Nonlinear Dimension Reduction Final Group Project Report

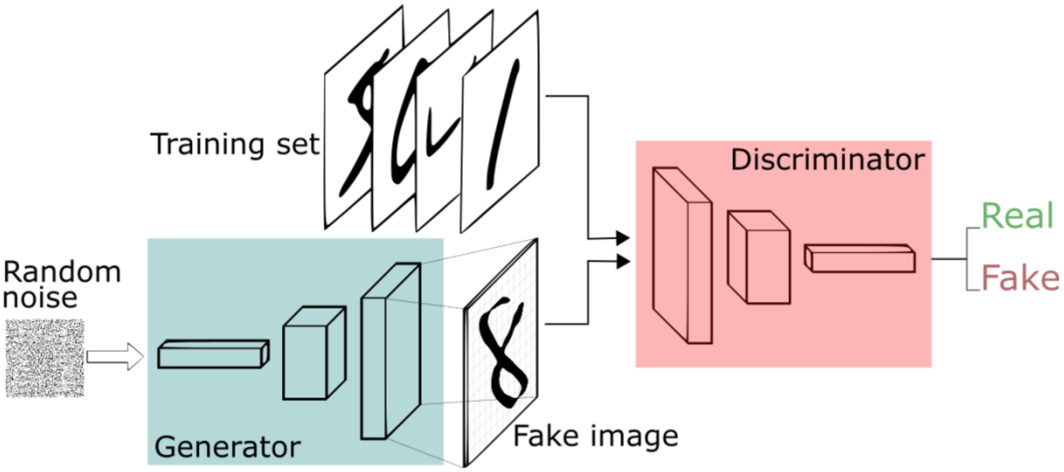
Yang Mo, Yixian Huang, Wenhao Qiu

Problem Description

The picture is valuable that follows a joint distribution function of different features and the nonlinear dimension reduction model can give us the latent space of the original picture. If there is a certain nonlinear dimension reduction model in which we could combine a continuous latent space with a continuous image space, we can change the latent space and then get different pictures in output. We hope to change pictures by changing features to make the images conform to the changed features.

Background

Unfortunately, we did not find a certain nonlinear dimension reduction model in which we could combine a continuous latent space with continuous image space. However, GAN can satisfy this requirement. A generative adversarial network(GAN) is a class of machine learning systems invented in 2014. Two neural work contest each other in a game, typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution. The generative network's training objective is to increase the error rate of the discriminative network.



GAN networks can give us a continuous latent space and continuous image space(at least to some degree), meanwhile, it can generate awesome image output.

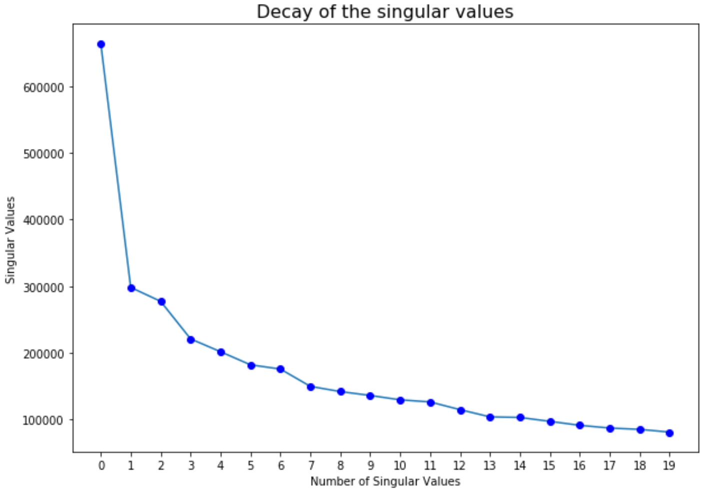
Dataset Description

Our dataset is Large-scale CelebFaces Attributes (CelebAHQ) Dataset.CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebAHQ has large diversities, large quantities, and rich annotations, including

- **10,177** number of **identities**,
- **202,599** number of **face images**, and
- **40 binary attributes** annotations per image like gender, age and so on.
- **Size 1024*1024**

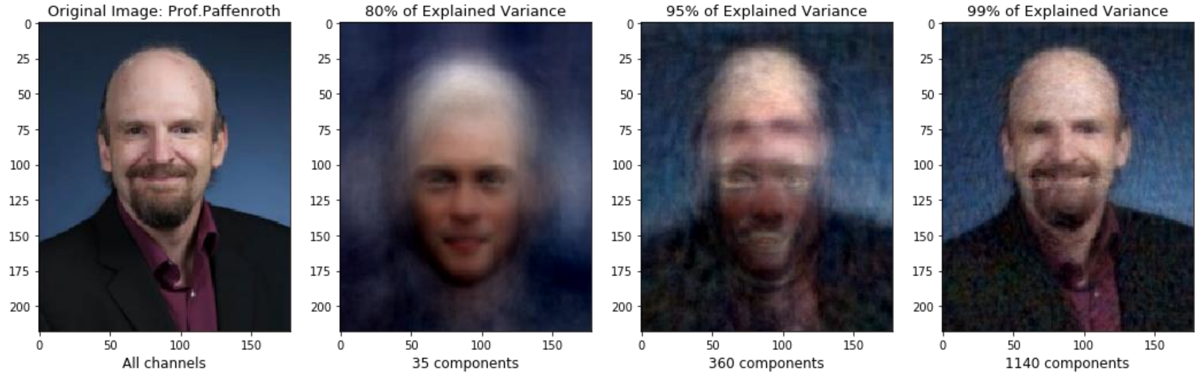
PCA Result

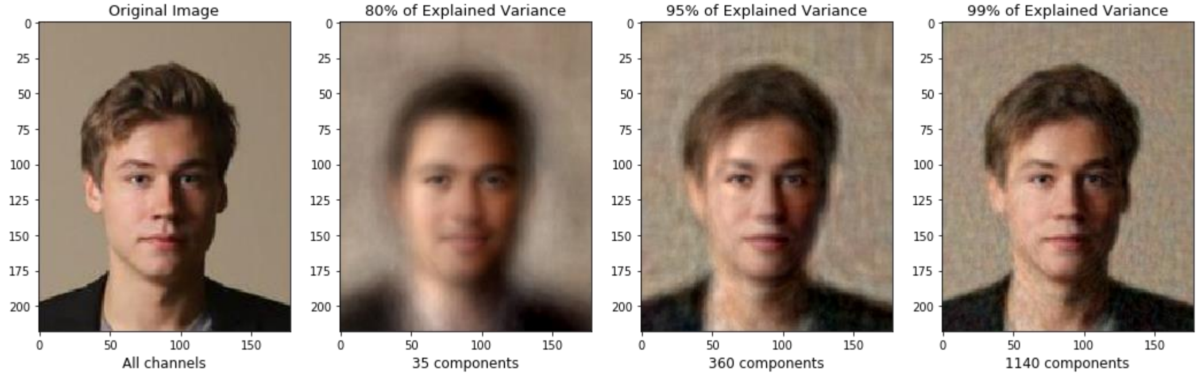
We apply PCA on CelebAHQ dataset. In order to save time, we only apply PCA to 2000 images with a smaller size (219*178*3) instead of all full images in the dataset. Here is the line chart of decay of the singular values (The chart only use the largest 20 of them):



After applying singular value decomposition, we reconstructed our portrait data using 35, 360 and 1140 principal components, which represent accordingly 80%, 95% and 99% of explained variance. Noticing that though the knee of the above chart is clearly lying on 3-10, in order to get an acceptable result, the number of components we have chosen will be much more than that due to the nonlinearity of the images.

Here are the two sets of results:

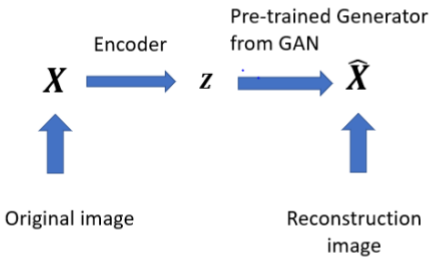

Original Image: Prof.Paffenroth
80% of Explained Variance
95% of Explained Variance
99% of Explained Variance
0 25 50 75 100 125 150 175 200
0 50 100 150
All channels
35 components
360 components
1140 components



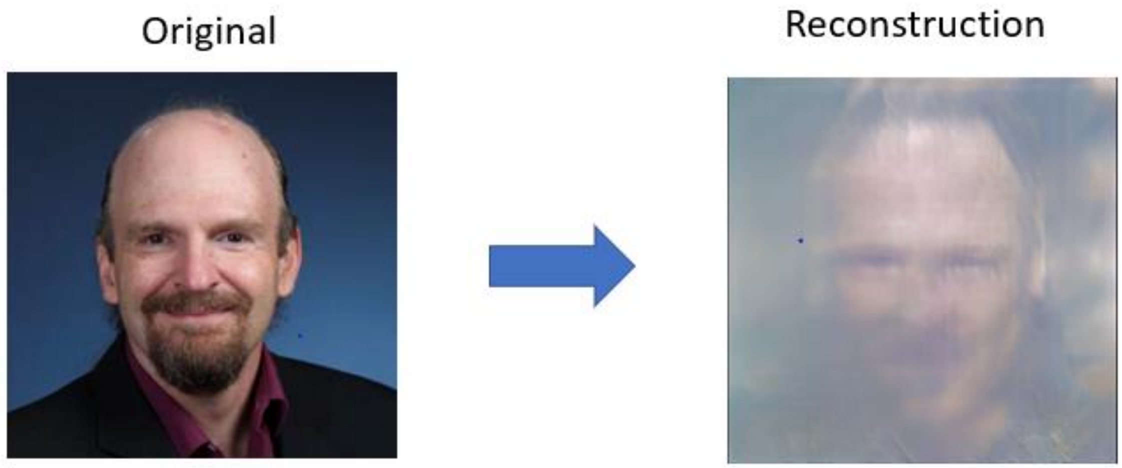
We speculate that since Prof. Paffenroth's image has the background of a different setting with other images (E.g. The original image is centered by the people's eyes and with light background.) in the CelebA-HQ, the reconstruction of his image needs a few more additional components.

Review of Proposal Approach

As we mentioned before, to solve the problem, we need to find a continuous latent space, while autoencoders' latent spaces are not always continuous and it is very hard to train a VAE which can generate awesome face images. So we plan to use a pre-trained generator trained by PG-GANs (A model which is able to generate high-quality images of a human face) as the decoder of our own autoencoder, specifically, to train our encoder. However, GAN networks do not offer an easy way to compute the encoder process based on the generator, actually, there are some papers doing research about this problem. For a certain picture, we cannot even find a good 'noise' in PG-GANs' latent space.



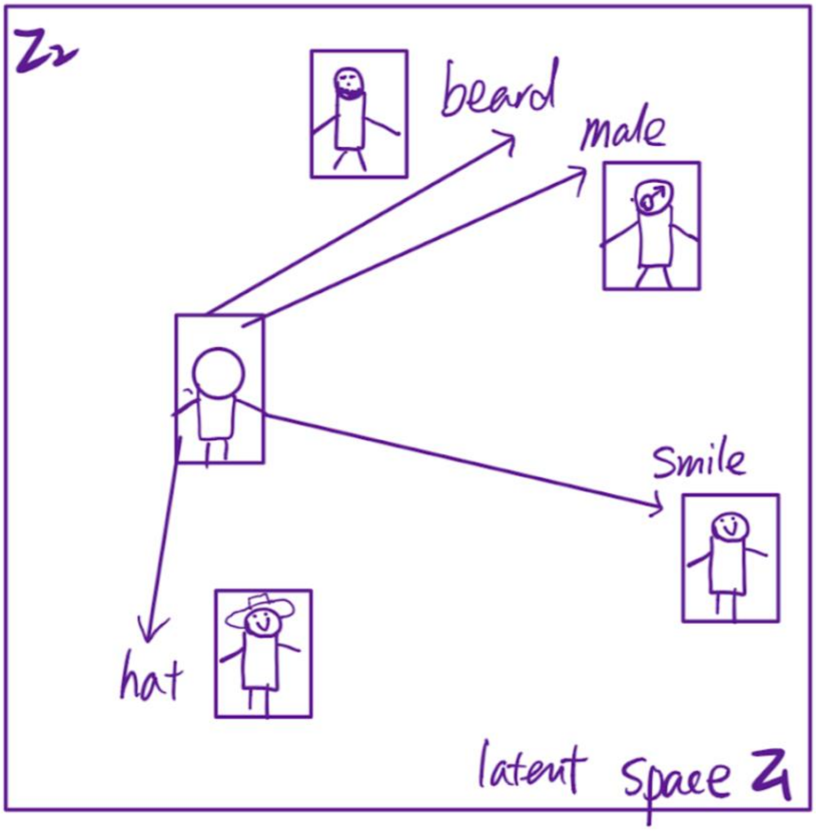
This is our best try:



Since the original idea of training an autoencoder from a GAN generator is not as easy as we thought, we switched to another potential approach and achieved pretty good results.

Methodology

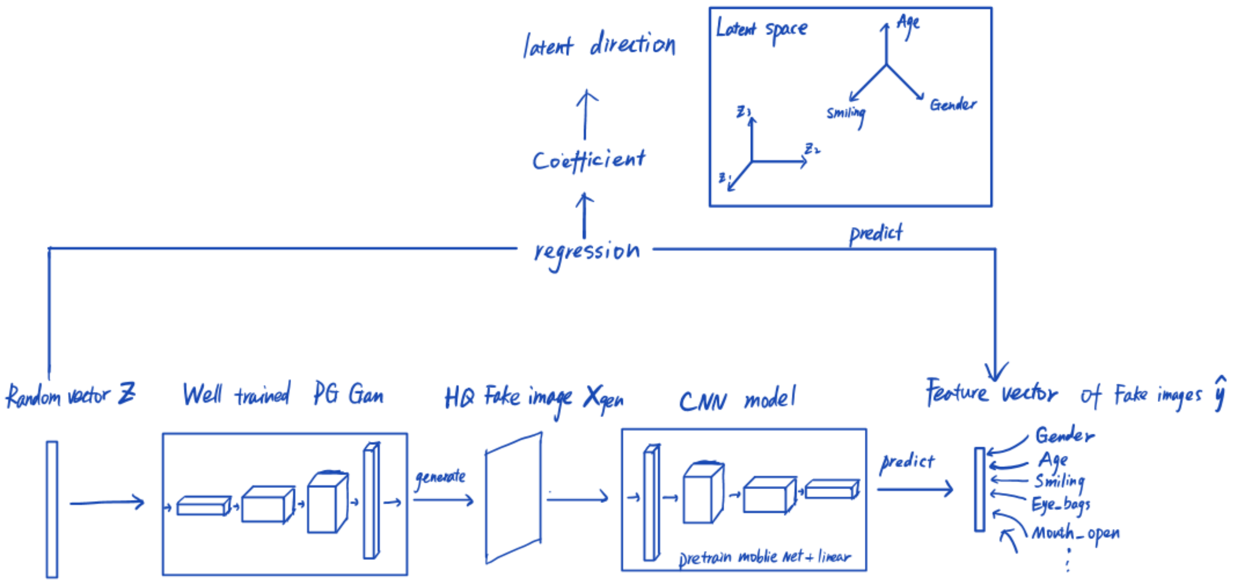
Walk Alone the Feature Axes



We already know that the latent space of PGGAN is quite continuous. So the objective is to find out the feature axes which is the fastest-changing direction of the corresponding feature within the latent space.

Bridge the Gap

Since we want to explore the relation between latent space and image's feature vector, we plan to do a linear regression between latent vectors and feature vectors. However, we fail to train an inverted net of the generator, which means we can not directly utilize the labeled data by finding the corresponding noise of each image. Instead, we bridge the gap between latent vectors and feature vectors through a supervised learning model trained on the CelebHQ dataset. Specifically, we pass a random noise Z through the generator to produce synthetic images X, then pass X through the pre-trained neural network to produce a label vector Y. Now we can collect a set (Z, Y) pairs to do the regression. The figure below describes the workflow of this approach.



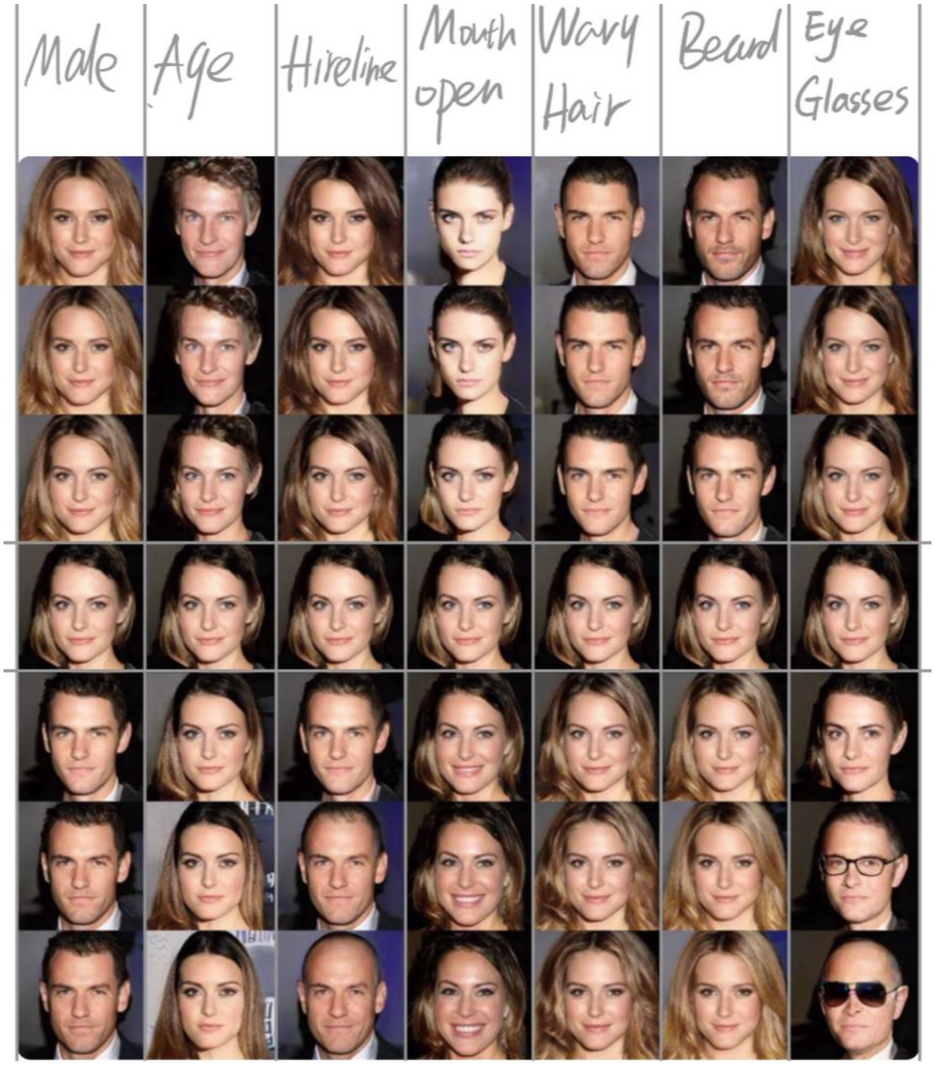
More specifically, this approach contains five steps:

1. Prediction: Build a feature extractor(a convolutional neural network)
2. Generation: Generate 30000 random latent vectors, pass through the well-trained PGGAN generator to produce synthetic images, then use a trained feature extractor to produce features for every image.
3. Correlation: Perform linear regression between latent vectors and features. The regression slope becomes the feature axes.
4. Orthogonalization: Orthogonalize each feature axes based on Gender and Age. (Detail see the result section)
5. Walk alone axes: Start from one random noise, move it along a feature axis, and check how this changes the generated images.

Results

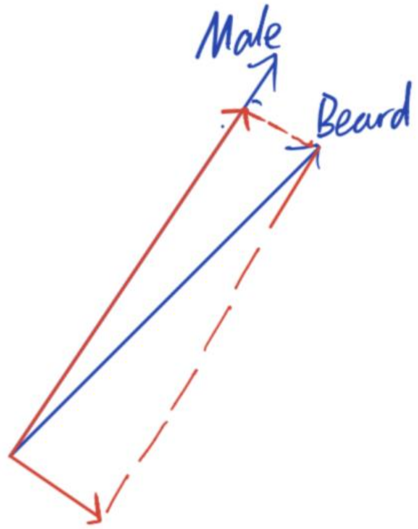
We first generated a fake image from random noise. After applied the set directions generate from regression, we plot these sets of results. We pick seven significant directions -- ‘Male’, ‘Age’, ‘Hairline’, ‘Mouth_open’, ‘Wavy_Hair’, ‘Beard’, and ‘Eyeglasses’. For each direction, we generated 6 different images with different steps.

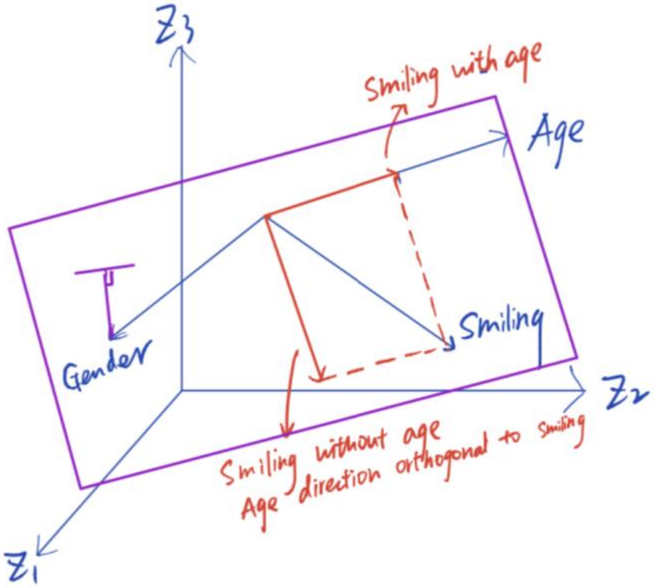
The experiment is quite successful --the vector “noise” plus vector “direction” generates the image as we expect. Such as, in the “hairline” direction, an attractive young woman graduate turn into a bald middle-aged man. Also, in the “mouth-open” direction, a serious teenage becomes a smiling lady.



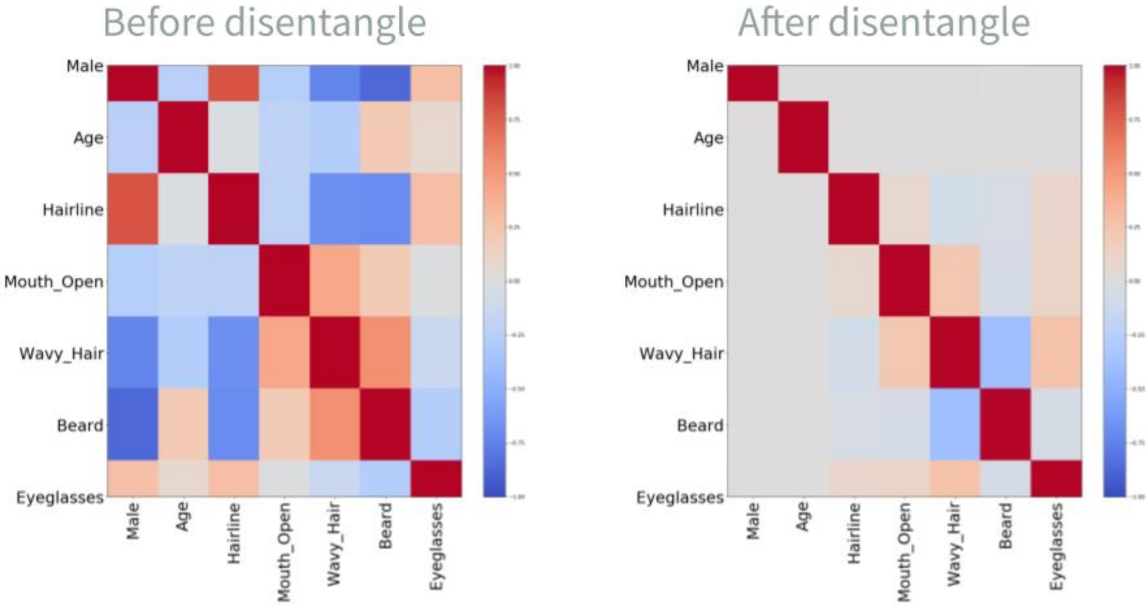
Notice we did not get a robust result without orthogonalization -- As a result, you can always see the gender of the people change while you walk another direction. This situation is due to the vectors of direction are not mutually orthogonal. In order to solve this problem, we picked a series of directions in the base direction. All of the other directions are orthogonal to these base directions. After orthogonalization, changing another direction will not change base directions anymore.

In practice, We choose “Male” and “Age” as our base direction, and as the following experiment showed, changing the other direction will not change the “Male” and “Age” anymore. Notice that in the column “beard”, the generator could not generate an image with Beard. We speculate that is because in the direction we deviated from regression, the “beard” vector and the “male” vector are extremely correlated. That is said they are likely to be changed at the same. If we enforce the orthogonalization trick on them, one of two directions becomes smaller and far from its original direction. That is why walk on the direction “beard” has less impact on the image after orthogonalization.



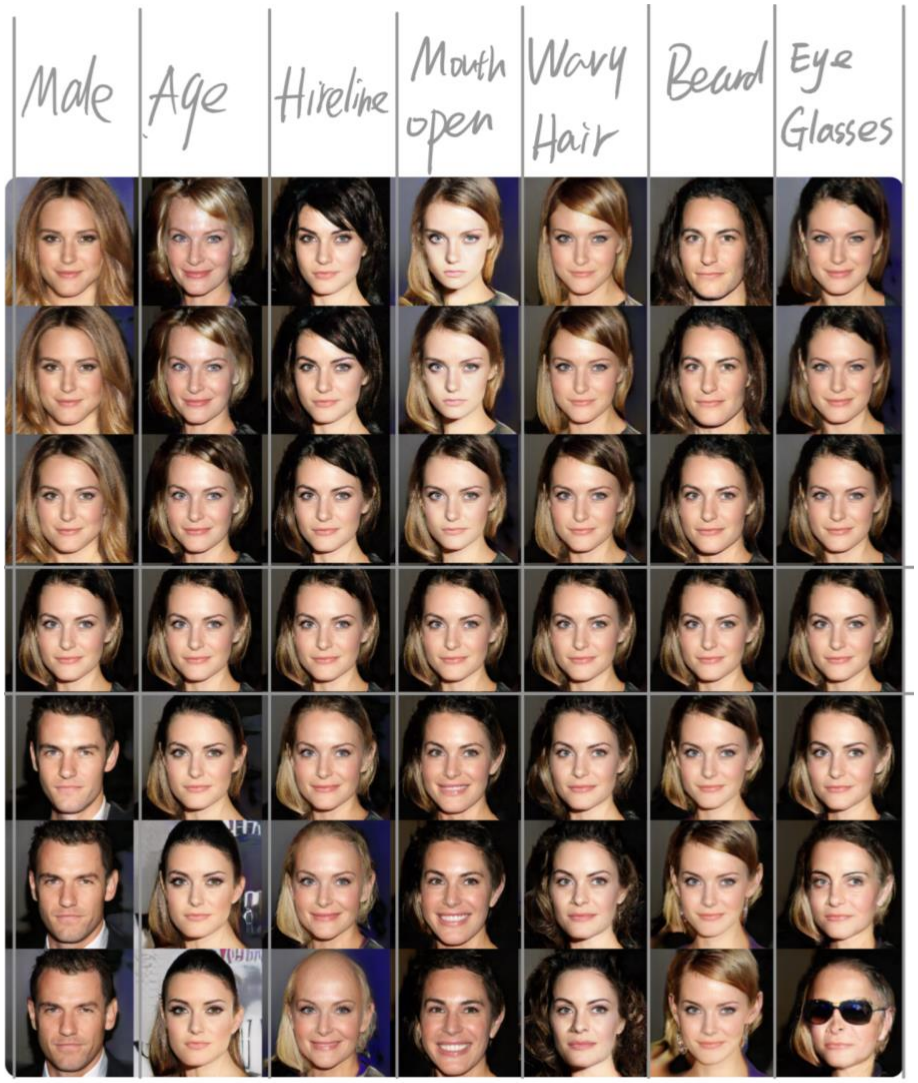


The following heatmap also shown the same result -- since we used “male” and “Age” as base directions, The correlation between other directions to this two directions should be 0 (See difference between the following two heatmaps).


Before disentangle

After disentangle

After orthogonalization, the result finally met our expectation. In the following sets of result, change the other directions, like “beard”, will not change person’s gender and age. However, follow the orthogonalized “bread” direction will not generate woman with beard. We speculate the reason behind this observation is on account of there are no young woman celebrities with bread in our dataset, the discriminator of PgGAN will classify this kind of image (woman celebrities with bread) as a fake image. Subsequently, the generator are less likely to generate the image like that.



Conclusion

For a specific well trained GAN generator and a image dataset with multi-labels, we can modify the noise(code) to manipulate the output. The directions could be learned through a simple linear regression. The orthogonalization of the directions will help to produce more robust stable result. However, currently we are not able to manipulate our own images, because of the code that could generate specific images may not existed. Also, since we learned the direction through a simple logistic regression, the estimate of that direction may not accurate.