ADL HW4 Report

Explain the structure of your networks and loss terms in detail (1.5%)

Basically I used the structure similar in DCGAN for both the generator and discriminator. I have added a conv layer to both the generator and discriminator to resize the output of the image to 128x128. I have also taken off the sigmoid function in the discriminator layer and batch normalization layers in the discriminator for WGAN-GP training and add 4 classifiers(Hair classifier, eye classifier, face classifier, glass classifier) to classify the conditions.

Generator:

```
Generator(
  (gen): Sequential(
    (0): ConvTranspose2d(115, 1024, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): ConvTranspose2d(1024, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace)
    (6): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace)
    (9): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace)
    (12): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace)
    (15): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (16): Tanh()
  )
)
```

Discriminator:

```
Discriminator(
  (conv_layers): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(negative_slope=0.2, inplace)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): LeakyReLU(negative_slope=0.2, inplace)
    (4): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (5): LeakyReLU(negative_slope=0.2, inplace)
    (6): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): LeakyReLU(negative_slope=0.2, inplace)
    (8): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): LeakyReLU(negative_slope=0.2, inplace)
  )
  (discriminator_layer): Sequential(
    (0): Conv2d(1024, 1, kernel_size=(4, 4), stride=(1, 1))
  )
  (bottleneck): Sequential(
    (0): Conv2d(1024, 1024, kernel_size=(4, 4), stride=(1, 1))
    (1): LeakyReLU(negative_slope=0.2)
  )
  (hair_classifier_layer): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=128, out_features=6, bias=True)
    (3): LogSoftmax()
  )
  (eye_classifier_layer): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=128, out_features=4, bias=True)
    (3): LogSoftmax()
  )
  (face_classifier_layer): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=128, out_features=3, bias=True)
    (3): LogSoftmax()
  )
  (glass_classifier_layer): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=128, out_features=2, bias=True)
    (3): LogSoftmax()
  )
)
```

Loss terms:

Adversarial loss:

GAN	Discriminator Loss	Generator Loss
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda \mathbb{E}_{\hat{x}\sim p_g}[(\|\nabla D(\alpha x + (1-\alpha\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$

$$\lambda = 10$$

Classifier loss:

Generator classifier loss
$CrossEntropyLoss\big(P\big(class = hair\big\vert X_{fake}\big), fake\_tag_{hair}\big)$ $+ CrossEntropyLoss\big(P\big(class = eye\big\vert X_{fake}\big), fake\_tag_{eye}\big)$ $+ CrossEntropyLoss\big(P\big(class = face\big\vert X_{fake}\big), fake\_tag_{face}\big)$ $+ CrossEntropyLoss\big(P\big(class = glasses\big\vert X_{fake}\big), fake\_tag_{glasses}\big)$

Discriminator classifier loss
$CrossEntropyLoss(P(class = hair\|X_{real}), real\_tag_{hair})$ $+ CrossEntropyLoss\big(P(class = eye\|X_{real}), real\_tag_{eye}\big)$ $+ CrossEntropyLoss\big(P(class = face\|X_{real}), real\_tag_{face}\big)$ $+ CrossEntropyLoss\big(P(class = glasses\|X_{real}), real\_tag_{glasses}\big)$

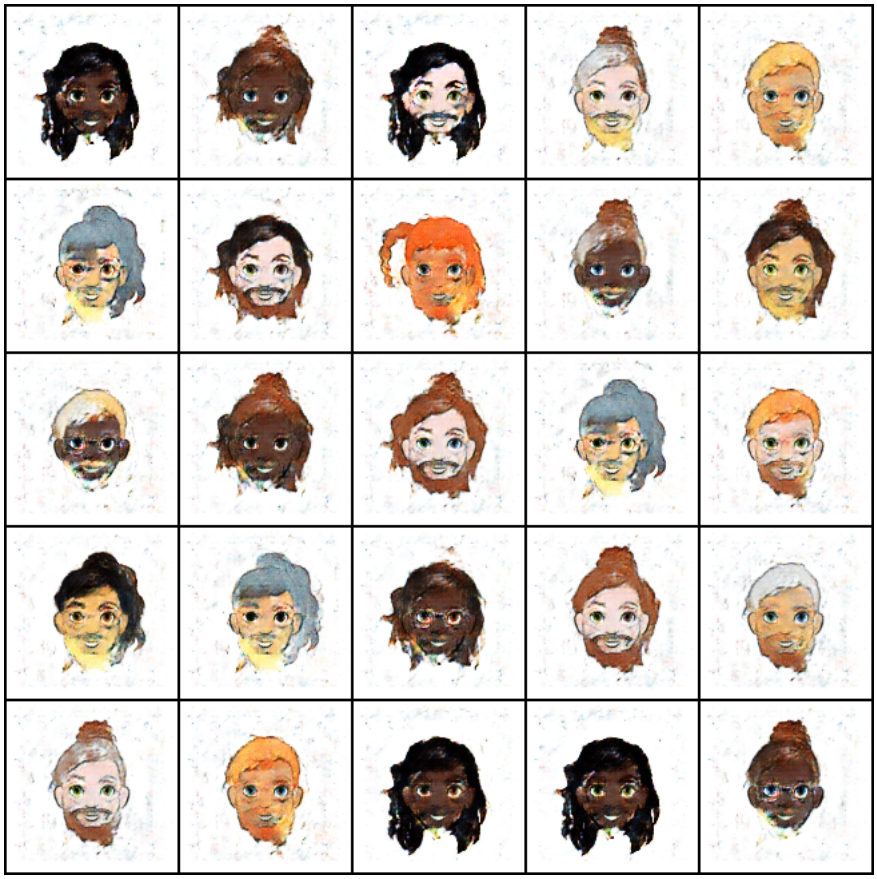
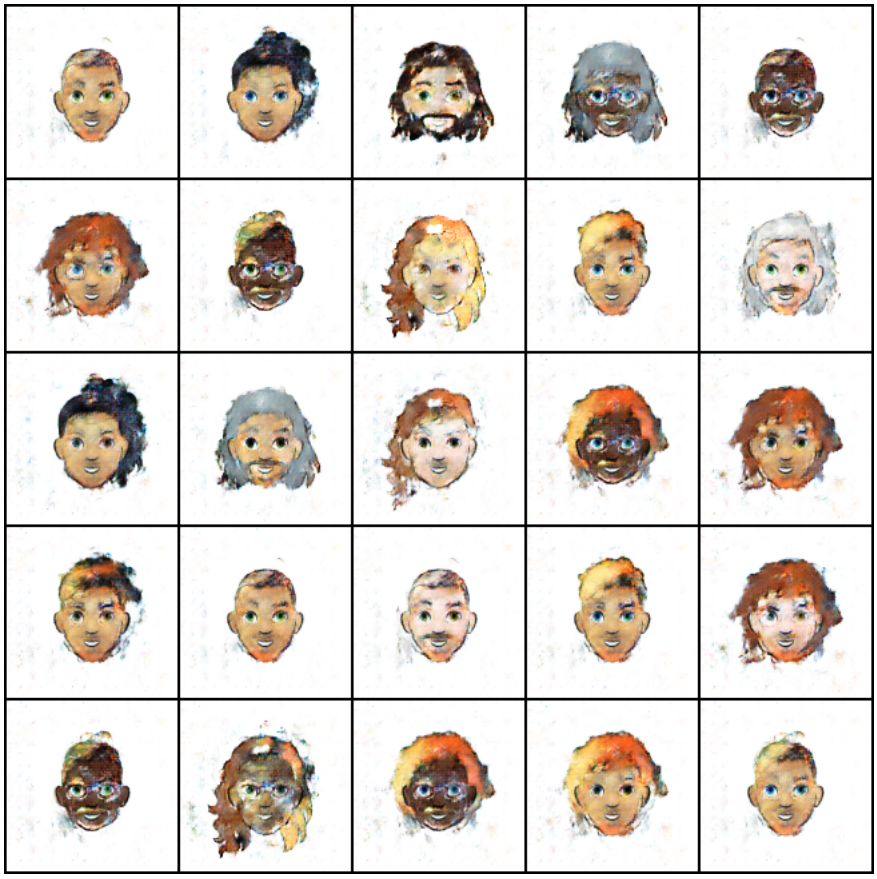
Experiment settings and results

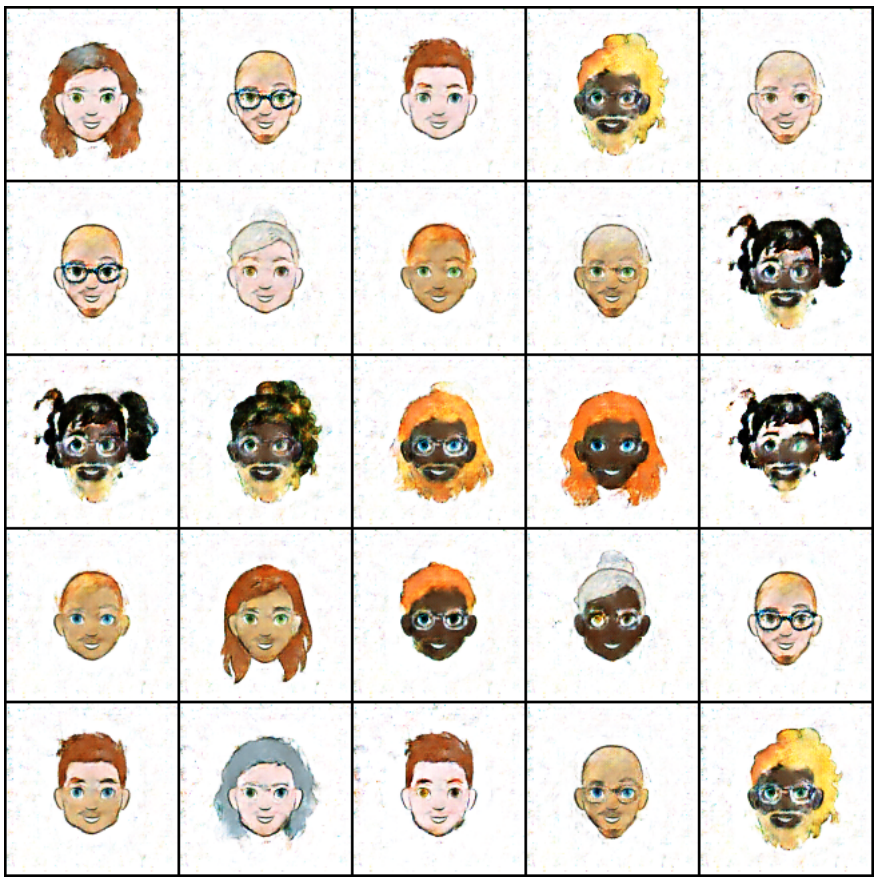
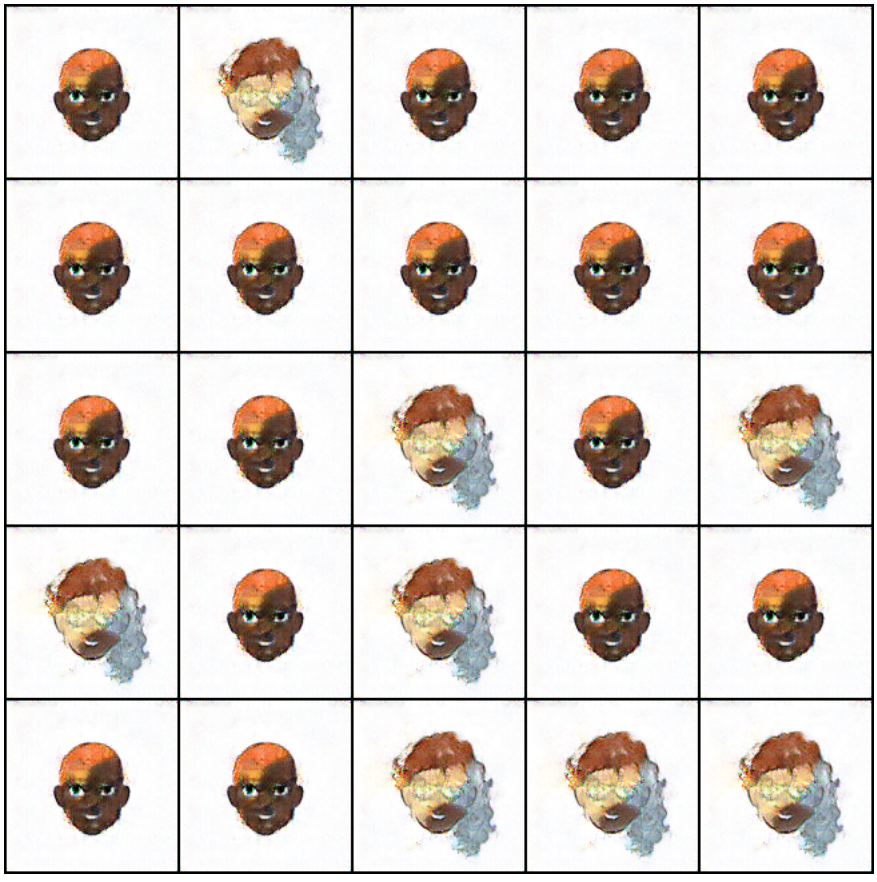
Iter	50000
Discriminator_iter: Generator_iter when training	5:1
Batch size	64
Learning Rate	0.0001
Optimizer	Adam
Latent dim	100
FID score	17.277

Plot your training progress (10 pics) (0.5%)

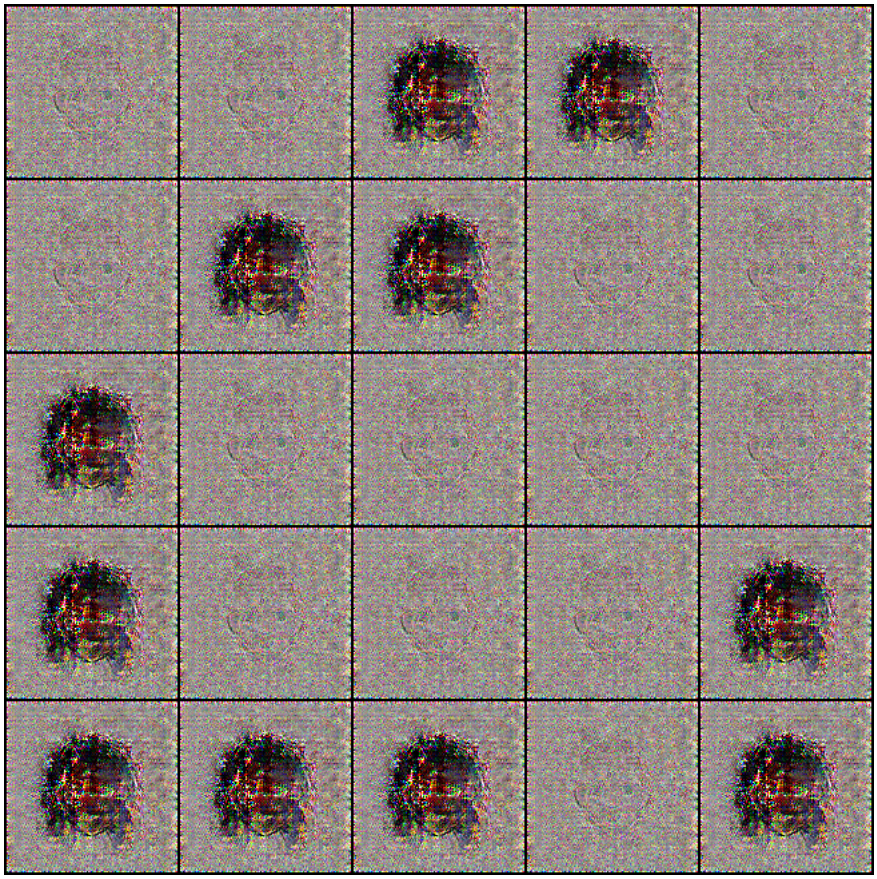
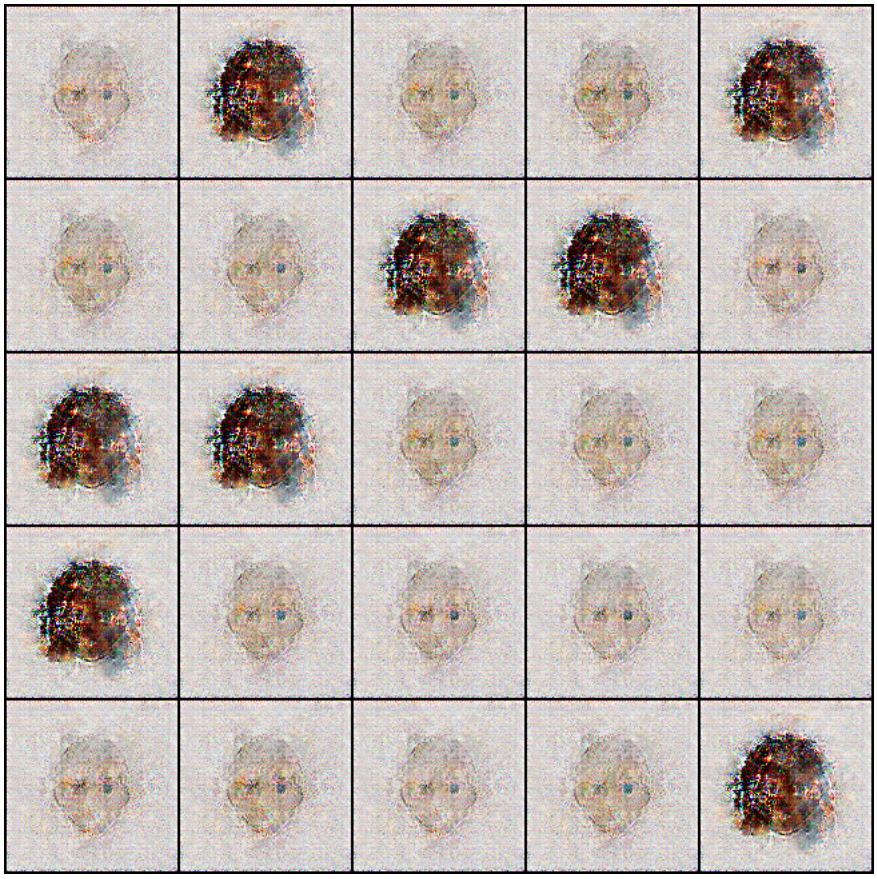
Step 500	

R07922118 資工碩一 翁子騰

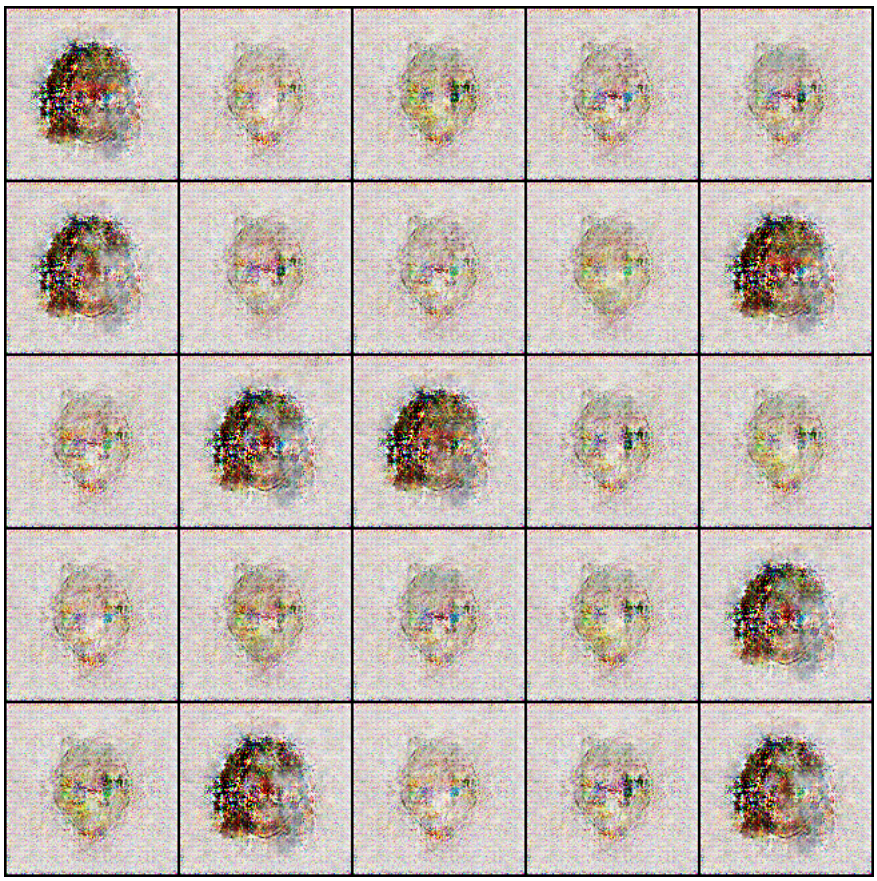
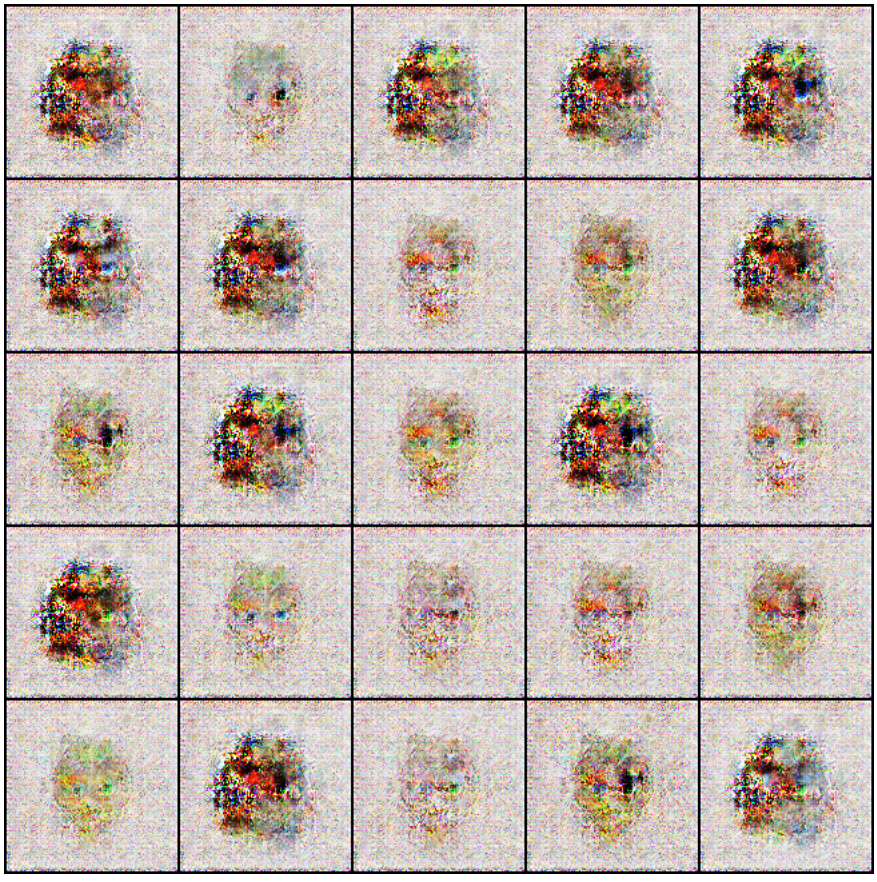
Step 5000	
Step 10000	

Step 15000	
Step 20000	

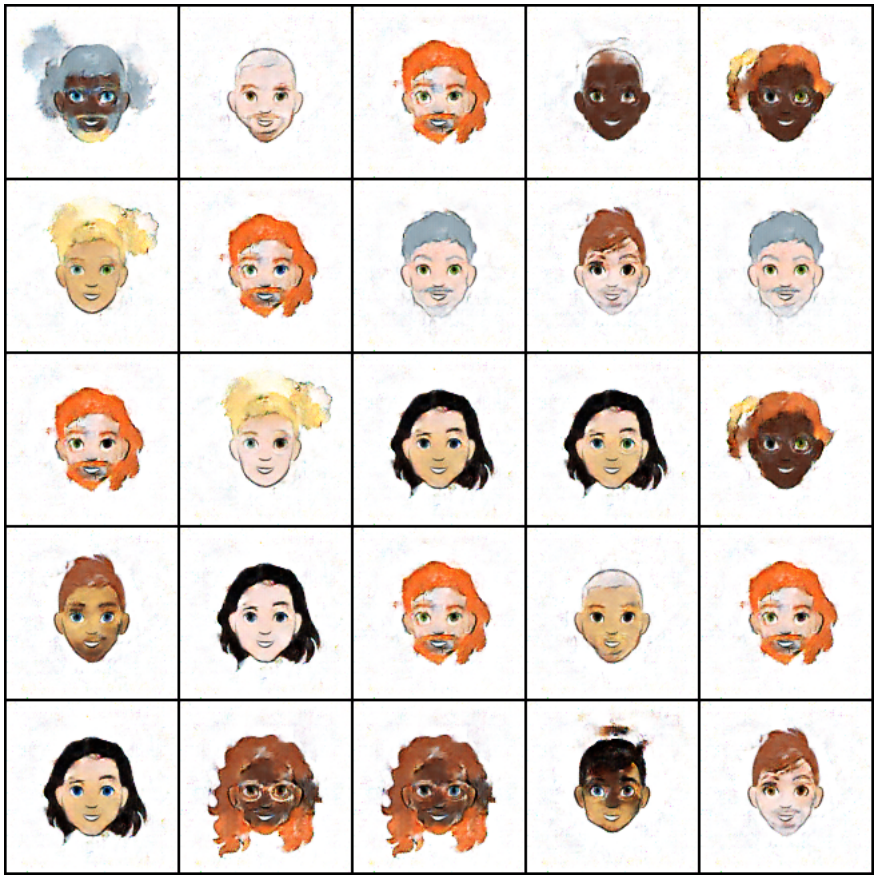
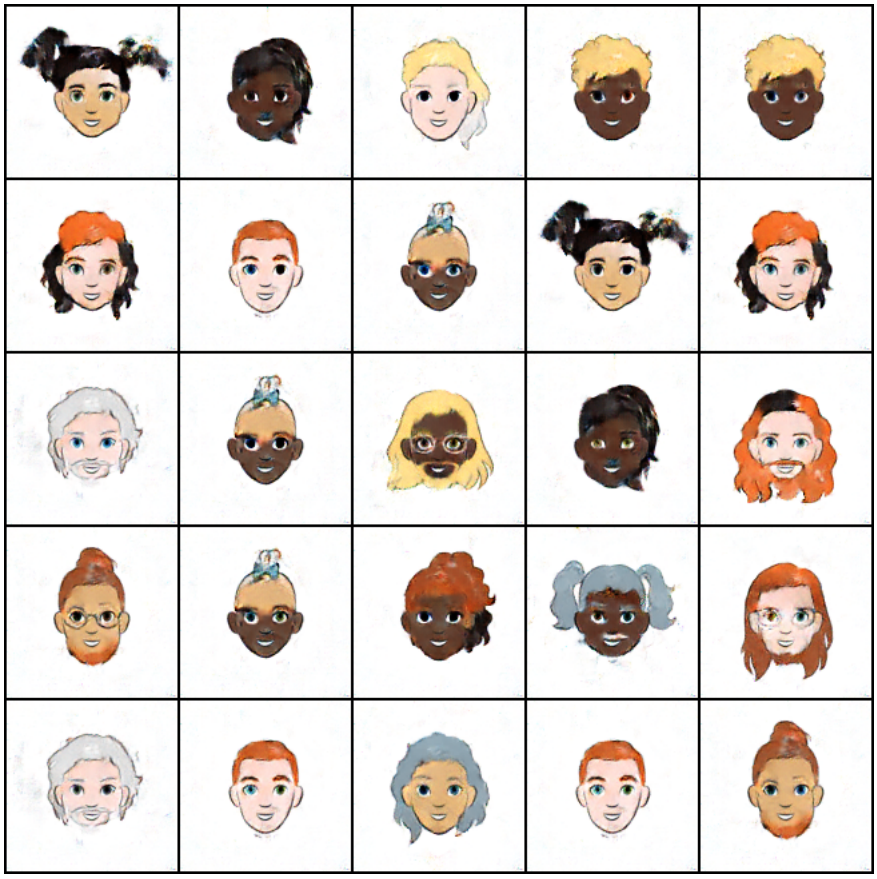
R07922118 資工碩一 翁子騰

Step 25000	
Step 30000	

R07922118 資工碩一 翁子騰

Step 35000	
Step 40000	

R07922118 資工碩一 翁子騰

Step 45000	
Step 50000	

可以觀察到訓練到一半時，generator 一度有崩潰的情形。但是後來有比較好一點。

Design at least 3 different experiments. Describe your settings, making comparisons and report your observations. (4% + 0.5% bonus)

$$X_{fake} = G(z, tag)\ is\ the\ fake\ image(or\ generated\ image),$$
$$and\ \ X_{real}\ is\ the\ real\ image$$

1.

Discriminator Loss function
$-\left(logP(S = real\|X_{real}) + \log\left(1 - P\left(S = fake\|X_{fake}\right)\right)\right) * \frac{1}{2}$ $-(logP(class = c\|X_{real}))$
Generator Loss function
$-\log \mathrm{P}\left(S = fake\middle\|X_{fake}\right) - logP(class = c\|X_{fake})$

The classification loss for fake images are omitted, since we believe that fake images will confuse the discriminator. I divide the adversarial loss for discriminator by 2 to balance the gradient descent schedule of G/D.

Iter	Batch size	FID score
50000	128	187.534

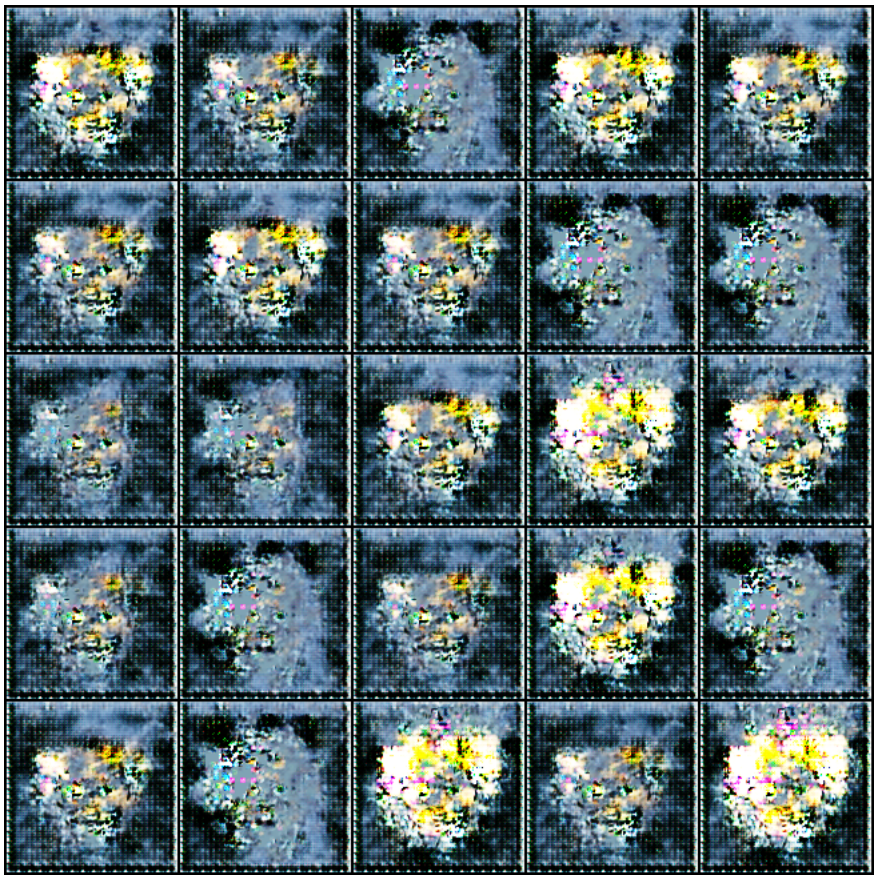
2.

Discriminator Loss function
$-\left(logP(S = real\|X_{real}) + \log\left(1 - P\left(S = fake\|X_{fake}\right)\right)\right) * \frac{1}{2}$ $-(logP(class = c\|X_{real}) + logP(class = c\|X_{fake}))$
Generator Loss function
$-\log \mathrm{P}\left(S = fake\middle\|X_{fake}\right) - logP(class = c\|X_{fake})$

Add classification loss for fake images in the discriminator loss function.

Iter	Batch size	FID score
300000	32	Generator crashed

訓練到約 25 萬 iters 時 generator 崩潰(見下圖)。所以終止訓練。



3.

Discriminator Loss function
$$-\left(logP(S=real\|X_{real})+\log\left(1-P\left(S=fake\|X_{fake}\right)\right)\right)$$ $$-(logP(class=hair\ \|X_{real})+logP(class=eye\ \|X_{real}) + logP(class=face\ \|X_{real})+logP(class=glass\|X_{real}))$$
Generator Loss function
$$-\log P\left(S=fake\middle\|X_{fake}\right)-logP\left(class=hair\middle\|X_{fake}\right) - logP\left(class=eye\middle\|X_{fake}\right)-logP\left(class=face\middle\|X_{fake}\right) - logP(class=glass\|X_{fake})$$

Split the classifier in discriminator into 4 different parts to classify hair colors, eye colors, face colors and with and without glasses.

Iter	Batch size	FID score
50000	128	133.792