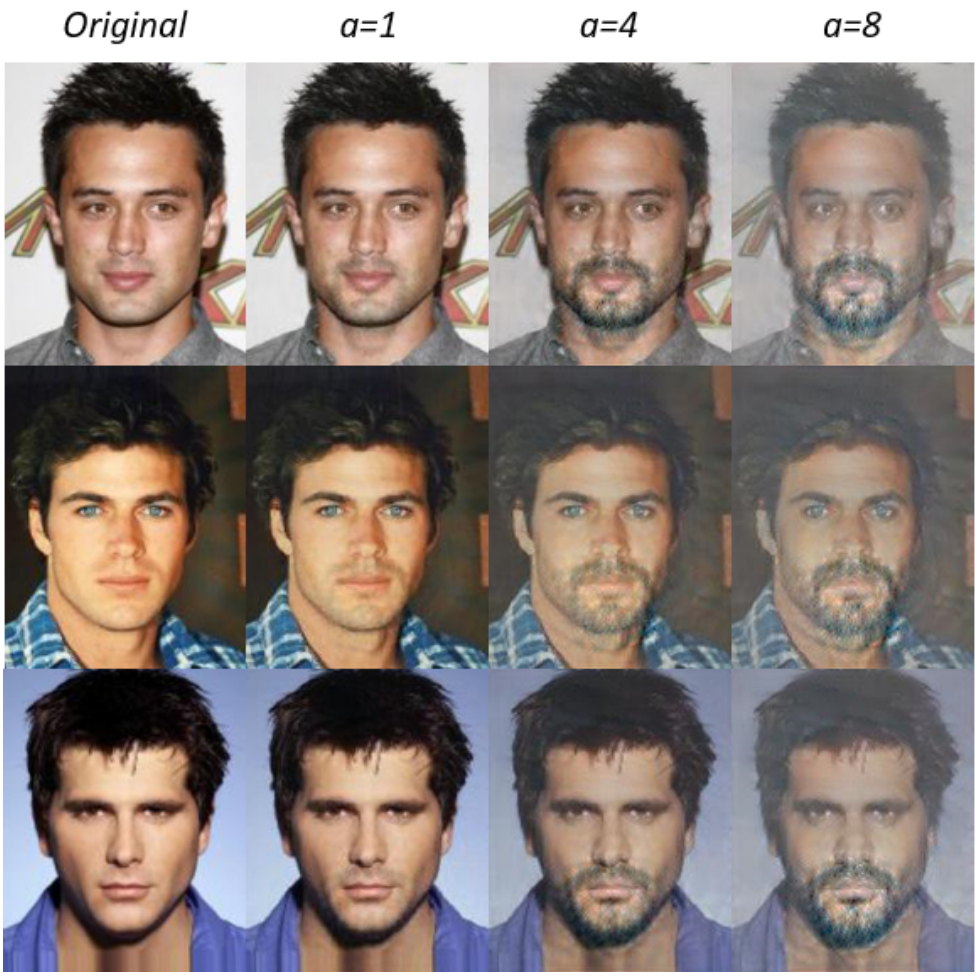

Figure 22

Total Variation Loss

Another important consideration during the modeling process is the inclusion of a Total Variation Loss expression. During the training of RDFI, the loss function only captures the distance between the masked image and the target image using their deep feature representations. Initially, the loss function from FRDFI was identical to that of RDFI. However, during the training of FRDFI, early results exhibited intense grid-like patterns of bright colors moving across the image. Closer examination of the pixels revealed how neighboring pixels are of significantly different intensities (Figure 23).

The cause of this mannerism is likely driven by the interaction between the convolutional layers and the randomly initialized weights which compels the learning process to gravitate towards local minimas that are characterized by repetitive pixel sequences.

The solution to this problem is to apply a Total Variation Loss expression to the loss function. This factor penalizes the difference in the intensity of neighboring pixels.

$$\sum \left(p_{i,j} - p_{i+1,j+1}\right)^2$$

The idea is that this constraint encourages smooth transitions across neighboring pixels. Interestingly, upon adding this expression into the loss function, the learning process quickly pushed out the bright patterns and the resulting image looked significantly improved (although not entirely solved).

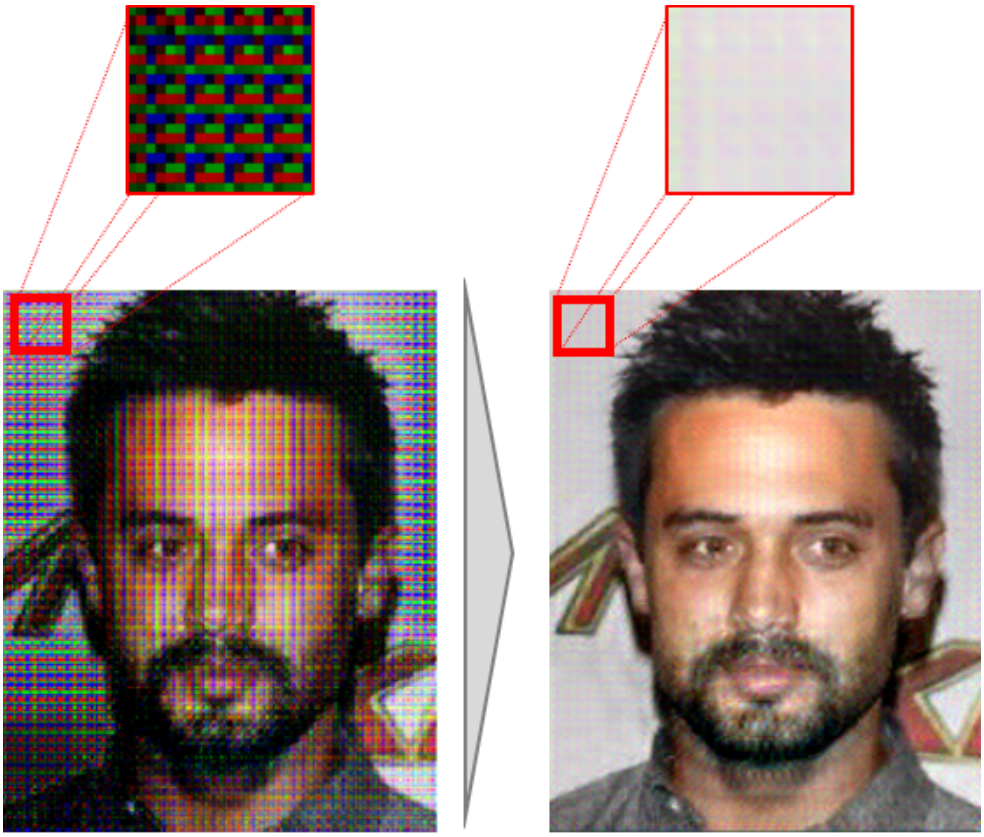

Figure 23

Next Steps

There are several next steps to be incorporated into this analysis in order to both improve the quality of the results as well as increase the breadth of application.

- In addition to training Fast RDFI networks to transform images by applying or removing beards, train a variety of additional networks to produce **different types of transformations** (clean to mustache, beard to goatee, etc.) including the ability to perform multiple transformations during the same forward-pass. Inputs would include a one-hot variable to communicate desired transformations.

- To improve the quality of the transformations, more individualized attribute vectors can be calculated and leveraged by using a **KNN image grouping technique** for the two attribute groups. The idea would be to compute the attribute vector by forming groups of similar characteristics to that of the image being transformed. By utilizing an attribute vector that is specific to the characteristics of the image being operated on, the quality of the transformation should benefit. This technique would be especially useful for beard removal, as discussed earlier.

- Experiment with **different alpha values** for different deep feature output layers. The output layers of VGG19 capture different characteristics of the image, specifically different spatial hierarchies of information. Layers likely have unequal importance to the capturing of information necessary for a facial transformation. Exploring different combinations of weights for these layers could produce more interesting and perceptually accurate results.

- Leveraging the power of **Cyclic GANs, Progressive GANs, or StarGANs [9]** could also lead to more accurate final outputs. A GAN is a training mechanism that learns to distinguish between a real image and a generated image, which can be baked into the learning optimizer to facilitate the generation of more realistic looking output images.

- Apply analysis to **images of stronger resolution** to test whether images can retain clarity following a transformation.

References

1. ***Deep Feature Consistent Variational Autoencoder***
   https://arxiv.org/abs/1610.00291

2. ***Fast Face-swap Using Convolutional Neural Networks***
   https://www.researchgate.net/publication/311106512_Fast_Face-swap_Using_Convolutional_Neural_Networks

3. ***Large-scale CelebFaces Attributes (CelebA) Dataset***
   http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

4. ***Facial landmarks with dlib, OpenCV, and Python;***
   https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

5. ***Face Alignment with OpenCV and Python;***
   https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/

6. ***Facial Landmark Detection: a Literature Survey***
   https://arxiv.org/pdf/1805.05563.pdf

7. ***Deep Feature Interpolation for Image Content Changes***
   https://arxiv.org/pdf/1611.05507.pdf

8. ***Perceptual Losses for Real-Time Style Transfer and Super-Resolution***
   https://arxiv.org/pdf/1603.08155.pdf

9. ***CycleGAN for Image-to-Image Translation***
   http://shikib.com/CycleGan.html