Manipulated Network and Feature Map Clustering

Project Report

Jasper Zheng / 21009460
Artificial Intelligence for Media - Mini project

Figure 1: Network manipulation includes unconventional ways to disrupting a trained model and therefore produce impossible, distorted but realistic images that divert from the original outputs.

1. Introduction

A generative neural network learns a set of parameters to approximately model a target distribution [1]. However, understanding the rules encoded in the intermediate layers is always an obscure problem [2]. This project explores three different ways of changing these encoded rules In a trained generative model: intermediate latent space truncation [3], layers' weights manipulation and network bending [4], aims to:

- Help understand how the model processes and generates images.
- Create novel image outcomes that are diverted from the original training data.
- Discover consistent and meaningful visual effects that could potentially be used as an artistic technique.

2. Implementation

The project implements StyleGAN in Tensorflow 2.3.0, explores and showcases changes on the generated images corresponding to different network operations. Finally, it creates a code interface that implements a set of 14 basic network bending operations, allows the chaining and masking of these operations on different network layers.

2.1 Intermediate Latent Space Truncation

StyleGAN generator has a mapping network and a synthesis network. The mapping network re-iterates the base latent vector with shape (1,512) and outputs an intermediate latent vector with shape (1,18,512) [6]. I separated these two networks and intercepted the intermediate vector. Then add deviations to different levels to create new variations to the generated image.

2.1.1 Swapping Vectors

I created two initial latent vectors and used the mapping network to generate the intermediate vector. Then, swap the first eight layers of the two vectors to perform a basic 'style transform': keep the foreground and transfer the 'style' from the other image.

Figure 2.1.1: Swap the first eight layers of the two vectors.

2.1.2 Adding Noise

First, I tried arbitrarily adding noise to each level of the latent vector. The resulting images are shown below.

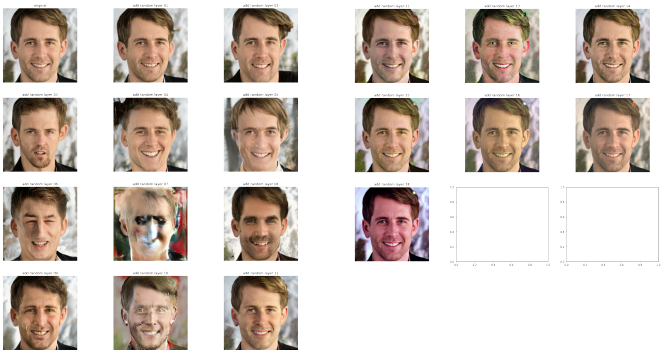


Figure 2.2.2: Adding noises

2.1.3 Invert

Then, I tried inverting each level of the latent vector.

$Invert(w_n) = -w_n$

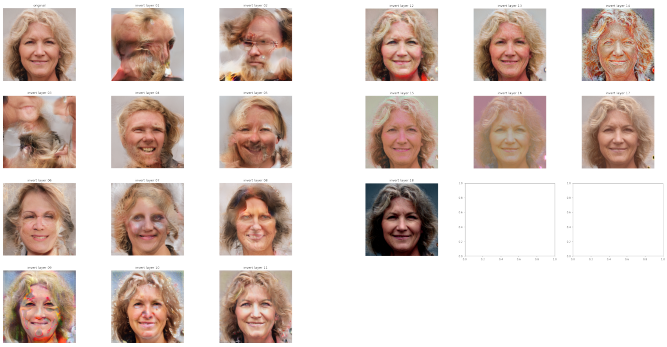


Figure 2.1.3: Inversion

2.1.4 Latent Space Interpolation

Although we started to see some unconventional effects happen, we hope to see if the achieved effects are consistent throughout every initial latent vector. Therefore, I tried interpolating from one initial latent vector to another, then applied the inversion effect on layer 7. The resulting images are shown below.

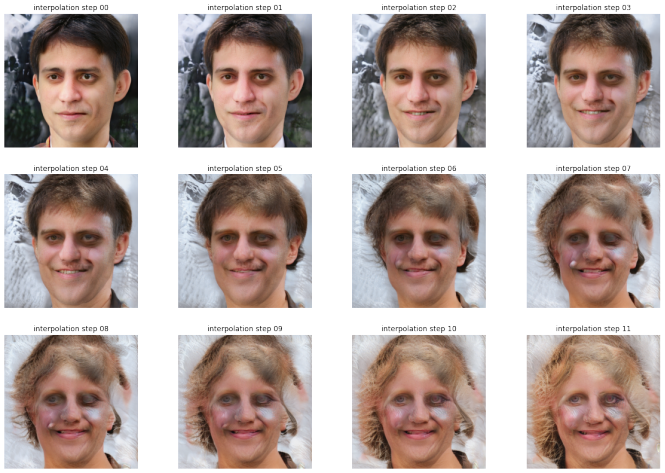


Figure 2.1.4: Latent Space Interpolation

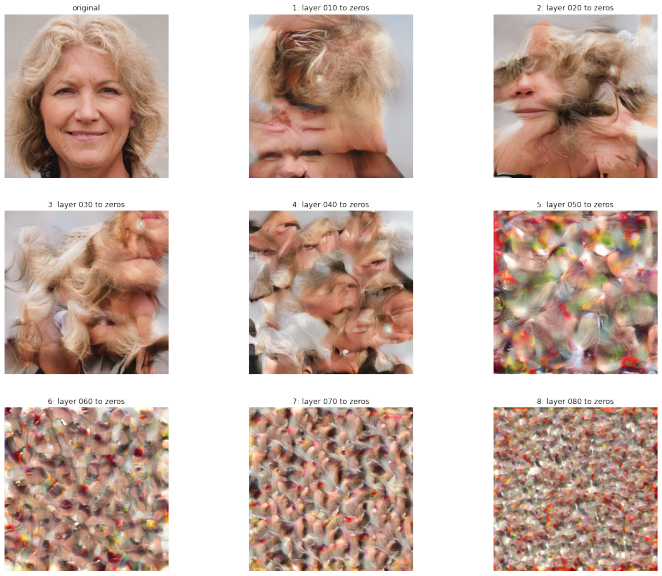
The effect shown in the last interpolation step did not happen in the first step. And after a few attempts, I found that the effects were not consistent throughout every trial. And the result is highly dependent on the initial latent vectors.

2.2 Layers' Weights Manipulation

StyleGAN model implements a set of convolution filters with the shape of a (3, 3) matrix. This section used the get and set method to manipulate the weights matrix encoded in the convolution filters.

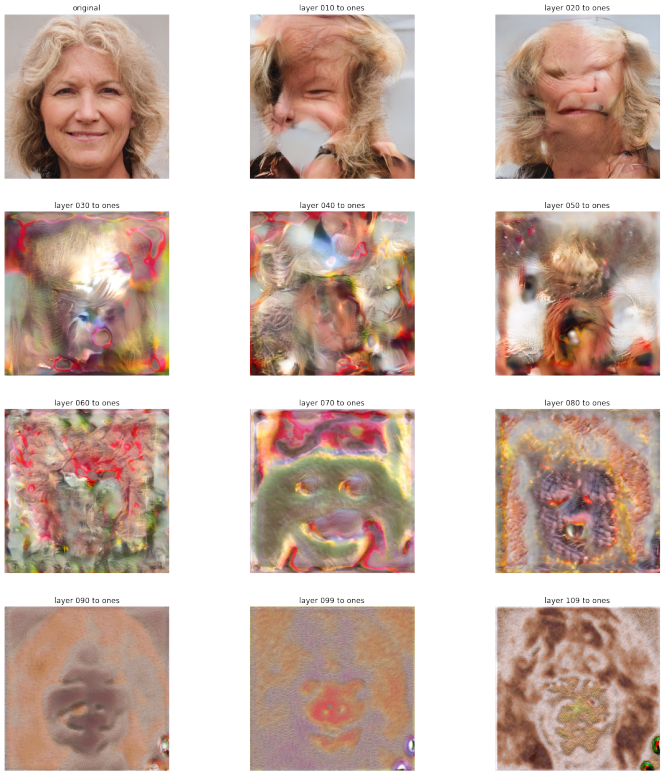
2.2.1 Zeros

Setting all the weights in each layer to zero.



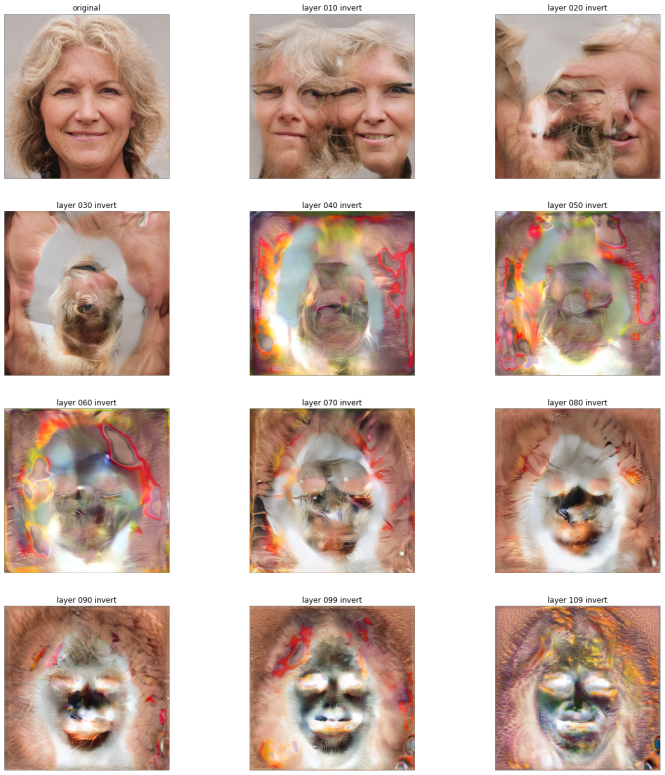
2.2.2 Ones

Setting all the weights in each layer to one.



2.2.3 Invert

Invert all the weights in each layer (w = -w).



2.2.4 Transpose

Flip a weight matrix over its diagonal.

2.2.5 Rotate (90 degrees clockwise and counter-clockwise)

2.2.6 Mean Filter

Apply a mean filter to the convolution kernels.

2.3 Network Bending

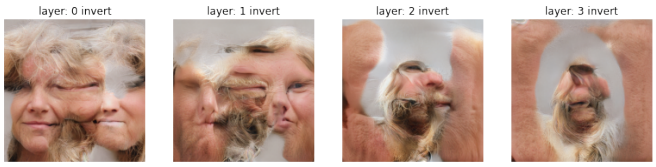
In this section, I re-implemented a few network bending operations [4]. Unlike weights manipulation, network bending inserts new layers to the model and operate the feature maps instead of the convolution filter.

In the file 'NetworkOperations.py', 14 basic operations were implemented in the 'Block' class, which could be inserted after specific model layers. The modified StyleGAN model takes a dictionary variable as an operation plan.

2.3.1 Scale

2.3.1 Translate

2.3.2 Ivert



2.3.3 Shuffle

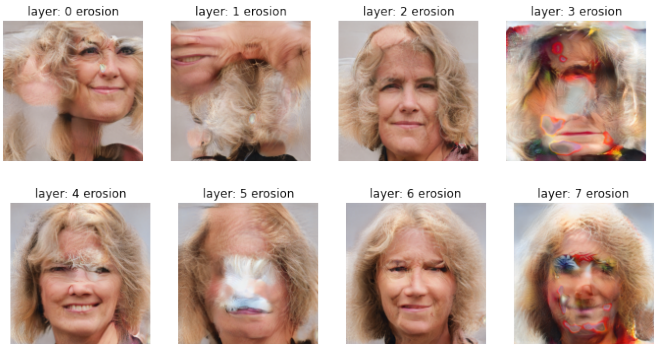
Randomly shuffle the order of a few feature maps.

2.3.4 Mean Filter

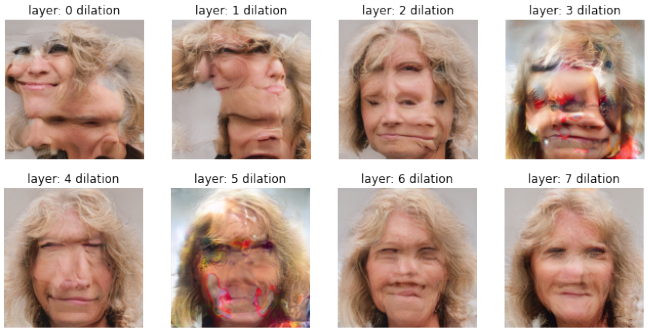
2.3.5 Rotation

2.3.6 Sharpen

2.3.7 Erosion



2.3.8 Dilation



2.4 Feature Map Clustering

This section attempts to cluster the feature maps in the intermediate layers to create groups. Based on the assumption that spatially similar features contribute to specific semantic properties of the generated images [3], operations on grouped layers might create more interpretable outcomes. Ideally, it might create a set of directions to manipulate the network.

First, feature maps from the first 0 to 7 layers (4x4 to 32x32) were generated by forward passing a single latent vector to a model without additional operations.

2.4.1 Settings

For the first two layers, I directly apply the KMean clustering algorithm to flattened data since they have relatively low resolution, and could be clustered based on spatial pixel information. And for the layers with resolution above 16, I first used the VGG16 [5] model to extract high-level features from the feature maps, then clustered the extracted feature by the KMean algorithm.

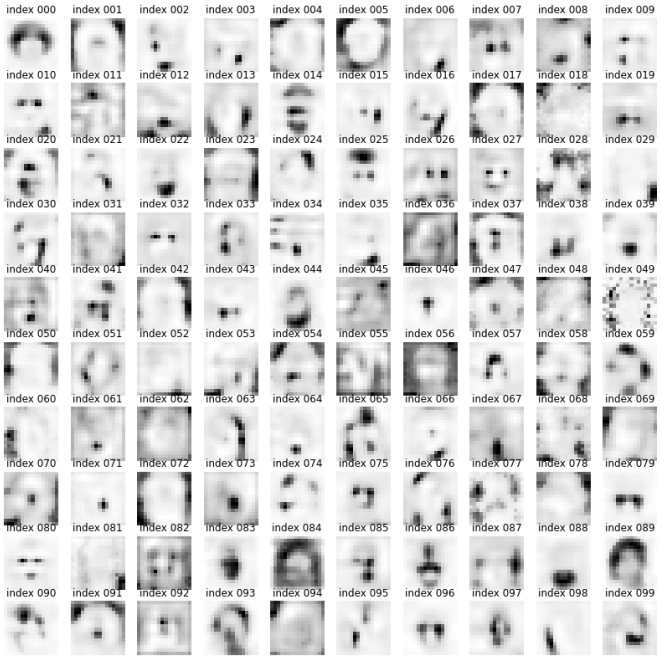


Figure 2.4.1a: Unclustered feature maps generated by layer 0 to 7 (512 for each layer, only showing the first 100).

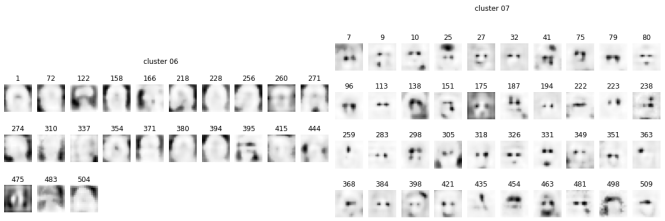


Figure 2.4.1b: Clustered Feature Maps from Layer 4 (Cluster 7 and 7) (16x16)

2.4.2 Results

The spatial information in clusters 7 (plots on the right in Figure 2.4.1b) indicated that it might have something to do with the formation of eyes. Therefore, I tried to add scale layers that masked to clusters 0 and 7.

Figure 2.4.2a: Scale operation applied on cluster 0 and 7 with scale factor: -8, -4, 0, 2

Figure 2.4.2b: Mean filter on cluster [0, 7], [0, 2, 3], [0, 2, 3, 7] and [1, 4, 5, 6]

Differennt from the operation layers in the unclustered operation, the clustered operation only affected specific elements.

Operations could also be combined to remove or emphasise certain elements. In this set of operations, I tried to remove all the facial elements and expand the features that generate hairs.

```
{'resolution':16,
  'install_after':'Conv0_up',
  'layers':[{'operation':'erosion',
             'name':'01',
             'with_norm':True,
             'clusters':[1,5,6]},
            {'operation':'mean_filter',
              'name':'02',
              'with_norm':True,
              'filter_shape':5,
              'clusters':[3]},
            {'operation':'mean_filter',
              'name':'02',
              'with_norm':True,
              'filter_shape':3,
              'clusters':[2,3]},
            {'operation':'scale',
              'name':'02',
              'with_norm':True,
              'scale':-6,
              'clusters':[3,7]},
            {'operation':'scale',
             'name':'02',
             'scale':8,
             'with_norm':True,
             'clusters':[1,5,6]},
            {'operation':'erosion',
             'name':'03',
             'with_norm':True,
             'clusters':[1,4,5,6]},
            {'operation':'dilation',
              'name':'02',
              'with_norm':True,
              'clusters':[3,7]},
            {'operation':'scale',
             'name':'02',
             'scale':-4,
             'with_norm':True,
             'clusters':[3,7]},
            {'operation':'scale',
             'name':'02',
             'scale':-8,
             'with_norm':True,
             'clusters':[0]},
            ]
},
```

Figure 2.4.2c and 2.4.2d: Chained operations on different layers to intentionally remove facial elements and emphasise hairs.

3. Discussion

Three ways of manipulation provide different levels of control when creating images.

Figure 3a: Unique effect created by invert layer 7 and 8 in the intermediate latent vector.

Latent space truncation gives more unexpected effects that did not show in the latter two methods (e.g. invert layer 7 or 8). We could create unconventional but consistent effects by controlling the intermediate latent vector while interpolating a part of the vectors. For example, I made the following moving image by inverting the third layer of the intermediate latent vector, then interpolating only the first four vectors to create movements while keeping the overall colouration. However, the effects are inconsistent throughout the interpolation and highly dependent on the initial latent vectors.

Figure 3b: Interpolating only the first four layers in the itermediate latent vector to maintain colour and visual style.

Layers' weights manipulation produces consistent effects while interpolating between different latent vectors. Operations on lower-level layers are likely to affect the formations of an image, resulting in dramatic distortions and permutations. In comparison, operations on higher-level layers are likely to affect the colouration, ending with unnatural colours or almost blank images.

However, operations on the latent vector or the weight matrix are not likely to be interpretable. For example, the rotate operation on any layers of weight matrix does not cause rotation to the final image. Instead, it only corrupts the formations and produces a unique style of distortions. Whereas the rotate and translation operation in the network bending technique, especially in latter layers, would instantly cause rotation or translation in the final output.

Overall, this project took an in-depth look into tweaking the StyleGAN model at different levels, showcasing a set of effects resulting from various operations. Network manipulation is an unconventional way of disrupting a trained model to produce impossible, distorted but realistic images that divert from the original outputs.

4. References

[1] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” *arXiv [cs.NE]*, 2018.

[2] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba, “Rewriting a deep generative model,” *arXiv [cs.CV]*, 2020.

[3] O. Katzir, V. Perepelook, D. Lischinski, and D. Cohen-Or, “Multi-level latent space structuring for generative control,” *arXiv [cs.CV]*, 2022.

[4] T. Broad, F. F. Leymarie, and M. Grierson, “Network bending: Expressive manipulation of deep generative models,” *arXiv [cs.CV]*, 2020.

[5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv [cs.CV]*, 2014.

[6] Y. Shen, J. Gu, X. Tang, and B. Zhou, “Interpreting the latent space of GANs for semantic face editing,” *arXiv [cs.CV]*, 2019.