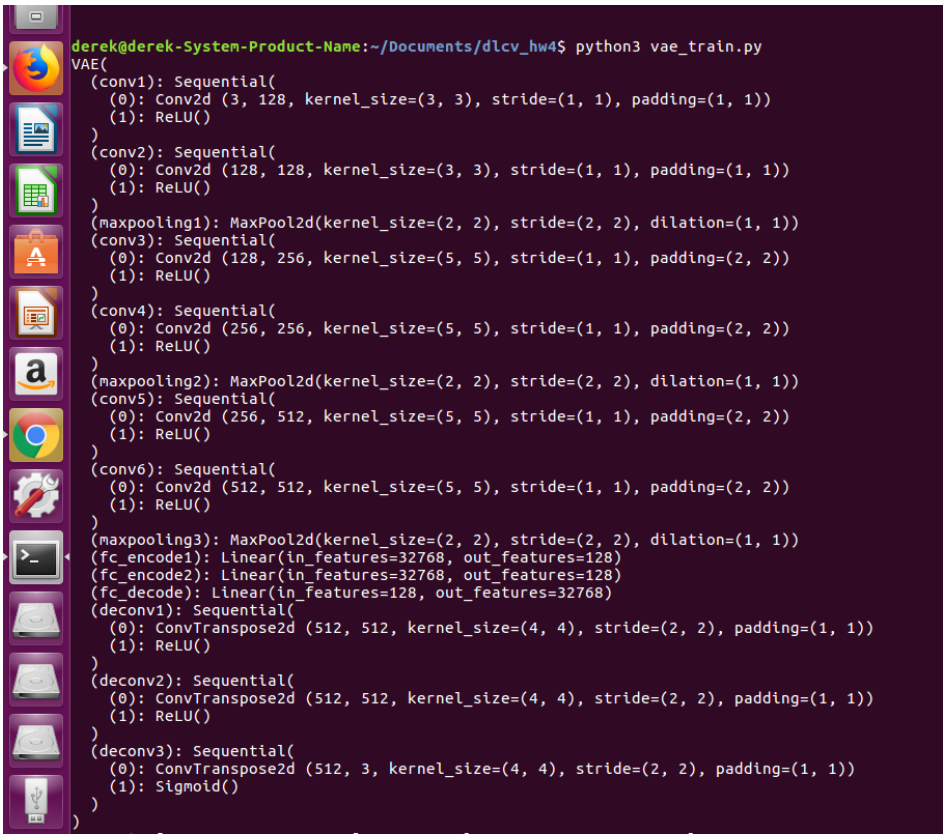
電機四 B03901161 楊耀程

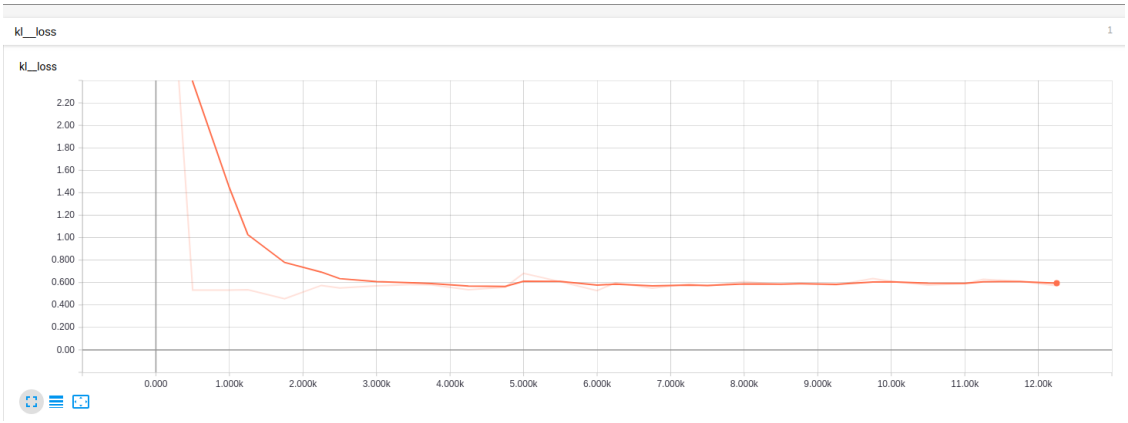
VAE

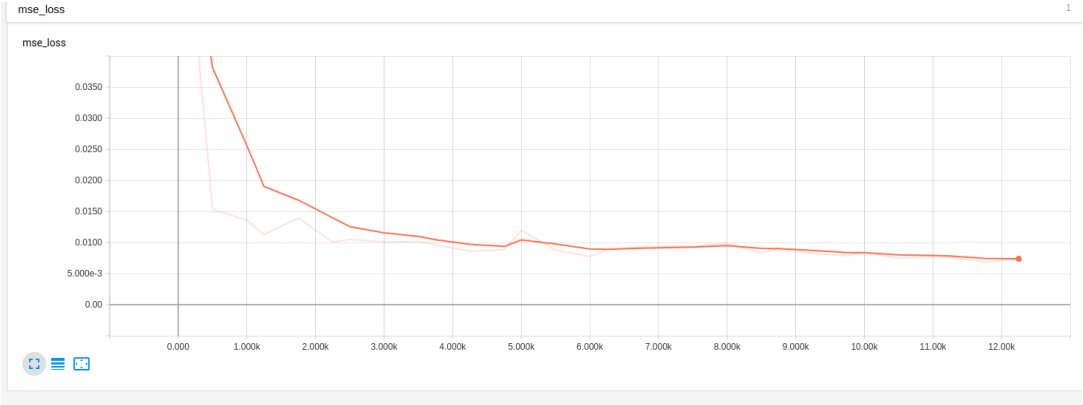
1. Model architecture:

   Latent dim: 128，loss = 5e-3*kl_loss + mse，optimizer = adamax, lr = 5e-4



2. Learning curve: (y 軸: loss，x 軸: number of steps)





Explain: 可以看到 mse 的 loss 以及 kl loss 都穩定的下降。

3. 10 pairs of testing images: (上: 原圖，下:VAE 重建的圖)

Total MSE on testing data : 0.0076

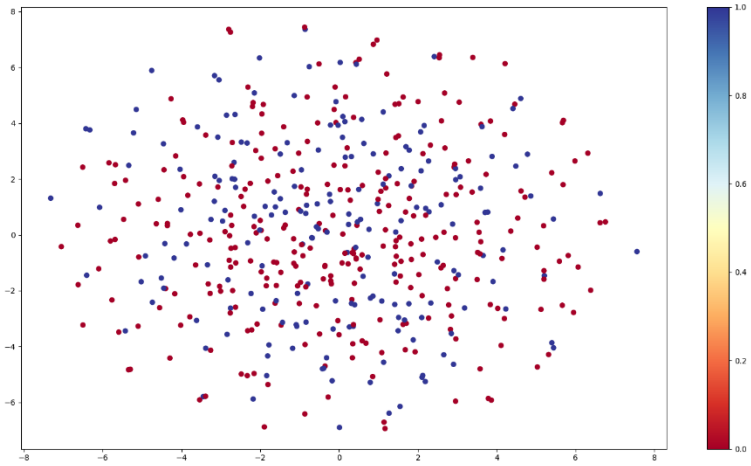
```
derek@derek-System-Product-Name:~/Documents/dlcv_hw4$ python3 vae_test.py
nubmer of testing image 2621
type of image out:  <class 'torch.cuda.FloatTensor'>
image saved
====> Test set loss: 0.0076
```

4. Random generate 32 images:

5. Visualization of latent space using TSNE

   Attribute: “Male”

   Red label: female, blue label: male



Explain: 男生跟女生的點並沒有明顯的分開，不過 VAE 不像 ACGAN，沒針對某一特定的 attribute 去做 disentangle，因此經過 VAE encoder 再經 TSNE 降維的圖片沒有把男生跟女生明顯分開，也還算合理的結果。

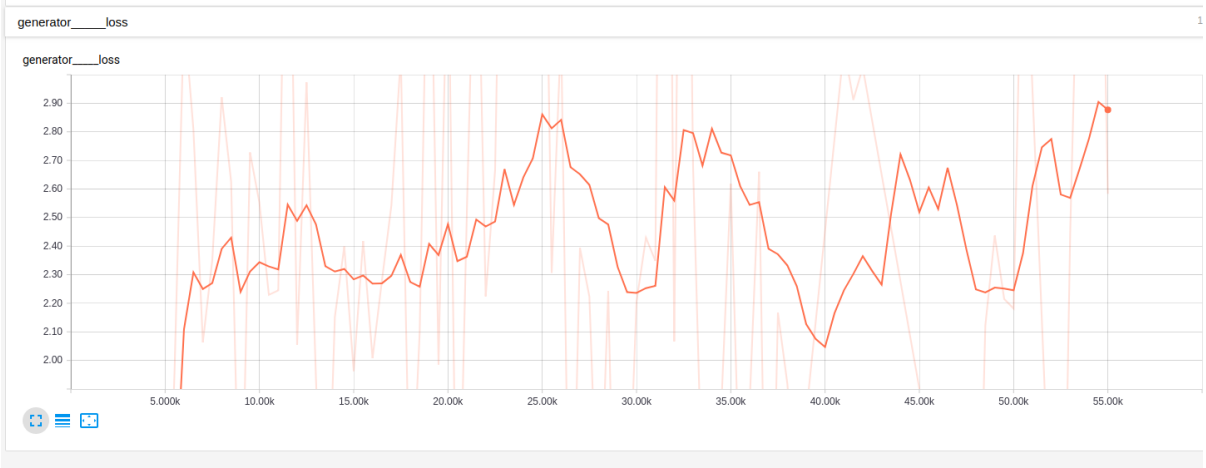
6. 討論從實作 VAE 觀察到的事:
   首先是 kl_loss 跟 mse 的比例，loss = lambda*kl_loss + mse，如果 lambda 太大，mse 就比較難降下來，重建的效果就不太好，但如果 lambda 太小，重建的 mse 非常小，但是 random sample 時就非常不像人臉，應該是 kl loss 沒降下來，造成 latent space 的分布非常不像 normal distribution。除了 lambda 之外，optimizer 跟 learning rate 也要挑選得當，才會使 kl loss, mse loss 都穩定下降到一定程度，這次我的經驗上來說 adamax 效果較好，learning rate 也不能太大。

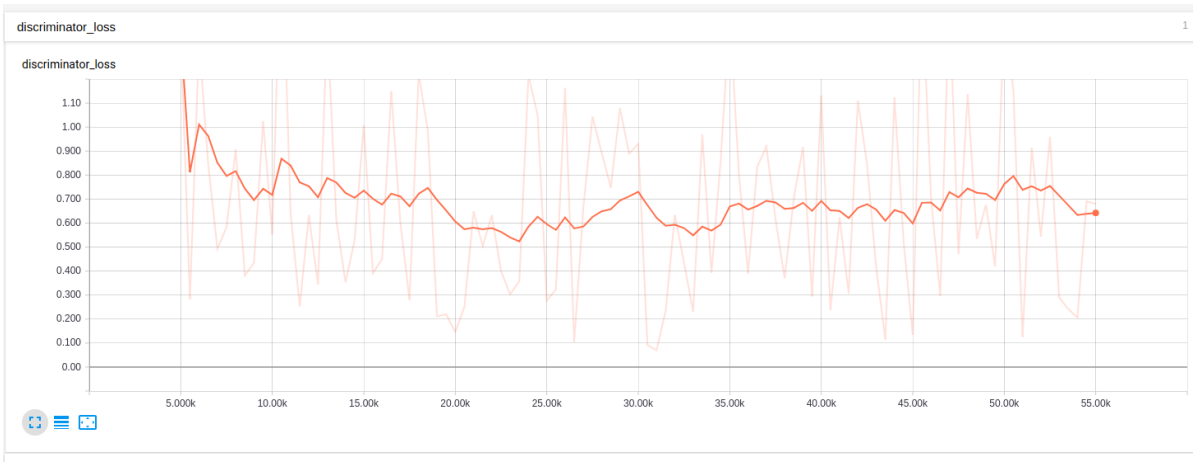
GAN

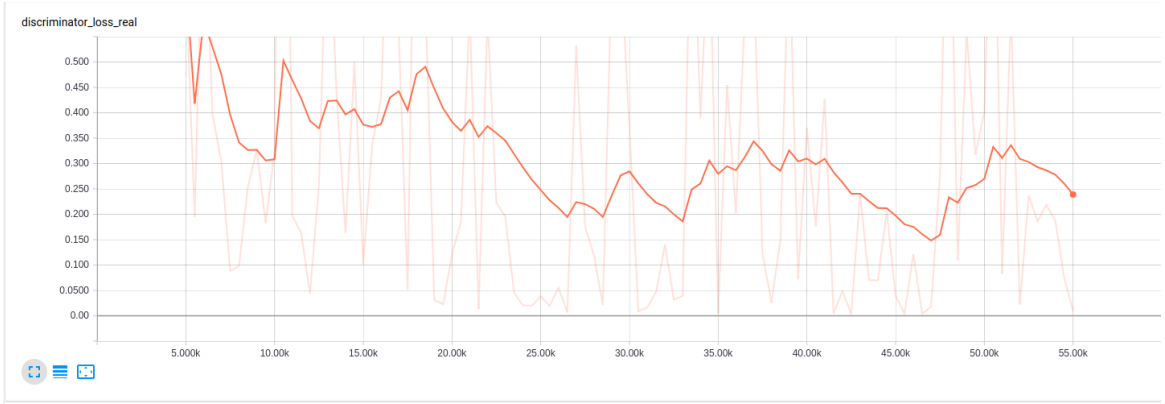
1. Model architecture:
   Latent dim: 128，optimizer = adam, lr = 5e-4

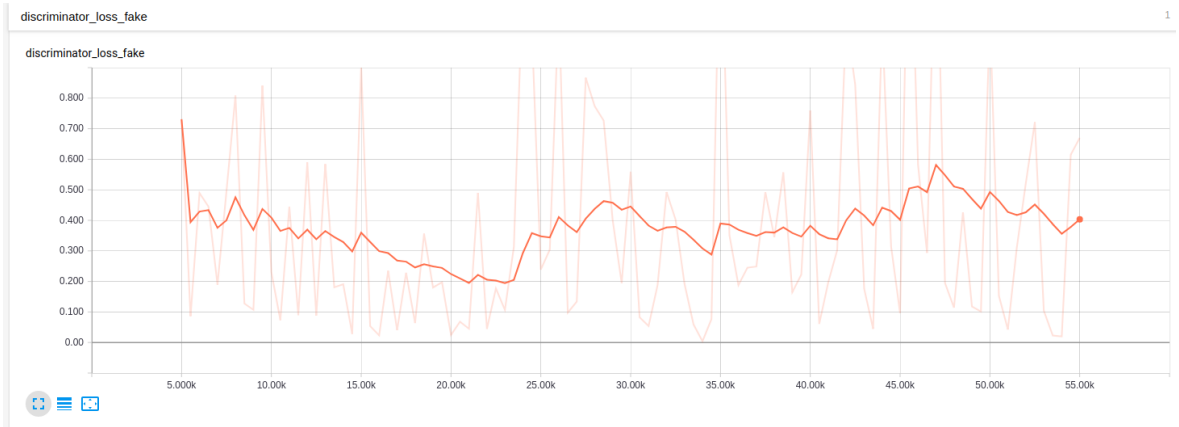
```
derek@derek-System-Product-Name:~/Documents/dlcv_hw4$ python3 dcgan_train.py
generator(
  (deconv1): Sequential(
    (0): ConvTranspose2d (128, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (deconv2): Sequential(
    (0): ConvTranspose2d (512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (deconv3): Sequential(
    (0): ConvTranspose2d (256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (deconv4): Sequential(
    (0): ConvTranspose2d (128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (deconv5): Sequential(
    (0): ConvTranspose2d (64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): Tanh()
  )
)
======================
discriminator(
  (conv1): Sequential(
    (0): Conv2d (3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (conv2): Sequential(
    (0): Conv2d (64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (conv3): Sequential(
    (0): Conv2d (128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (conv4): Sequential(
    (0): Conv2d (256, 512, kernel_size=(4, 4), stride=(4, 4), padding=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
  )
  (conv5): Sequential(
    (0): Conv2d (512, 1, kernel_size=(4, 4), stride=(4, 4), padding=(1, 1), bias=False)
    (1): Sigmoid()
  )
)
```

2. Learning curve: (y 軸: loss，x 軸: number of steps)









Explain:

首先看 generator loss (簡稱 G loss)以及 discriminator loss (D loss)，本圖片有 smooth 過，可以看到即使 smooth 過，兩個 loss 的值都還是變動很大，但是沒有一方變特別高，只是到了後期(約 55k step 後) D 似乎還是比 G 更強，G loss 漸漸變高不少，失去了平衡，因此我就沒繼續 train 下去。

至於 D loss real 和 D loss fake 在這邊也分別列出，可以看到兩者也沒有變特別低或特別高，算是沒有壞掉的狀態。

3. Random generate 32 images:

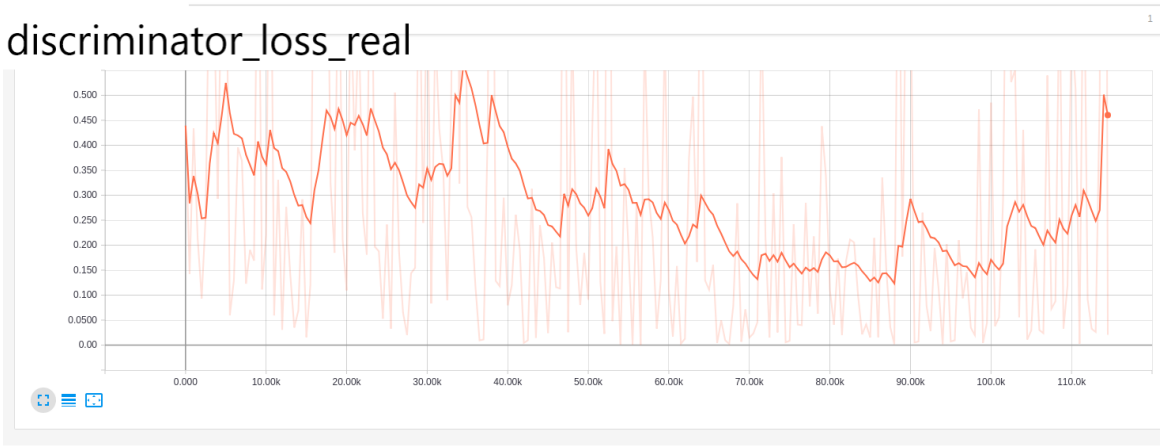
4. 討論從實作 GAN 觀察到的事:
   我實作的是 DCGAN，原本在 VAE 還有 fc，但是我用那樣的 generator(之前 VAE 的 decoder)效果很不好，直到改成 fully convolution 才長出品質比較好的照片。此外 G loss 與 D loss 的平衡非常困難，除了層數跟參數量要設計得當之外，learning rate 也要控制好，才不會使 generator 的 loss 很高，而 discriminator loss 降很低。
5. 比較與 VAE 的不同
   以 random sample 的結果來看，我認為 GAN 比較有機會長出像真實拍攝的照片，GAN 長的照片的邊緣銳利很多，VAE 有一種模糊類似柔焦的感覺，有 discriminator 的架構在真的有很大幫助。不過也因為 G,D 兩種相互競爭的關係，GAN 比 VAE 難 train 很多，VAE 只要兩個 loss 都有下降就好。但是 GAN 就很容易 train 壞，因為 loss 跳動很大，不能有單方一直下降，要小心有 discriminator 的強度壓過 generator 的情形發生，因此需要耐心調整參數與觀察 loss 的變化。

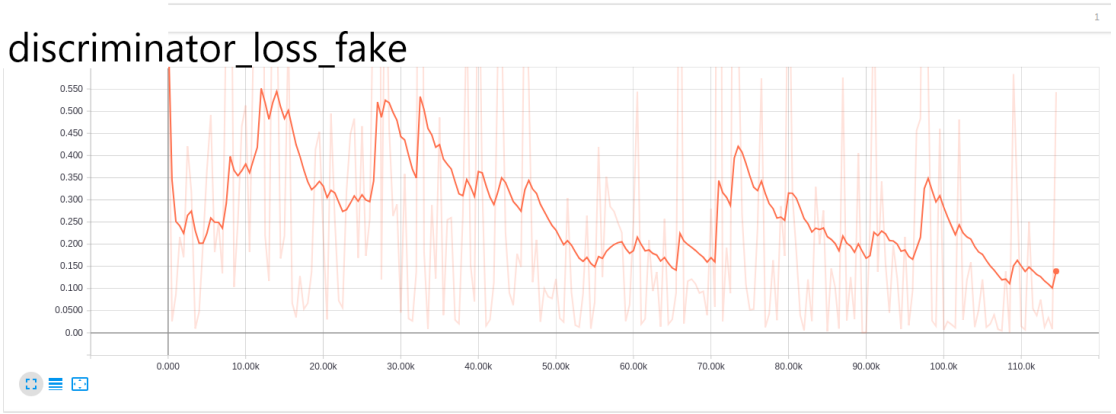
ACGAN

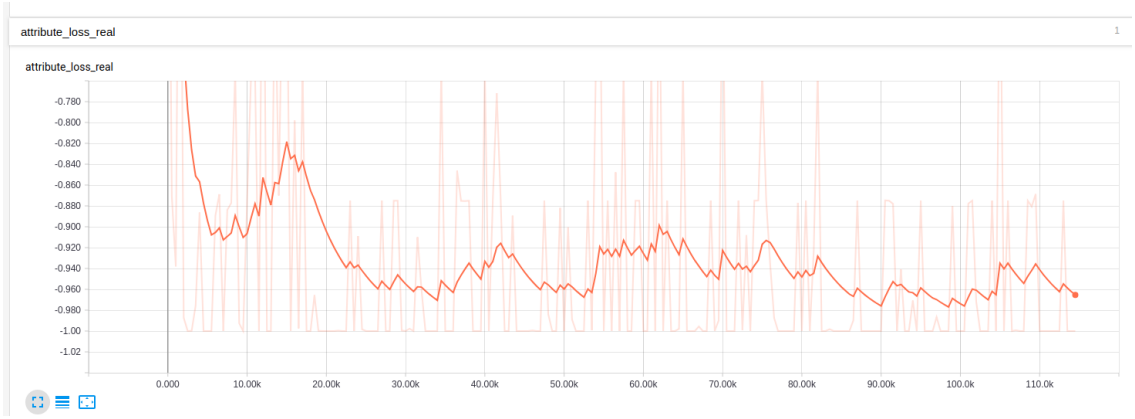
1. Model architecture:
   Latent dim: 100，optimizer = adam, lr = 0.0002

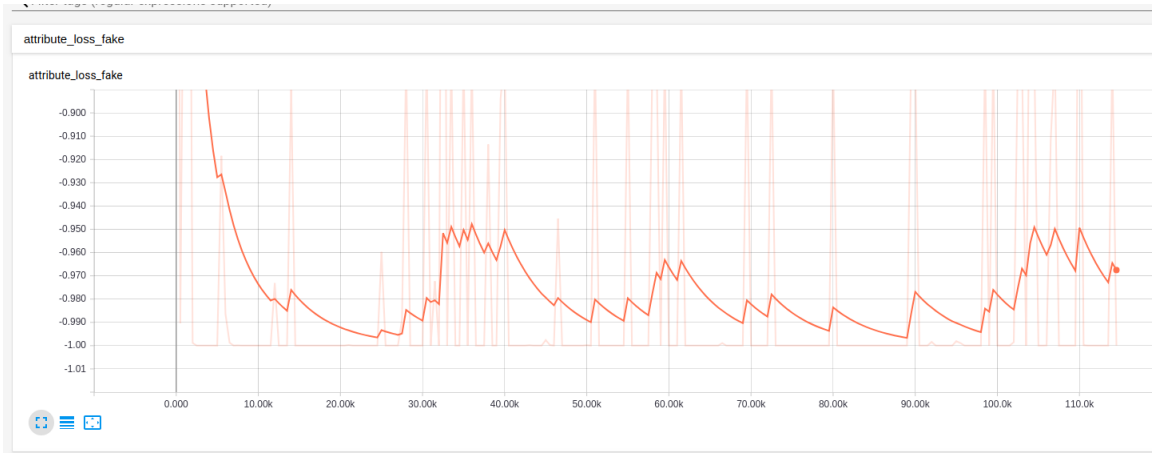
```
derek@derek-System-Product-Name: ~/Documents/dlcv_hw4
derek@derek-System-Product-Name:~/Documents/dlcv_hw4$ python3 acgan_train.py
Namespace(batchSize=8, beta1=0.5, dataroot='hw4_data', imageSize=64, lr=0.0002, manualSeed=5414, ndf=64
/usr/local/lib/python3.6/dist-packages/torchvision/transforms/transforms.py:188: UserWarning: The use
  "please use transforms.Resize instead.")
netG(
  (ReLU): ReLU(inplace)
  (Tanh): Tanh()
  (conv1): ConvTranspose2d (100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
  (BatchNorm1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (conv2): ConvTranspose2d (512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (conv3): ConvTranspose2d (256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (conv4): ConvTranspose2d (128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
  (conv5): ConvTranspose2d (64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
)
==========================
netD(
  (LeakyReLU): LeakyReLU(0.2, inplace)
  (conv1): Conv2d (3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (conv2): Conv2d (64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (conv3): Conv2d (128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (conv4): Conv2d (256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (BatchNorm4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (conv5): Conv2d (512, 64, kernel_size=(4, 4), stride=(1, 1), bias=False)
  (disc_linear): Linear(in_features=64, out_features=1)
  (aux_linear): Linear(in_features=64, out_features=2)
  (softmax): Softmax()
  (sigmoid): Sigmoid()
)
```

2. Learning curve: (y 軸: loss，x 軸: number of steps)









Explain:

此處列了四種 loss 的圖，都是 discriminator 端的，前兩個是判斷真假圖片的 loss，後兩個是判斷 classification (是否為選定的 attribute)的 loss。雖然四個 loss 都跳動很大，但我觀察到 attribute loss 下降較 discriminator loss 快。首先是 discriminator 的 loss 本來就不應該下降太快，應該是持續跳動的，否則就會造成 discriminator 壓過 generator 的情況(DCGAN 那邊有提到了)。而 attribute loss 下降較快的原因，我推測可能是二元分類的問題比較好學，generator 學到 latent dim 中 attribute 那個維度的意義(長出的圖片都有包含此 attribute 的特性)，而且 discriminator 的 layer 也學到如何 classify 此 attribute 後，attribute 的 loss 就能下降了。

3. 10 pairs of random generated images:使用性別 attribute (上: Female，下: Male)