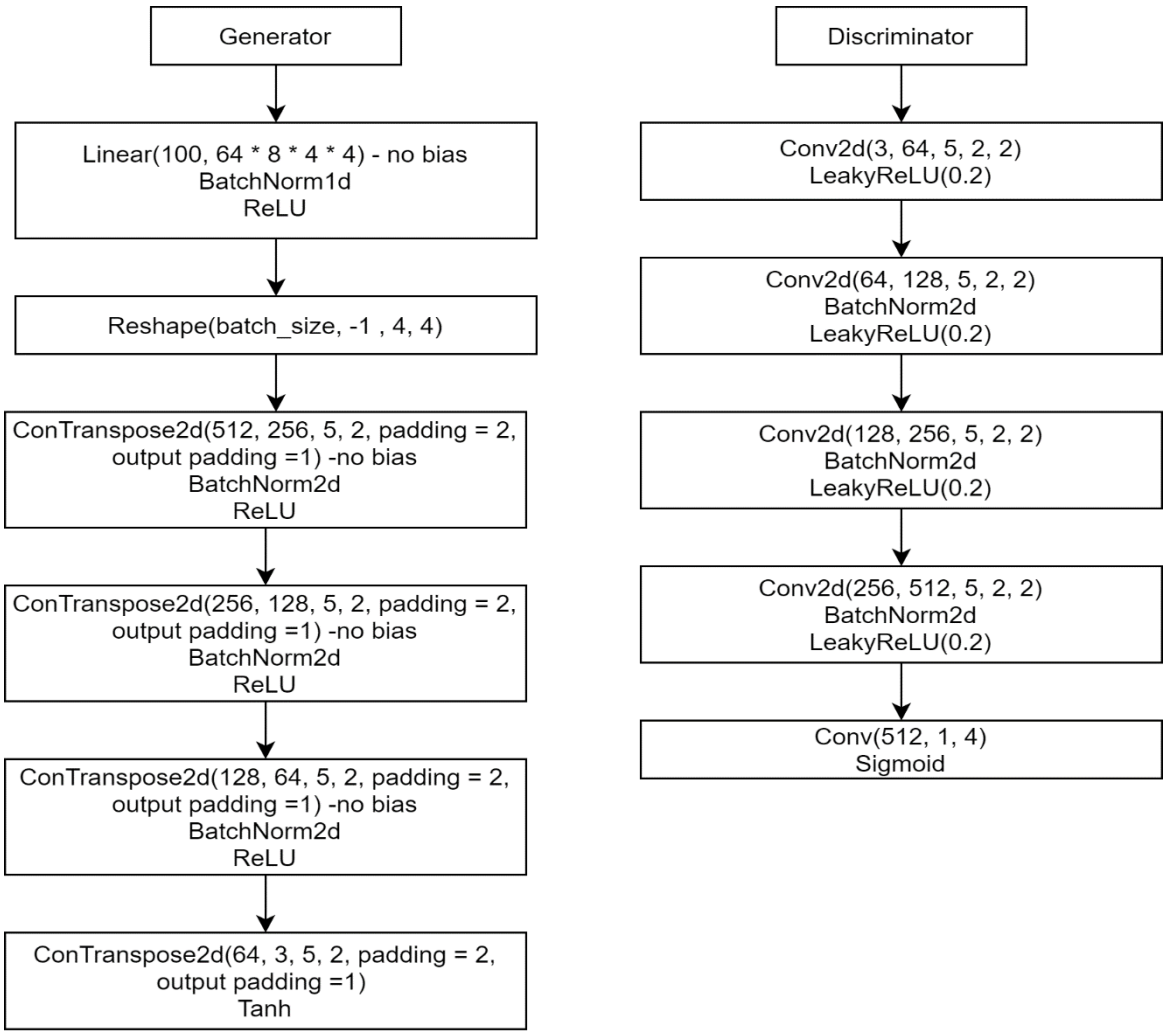
學號：R08946006　　系級：資料科學學程碩一　姓名：　周逸平

1. (2.5%) 訓練一個 model。
   a. (1%) 請描述你使用的 model（可以是 baseline model）。包含 generator 和 discriminator 的 model architecture、loss function、使用的 dataset、optimizer 參數、以及訓練 step 數（或是 epoch 數）。
   b. (1.5%) 請畫出至少 16 張 model 生成的圖片。

此處使用助教 tutorial 中相同的參數以及模型。如下所示：

Batch size	128
Learning Rate	1e-4
Optimizer	Adam(beta = (0.5,0.999))
Epochs	20 (558 iters/epoch)
Criterion	BCELoss
Dataset	Faces (作業給的)



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）

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

此處我使用 WGAN 作為嘗試，而由於 WGAN 是基於 loss 算法做更動的，所以我沿用 Q1 中 DCGAN 的 architecture，參見 Q1 圖。另做以下四點修改

1. 移除 Discriminator 中最後的 sigmoid (只有此處更動 architecture)
2. 不對 Discriminator 以及 Generator 的 Loss 取 log
3. 對每次更新 Discriminator 後之參數做 clip
4. 不使用 momentum-based 的 optimizer，此處採用 RMSprop

故參數調整如下：

Batch size	128
Learning Rate	1e-4
Optimizer	**RMSprop**
Epochs	20 (558 iters / epoch)
Criterion	**None**
Dataset	Faces (作業給的)
Discriminator Clip	**[-0.01, 0.01]**

更改處以灰底顯示。而計算 loss 的方法改為以下 code：

Discriminator	
DCGAN (tutorial)	**WGAN**
r_loss = criterion(r_logit, r_label) f_loss = criterion(f_logit, f_label) loss_D = (r_loss + f_loss) / 2 D.zero_grad() loss_D.backward() opt_D.step()	one = torch.FloatTensor([1]).cuda() mone = torch.FloatTensor([-1]).cuda() D.zero_grad() r_loss = D(r_imgs.detach()) f_loss = D(f_imgs.detach()) r_loss = r_loss.mean(0).view(1) r_loss.backward(one) f_loss = f_loss.mean(0).view(1) f_loss.backward(mone) opt_D.step()

Generator	
DCGAN (tutorial	**WGAN**
f_logit = D(f_imgs) loss_G = criterion(f_logit, r_label) G.zero_grad() loss_G.backward() opt_G.step()	one = torch.FloatTensor([1]).cuda() G.zero_grad() g_loss = D(f_imgs) g_loss = g_loss.mean().mean(0).view(1) g_loss.backward(one) opt_G.step()

WGAN 使用了 Wasserstein distance 作為更新的手段，而非傳統的 BCELoss，而利用這個特性他解決了傳統 GAN 常見的幾個問題。
以下 Discriminator 簡稱 D 而 Generator 簡稱 G。
1. GAN 訓練不穩定，D 與 G 的訓練程度不一，造成其一方之 loss 過小，使得該方得不到正確的訓練方向，得到好的結果通常是「運氣好」。而此情形大多發生在 D 過優，G 的梯度消失嚴重導致 G 完全沒有成長。
2. 傳統 GAN 帶有 mode collapse 的問題，使其喪失樣本多樣性，且通常是無法辨認之圖像。
3. 傳統 GAN 之 G 以及 D 的 Loss 不帶有指標性，無法透過 loss 來得知當前訓練狀況是否是好的，或者模型是否還有繼續在學習。

而透過 WGAN 以上幾點獲得改善
1. Wasserstein 的算法徹底解決了兩方訓練不一致的問題。
2. 避免了 mode collapse 的問題。
3. 訓練中有個指標可以觀察當前模型學習情況。
且以上修正不需要依賴龐大的模型架構，簡單的全連接網路就可以達成。

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

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

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

c. (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。

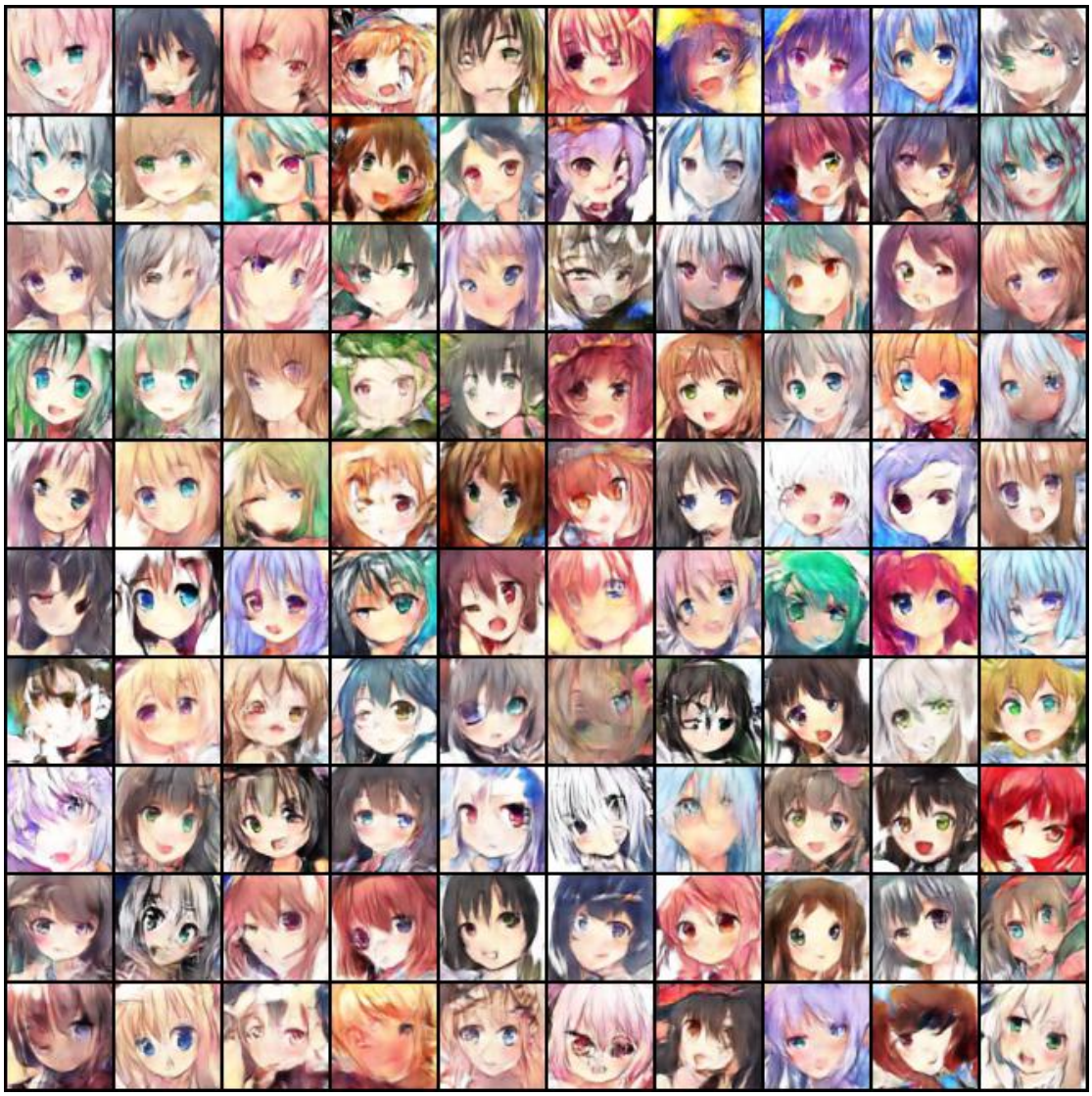
此處直接沿用 Q1 的配置，觀察訓練至第 epoch 32 時，直接發生 mode collapse 的情況。

此時 G 的 Loss 直接歸 0，如同 Q2 中 C 小題的敘述，G 的梯度消失。

此時我們改用 WGAN 來改進這個問題，其中雖然小題希望不要更改 optimizer，

但由於若是使用 Adam 等基於動量的 optimizer，根據作者所述其 loss 有可能會爆炸，理由作者推估是由於 GAN 本身的 loss 是不穩定的，其參數空間並非固定使得基於動量的算法容易給出與當前空間梯度南轅北轍的更新方向，所以我們應該要使用非基於動量的算法，此處採用 RMSprop
訓練至同樣 32epoch 圖如下：

為了確保真的不會發生 mode collapse，我持續訓練到了 epoch 150



可以發現完全沒有 mode collapse 問題