# DLCV HW4 Report

**Name:** 黃宇平　**Dep.:**電信碩一　**Student ID:**R06942065

## Problem 1: VAE

1. Describe the architecture & implementation details of your model(1%)

我的Encoder及Decoder架構分別如下：

Encoder:

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| vae_in (InputLayer) | (None, 64, 64, 3) | 0 | |
| vae_enc_conv_0 (Conv2D) | (None, 32, 32, 64) | 4864 | vae_in[0][0] |
| batch_normalization_1 (BatchNor | (None, 32, 32, 64) | 256 | vae_enc_conv_0[0][0] |
| activation_1 (Activation) | (None, 32, 32, 64) | 0 | batch_normalization_1[0][0] |
| dropout_1 (Dropout) | (None, 32, 32, 64) | 0 | activation_1[0][0] |
| vae_enc_conv_1 (Conv2D) | (None, 16, 16, 128) | 204928 | dropout_1[0][0] |
| batch_normalization_2 (BatchNor | (None, 16, 16, 128) | 512 | vae_enc_conv_1[0][0] |
| activation_2 (Activation) | (None, 16, 16, 128) | 0 | batch_normalization_2[0][0] |
| dropout_2 (Dropout) | (None, 16, 16, 128) | 0 | activation_2[0][0] |
| vae_enc_conv_2 (Conv2D) | (None, 8, 8, 256) | 819456 | dropout_2[0][0] |
| batch_normalization_3 (BatchNor | (None, 8, 8, 256) | 1024 | vae_enc_conv_2[0][0] |
| activation_3 (Activation) | (None, 8, 8, 256) | 0 | batch_normalization_3[0][0] |
| dropout_3 (Dropout) | (None, 8, 8, 256) | 0 | activation_3[0][0] |
| flatten_1 (Flatten) | (None, 16384) | 0 | dropout_3[0][0] |
| vae_z_mean (Dense) | (None, 1024) | 16778240 | flatten_1[0][0] |
| vae_z_log_var (Dense) | (None, 1024) | 16778240 | flatten_1[0][0] |
| vae_z (Lambda) | (None, 1024) | 0 | vae_z_mean[0][0]<br>vae_z_log_var[0][0] |

Total params: 34,587,520
Trainable params: 34,586,624
Non-trainable params: 896

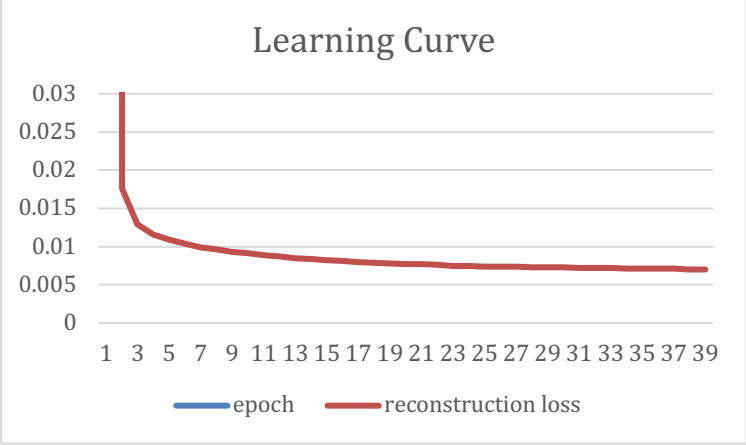
Decoder:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| vae_decoder_in (InputLayer) | (None, 1024) | 0 |
| dense_1 (Dense) | (None, 16384) | 16793600 |
| reshape_1 (Reshape) | (None, 8, 8, 256) | 0 |
| vae_dec_conv_2 (Conv2DTransp | (None, 16, 16, 256) | 1638656 |
| batch_normalization_4 (Batch | (None, 16, 16, 256) | 1024 |
| activation_4 (Activation) | (None, 16, 16, 256) | 0 |
| vae_dec_conv_1 (Conv2DTransp | (None, 32, 32, 128) | 819328 |
| batch_normalization_5 (Batch | (None, 32, 32, 128) | 512 |
| activation_5 (Activation) | (None, 32, 32, 128) | 0 |
| vae_dec_conv_0 (Conv2DTransp | (None, 64, 64, 64) | 204864 |
| batch_normalization_6 (Batch | (None, 64, 64, 64) | 256 |
| activation_6 (Activation) | (None, 64, 64, 64) | 0 |
| vae_decoded (Conv2D) | (None, 64, 64, 3) | 4803 |

Total params: 19,463,043
Trainable params: 19,462,147
Non-trainable params: 896

其中，兩者convolution layer後面皆有batch normalization，沒有pooling 或 up-sampling layers，另外encoder還有dropout。
我的latent dimension是1024維，kl-loss-lambda是1e-5，影像讀入後先轉為介於[0,1]間的值才輸入model。使用Early-stopping callback下39個Epoch便結束training。

2. Plot the learning curve (reconstruction loss & KL divergence) of your model(1%)

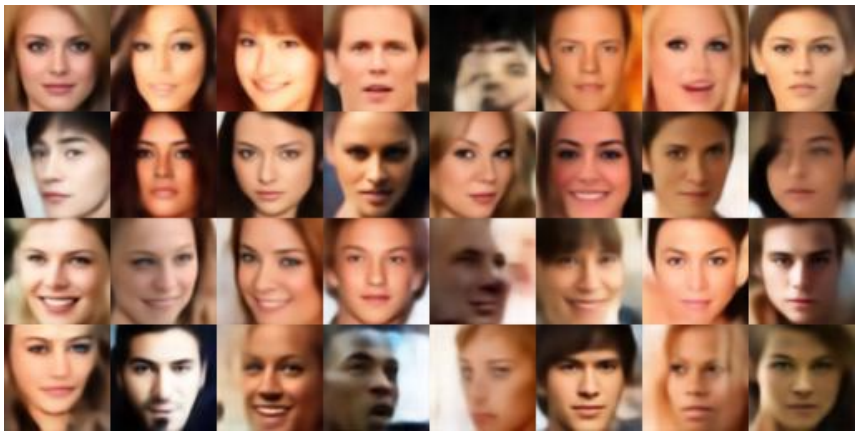


3. Plot 10 testing images and their reconstructed result of your model and report your testing MSE of the entire test set

(1%)

MSE of the entire test set: 0.0064

4. Plot 32 random generated images of your model(1%)



5. Visualize the latent space by mapping test images to 2D space (with tSNE) and color them with respect to an attribute of your choice(1%)

6. Discuss what you’ve observed and learned from implementing VAE(1%)

這次實作VAE之前，我先實作了Dense AE和Conv AE，效果為VAE >> Dense AE > Conv AE，其中VAE和Conv AE的差別只在於Encoder output及Decoder Input。可見’限制latent space distribution’ 確實能為Auto-Encoder Performance帶來大幅度的提升。

Problem 2: GAN

1. Describe the architecture & implementation details of your model(1%)

我的Discriminator及Generator架構分別如下：

discriminator:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dcgan_discrim_in (InputLayer | (None, 64, 64, 3) | 0 |
| dcgan_discrim_conv_0 (Conv2D | (None, 32, 32, 128) | 9728 |
| leaky_re_lu_1 (LeakyReLU) | (None, 32, 32, 128) | 0 |
| dropout_1 (Dropout) | (None, 32, 32, 128) | 0 |
| dcgan_discrim_conv_1 (Conv2D | (None, 16, 16, 256) | 819456 |
| leaky_re_lu_2 (LeakyReLU) | (None, 16, 16, 256) | 0 |
| dropout_2 (Dropout) | (None, 16, 16, 256) | 0 |
| dcgan_discrim_conv_2 (Conv2D | (None, 8, 8, 512) | 3277312 |
| leaky_re_lu_3 (LeakyReLU) | (None, 8, 8, 512) | 0 |
| dropout_3 (Dropout) | (None, 8, 8, 512) | 0 |
| flatten_1 (Flatten) | (None, 32768) | 0 |
| dense_1 (Dense) | (None, 256) | 8388864 |
| leaky_re_lu_4 (LeakyReLU) | (None, 256) | 0 |
| dropout_4 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 1) | 257 |
| activation_1 (Activation) | (None, 1) | 0 |

```
/usr/local/lib/python3.6/site-packages/keras/engine/training.py:9
75: UserWarning: Discrepancy between trainable weights and collec
ted trainable weights, did you set `model.trainable` without call
ing `model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'
Total params: 24,991,234
Trainable params: 12,495,617
Non-trainable params: 12,495,617
```

generator:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dcgan_gen_in (InputLayer) | (None, 100) | 0 |
| dense_3 (Dense) | (None, 8192) | 827392 |
| batch_normalization_1 (Batch | (None, 8192) | 32768 |
| activation_2 (Activation) | (None, 8192) | 0 |
| reshape_1 (Reshape) | (None, 8, 8, 128) | 0 |
| dcgan_gen_conv_0 (Conv2DTran | (None, 16, 16, 512) | 1638912 |
| batch_normalization_2 (Batch | (None, 16, 16, 512) | 2048 |
| activation_3 (Activation) | (None, 16, 16, 512) | 0 |
| dcgan_gen_conv_1 (Conv2DTran | (None, 32, 32, 256) | 3277056 |
| batch_normalization_3 (Batch | (None, 32, 32, 256) | 1024 |
| activation_4 (Activation) | (None, 32, 32, 256) | 0 |
| dcgan_gen_conv_2 (Conv2DTran | (None, 64, 64, 128) | 819328 |
| batch_normalization_4 (Batch | (None, 64, 64, 128) | 512 |
| activation_5 (Activation) | (None, 64, 64, 128) | 0 |
| conv2d_1 (Conv2D) | (None, 64, 64, 3) | 387 |

```
Total params: 6,599,427
Trainable params: 6,581,251
Non-trainable params: 18,176
```

其中，兩者皆沒有pooling 或 up-sampling layers，generator有batch normalization 及dropout，Discriminator則無，但是最後有兩層分別為256和1的Dense layer。

我的latent dimension是100維，影像讀入後先轉為介於[-1,1]間的值才輸入discriminator。(Generator最後的activation為tanh)

2. Plot the learning curve (in the way you prefer) of your model and briefly explain what you think it represents(1%)

3. Plot 32 random generated images of your model(1%)

4. Discuss what you’ve observed and learned from implementing GAN(1%)

   我發現在不調整loss metric的情形下，DCGAN很難Train得起來，且Model Structure稍有變動結果便會差很多。此外，一般DCGAN Train好後即便能成功生成圖片，所生成的圖仍會非常接近，這可能是因為Mode Collapse的關係。

5. Compare the difference between image generated by VAE and GAN, discuss what you’ve observed(1%)

   VAE生成的結果不論是在圖像品質或是變化上都較一般的GAN好，且參數較少、收斂速度較快，因此VAE在Generation上並不失為一個好的Base Line。

Problem 3: ACGAN

1. Describe the architecture & implementation details of your model(1%)

   我的Discriminator及Generator架構分別如下：

discriminator:

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| acgan_discrim_in (InputLayer) | (None, 64, 64, 3) | 0 | |
| acgan_discrim_conv_0 (Conv2D) | (None, 32, 32, 128) | 9728 | acgan_discrim_in[0][0] |
| leaky_re_lu_1 (LeakyReLU) | (None, 32, 32, 128) | 0 | acgan_discrim_conv_0[0][0] |
| dropout_1 (Dropout) | (None, 32, 32, 128) | 0 | leaky_re_lu_1[0][0] |
| acgan_discrim_conv_1 (Conv2D) | (None, 16, 16, 256) | 819456 | dropout_1[0][0] |
| leaky_re_lu_2 (LeakyReLU) | (None, 16, 16, 256) | 0 | acgan_discrim_conv_1[0][0] |
| dropout_2 (Dropout) | (None, 16, 16, 256) | 0 | leaky_re_lu_2[0][0] |
| acgan_discrim_conv_2 (Conv2D) | (None, 8, 8, 512) | 3277312 | dropout_2[0][0] |
| leaky_re_lu_3 (LeakyReLU) | (None, 8, 8, 512) | 0 | acgan_discrim_conv_2[0][0] |
| dropout_3 (Dropout) | (None, 8, 8, 512) | 0 | leaky_re_lu_3[0][0] |
| flatten_1 (Flatten) | (None, 32768) | 0 | dropout_3[0][0] |
| dense_1 (Dense) | (None, 256) | 8388864 | flatten_1[0][0] |
| leaky_re_lu_4 (LeakyReLU) | (None, 256) | 0 | dense_1[0][0] |
| dropout_4 (Dropout) | (None, 256) | 0 | leaky_re_lu_4[0][0] |
| dense_2 (Dense) | (None, 1) | 257 | dropout_4[0][0] |
| dense_3 (Dense) | (None, 14) | 3598 | dropout_4[0][0] |
| activation_1 (Activation) | (None, 1) | 0 | dense_2[0][0] |
| activation_2 (Activation) | (None, 14) | 0 | dense_3[0][0] |

```
/usr/local/lib/python3.6/site-packages/keras/engine/training.py:975: UserWarning: Discrepancy betwe
 weights and collected trainable weights, did you set `model.trainable` without calling `model.comp
  'Discrepancy between trainable weights and collected trainable'
Total params: 24,998,430
Trainable params: 12,499,215
Non-trainable params: 12,499,215
```

generator:

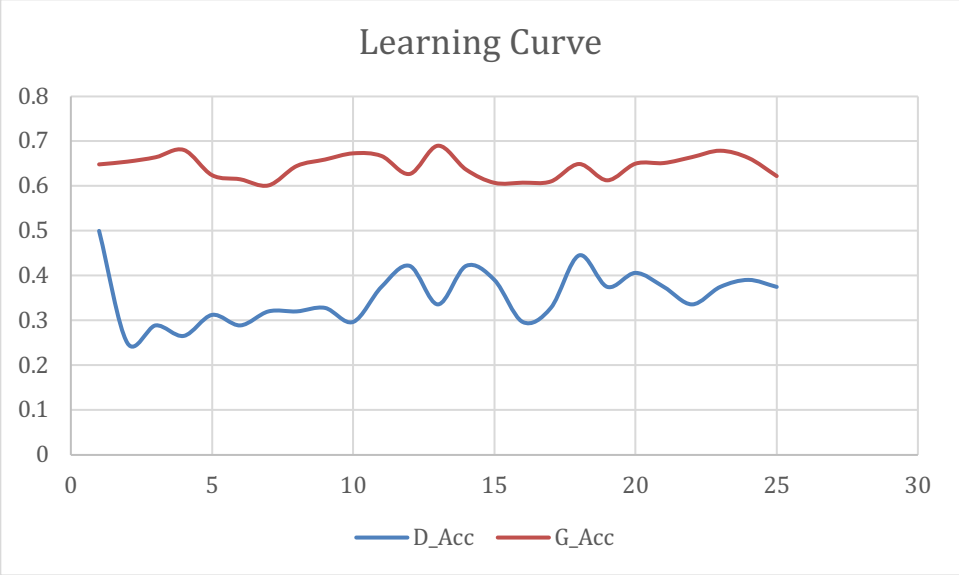
| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| acgan_gen_class_in (InputLayer) | (None, 14) | 0 | |
| acgan_gen_in (InputLayer) | (None, 100) | 0 | |
| dense_4 (Dense) | (None, 100) | 1500 | acgan_gen_class_in[0][0] |
| multiply_1 (Multiply) | (None, 100) | 0 | acgan_gen_in[0][0]<br>dense_4[0][0] |
| dense_5 (Dense) | (None, 8192) | 827392 | multiply_1[0][0] |
| batch_normalization_1 (BatchNor | (None, 8192) | 32768 | dense_5[0][0] |
| activation_3 (Activation) | (None, 8192) | 0 | batch_normalization_1[0][0] |
| reshape_1 (Reshape) | (None, 8, 8, 128) | 0 | activation_3[0][0] |
| acgan_gen_conv_0 (Conv2DTranspo | (None, 16, 16, 512) | 1638912 | reshape_1[0][0] |
| batch_normalization_2 (BatchNor | (None, 16, 16, 512) | 2048 | acgan_gen_conv_0[0][0] |
| activation_4 (Activation) | (None, 16, 16, 512) | 0 | batch_normalization_2[0][0] |
| acgan_gen_conv_1 (Conv2DTranspo | (None, 32, 32, 256) | 3277056 | activation_4[0][0] |
| batch_normalization_3 (BatchNor | (None, 32, 32, 256) | 1024 | acgan_gen_conv_1[0][0] |
| activation_5 (Activation) | (None, 32, 32, 256) | 0 | batch_normalization_3[0][0] |
| acgan_gen_conv_2 (Conv2DTranspo | (None, 64, 64, 128) | 819328 | activation_5[0][0] |
| batch_normalization_4 (BatchNor | (None, 64, 64, 128) | 512 | acgan_gen_conv_2[0][0] |
| activation_6 (Activation) | (None, 64, 64, 128) | 0 | batch_normalization_4[0][0] |
| conv2d_1 (Conv2D) | (None, 64, 64, 3) | 387 | activation_6[0][0] |

```
Total params: 6,600,927
Trainable params: 6,582,751
Non-trainable params: 18,176
```

Convolutional Layers的維度、kernel數和DCGAN維持一致。其中，兩者皆沒有pooling 或 up-sampling layers，generator有batch normalization，Discriminator則有dropout。

Discriminator的output先經過一層Dense(256)，再分別通過Dense(14) 和 Dense(1) 得到Class prediction和 Real/Fake Label predictionGenerator的部分，Class label input先通過Dense(100) 再與Latent vector以點乘的方式Merge，之後便和DCGAN一樣Reshape後通入Convolution layers。

2. Plot the learning curve (in the way you prefer) of your model and briefly explain what you think it represents(1%)




在上圖中，我畫出每個 Epoch Discriminator和Generator 的Accuracy，由此圖大約可以看出兩個Accuracy始終維持互相對抗的關係，並沒有誰能上升至接近100%，且若一者上升，另一者有可能會略為下降。

3. Plot 10 pair of random generated images of your model, each pair generated from the same random vector input but with different attribute. This is to demonstrate your model's ability to disentangle feature of interest(2%)