Problem1

- **DCGAN model architecture**

```
Generator(
  (decoder): 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()
  )
)
Discriminator(
  (decoder): 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()
  )
)
```

- **Implementation details**

Batch_size = 128

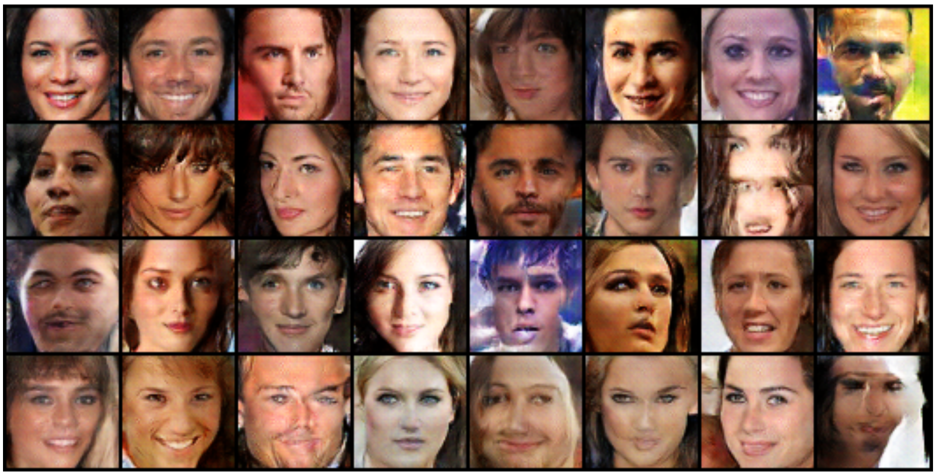
Epoch = 100

Learning rate = 0.0002

Optimizer = Adam

Scheduler = ReduceLROnPlateau

- **The first 32 images result**



- **Evaluation result**

FID : **31.798**

IS : **1.9803**

- **Discuss what you’ve observed and learned from implementing GAN.**

一開始嘗試使用兩種模型分別是 DCGAN 和 WGAN，但 WGAN 好難 train 我也不知道發生什麼事所以後面就放棄ㄌ，之後就都用 DCGAN 來 train。和其他兩題比起來這題要 train 好久，只好偷連別人的 sever 來用。在 training tips 上有看到可以做 Label smoothing，把原本 real=1 和 fake=0 的改成 real=1.2 和 fake=0.3 之類的，但改了之後結果變差很多，好怪呢所以最後就沒有這樣做。然後再 train 的時候 Generator 的 loss 一直爆炸，很煩，所以放了 scheduler，但感覺也沒有比較好，GAN 好難喔。

Problem2

- **ACGAN model architecture**

```
Generator(
  (decoder): 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()
  )
)
Discriminator(
  (decoder): 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()
  )
)
```

- **Implementation details**

batch_size = 32

epoch = 100

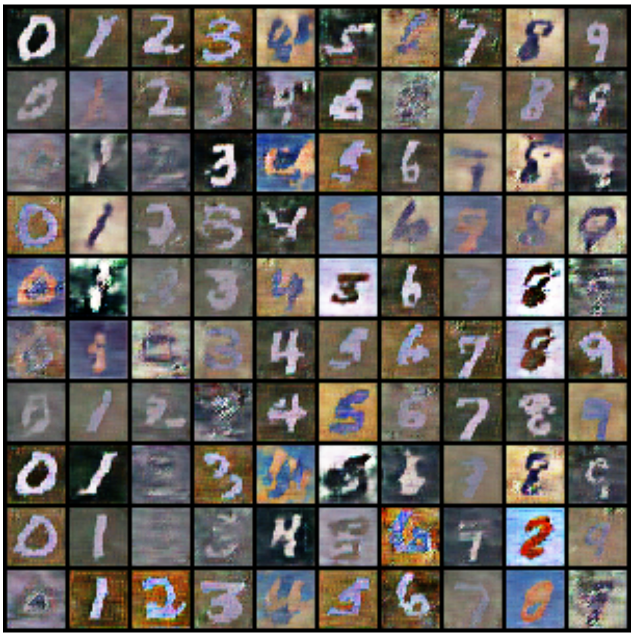
learning rate = 0.0002

optimizer = Adam

- **Evaluation**

Classifier accuracy = **0.834**

- **Result**



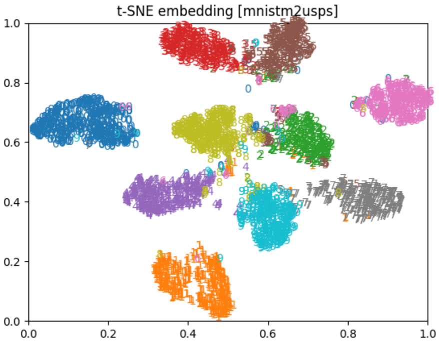
Problem3

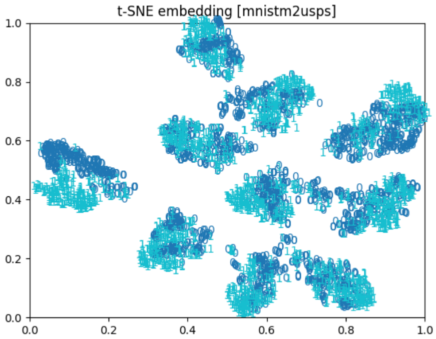
- **Accuracy**

	MNIST-M - USPS	SVHN – MNIST-M	USPS - SVHN
Trained on source	**0.786**	**0.506**	**0.145**
Adaptation	**0.882**	**0.548**	**0.169**
Trained on target	**0.964**	**0.905**	**0.813**

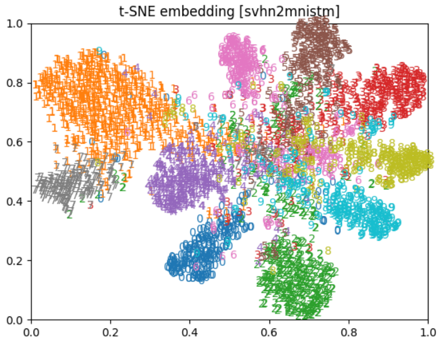
- **t-SNE**

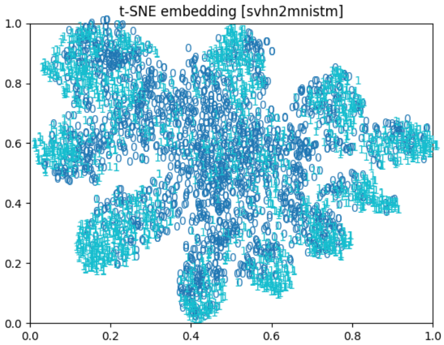
MNIST-M - USPS



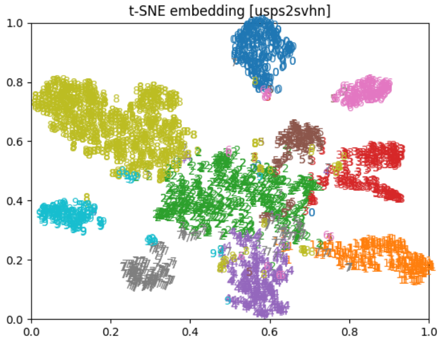


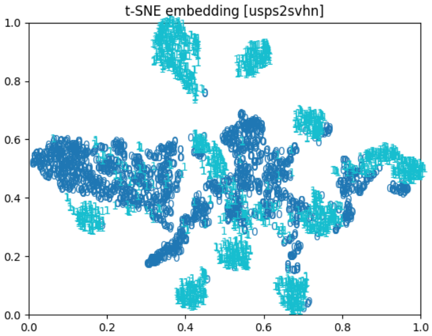
SVHN – MNIST-M





USPS - SVHN





- **Describe the implementation details of your model and discuss what you've observed and learned from implementing DANN.**

Implementation details

Batch_size = 64

Epoch = 100

Learning rate = 0.0002

Lamba = 0.1

Data augmentation:

RandomRotation(25)

RandomPerspective(distortion_sclae=0.3, p=0.5)

ColorJitter(contrast=(1, 1,5))

RandomHorizontaFlip(p=0.8)

首先！這題比起第一題的 GAN 好 train 多了，而且半小時內就可以 train 完一個 model，舒服。Model 架構的部分先用 feature extractor 把 source 和 target 影像卷積拉平後輸出成 512 維，再分別接上 domain classifier 和 label predictor 去預測 domain 以及辨識類別，這兩個模型架構都是直接 MLP 到尾。對於前兩個 scenario，DANN 結果有高於 lower bound，但第三個 DANN 的結果和 lower 一樣爛，可能是照片看起來真的差很多，那我也是沒辦法。

###

Discussing with:

R09521601, R09521603, R09521608

Reference:

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/acgan/acgan.py

https://github.com/ga642381/ML2021-Spring/tree/main/HW11

https://github.com/fungtion/DANN_py3

###