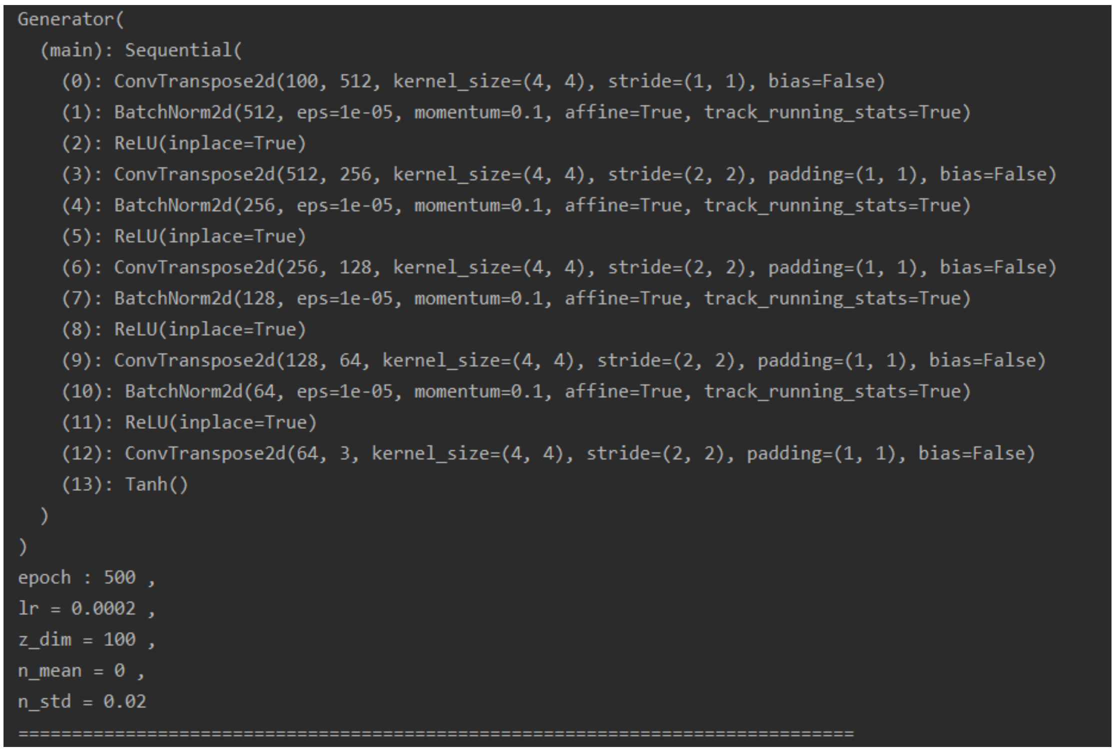
DLCV HW2

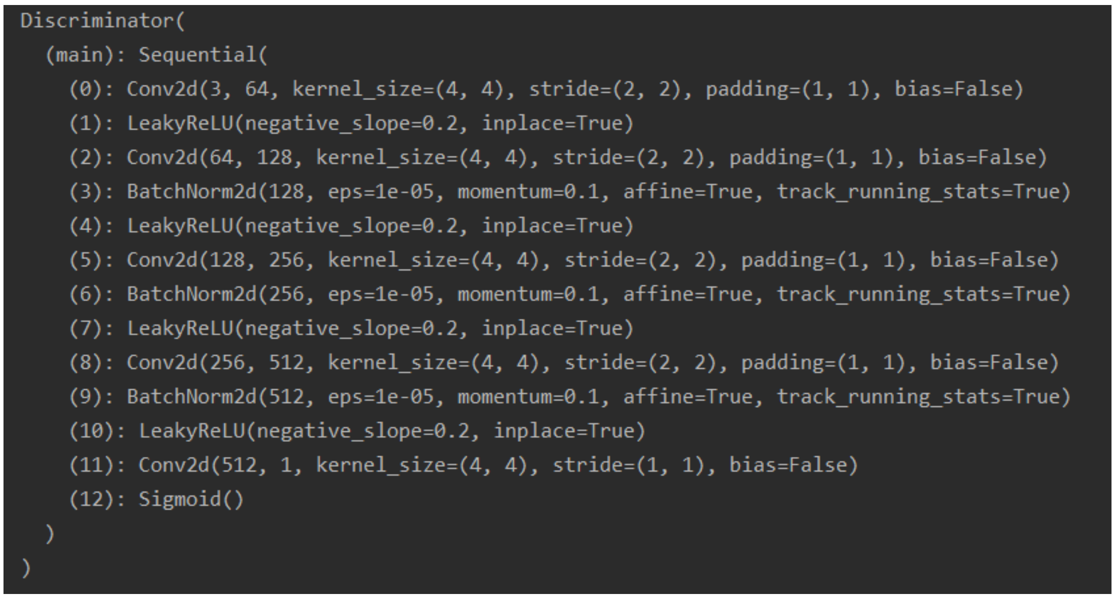
學號:r08945050
姓名:蔡宇晴
系所:生醫電資所

Problem 1: GAN

1. your generator and discriminator from scratch and show your model architecture in your report (You can use "print(model)" in PyTorch directly). Then, train your model on the face dataset and describe implementation details. (Include but not limited to training epochs, learning rate schedule, data augmentation and optimizer) (5%)



```
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
epoch : 500 ,
lr = 0.0002 ,
z_dim = 100 ,
n_mean = 0 ,
n_std = 0.02
===============================================================================
```

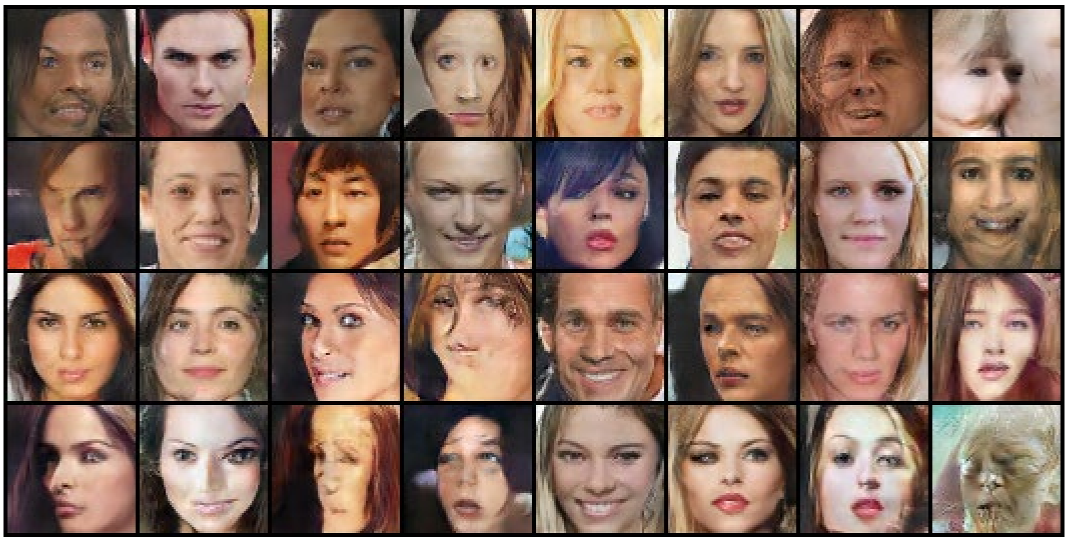


```
Discriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

原先 generator 以 fc layer進行特徵抽取儘管以訓練200epochs，但 FID結果仍約為 40，接著試著以 convTranspose 進行訓練，發現 FID 有明顯下降，此外 DCGAN 作者認為應該對所有 conv 層進行初始化動作 mean=0, stdev=0.02，訓練結果會較穩

定。在初始化conv layer後，模型在epochs = 175時，則達到baseline。

2. Now, we can use the Generator to randomly generate images. Please samples 1000 noise vectors from Normal distribution and input them into your Generator. Save the 1000 generated images in the assigned folder path for evaluation, and show the first 32 images in your report.[see the below example] (5%)



3. Evaluate your 1000 generated images by implementing two metrics:
(1)Fréchet inception distance (FID) : use all "test" images in the face dataset as reference

(2)Inception score (IS)

4. We will calculate FID and IS to evaluate your generated images. (20% total)
FID:24.74
IS:2.07

5. Discuss what you've observed and learned from implementing GAN. (5%)
我認為 GAN 是一個受到超參數影響較大的模型，因此 conv 參數似乎不能如同以往使用較隨機的數值，此外 Generator 與 Discriminator 的能力需要差不多，若有其中一者的能力較弱，則會造成另一個架構的能力也較弱，會影響另一個架構的表現。而在 latent space 中，原本選擇 50~150，但 FID 均無法通過，而將其設為 1024 以後卻有較好的表現，然而在數字生成中，50~150 的 latent space 卻已有不錯的表現，因此我認為在生成細緻的影像生成中(如人臉)，latent space 不宜設置太小。

Problem 2: ACGAN

1. Build your ACGAN model from scratch and show your model architecture in

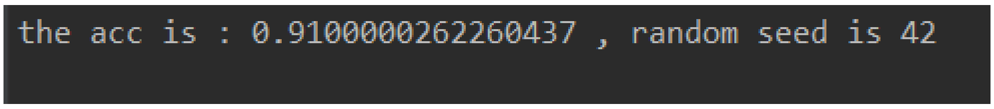
your report (You can use “print(model)” in PyTorch directly). Then, train your model on the mnistm dataset and describe implementation details. (e.g. How do you input the class labels into the model?) (10%)

我使用 embedding layer 將 labels 和 noise 輸入 model 裡面。而原作者是將 labels 以串接的方式接在 noise 後，此方法會造成大量計算上的浪費，因此改進的方法是使用 embedding layer，此方法是將 labels 與 Linear layer 的 weight 先做一次內積，接著會產生一個由 weight 組成的矩陣，此矩陣同時代表查表功能以及 noise 與 weight 運算的能力。這樣的改進方式在面對 latent space 非常大的時候具有良好節省運算的能力。

2. Sample random noise and generate 100 conditional images for each digit (0-9). Save total 1000 outputs in the assigned folder path for further evaluation. We will provide a digit classifier to evaluate your output images.

3. You should name your output digit images as the following format:(The first number of your filename indicates the corresponding digit label)

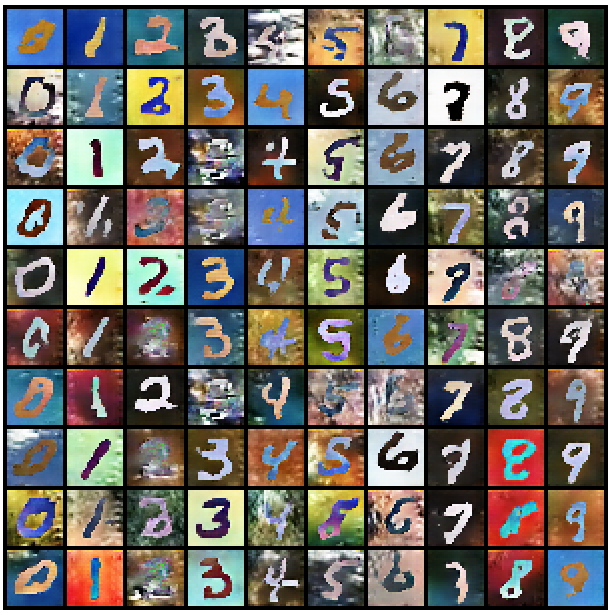
4. We will evaluate your generated output by the classification accuracy with a pretrained digit classifier, and we have provided the model architecture [digit_classifer.py] and the weight [Classifier.pth] for you to test. (15%)



```
the acc is : 0.9100000262260437 , random seed is 42
```

5. Show 10 images for each digit (0-9) in your report. You can put all 100 outputs in one image with columns indicating different digits and rows indicating different noise inputs. [see the below example] (5%)

4.



Problem 3: DANN

1. Compute the accuracy on target domain, while the model is trained on source domain only. (lower bound) (3%)

   Source Domain、target Domain 完全不相同時:lowerbound

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Baseline	75.61%	42.03%	18.2%

2. Compute the accuracy on target domain, while the model is trained on source and target domain. (domain adaptation) (4+9%)

   Source Domain、target Domain 部分相同時:Mid

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Baseline	78.38%	47.02%	28.75%

3. Compute the accuracy on target domain, while the model is trained on target domain only. (upper bound) (3%)

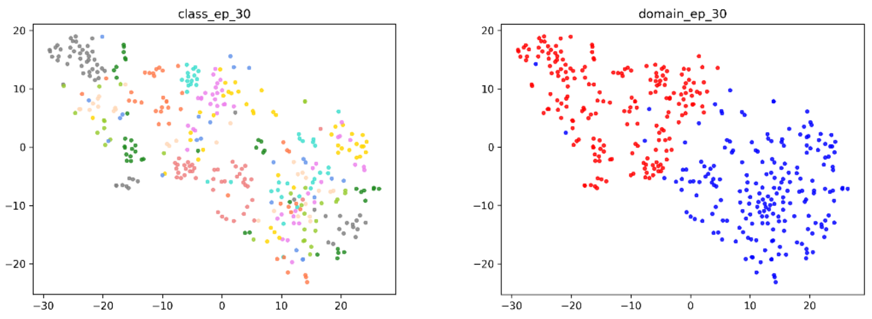
   Source Domain、target Domain 完全相同時:upperbound

	MNIST-M → MNIST-M	SVHN → SVHN	USPS → USPS
Baseline	96.9%	95.71%	89.7%

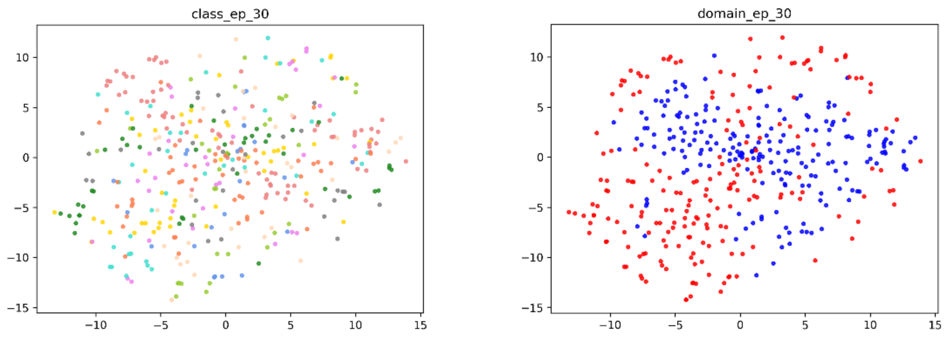
4. Visualize the latent space by mapping the testing images to 2D space with t-SNE and use different colors to indicate data of (a) different digit classes 0-9 and (b)

different domains (source/target). (9%)

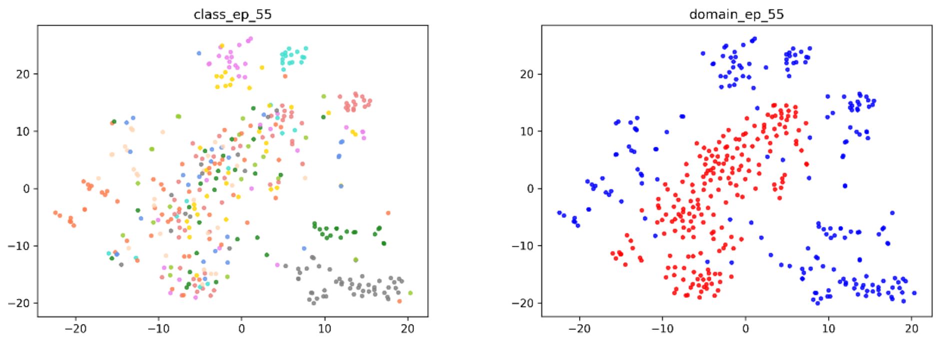
MU:



SM:



US:



5. Describe the implementation details of your model and discuss what you've observed and learned from implementing DANN. (7%)

此 3 個 DOMAIN 轉換均使用相同的 MODEL 架構，但是在一開始 usps->svhn 時的表現卻遠低於 baseline，後來我觀察了兩個 dataset 的差別，發

現 svhn 的風格與其他 2 者的差別較大，且圖片中的數字通常帶有別的數字(部分)，在抽取特徵時，非標準答案的數字可能一樣會被提取出來並更新參數，這樣的影響我認為 usps->svhn 效果不佳的原因之一。我試著將 Loss Fucntion 從原本的 CrossEntropyLoss()改為 NLLLoss()，結果則在前 10 個 epoch 出現了 28.75%的正確率，然而隨著更新次數增加，正確率卻隨之下將，最後在正確率約 20%的時候，出現正確率停滯不前的情況。