HW3 Report

分工表（0.5%）

組員	分工
B04901040 電機三 林哲賢	HW3-1、報告撰寫
B04901117 電機三 毛弘仁	HW3-3、報告撰寫
B04901118 電機三 王克安	HW3-2、報告撰寫

Model Description

Describe the models you used, including the model architecture and objective function for G and D

Image Generation (2%)

Model 是使用 DCGAN，每一個iteration裡generator和discriminator各更新一遍。

Generator:

1. Input 為 100 維0到1之間的noise向量
2. 中間的架構如下圖，而在output的地方會通過hyperbolic tangent使得值在-1到1之間
3. Objective function為binary cross entropy loss(BCE)，random產生出來的圖經過discriminator後得到的分數當成real data去update參數。
4. optimizor: Adam, lr = 0.0002, beta = 0.5
5. Batch size為64

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         819,200
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
         LeakyReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
         LeakyReLU-6            [-1, 256, 8, 8]               0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524,288
       BatchNorm2d-8          [-1, 128, 16, 16]             256
         LeakyReLU-9          [-1, 128, 16, 16]               0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131,072
      BatchNorm2d-11           [-1, 64, 32, 32]             128
        LeakyReLU-12           [-1, 64, 32, 32]               0
  ConvTranspose2d-13            [-1, 3, 64, 64]           3,072
================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
----------------------------------------------------------------
```

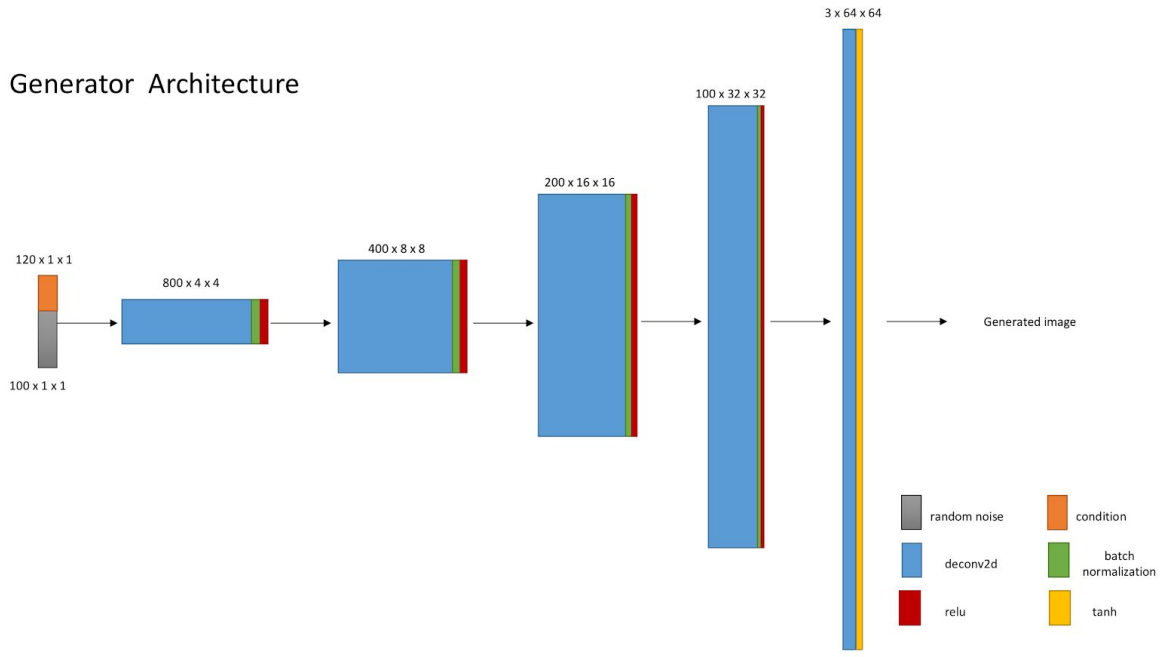
Discriminator:

1. Input 為 (3, 64, 64), 值在-1到1之間的向量
2. 中間的架構如下圖，而output會通過sigmoid function使得值在0到1之間(original GAN)
3. Objective function為binary cross entropy loss(BCE)，random產生出來的圖為fake data(0), 而training data的圖當做real data(1)
4. optimizor: Adam, lr = 0.0002, beta = 0.5

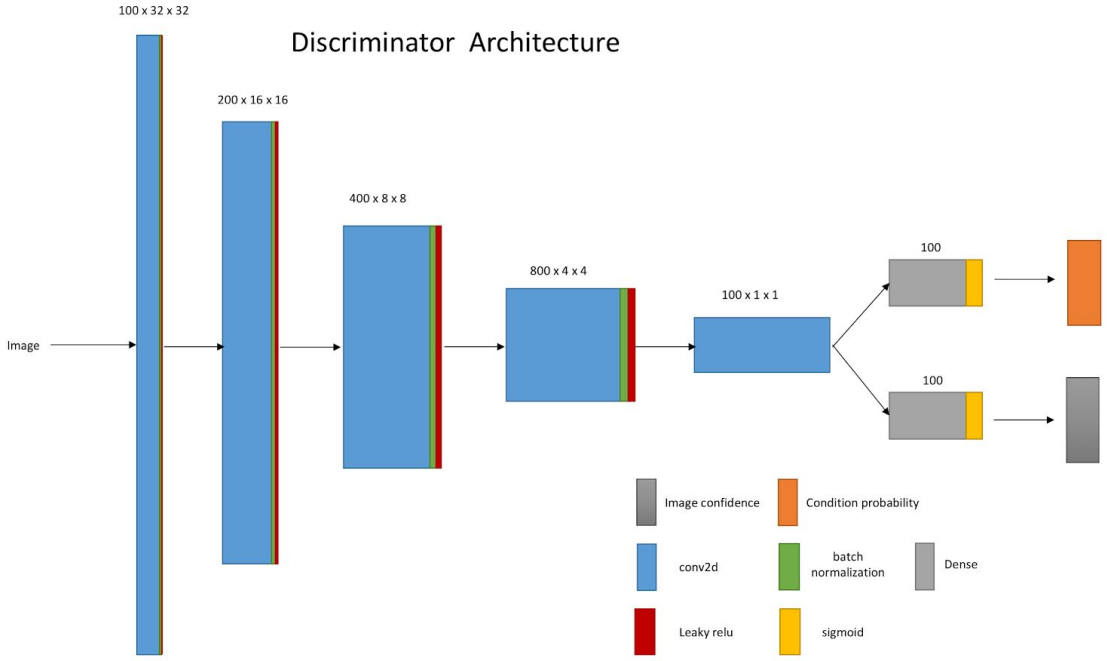
Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	3,072
LeakyReLU-2	[-1, 64, 32, 32]	0
Conv2d-3	[-1, 128, 16, 16]	131,072
BatchNorm2d-4	[-1, 128, 16, 16]	256
LeakyReLU-5	[-1, 128, 16, 16]	0
Conv2d-6	[-1, 256, 8, 8]	524,288
BatchNorm2d-7	[-1, 256, 8, 8]	512
LeakyReLU-8	[-1, 256, 8, 8]	0
Conv2d-9	[-1, 512, 4, 4]	2,097,152
BatchNorm2d-10	[-1, 512, 4, 4]	1,024
LeakyReLU-11	[-1, 512, 4, 4]	0
Conv2d-12	[-1, 1, 1, 1]	8,192

Total params: 2,765,568
Trainable params: 2,765,568
Non-trainable params: 0

<u>Text-to-image Generation (2%)</u>
Generator input: random noise, condition vector
Generator output: image



Discriminator input: image
Discriminator output: confidence of image, probability distribution of condition vector



Both optimizers for G and D are Adam with beta1 = 0.5
Both objective functions are BCE.
Random noise dimension = 100, and trained for 140 epochs

Experiment settings and observation (show generated images)

Image Generation (1%)

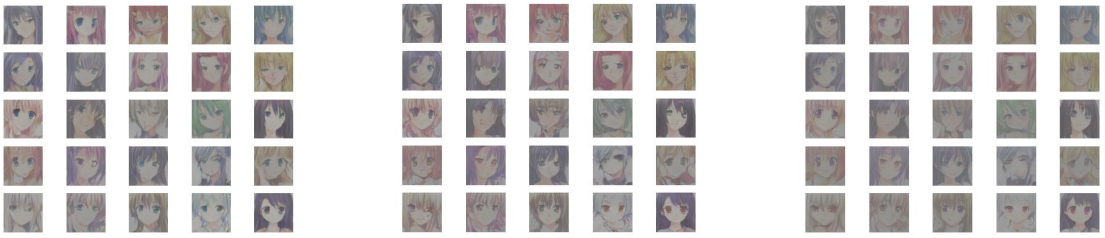
Result:

Observation:

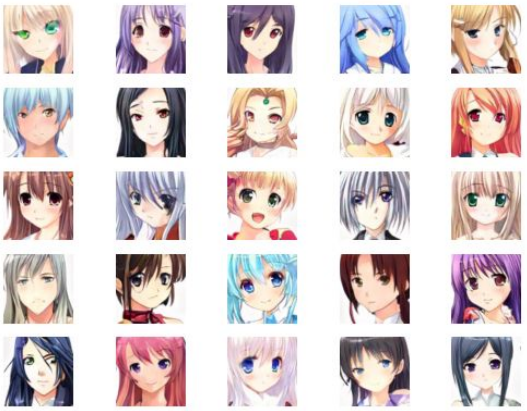
在training的過程中我們發現一件事，那就是圖的彩度會一直下降，直到上面那張結果，如下面三張圖為10、15以及第20個epoch的結果(大概是6000，9000，12000個iteration)

這個結果並非mode collapse，而且網路上並沒有找到類似的說法，和老師討論過後得到的結論是discriminator沒有學到彩度的資訊，而造成這個原因的主要原由可能是因為batch normalization，於是我就試著把discriminator的batch normalization拔掉train看看結果

因為收斂速度比較慢，所以採用了第20、25以及30個epoch的結果，發現彩度下降得更嚴重了，而且從第0個epoch開始彩度就相當的低，而且直到mode collapse彩度都沒有上來，這個結果令我還蠻納悶的，因此我再試著把generator的batch normalization拔掉。

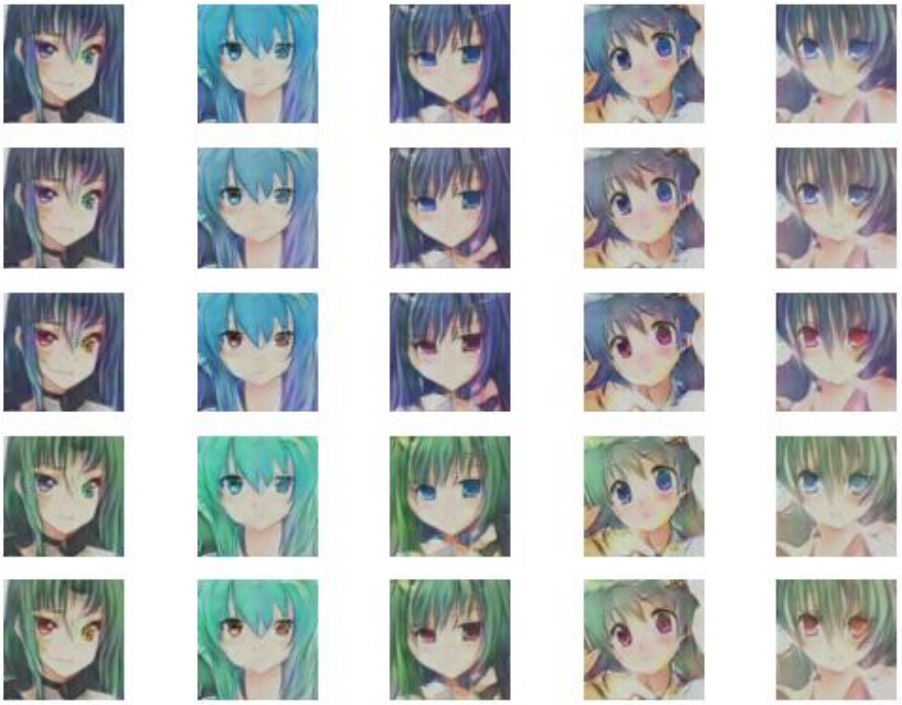


結果就是彩度變得更低了，迄今還沒有找到原因，理論上並非程式問題，因為我用同樣的方式print出training data，如下圖，發現彩度是沒有問題的。



Text-to-image Generation (1%)

Result:



Observation:
生成的圖片基本上可以分成頭髮跟眼睛的顏色，但是在blue eyes跟green eyes的task中會稍微沒辦法分辨，推測是因為藍色跟綠色太像了，所以model生成上會比較接近。

Compare your model with WGAN, WGAN-GP or LSGAN (Image Generation Only)

Description of the chosen model (1%)

我所採用的model是WGAN，基本上的架構都和先前使用的 DCGAN 相同，只差在以下幾點:

1. Discriminator的最後一層為線性output，要把original GAN最後一層的sigmoid拔掉。
2. Loss function 修改，原本是用binary cross entropy，但是在WGAN則是直接取mean，在updata D時，real 的data取的是負號而generator生成的data在取mean之後要取正號，在update G 時，生成的data是要去騙過discriminator所以取mean之後要取負號。
3. 要把 D 的 weight clip住，而我這裡取的值是(-0.01, 0.01)
4. 用RMSprop來更新D，而G則是沿用Adam。
RMSprop : lr = 0.0002
5. 每個iteration裡D update 5 次，而G 只update 1 次。

其他的部分都和第一題一樣，詳見第一題的模型架構。

Result of the model (1%)

第30個epoch出來的結果最好(如下圖)，經過辨識model後可以認出24張臉。
第100個epoch都還沒有mode collapse。

Comparison Analysis (1%)

1. 和原本的 DCGAN相比，結果差了一些，有蠻多張圖片都糊糊的，如上圖。
2. converge的速度很快，收斂的epoch數比原本的 DCGAN少一些，但是因為discriminator要更新五次所以total training time慢了差不多2.5倍。
3. 從圖上很明顯得可以看出來彩度並沒有下降，因此可以推測彩度下降的理由可能是因為discriminator output的sigmoid使得D學不到辨識彩度這件事。

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

Which tip & implement details(3%)

Tip1: Normalize the inputs

我把real的圖片從0~255的數值經過-127.5再除以127.5轉到-1和1之間來當做discriminator 的input，而另一方面，我把generator的output加上一層hyperbolic tangent使得output在-1和1之間，再餵進discriminator。

比較的方面我把tanh拔掉，並把batch normalization拔掉，然後直接用0~255丟進去train看看結果。

Tip5: Avoid Sparse Gradients: ReLU, MaxPool

可以參考report最上面的架構圖，我從頭到尾都是使用leaky relu(0.2)而且沒有做pooling，比較的方面我把leaky relu(0.2)換成ReLU試試看結果會如何。

Tip9: Use the ADAM Optimizer

使用Adam當做optimizor來train，G和D都用，而比較的對象我使用了常見的SGD以及RMSprop來做。

在使用的時候的lr設定如下:

Adam : lr = 0.0002, beta = 0.5

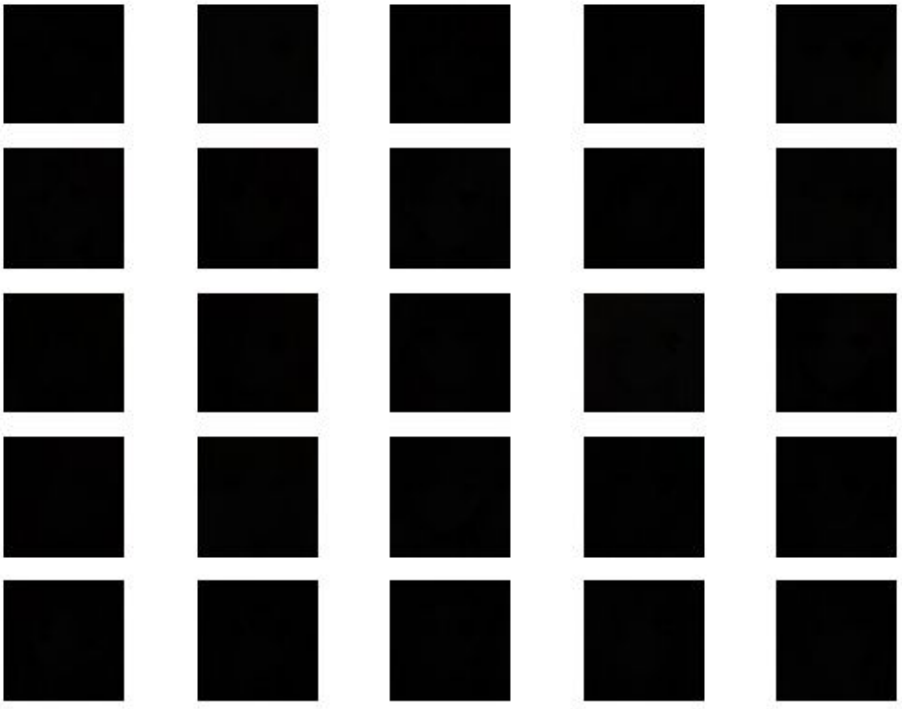
SGD : lr = 0.002

RMSprop : lr = 0.0005

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

Tip1: Normalize the inputs

我發現即使把input data換成0~255之間並把generator的tanh拿掉，generator出來的值還是在1以下，原因可能是有batch normalization的關係，因此我把batch normalization拔掉再train，得到以下結果



把值拿出來會發現集中在3~5附近，然後在第14個epoch時壞掉了(loss炸掉)，推測可能的原因是這是original GAN，所以discriminator的output會掛上sigmoid，造成了gradient vanishing，沒辦法處理數值那麼大(0~255)的情形，因此必須要normalize才能train。

Tip5: Avoid Sparse Gradients: ReLU, MaxPool

在training的過程中會發現一些事情:
1. 如果去看它的gradNorm，會發現比起leaky relu，ReLU在training時的gradient norm會大了大概1.5倍(G和D都是)
2. Train 到第50個epoch左右就會爛掉，意思是loss突然就卡住不動了(並非 mode collapse)，gradient也變成0，非常的unstable，如果是leaky relu的話到第500個epoch都不會爛掉(雖然會mode collapse)，這個原因應該是因為ReLU在負數的地方gradient是0，因此會變得很不穩定。
3. 在train到一定程度後，破圖情況比較嚴重，如下圖，這應該並非是random結果造成的問題，我ReLU和leakyReLU都各生成了10*25張，挑選最差和最好的25張都會發現使用ReLU比leakyReLU破圖的情況要嚴重

4. 彩度依然會下降，但是下降的比leaky ReLU慢一些。
5. 順帶一提，如果用selu的話比ReLU的效果還糟。

Tip9: Use the ADAM Optimizer

1. G和D都使用SGD，下圖是train了30個epoch的結果，可以發現圖片相較於使用ADAM當optimizor會模糊很多，可能的原因是因為是stochastic更新，比較難學到generate清晰的圖像，如果把G換成Adam而D仍維持SGD，會發現performance還不錯，雖然要train得比較久，但是圖片並沒有比較模糊，因此可以得知下圖模糊的原因是因為G而非D的關係。
P.S. train到35個epoch就mode collapse了，比起adam 到了200個epoch才mode collapse而言快了許多。

2. G和D都使用RMSprop，會發現非常容易mode collapse，第20個epoch就發生了，比SGD還要快，以下是第15個epoch的結果，結果也是很模糊，所以得到的結論是optimizor還是要選Adam最好。

HW3-3 Bonus

Show two domains and their transfer results (1%)

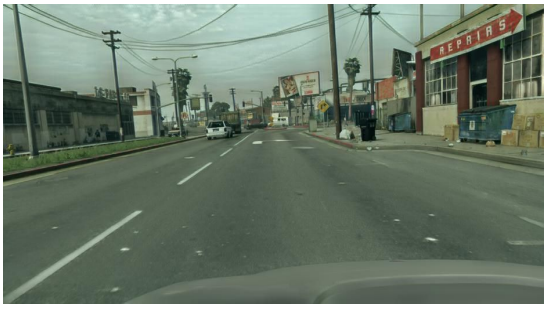
Pretrained 好的 model，是用「真實城市街景」以及「電玩城市街景」這兩個 domain 來互相轉換，成果如下：

Input: 真實場景

Output: 電玩場景

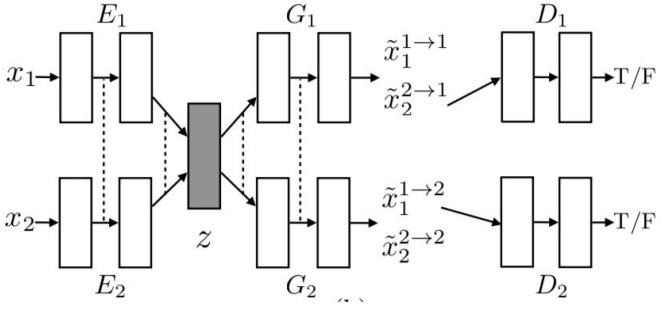
Input: 電玩場景

Output: 真實場景



Which model you used and your observations (1%)

我們使用的是 UNIT model，它拿 VAE 當作 GAN 的 generator，並且讓兩個 domain 各有自己的 encoder、decoder、discriminator。使用的 constraints 有 weight-sharing（encoder with encoder, decoder with decoder）、cycle consistency 以及 same-domain reconstruction loss。Shared latent space 的各個維度假設是 conditionally independent and Gaussian with unit variance，在這樣的架構下，encoder 會 output 出 mean vector（細節可看 paper）。

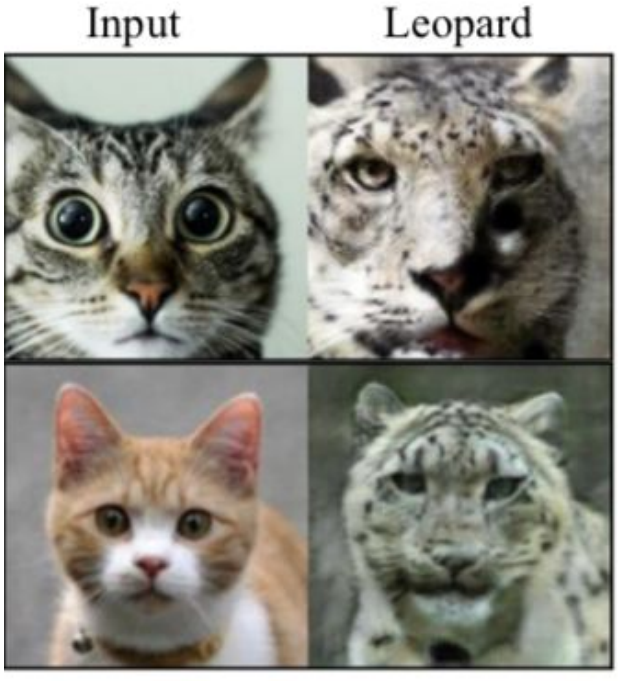


我們試著使用 Google Maps 抓下來的街景來 transfer 成電玩場景，結果如下：



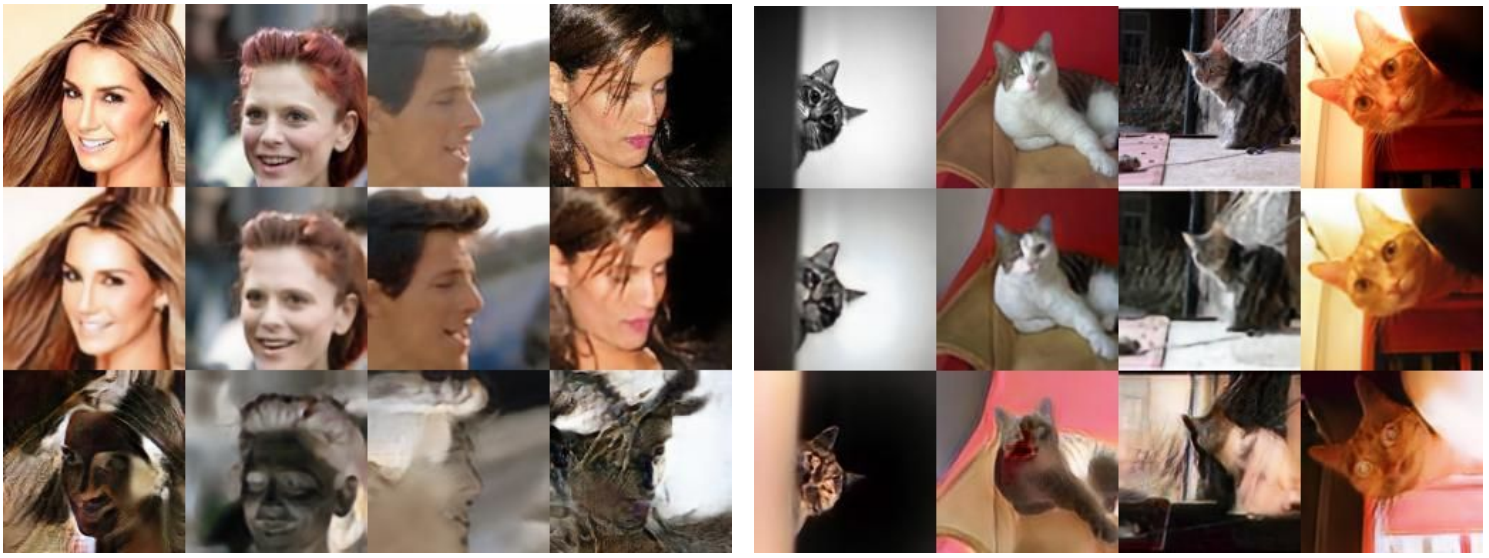
左邊的圖都是「真實場景」，右邊的圖都是生成出來的「電玩場景」。其實好像沒有什麼大變化，主要就是亮度、對比變高，以及一些比較細的 texture 變得更平滑，這和 paper 作者提供的範例圖非常相似。

以這個 pre-trained model 來說，我們認為達到的效果感覺跟 CycleGAN 很像，比較沒有看到老師上課所提到的 “larger change, only keep the semantics” 現象。不過 UNIT 的 paper 裡面，確實是有做到比較大幅度的 transfer，像是「貓」轉「豹」，如下：

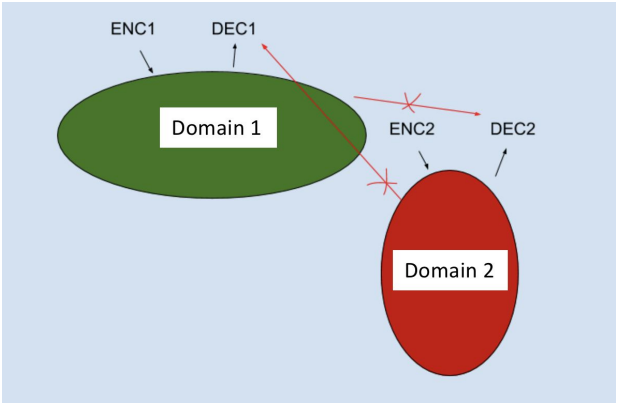


看到這樣的成果，我們想挑戰拿兩個差異較大的 domain 來互相轉換，決定自己訓練「人貓互轉」的 UNIT model，train 了 24 小時（67000 個 iterations）過後的成果如下：

上：原圖　　中：原 domain 的 reconstruction　　下：domain transfer



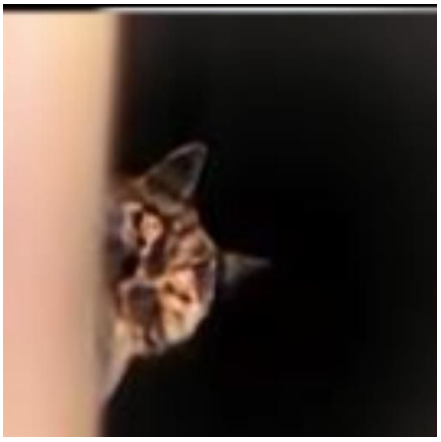
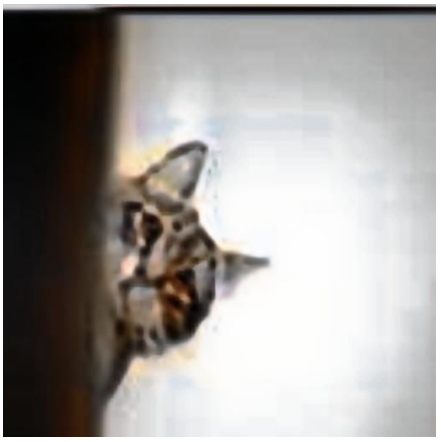
可以發現，transfer 成果是沒有像「貓」轉「豹」那麼 impressive 的，但是原 domain 的 reconstruction 看起來不錯的，代表 latent space encoding 確實有儲存到豐富的圖像資訊。我們推測，即便將兩個 domain 放到同一個 latent space，transfer 能力還是有限，尤其當 domain 性質差異很大的時候。如下圖所示，儘管有了 discriminator 和 cycle consistency 來拉近兩個 domain 的 encoding，它們在同個 latent space 上面可能還是兩個分開的 distribution，這會使得 domain B 的 decoder 不適合用在 domain A 的 encoding 上面。



再仔細一想，cycle consistency constraint 會鼓勵這件事：轉 B domain 時，保留足夠多 A domain的特色。這似乎會和 B 的 discriminator loss 有衝突，model 必須在兩個 loss 中間取個平衡點，如果在 A、B domain 相差較大時，很難兩者同時滿足。

經由實驗，我們發現 cycle consistency 感覺被滿足得還可以（如下圖），達到這個效果所付出的代價，可能就是犧牲一些掉欺騙 discriminator 的能力。

貓 (input) → 人 (transfer) → 貓 (cycle reconstruction)

人 (input) → 貓 (transfer) → 人 (cycle reconstruction)