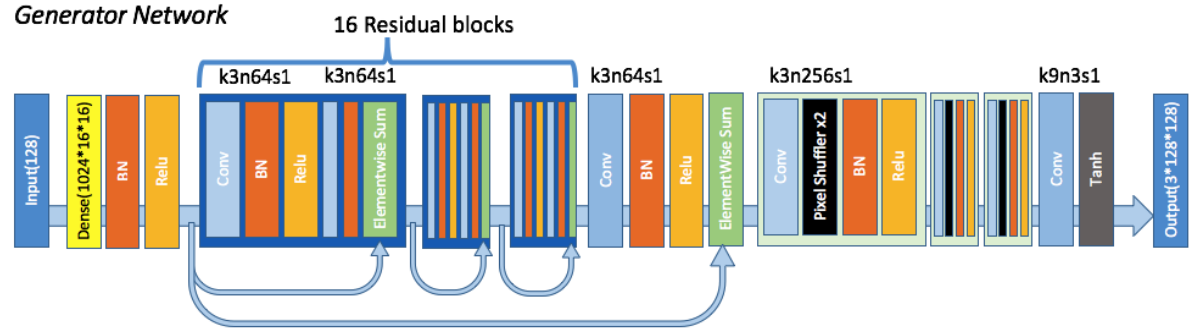
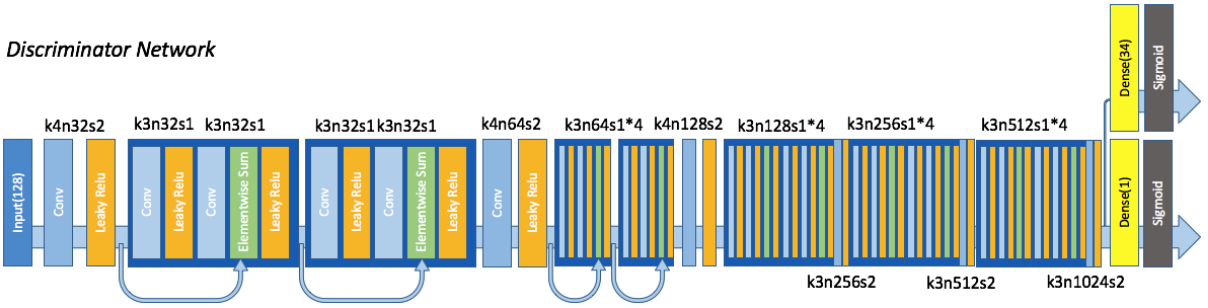
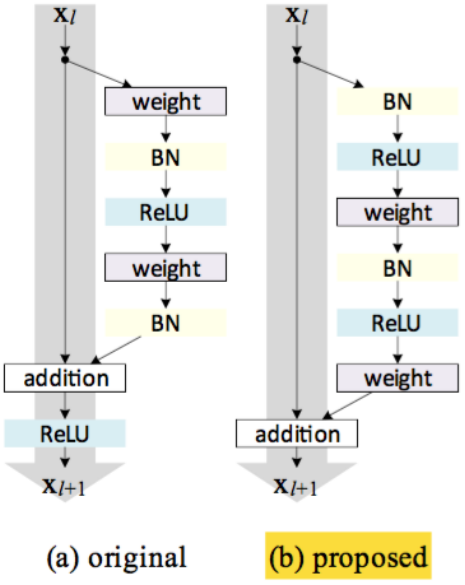
Model Description

- Image generation
  - Discriminator : max(E[D(x)] + E[1-D(G(z))])，模型為4層Conv2D且每層後面都接BatchNorm以及LeakyReLU，最後Activation function 使用 sigmoid
  - Generator : min(E[D(G(z))])，模型為5層Conv2D且每層後面都接BatchNorm以及LeakyReLU，最後的Activation function 使用 tanh
  - model採用上述組成的DCGAN
- Text-to-image generation
  - G : E[D(G(z))]
  - D : E[D(x)] - E[D(G(z))] + $\lambda E[(\|\nabla D(\alpha x + (1-\alpha)G(z))\|_2 - 1)^2]$
  - model 主要參考，基本上就是用 ResNet 組成的 ACGAN
    https://arxiv.org/pdf/1708.05509.pdf





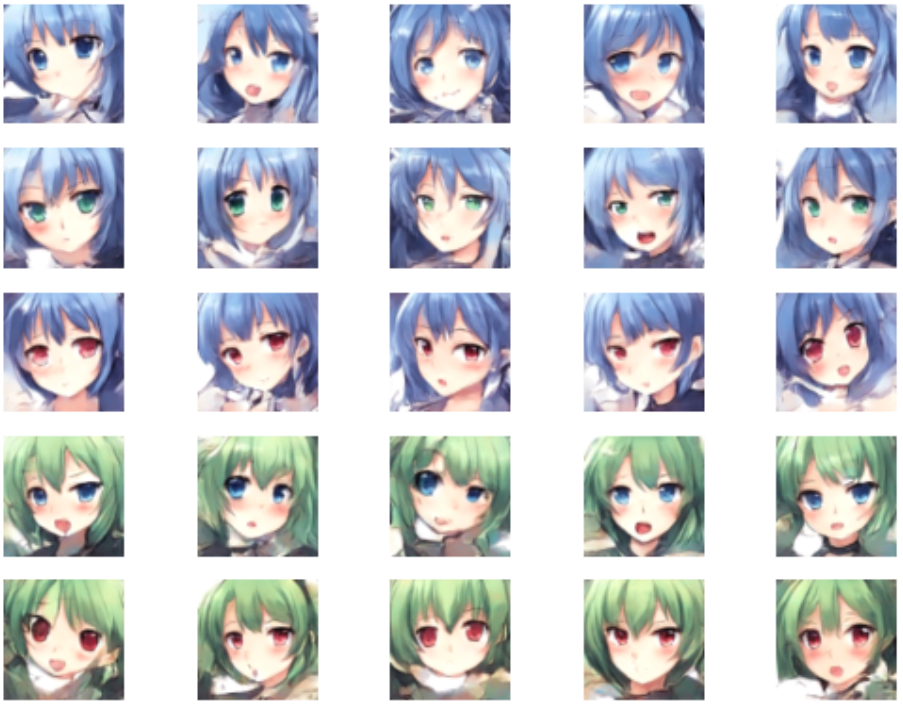
並使用以下的Residual block的架構
https://arxiv.org/pdf/1603.05027.pdf



(a) original (b) proposed

Experiment settings & observation

- Image generation
  - 使用DCGAN
  - 嘗試使用extra dataset (提供給3-2使用的)，
    - 原本以為訓練資料變多應該要比原本好，但結果不然，圖片都變成noise(沒有明顯的臉的輪廓)。我猜測是因為新增的dataset都長得太像了，沒有額外label一起丟入discriminator導致結果變得很差。
- Text-to-image generation
  - 使用 WGAN-gp
  - 使用experience replay
  - 使用額外 datasets
  - 生成圖片時降低noise的std
  - 使用兩個 models 分別做灰階圖片產生和上色，實驗中最主要進步就在於使用灰階圖片訓練可以得到非常好的細節和構圖，猜測因資料顏色非常不均所以彩圖訓練效果沒那麼好



Compare Model with WGAN

- 我們選擇的是WGAN
  - model架構與第一題之DCGAN差不多，差別只在於discriminator的optimizer使用RMSprop以及在每個iteration的時候discriminator會訓練3次，且在每次訓練前會使用np.clip將每層layer的weights clip到-0.01~0.01。
  - Results:左圖為原本的DCGAN結果，右圖為WGAN結果

  - 在loss方面因為WGAN訓練了discriminator較多次，因此不論是在real or fake label上都得到較好的準確率而GAN的loss也降到2.9，表示noise generate出來的圖越接近training data而讓GAN認為他是real image。

Training tips for improvement

- tip 5 Avoid Sparse Gradients
  - Implement details: 將第一部分image generation的DCGAN中的LeakyReLU改成RELU
  - Results: 雖然使用RELU的GAN loss降到了2.1，但是我實際去看生出來的圖片卻不如原本的好，實際如下，我推斷是因為RELU的出來的output都是>0的，導致最後的tanh出來之後也全是介於0,1之間，導致最後在將tanh的 -1~1 match到0-255的時候全部都變成偏大的數字。

- tip 6 Use soft and noisy labels
  - Implement details: 產生real與fake的label時，將real 的label控制在0.8-1.2，fake的控制在0-0.2。
  - Results: 在有用這個tip的情況下，GAN的loss 平均是3.多，而沒有使用的情況下是6.多，代表這有實作的話G所生出來的圖片會比較接近原本的圖片。
- tip 17 Use Dropout in G
  - Implement details: 在G中model呈現與 https://arxiv.org/pdf/1611.07004v1.pdf 這篇paper論述的一樣，Conv->BatchNorm->Dropout->LeakyRelu

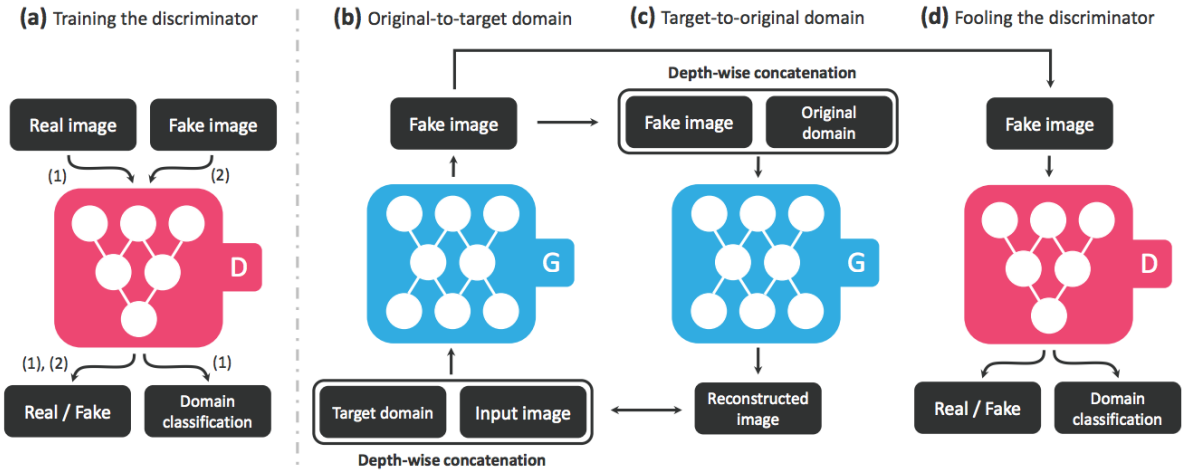
- Results: 相同batch_size與train到最好情況下，上圖為原本下圖為實作結

果

- 發現因為加入Dropout的關係，導致結果變差。

Style transfer

- Model: 我用的是 StarGAN



並使用 https://github.com/yunjey/StarGAN 中的 pretrained model 生成圖片。

- Result:

由左至右分別是：原圖、黑髮、金髮、棕髮、女性化、變老

- Analysis:
  大致上都有達到轉換的目標，而生成出來圖片品質有不少差距，例如金髮的結果就很好，黑髮和金髮在下面那張圖的結果相較之下就比較差，推測可能是因為訓練資料量的差異所造成的結果。

分工表

3-1: 謝宏祺
3-2: 蔡昀達
3-3: 朱柏澄