ADLxMLDS HW4 Report　b05902052 劉家維

- Model description(2%)
    - model structure - Generator
        - Input = (1, 1, 100) 的 noise ，以及 (1, 1, 23) 的標籤（12 種髮色和 11 種瞳色的 one-hot encoding）
        - Concatenate 層，使輸入變成 (1, 1, 123) 的向量
        - Conv2DTranspose 層（反卷積）
            - filters = 2048, kernel_size = (4,4), strides = (1,1)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2DTranspose 層（反卷積）
            - filters = 1024, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2DTranspose 層（反卷積）
            - filters = 512, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2DTranspose 層（反卷積）
            - filters = 256, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2D 層
            - filters = 256, kernel_size = (3,3), strides = (1,1)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2DTranspose 層（反卷積）
            - filters = 3, kernel_size = (4,4), strides = (2,2)
        - Activation (tanh)
        - optimizer = Adam, lr = 0.00015, beta_1 = 0.5

- model structure – Discriminator
    - Input = (64, 64, 3) 的 image ，以及 (1, 1, 23) 的標籤（12 種髮色和 11 種瞳色的 one-hot encoding）
    - 此部份只拿 image 當輸入
        - Conv2D 層
            - filters = 256, kernel_size = (4,4), strides = (2,2)
        - LeakyReLU (alpha = 0.2)
        - Conv2D 層
            - filters = 512, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2D 層
            - filters = 1024, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
        - Conv2D 層
            - filters = 2048, kernel_size = (4,4), strides = (2,2)
        - BatchNormalization (momentum = 0.5)
        - LeakyReLU (alpha = 0.2)
    - 此部份只拿標籤當輸入
        - 將原本 (1, 1, 23) 的標籤擴增 (keras backend 的 tile) 成 (4, 4, 23)
    - Concatenate 上述兩部份的結果，變成 (4, 4, 2071) 的向量
    - Conv2D 層
        - filters = 2048, kernel_size = (1,1), strides = (1,1)
    - BatchNormalization (momentum = 0.5)
    - LeakyReLU (alpha = 0.2)
    - Conv2D 層
        - filters = 1, kernel_size = (4,4), strides = (4,4)
    - Activation (sigmoid)
    - optimizer = Adam, lr = 0.0002, beta_1 = 0.5

    - objective function – Generator
        - 用最基本的 GAN
        - binary (real / fake) crossentropy
    - objective function – Discriminator
        - 用最基本的 GAN
        - binary (real / fake) crossentropy
- How do you improve your performance
    - 將 filter 數增多，及卷積層數增多。
    - 在 Generator 最後先卷積一次再反卷積回來。推測此舉是為了能更好抓到反卷積後圖片的特徵，再以此反卷積回圖片。
- Experiment settings and observation
    - 在訓練 Discriminator 時，若加入一些輸入是原本的圖片，只是配上錯誤標籤，會很容易訓練失敗，連正常的人臉都畫不出。曾經試過佔所有正確圖片的 10%, 20% 都是如此。所以最後就沒有讓正確圖片配上錯誤標籤，直接配上正確的標籤訓練，效果還算能接受。
    - 在訓練 Generator 時，若濾鏡數開太少，生出來的照片很容易是糊糊的，但是濾鏡數一增大可以明顯感受到清晰程度的差距。

      左圖為原本濾鏡數的 ¼ ，右圖為本報告所用的的模型，兩者皆讓 Generator 和 Discriminator 交叉訓練 10000 次，可以看出右邊的頭髮部份比較細膩。

- 沒有 BatchNormalization 差很多，會到完全訓練不起來的程度。
- 試過讓 Generator 和 Discriminator 的訓練次數為 1:1 或 2:1 ，沒有顯著差異。
- 不管用什麼方法，瞳色好像都比較難訓練成功。推測是因為相對於大面積的髮色，眼睛面積相對較小而且位置比較不固定。
- 在訓練 Discriminator 時，曾經學論文試過最後一次卷積前再加入標籤 Concatenate ，結果模型最後居然變成複製人。原本以為是 collapse 問題，試了兩三次，最後我把標籤改成早一點加進去就好了。雖然我不太確定是不是我那三次運氣都不太好，但是我就沒繼續試了。