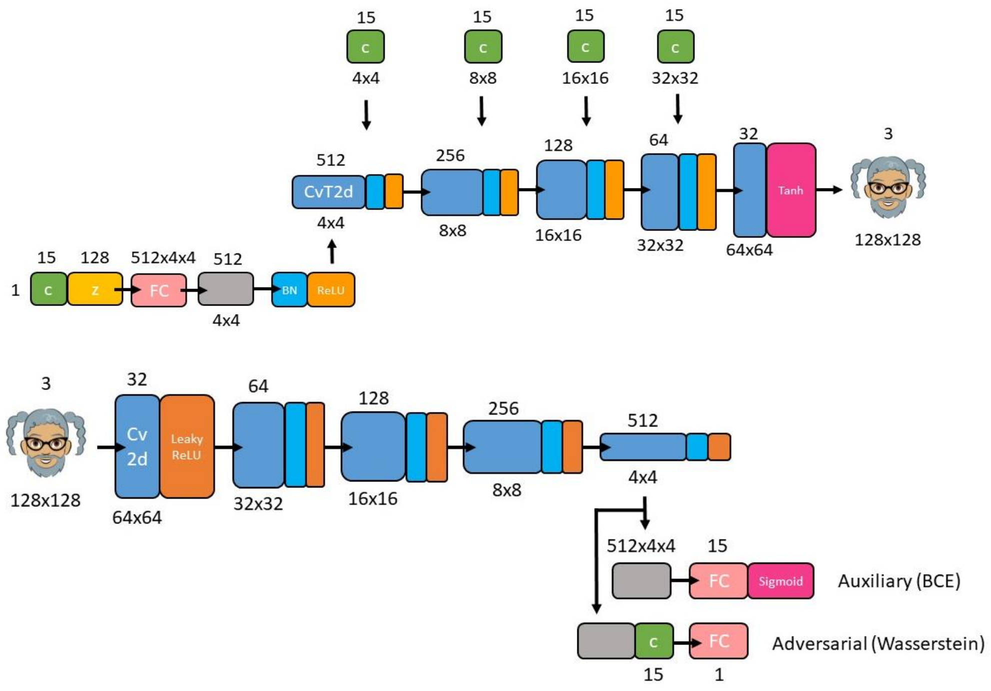
Applied Deep Learning HW4-Report

B04901020 電機四 解正平

Q1：Explain the structure of your networks and loss terms in detail.

1. Network structure

I simply use ACGAN structure but some modifications to build my networks. We only need real images, one-hot condition and gaussian noise for the whole training procedure.



- Generator：

I first sample a normal gaussian noise (128) and concatenate with one-hot condition (6+4+3+2) as the input. After then, linear project the input to 512 * 4 * 4 size and use 5 ConvTranspose2d layer to upsample input into a real image 3 * 128 * 128. Before each layer input, I concatenate a condition map (15 * 2D map size) with the previous layer output. After each layer output, I add BatchNorm 2D and ReLU but Tanh for last layer.

ConvTranspose2d			
Num layer	Kernel size	Stride	Padding
5	4	2	1

- Discriminator：

The inputs are an image (3x128x128) and a one-hot condition. I use 5 Conv2d layer to project the image into a vector (512x4x4). After each layer output, I add BatchNorm 2D and LeakyReLU(0.2) but no batch normalization for the first layer. With an image representation, I implement a linear layer with representation and condition to show the adversarial score and a linear layer only with representation to show the classifier score between one-hot condition.

Conv2d			
Num Layer	Kernel Size	Stride	Padding
5	4	2	1

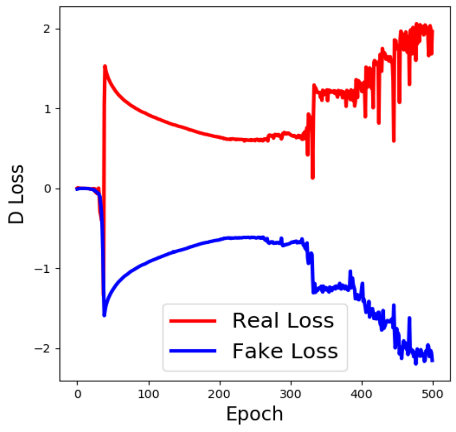
2. Loss design

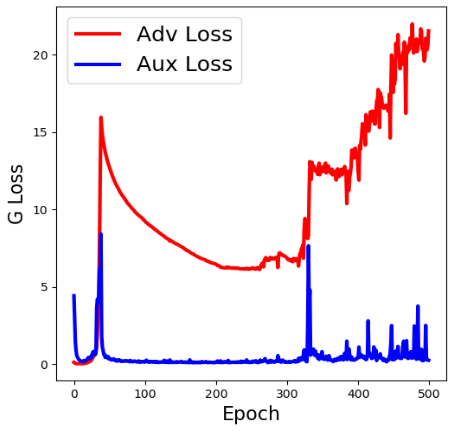
- Adversarial Loss：Wasserstein distance with gradient penalty

$$\mathcal{L}_D^{\mathrm{WGANGP}} = \mathcal{L}_D^{\mathrm{WGAN}} + \lambda \mathbb{E}_{\hat{x}\sim p_g}[(||\nabla D(\alpha x + (1-\alpha\hat{x})||_2 - 1)^2] \quad \mathcal{L}_G^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$$

- Auxiliary Loss：Binary cross entropy for one-hot condition

I use WGAN-GP as my adversarial loss and the BCE as my auxiliary loss. The reason for using BCE is that we can simply view the condition as multi-label problems for each class and it is easier to implement. From the following image, we can find the discriminator has real loss going down and fake loss going up while the generator has a convergent auxiliary loss and going down adversarial loss. However, it seems to be unstable after 300 epochs, which shows that we cannot train GAN for a long time.





3. Hyper parameters

Learning Rate	Lamda for GP	Lamda for Aux	Batch Size
0.001	10	20	32

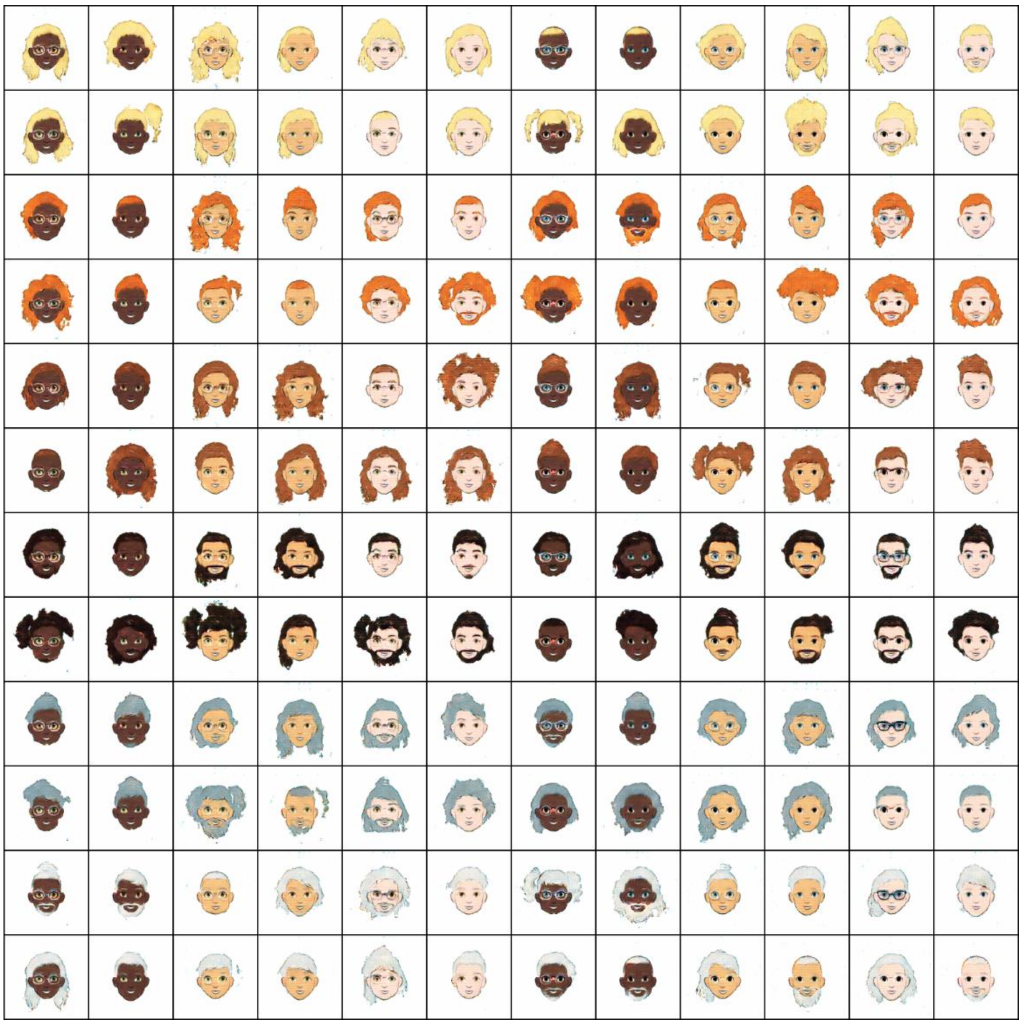
Adversarial Loss = Wasserstein distance + GP * 10

Loss = Adversarial Loss + Auxiliary Loss * 20

Q2：Plot your training progress (10 pics).

- Best FID：89.276 (epoch 300)
- Range：epoch (50, 100, 150 … 500)
- GIF：https://bit.ly/2JPKSVx (10 pics in GIF)

The last picture is in the following, which is generated from the model training last 500 epochs. We can find the model will have collapsed problems when training for a long time. Moreover, the picture from the model training on 50 epochs is good enough.



Q3：Design at least 3 different experiments.

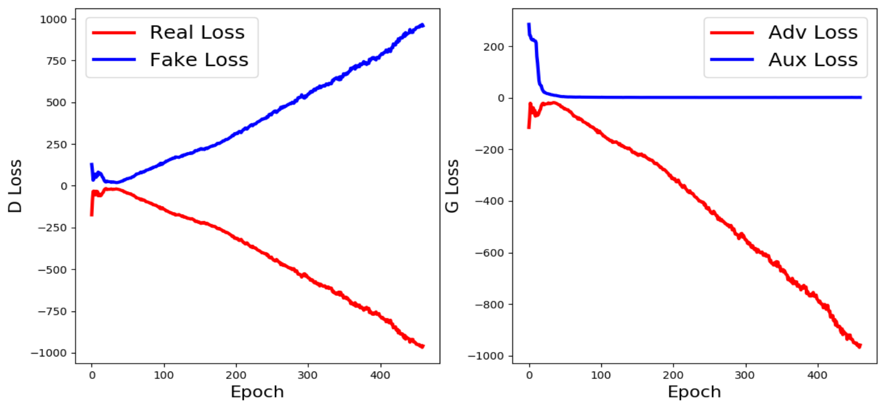
1. Spectral Normalization

a、 Settings

I remove the batch normalization layer from my original discriminator introduced in above section. After then, I add spectral normalization layer after each convolution and linear layer. The parameters are all the same with Q1.

b、 Comparison/Observation on training procedure

From the following image, we can find use spectral normalization can lead to a more stable training procedure. The discriminator real loss goes lower while fake loss goes higher with a more real fake image. When it comes to generator, the adversarial loss goes down to enable images to fool discriminator and the auxiliary loss converges fast at the beginning.



c、 Training Progress

- Best FID：63.240 (epoch 450)

- Range：epoch (50, 100, 150 … 450)

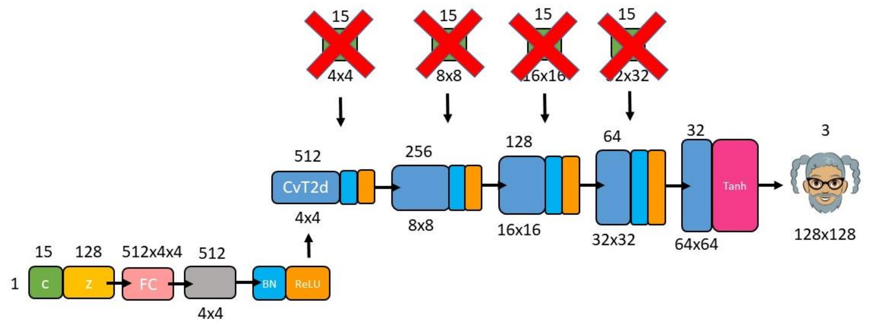
- GIF：https://bit.ly/2I6SmQi

The last picture is generated from the model training last 450 epochs. We can find that spectral normalization has the ability to reduce collapsed problems. It is difficult for human to find a similar face, showing the sample noise can have effective difference.

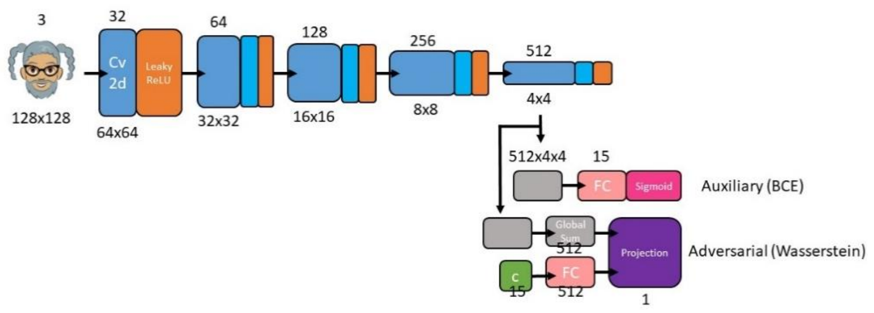
2. Model Condition on Label

a、 Settings

I have two settings about model condition on label in this section.



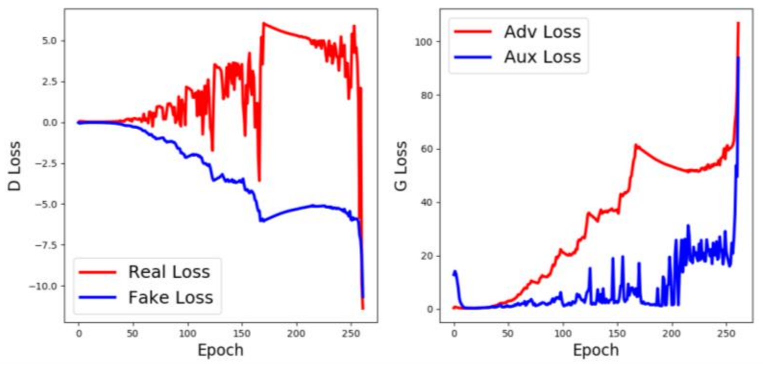
First, I compare the generators, which I simply remove all the 2D condition map but only use the input condition.



For the other, I compare the discriminator with spectral normalization, which I don't concatenate condition for the adversarial score but use projection layer on condition.

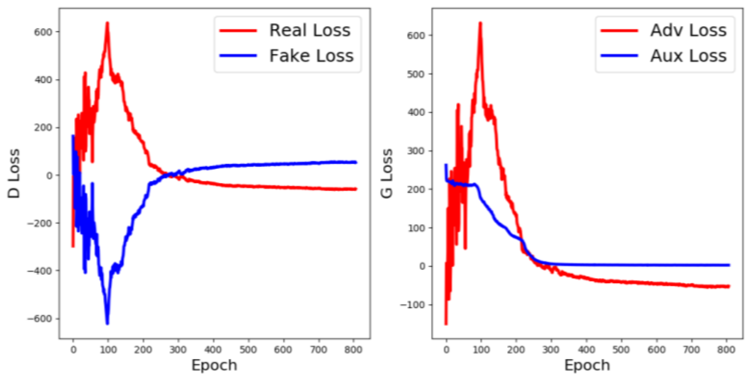
b、 Comparison/Observation on training procedure

- Generator (without concatenation)



The loss without label concatenation is more unstable than with. Moreover, there is a strange phenomenon that the generator adversarial loss goes higher and the discriminator fake loss goes lower, which doesn't meet my previous result.

- Discriminator (projection)



The concatenation with spectral normalization had showed stable loss during training while the projection had an unstable loss at the beginning. However, the result goes better when time passing. I think it is the benefits of the spectral normalization.

c、 Training Progress

- Generator (wo concatenation)
    - Best FID：131.046 (epoch 200)
    - Range：epoch (50, 100, 150 … 250)
    - GIF：https://bit.ly/2MfG11Q

    We can find that the unstable loss may lead to a bad quality of images. The last picture shows a lots noise just like a coming explosion on the image. Moreover, the FID score is worse than the generator with concatenation.

- Discriminator (projection)
    - Best FID：44.292 (epoch 800)
    - Range：epoch (50, 100, 150 … 800)
    - GIF：https://bit.ly/2W86aEh (Need download to view)

    We can find a strange picture at the beginning. However, it goes better after 150 epochs, which meets the loss results we had discussed in the above section.

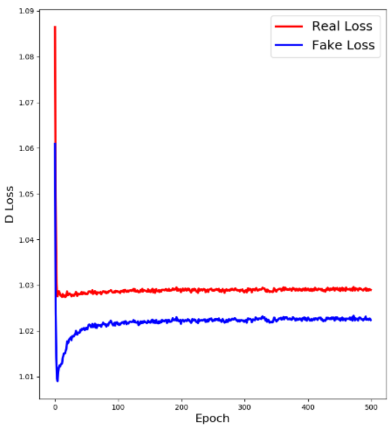
3. Methods for Wasserstein Distance

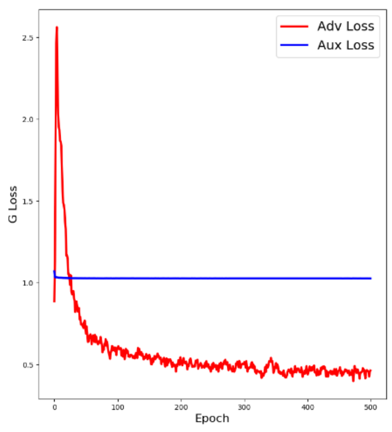
a、 Settings

I use different tricks on Wasserstein Distance. First is the methods in section Q1 with gradient penalty for training. The other is the clipping methods and Wasserstein divergence methods.

b、 Comparison/Observation on training procedure

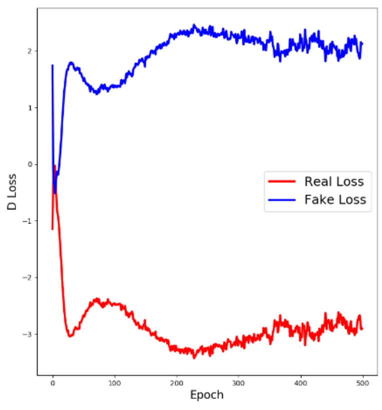
- Wasserstein Clipping

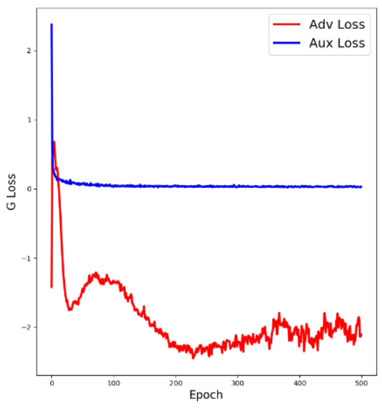




The discriminator loss and generator adversarial loss is stable and can converge gradually. However, the generator auxiliary loss decreased little and almost stayed static. I think the generator loss is dominated by the adversarial rather than auxiliary.

- Wasserstein Divergence





This method can train a stable procedure without gradient penalty and spectral normalization. We can find the discriminator fake loss decreased and real loss increased, which is a correct phenomenon. Nevertheless, it may turn unstable for a long training time. Fortunately, we didn't see a loss explosion.

c、 Training Progress

- Wasserstein Clipping
    - Best FID：216.239 (epoch 400)
    - Range：epoch (50, 100, 150 … 500)
    - GIF：https://bit.ly/30Qz3U3 (Need download to view)

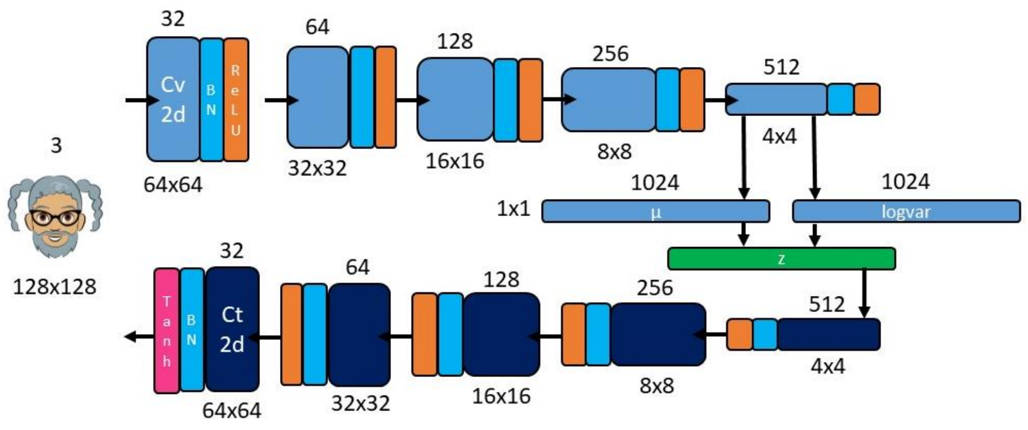
    This method cannot learn the condition well. The given class didn't match the reasonable image. Although it can generate a face, a clear and conditioned image is hard to obtain. Perhaps this method need more parameters tuning.

- Wasserstein Divergence
    - Best FID：55.371 (epoch 500)
    - Range：epoch (50, 100, 150 … 500)
    - GIF：https://bit.ly/2EHuCBz

    The image looks better when training for a long time. The results barely have the collapsed problem, which is a good method to train GAN.

Bonus：Unsupervised Conditional Generation

1. Model architecture (VAE)



I use VAE for my unsupervised generation model. Due to the purpose of only considering image processing, I build the structure with all convolution layers. The latent code z is set based on the reparameterization trick during training and we can sample some gaussian noise for random generation on testing.

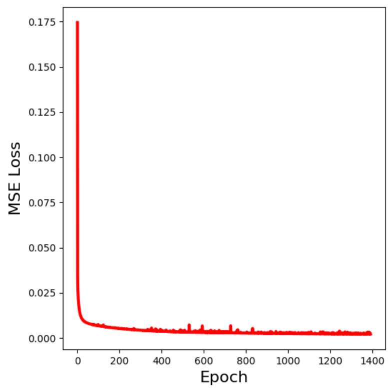
2. Loss terms

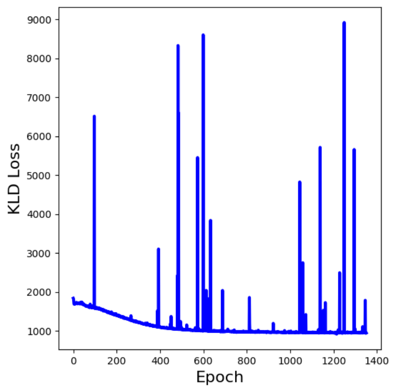
$$\text{recon. loss} = \text{MSE}(\mathbf{x}, \mathbf{x}_r) \qquad \text{Compute reconstruction loss}$$

$$\text{var. loss} = -\text{KL}[\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x) \| \mathcal{N}(0, I)] \qquad \text{Compute variational loss}$$

$$\text{L} = \text{recon. loss} + \text{var. loss} \qquad \text{Combine losses}$$

I use pixel-wised averaged mean square error (MSE) and KL divergence as my loss function. MSE is for the minimization of reconstruction error while KLD is for the latent space similar to normal distribution.





From the above image, we can find the MSE loss is stable but KLD loss not. MSE loss can decrease gradually while KLD loss may arise some peaks. However, both of them have the trend for minimization.

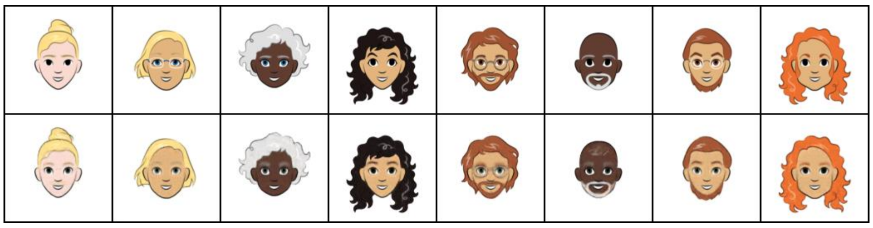
3. Experiment settings

Learning Rate	Beta1	Beta2	Batch Size
0.0002	0.5	0.9	32

4. Results

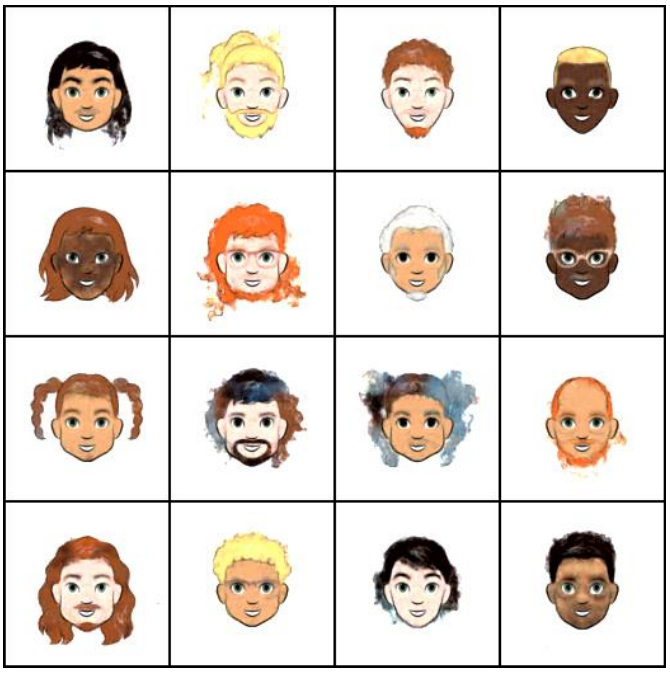
a、 Reconstruction (8)

GIF：https://bit.ly/2VW9f5e



b、 Random Generation (4x4)

GIF：https://bit.ly/2I52IAi



From the images reconstruction, we can find the quality is more and more better. However, there are still some details cannot maintain. From the images random generation, we can also find high quality face, which can be recognized easily. Since it is hard to learn the latent code to represent each attribute, the face may mix some color on hair, face, eyes, and glasses.