Manipulated Network and Feature Map Clustering

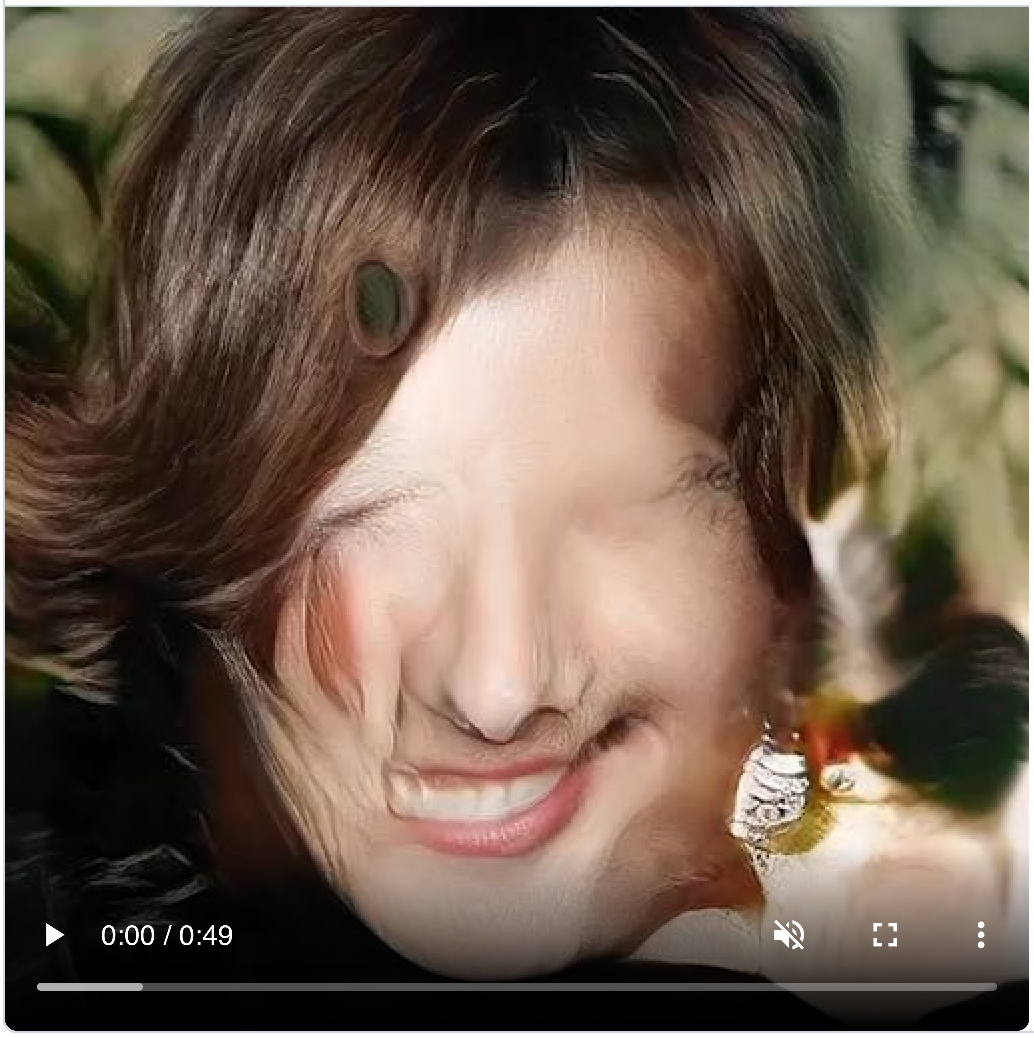
Msc Artificial Intelligence for Media

Jasper Zheng (Shuoyang) / 21009460

This project explores three methods of manipulating pre-trained StyleGAN [1] models: intermediate latent space truncation [2], layers' weights manipulation and Network Bending [3]. It also used VGG16 feature extraction model and KMean algorithm to cluster the feature maps in the intermediate layers to create more interpretable outcomes. Finally, it re-implemented a set of network bending operations to a code interface and showcased a series of novel images produced by the manipulated models.

★★★ Project Report ★★★

interpolation_video.mp4



Implementation

Requirements

The code explicitly require `python 3.7`, `tensorflow==2.3.0`, `tensorflow-addons==0.13.0`, `numpy==1.19.0`.

Intermediate Latent Space Truncation

Codes in the second section in Manipulated_Network_pt1.ipynb

Layers’ Weights manipulation

Codes in the third section in Manipulated_Network_pt1.ipynb

Network Bending

Codes in Manipulated_Network_pt2.ipynb

Feature Map Clustering

Clustering Process: Feature_Map_Clustering_pt1.ipynb
Network Bending on Clustered Model: Feature_Map_Clustering_Pt.2.ipynb

Template

In the file `NetworkOperations.py`, I prepared 14 basic operations that could be inserted after specific model layers.
The implemented StyleGAN model takes a dictionary variable as an operation template.

`resolution` and `install_after` together define the Conv layer that is going to be operated. In the StyleGAN architecture, each resolution has two convolutional layers, `Conv0_up` is the one that scale up the feature maps, `Conv1` is the accompanied layer after the upscaling layer.

`layers` defines a list of bending operations that is going to be inserted.

`operation` takes a string defining the type of operation, including `scale`, `invert`, `shuffle`, `brightness`, `translate`, `vanish`, `mean_filter`, `rotate`, `sharpen`, `erosion`, `dilation`, `mirrorY`, `sin_disrupt`.

An example operation template:

```
operations = [{'resolution':4,
               'install_after':'Conv0_up',
               'layers':[{'operation':'none',
                           'name':'002',
                           'clusters':[]}]
               },
              {'resolution':4,
                'install_after':'Conv1',
                'layers':[{'operation':'none',
                           'name':'002',
                           'clusters':[]}]
               },
              {'resolution':8,
                'install_after':'Conv0_up',
                'layers':[{'operation':'none',
                           'name':'002',
                           'clusters':[]}]
               },
              {'resolution':8,
                'install_after':'Conv1',
                'layers':[{'operation':'none',
                           'name':'002',
                           'clusters':[]}]
               },
              {'resolution':16,
                'install_after':'Conv0_up',
                'layers':[{'operation':'scale',
                           'name':'001',
                           'scale':-3,
                           'clusters':[0,7]},
                          {'operation':'mean_filter',
                           'name':'002',
                           'kernel_size':3,
                           'clusters':[1,5,6]}]
               },
              {'resolution':16,
                'install_after':'Conv1',
                'layers':[{'operation':'sharpen',
                          'name':'003',
                          'sharpen_factor':5,
                          'with_norm':True,
                          'clusters':[]}]
               },
              {'resolution':32,
                'install_after':'Conv0_up',
                'layers':[]
               },
              {'resolution':32,
                'install_after':'Conv1',
                'layers':[]
               },
              ]
```

Then we'll use `rebuild_operations()` to insert operations to the model.

```
model.rebuild_operations(clusters, operations)
y = model.generate_from_vector_fast(latents,is_visualize=False)
```