Homework 2 , m11015q11, 柯鐘奇

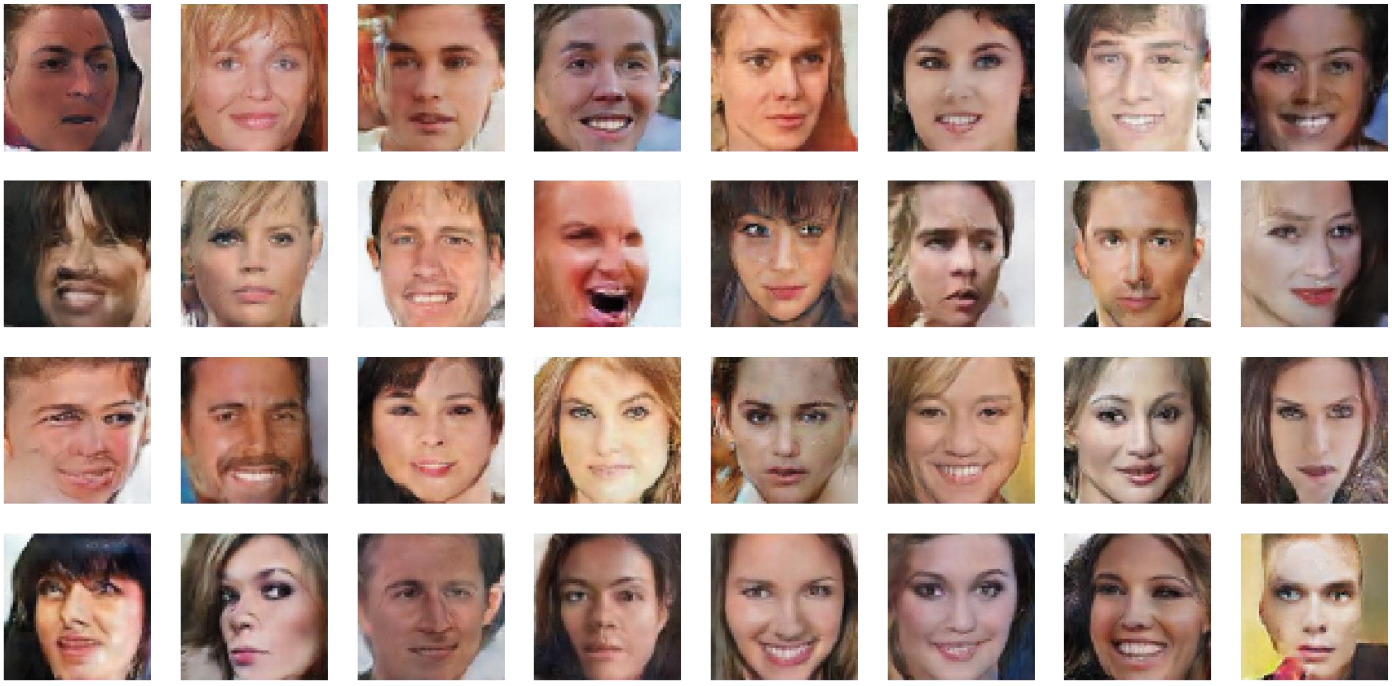
Problem 1. GAN

1. Model

經過查詢相關文獻，基本上還是依照 pytorch 的 DCGAN 教學的 hyperparameter，
因為這組超參數算是經過調教後相對適合的。參考官網的超參數後即達到 Strong Baseline
Epoch:139, learning rate:0.0002, optimizer:Adam, Argumentation:None, Randomseed:999

```
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()
  )
)
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()
  )
)
```

2.



3. In inference code FID = 27, IS = 2.06
4. In inference code FID = 27, IS = 2.06
5. GAN 是一個非常不穩定的模型，若沒有特別了解理論，別隨便亂改架構比較好，
   因此這個 Model 的 hyperparameter 和 Net 的建構是基本上還是參考 Pytorch 官網的 DCGAN。

Problem 2 ACGAN

1. Model

Discriminator 從倒數第二個 Convolution 分岔出 realfake 及 class 層。

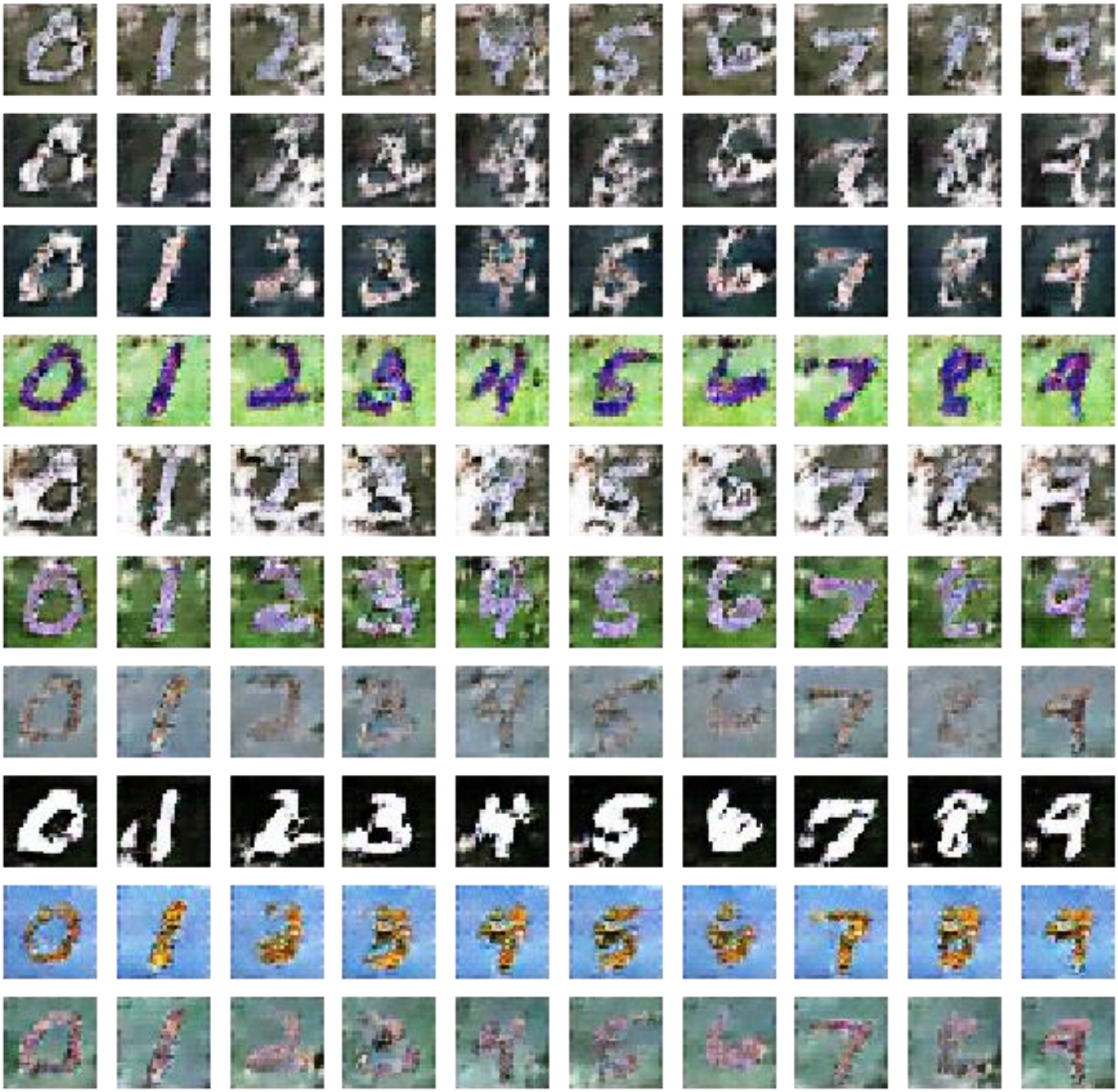
Generator cat 上 10 個 one-hot encoding vector 當作 label vector。

基本上 Model 只是從 DCGAN 拿出來稍做微調，input size 設為 28，conv 另外捲。

```
Discriminator(
  (main): Sequential(
    (0): Conv2d(3, 28, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(28, 56, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(56, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(56, 112, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (realfake): Sequential(
    (0): Conv2d(112, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
  (cls): Sequential(
    (0): Conv2d(112, 10, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Softmax(dim=None)
  )
)
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(110, 224, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(224, 112, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(112, 56, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(56, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(56, 28, kernel_size=(2, 2), stride=(1, 1), padding=(1, 1), bias=False)
    (10): BatchNorm2d(28, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(28, 3, kernel_size=(2, 2), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
```

)

2. In inference code (acc 90%)
3. in inference code (acc 90%)
4. in inference code (acc 90%)
5.

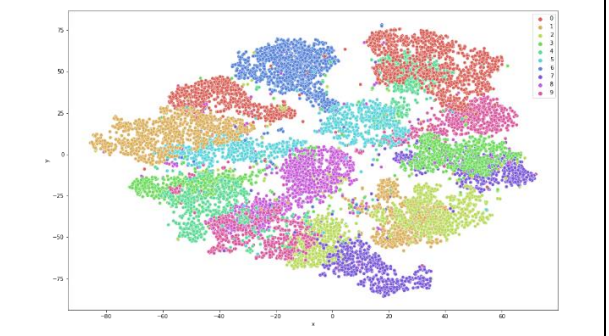
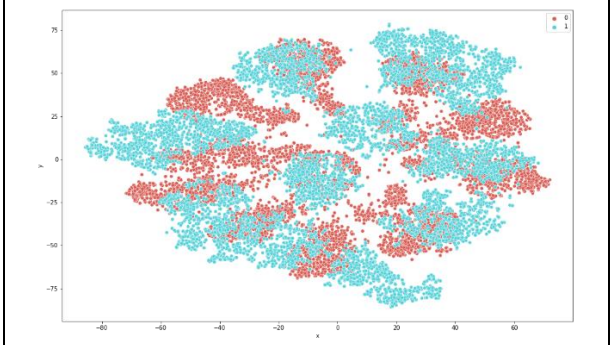
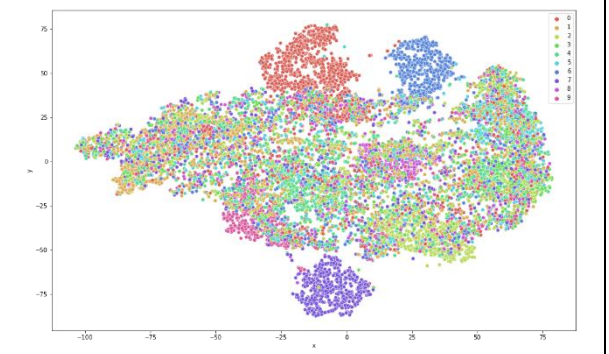
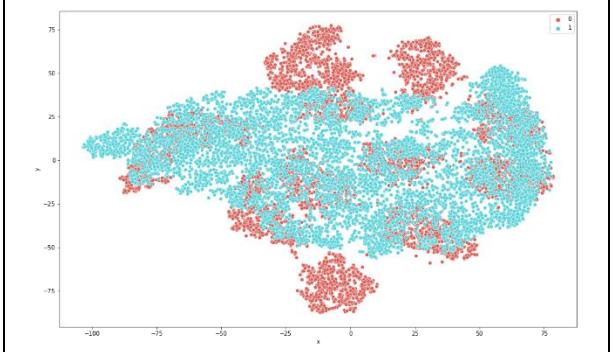
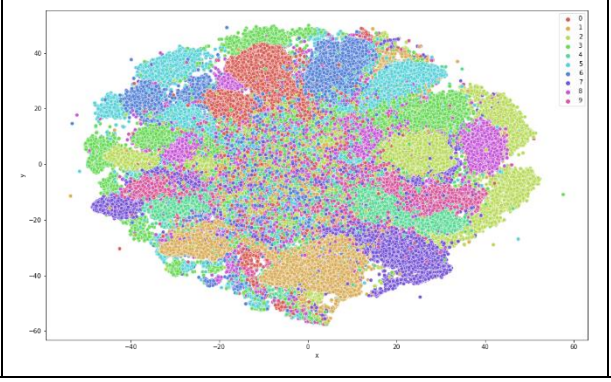
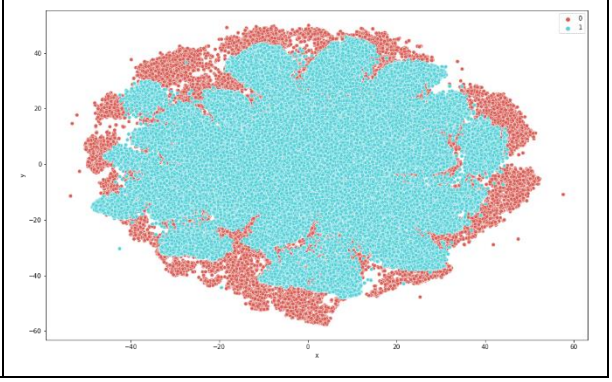


Problem 3. DANN

1&2&3.

ACC	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Trained on source	69	40	10
Adaptation (DANN/Improved)	73	52	14
Trained on target	95	95	88

4. T-SNE

MNIST-M → USPS		
USPS → SVHN		
SVHN → MNIST-M		

5. 基本上還是以 pytorch 的 DCGAN 為基礎下去做修改，但把 realfake 的部分加了一層 gradient reverse layer 混淆原本的模型，前兩個有達到成 baseline，但在 USPS->SVHN 上似乎效果不彰，應另外做 data argumentation，但因為時間來不及就先放掉了 QQ。

參考資料

DCGAN	https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
Gradient reverse layer	https://discuss.pytorch.org/t/solved-reverse-gradients-in-backward-pass/3589
	https://towardsdatascience.com/understanding-acgans-with-code-pytorch-2de35e05d3e4
	https://github.com/clvrai/ACGAN-PyTorch
	https://machinelearningmastery.com/how-to-train-stable-generative-adversarial-networks/
	https://discuss.pytorch.org/t/categorical-cross-entropy-loss-function-equivalent-in-pytorch/85165
	https://zhuanlan.zhihu.com/p/91592775
	https://zhuanlan.zhihu.com/p/44177576
	https://blog.csdn.net/weixin_37993251/article/details/87260372
	https://blog.csdn.net/LOVEmy134611/article/details/109094647