隊長：R06922002 姚嘉昇　組員：R06942054 潘仁傑、R06522620 王仁蔚

0. 分工表 (0.5%)

	HW3-1	HW3-2	HW3-3	Report 3-1	Report 3-2	Report 3-3
姚嘉昇 R06922002						
王仁蔚 R06522620						
潘仁傑 R06942054						

1. Model Description (4%)

- **Describe the models you use to, including the model architecture, objective function for G and D.**

以下 model 之 Optimizer 都使用 Adam (lr=0.0001, beta1=0.5, beta2=0.9)，我們使用 WGAN-GP + 5 個 Tips 實作這次的作業 3-1 及 3-2。

- **Image Generation (2%)**

表 1.1 Image Generation 的 G 的 Model 架構

Input：z (62)
fully_connected(4 * 4 * 384) + batch_normalization
Conv2d_transpose(256, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(128, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(64, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(3, (5, 5), stride=2)
Output：Tanh()

表 1.2 Image Generation 的 D 的 Model 架構

Input：image (64 * 64)
Conv2d(64, (5, 5), stride=2) + leakly_relu(leak=0.2)
Conv2d(128, (5, 5), stride=2) + leakly_relu(leak=0.2)
Conv2d(256, (5, 5), stride=2) + leakly_relu(leak=0.2)
Conv2d(384, (5, 5), stride=2) + leakly_relu(leak=0.2)
Conv2d(384, (1, 1), stride=2, padding='valid') + leakly_relu(leak=0.2)
Flatten()
Output：fully_connected(1)

■ **Text-to-image Generation (2%)**

表 1.3 Text-to-image Generation 的 G 的 Model 架構

Input：z (128) + tag (23)
fully_connected(4 * 4 * 384) + batch_normalization
Conv2d_transpose(256, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(128, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(64, (5, 5), stride=2) + batch_normalization + relu
Conv2d_transpose(3, (5, 5), stride=2)
Output：Tanh()

表 1.4 Text-to-image Generation 的 D 的 Model 架構

<table>
<tr><td>Input 1：image (64 * 64)</td><td>Input 2：tag (23)</td></tr>
<tr><td>Conv2d(64, (5, 5), stride=2) + leakly_relu(leak=0.2)</td><td>embed_tag = tf.expand_dims(tag, 1)</td></tr>
<tr><td>Conv2d(128, (5, 5), stride=2) + leakly_relu(leak=0.2)</td><td>embed_tag = tf.expand_dims(embed_tag, 2)</td></tr>
<tr><td>Conv2d(256, (5, 5), stride=2) + leakly_relu(leak=0.2)</td><td rowspan="2">tiled_embeds = tf.tile(embed_tag, [1, 4, 4, 1])</td></tr>
<tr><td>Conv2d(384, (5, 5), stride=2) + leakly_relu(leak=0.2)</td></tr>
<tr><td colspan="2">tf.concat([image_net, tiled_embeds], axis=-1)</td></tr>
<tr><td colspan="2">Flatten()</td></tr>
<tr><td colspan="2">Conv2d(384, (1, 1), stride=2, padding='valid') + leakly_relu(leak=0.2)</td></tr>
<tr><td colspan="2">Output：fully_connected(1)</td></tr>
</table>

2. Experiment settings and observation (2%)

- **Show generated images**

■ **Image Generation (1%)**

A. 使用 WGAN_GP，其 lambd 為 0.25。
B. Batch size 為 64，共 train 50 個 epochs。
C. 使用 Tip 1，Normalize the inputs。
D. 使用 Tip 3，Use a spherical Z，且 Z 為 62 維。
E. 使用 Tip 4，BatchNorm。
F. 使用 Tip 5，Avoid Sparse Gradients: ReLU, MaxPool。
G. 使用 Tip 14，每一個 epoch train D 兩次 G 一次。
H. 使用 Data Augmentation，隨機左右翻轉、旋轉-15～+15 度。

圖 2.1 Result of Image Generation

■ **Text-to-image Generation (1%)**

A. 使用 WGAN_GP，其 lambd 為 0.25。
B. Batch size 為 128，共 train 200 個 epochs。
C. 使用 Tip 1，Normalize the inputs。
D. 使用 Tip 3，Use a spherical Z，且 Z 為 128 維。
E. 使用 Tip 4，BatchNorm。
F. 使用 Tip 5，Avoid Sparse Gradients: ReLU, MaxPool。
G. 使用 Tip 14，每一個 epoch train D 兩次 G 一次。
H. 使用 Data Augmentation，隨機左右翻轉、旋轉-15～+15 度。

Testing Tags	./samples/cgan_original.png
blue hair blue eyes	
blue hair green eyes	
blue hair red eyes	
green hair blue eyes	
green hair red eyes	

圖 2.2 Result of Text-to-image Generation

3. Compare your model with WGAN, WGAN-GP, LSGAN (choose 1) (Image Generation Only) (3%)

- **Model Description of the choosed model (1%)**

  我們使用 WGAN-GP，並與 WGAN 比較，這兩者的 model setting，只差在 Gradient Penalty。

- **Result of the model (1%)**

表 3.1 WGAN-GP 與 WGAN 比較表

	WGAN-GP	WGAN
Epoch 25		
Epoch 50		

- **Comparison Analysis (1%)**

  A. 我們所使用的 model，WGAN-GP 較 WGAN 來的清楚許多。

  B. 不會有左右兩顆眼睛顏色或整顆頭的頭髮顏色不相同的情況。

  C. WGAN-GP 較 WGAN 較快收斂，可參考 README.md 中的.gif。

4. Training tips for improvement (Image generation Only) (6%)

- **Which tip & implement details (1% * 3)**

我們使用 5 個 Tips，並將透過抽掉其中一個 Tip 後，訓練 50 個 epoch 得到的結果，與未抽掉 Tips 的結果比較，以此方法嘗試推斷出該 Tip 對整個 GAN 的影響，以下是我們所使用的 5 個 Tips。

- Tip 1：Normalize the inputs

  將照片 Normalize 到-1 ~ +1 之間，並用 tanh 作為 Generator 的最後一層輸出。

- Tip 3：Use a spherical Z

  不從 Uniform distribution 取樣，改為從 Gaussian distribution 取樣。

- Tip 4：BatchNorm

  Construct different mini-batches for real and fake, i.e. each mini-batch needs to contain only all real images or all generated images. When BatchNorm is not an option use instance normalization (for each sample, subtract mean and divide by standard deviation).

- Tip 5：Avoid Sparse Gradients: ReLU, MaxPool

  Down sampling 使用 Conv2d + stride、Up sampling 使用：ConvTranspose2d + stride，並使用 LeakyReLU 為 Activation Function。

- Tip 14：Train discriminator more (sometimes)

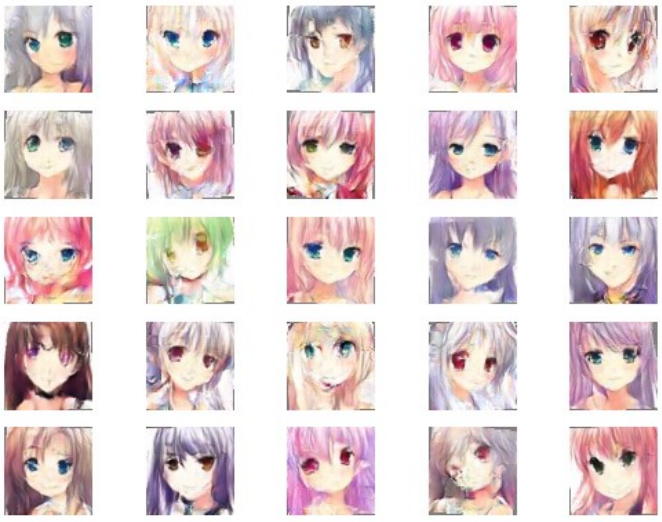
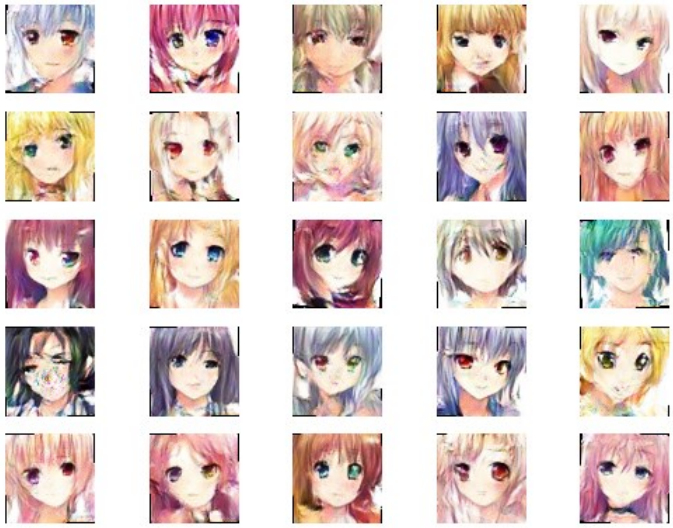
  我們每一個 epoch 中 D 更新兩次，G 更新一次

- **Result (image or loss...etc.) and Analysis (1% * 3)**

- Without Tip 1：Normalize the inputs

  拿掉 Tip 1 之後，我們發現右邊的圖片中，有些眼睛左右顏色不一，有些的眼睛大小也不同。

表 4.1 Tip 1 比較表

With Tip 1, 3, 4, 5, 14	Without Tip 1
	

■ Without Tip 3：Use a spherical Z

拿掉 Tip 3 後，會發現有的臉比較歪，頭髮也比較亂，不像左邊頭髮較順。

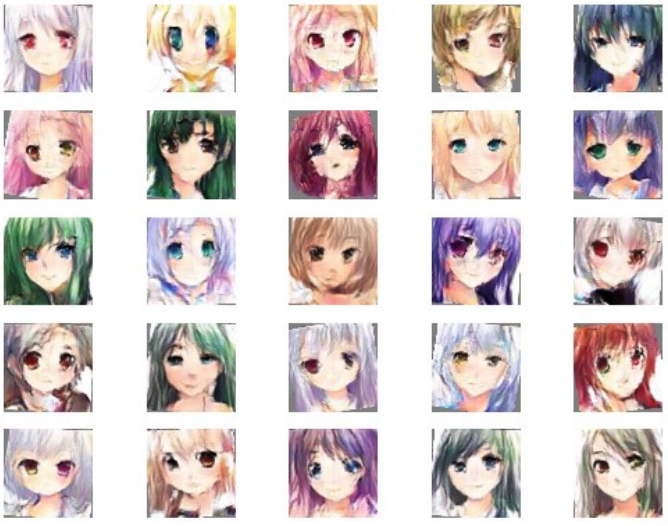
表 4.2 Tip 3 比較表

With Tip 1, 3, 4, 5, 14	Without Tip 3
	

■ Without Tip 14: Train discriminator more (sometimes)

拿掉 Tip 14 後，右邊圖片一樣有大小眼問題，以及頭髮看起來不順，眼睛顏色不同，但沒有像拿掉 Tip3 有臉型不對稱的問題。

表 4.3 Tip 14 比較表

With Tip 1, 3, 4, 5, 14	Without Tip 14
	

5. Style Transfer (2%)

- **Show your result (1%)**
  - **CycleGAN**

    Two domain and transfer result: monet2photo and winter2summer.

圖 5.1 Results of monet2photo

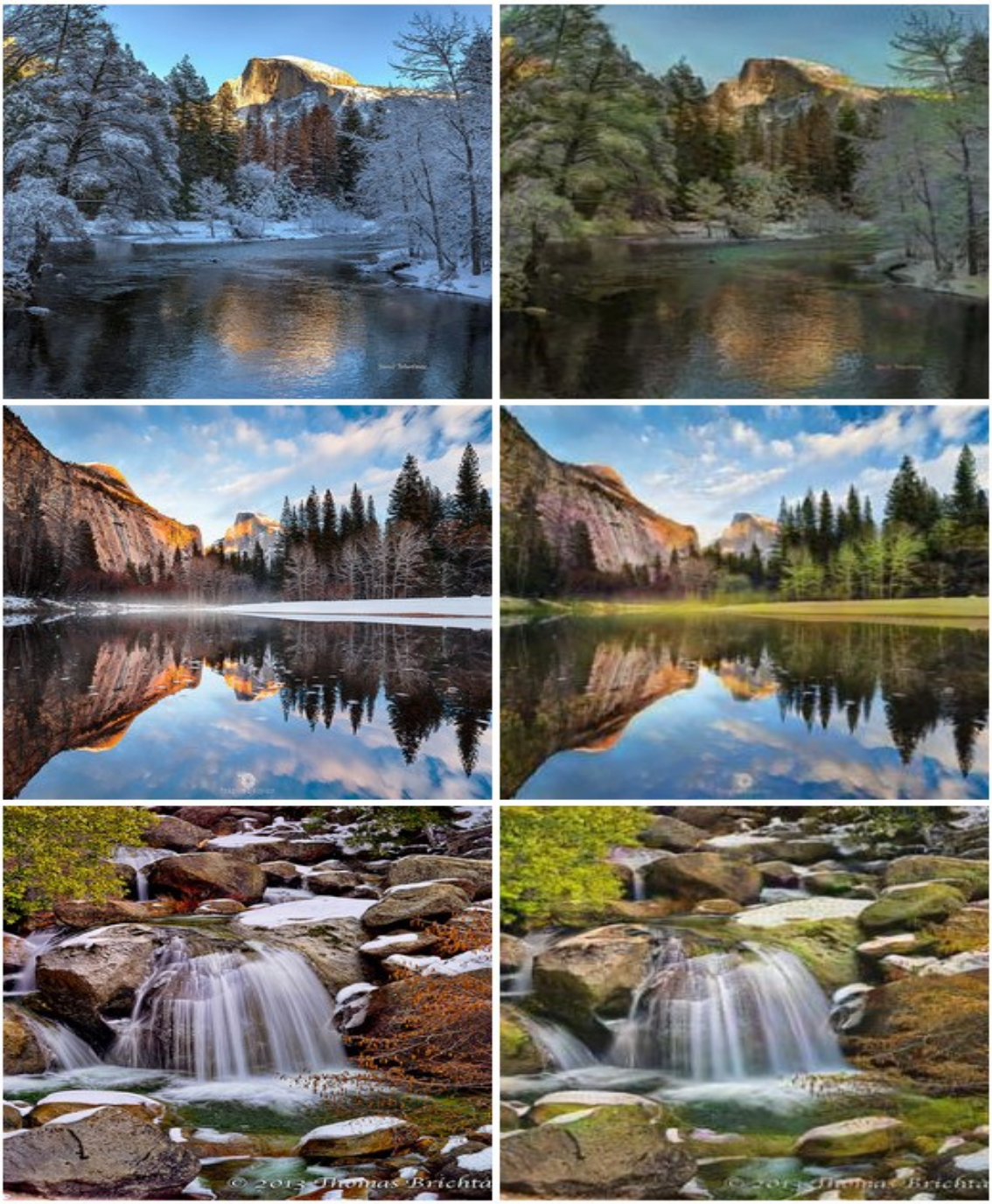


圖 5.2 Results of winter2summer

■ StarGAN

Seven domain and transfer result: 'Black Hair', 'Blond Hair', 'Brown Hair', 'Male', 'Young', 'Mustache', and 'Pale Skin'.

表 4.3 Tip 14 比較表

Black Hair Male Young Mustache	
0101001 Blond Hair Male Pale Skin	
0011111 Brown Hair Male, Young Mustache Pale Skin	

■ UnitGAN

圖 5.3 Result of UnitGAN

圖 5.4 Result of UnitGAN

- **Analysis (1%)**

  Which model you use and your observation

  - CycleGAN

    同時 Train 四個 model，包括 G,、F、Dx、Dy，G 和 F 是同個架構，都是有八層 Convolution Layer 加上八層 De-convolution Layer，整張照片從第一層的 64 個 Filter 到第八層的 512 個 Filter 最後再回到 64 個 Filter，Dx 和 Dy 也是同一種架構，都有五層 Convolution Layer，最後一層的 Unit 數是 1。

    整體的 loss 是用 LSGAN 來定義，在 Train Generator 時，除了使 Discriminator 的分數越高外，重建回來的一致性也要越高，在 Train Discriminator 時，要使 real data 和 fake data 區分的越好，最後我們 Train 在兩組不同的 dataset 上：monet2photo、winter2summer，總共 Train 了 200 個 Epoch，batch_size 設 4。

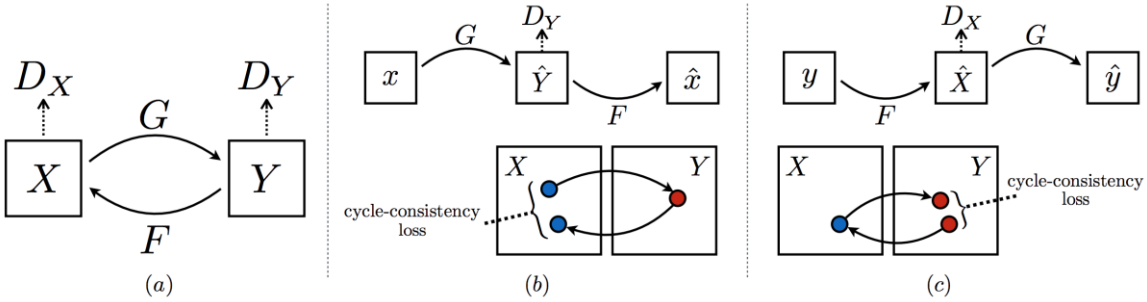


圖 5.5 Structure of CycleGAN

  - StarGAN

    整體使用 WGAN 來定義 loss。Discriminator 是一個七層的 CNN，主要在區分 real image 和 fake image，並且能知道目前在區分的影像是屬於哪一個 Domain。Generator 是九層 Convolution Layer 和三層 De- Convolution Layer 組成，除了使 Discriminator 評分較高之外，更要使重建的 error 小。上面所示，與 CycleGAN 最大的差別在於 StarGAN 會將一串 Binary label 和影像一起送進 Generator，這也是造成他不需要很多個 Generator 就能達到多 Domain 之間的轉換。

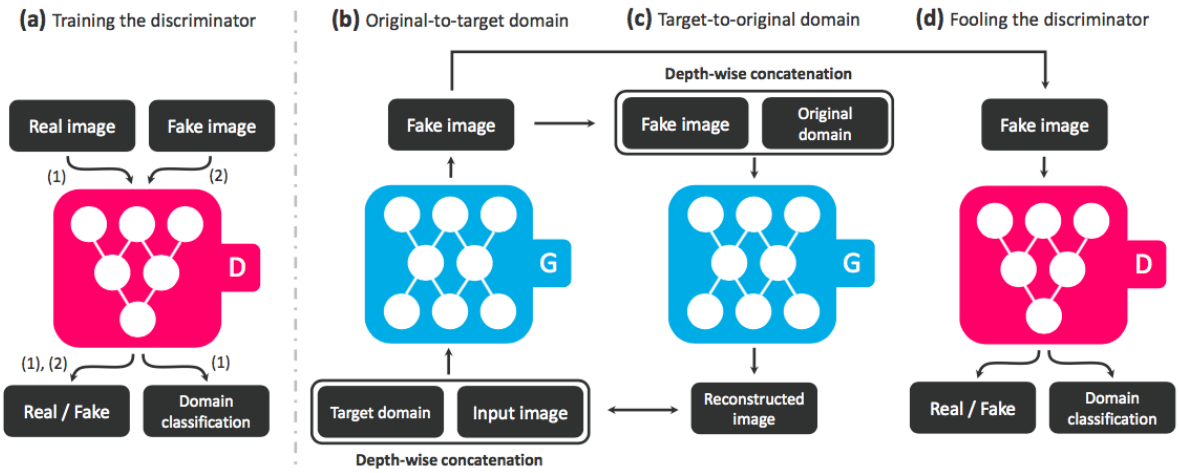


圖 5.6 Structure of StarGAN

■ UnitGAN

與 CycleGAN 主要的差別在於 UnitGAN 會先將不同 Domain 的圖片轉到同一種 Latent space，所以能夠轉換的 Domain 差距可以比較大，不過礙於時間限制，我們 CycleGAN 並沒有跑太久，所以結果沒有很好。

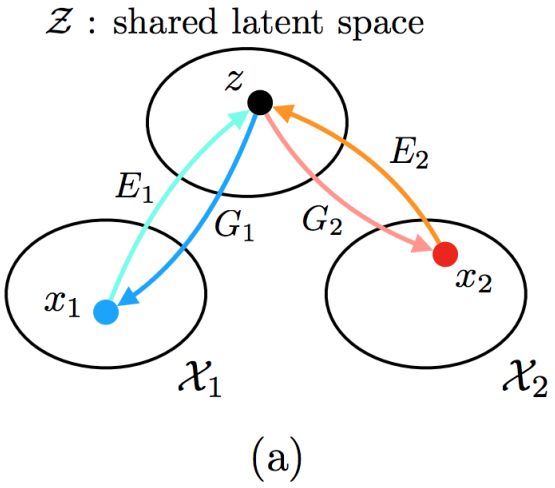


圖 5.7 Structure of UnitGAN