Machine Learning - Homework 11

資工四 B05902023 李澤諺

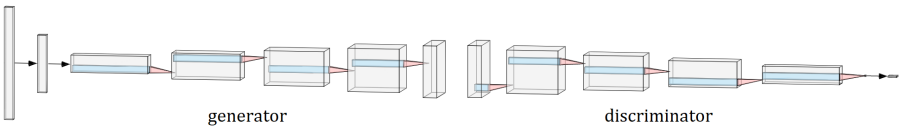
June 11, 2020

1. (2.5%) 訓練一個 model。

(1) (1%) 請描述你使用的 model (可以是 baseline model)，包含 generator 和 discriminator 的 model architecture、loss function、optimizer 參數、以及訓練 step 數 (或是 epoch 數)。

(2) (1.5%) 請畫出至少 16 張 model 生成的圖片。

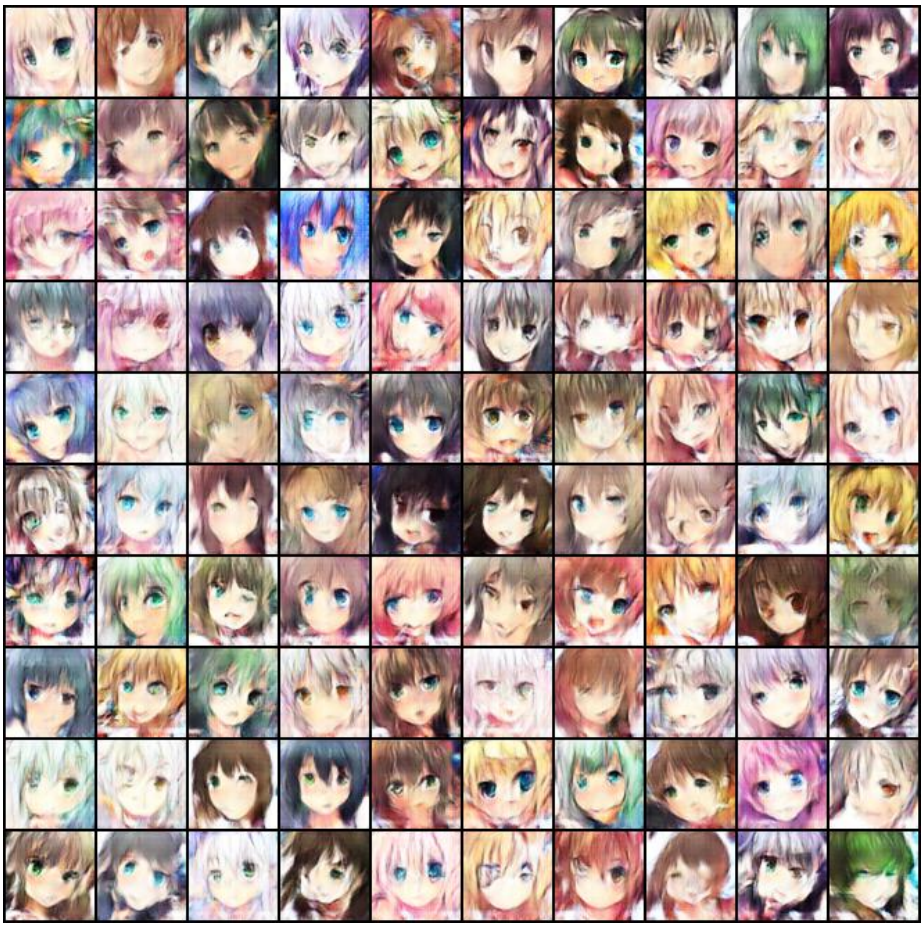
以下爲我於本次作業中使用 PyTorch 所實作的 DCGAN 架構：



DCGAN	
generator	Linear(100 , 8192 , bias = False)
	BatchNorm1d(8192)
	ReLU()
	ConvTranspose2d(512 , 256 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(256)
	ReLU()
	ConvTranspose2d(256 , 128 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(128)
	ReLU()
	ConvTranspose2d(128 , 64 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(64)
	ReLU()
	ConvTranspose2d(64 , 3 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1)
	Tanh()
discriminator	Conv2d(3 , 64 , kernel_size = 5 , stride = 2 , padding = 2)
	LeakyReLU(0.2)

discriminator	Conv2d(64 , 128 , kernel_size = 5 , stride = 2 , padding = 2)
	BatchNorm2d(128)
	LeakyReLU(0.2)
	Conv2d(128 , 256 , kernel_size = 5 , stride = 2 , padding = 2)
	BatchNorm2d(256)
	LeakyReLU(0.2)
	Conv2d(256 , 512 , kernel_size = 5 , stride = 2 , padding = 2)
	BatchNorm2d(512)
	LeakyReLU(0.2)
	Conv2d(512 , 1 , kernel_size = 4)
	Sigmoid()

其中，我將所有的 convolutional layer 和 transposed convolutional layer 的 weight 以 mean 爲 0 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，並將所有的 batch normalization layer 的 weight 以 mean 爲 1 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，而 batch normalization layer 的 bias 則皆 initialize 爲 0。接著，我將 training dataset 中所有的圖片皆 resize 爲 64×64，再將其 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。最後，我使用了 Adam 作爲 optimizer，並使用 binary cross entropy 作爲 loss function，以此訓練 DCGAN，其中 batch size 爲 64，learning rate 爲 0.0001，β_1 和 β_2 分別爲 0.5 和 0.999，訓練了 10 個 epoch，以此得到最後的 model。我使用訓練好的 DCGAN 生成了 100 張圖片，如下圖所示：



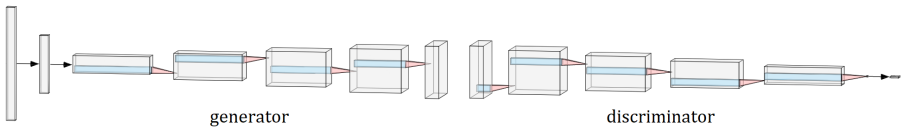
2. (3.5%) 請選擇下列其中一種 model：WGAN、WGAN-GP、LSGAN、SNGAN (不要和 1. 使用的 model 一樣，至少 architecture 或是 loss function 要不同)。

(1) (1%) 同 1.a，請描述你使用的 model，包含 generator 和 discriminator 的 model architecture、loss function、optimizer 參數、以及訓練 step 數 (或是 epoch 數)。

(2) (1.5%) 和 1.b 一樣，就你選擇的 model，畫出至少 16 張 model 生成的圖片。

(3) (1%) 請簡單探討你在 1. 使用的 model 和 2. 使用的 model，他們分別有何性質，描述你觀察到的異同。

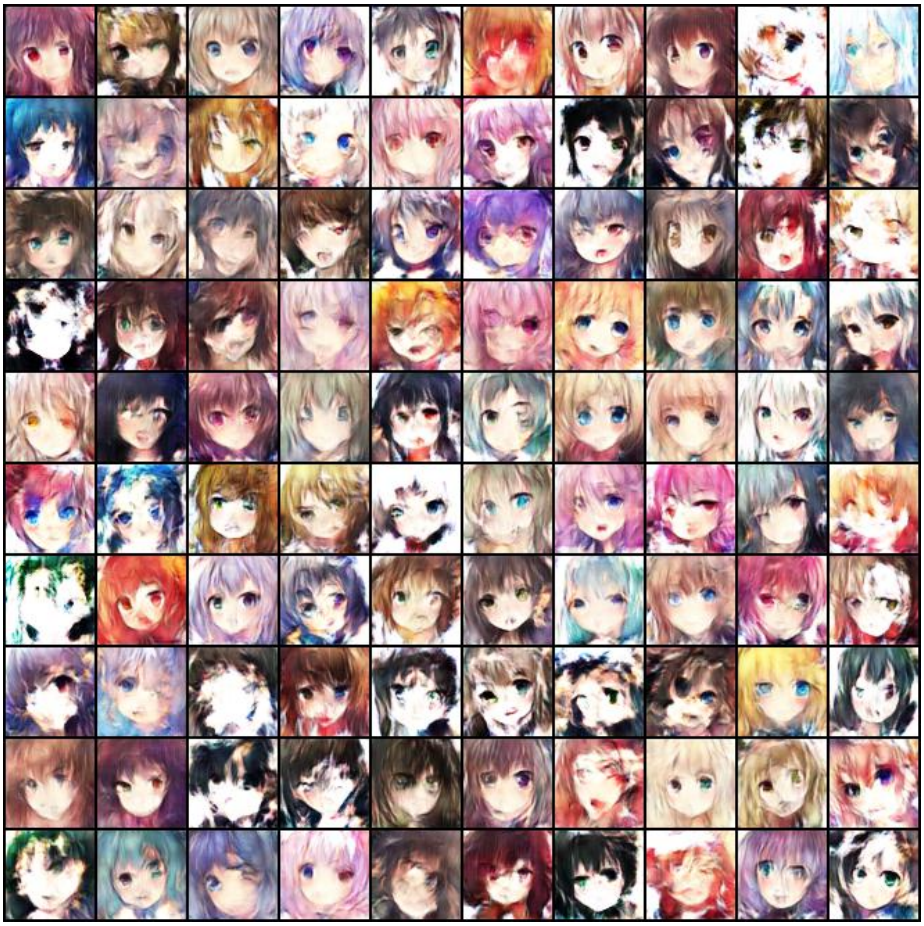
我於本次作業中使用 PyTorch 實作了WGAN、WGAN-GP、LSGAN、SNGAN。首先，以下爲我所實作的 WGAN 架構：



WGAN	
generator	Linear(100 , 8192 , bias = False)
	BatchNorm1d(8192)
	ReLU()
	ConvTranspose2d(512 , 256 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(256)
	ReLU()
	ConvTranspose2d(256 , 128 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(128)
	ReLU()
	ConvTranspose2d(128 , 64 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(64)
	ReLU()
	ConvTranspose2d(64 , 3 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1)
	Tanh()
discriminator	Conv2d(3 , 64 , kernel_size = 5 , stride = 2 , padding = 2)
	LeakyReLU(0.2)
	Conv2d(64 , 128 , kernel_size = 5 , stride = 2 , padding = 2)
	InstanceNorm2d(128)
	LeakyReLU(0.2)
	Conv2d(128 , 256 , kernel_size = 5 , stride = 2 , padding = 2)
	InstanceNorm2d(256)
	LeakyReLU(0.2)
	Conv2d(256 , 512 , kernel_size = 5 , stride = 2 , padding = 2)
	InstanceNorm2d(512)
	LeakyReLU(0.2)
	Conv2d(512 , 1 , kernel_size = 4)

其中，我將所有的 convolutional layer 和 transposed convolutional layer 的 weight 以 mean 爲 0 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，並將所有的 batch normalization layer 的 weight 以 mean 爲 1 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，而 batch normalization layer 的 bias 則皆 initialize 爲 0。接著，我將 training dataset 中

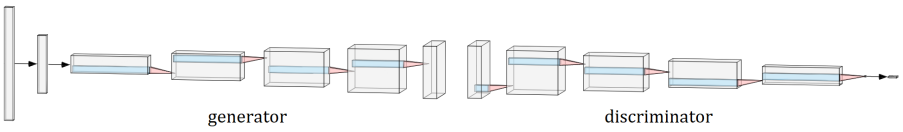
所有的圖片皆 resize 爲 64×64，再將其 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。最後，我使用了 RMSprop 作爲 optimizer，並使用 Wasserstein distance 作爲 loss function，以此訓練 WGAN，其中 batch size 爲 64，learning rate 爲 0.0001，訓練了 50 個 epoch ，並且，在每次訓練完 discriminator 之後，會將 discriminator 中所有的 weight 皆 clip 到 -0.01 和 0.01 之間，而 generator 則會每隔 5 個 batch 才訓練一次，以此得到最後的 model。我使用訓練好的 WGAN 生成了 100 張圖片，如下圖所示：



接著，在 WGAN-GP 之中，其 model architecture 和 initialization 與 WGAN 相同，在 data preprocessing 上，亦是將圖片 resize 爲 64×64 再將 pixel 的值 normalize 到 -1 和 1 之間進行相同的處理，而在訓練上，我使用了 Adam 作爲 optimizer，並使用 Wasserstein distance 作爲 loss function，而在訓練 discriminator 時 loss 還會再加上 gradient penalty，以此訓練 WGAN-GP，其中 batch size 爲 64，learning rate 爲 0.0001，β_1 和 β_2 分別爲 0.5 和 0.999，訓練了 50 個 epoch ，此外，generator 會每隔 5 個 batch 才訓練一次，以此得到最後的 model。我使用訓練好的 WGAN-GP 生成了 100 張圖片，如下圖所示：

接著，在 LSGAN 之中，除了 loss function 換爲 mean square error，並將 epoch 數量提高到 20 次以外，其它的 data preprocessing、model architecture、initialization、optimizer、hyper-parameter 等等皆和第 1 題所述的 DCGAN 相同。我使用訓練好的 LSGAN 生成了 100 張圖片，如下圖所示：

最後，以下爲我實作的 SNGAN 架構：



SNGAN	
generator	Linear(100 , 8192 , bias = False)
	BatchNorm1d(8192)
	ReLU()
	ConvTranspose2d(512 , 256 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(256)
	ReLU()
	ConvTranspose2d(256 , 128 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(128)
	ReLU()
	ConvTranspose2d(128 , 64 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1 , bias = False)
	BatchNorm2d(64)
	ReLU()
	ConvTranspose2d(64 , 3 , kernel_size = 5 , stride = 2 , padding = 2 , output_padding = 1)
	Tanh()
discriminator	spectral_norm(Conv2d(3 , 64 , kernel_size = 5 , stride = 2 , padding = 2))
	LeakyReLU(0.2)
	spectral_norm(Conv2d(64 , 128 , kernel_size = 5 , stride = 2 , padding = 2))
	LeakyReLU(0.2)
	spectral_norm(Conv2d(128 , 256 , kernel_size = 5 , stride = 2 , padding = 2))
	LeakyReLU(0.2)
	spectral_norm(Conv2d(256 , 512 , kernel_size = 5 , stride = 2 , padding = 2))
	LeakyReLU(0.2)
	spectral_norm(Conv2d(512 , 1 , kernel_size = 4))
	Sigmoid()

其中，我將所有的 convolutional layer 和 transposed convolutional layer 的 weight 以 mean 爲 0 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，並將所有的 batch normalization layer 的 weight 以 mean 爲 1 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，而 batch normalization layer 的 bias 則皆 initialize 爲 0。接著，我將 training dataset 中所有的圖片皆 resize 爲 64×64，再將其 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。最後，我使用了 Adam 作爲 optimizer，並使用 binary cross entropy 作爲 loss function，以此訓練 SNGAN，其中 batch size 爲 64，learning rate 爲 0.0001，β_1 和 β_2 分別爲 0.5 和 0.999，訓練了 20 個 epoch，以此得到最後的 model。我使用訓練好的 SNGAN 生成了 100 張圖片，如下圖所示：

在以上所有 model 的訓練過程中，我在每個 epoch 中都會使用目前所得到的 model 生成 100 張圖片，以檢視訓練過程 (由於圖片數量極多，無法呈現在 report 之中，因此我將其放在命名爲 image 的資料夾裡以供檢視)，由這些圖片可以看出，WGAN 和 WGAN-GP 需要較多的 epoch 來進行訓練，其可能是因爲在 WGAN 和 WGAN-GP 的訓練過程中，generator 每過數個 batch 才會被訓練一次，其訓練頻率較低所導致，而 WGAN-GP 因爲要計算 gradient penalty 的關係，其每一個 epoch 所需要的時間比其它 model 的還要來得更多，接著，在 DCGAN、LSGAN、SNGAN 的訓練過程中，DCGAN 沒辦法訓練太多個 epoch，否則會發生 mode collapse (詳見第 3 題)，而 LSGAN 在訓練過程的初期看起來收斂較慢，此時所生成的圖片看起來仍爲雜訊，但之後仍能生成質量較高的圖片，而 SNGAN 的訓練過程則相較之下較爲平穩，最後，以我的主觀判斷來看，大部分的 model 所生成的圖片其質量都差不多，僅有 WGAN 所生成的圖片質量很差，其可能是因爲強行將 discriminator 的 weight 進行 clip 所導致的不良後果，或可能是因爲所使用的 hyper-parameter 不佳所導致。

3. (4%) 請訓練一個會導致 mode collapse 的 model。

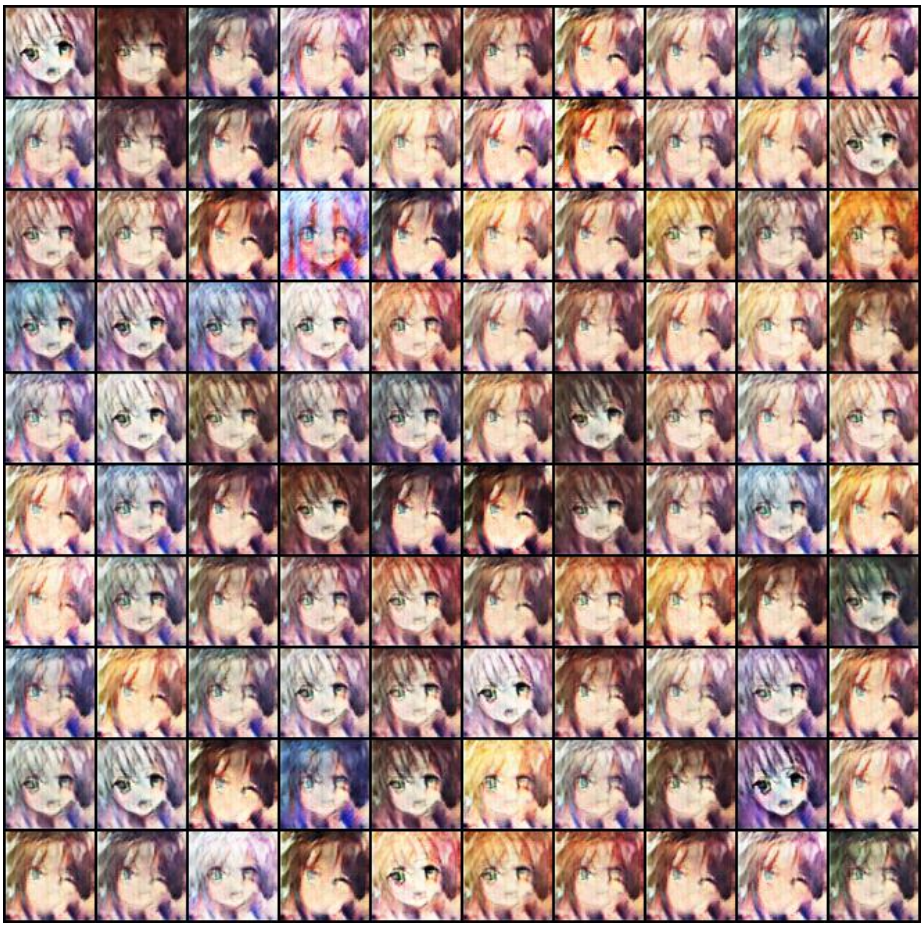
(1) (1%) 同 1.a，請描述你使用的 model，包含 generator 和 discriminator 的 model architecture、loss function、optimizer 參數、以及訓練 step 數 (或是 epoch 數)。

(2) (1.5%) 請畫出至少 16 張 model 生成且具有 mode collapse 現象的圖片。

(3) (1.5%) 在不改變 optimizer 和訓練 step 數的情況下，請嘗試使用一些方法來減緩 mode collapse。說明你嘗試了哪些方法，請至少舉出一種成功改善的方法，若有其它失敗的方法也可以記錄下來。

我將第 1 題之中所述的 DCGAN 其 epoch 數量提高到 20 次，而其它的 data preprocessing、model architecture、initialization、loss function、optimizer、hyperparameter 等等皆不變，以此進行訓練，結果在訓練到第 14 個 epoch 時便發生了嚴重的 mode collapse，如下圖所示：

以下爲在 20 個 epoch 全部訓練完之後，DCGAN 所生成的 100 張圖片，仍然可以從中看出 mode collapse 的現象：



由第 2 題可以看出，在同樣訓練 20 個 epoch 的情況下，使用 LSGAN (將 loss function 換爲 mean square error) 或是 SNGAN (在 discriminator 的每個 module 上都使用 spectral normalization) 可以改善 mode collapse 現象。