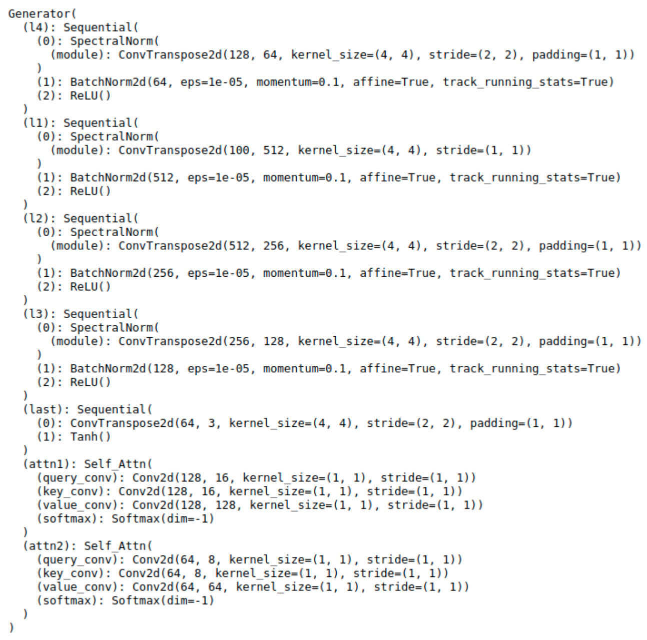
2021 Deep Learning for Computer Vision hw2

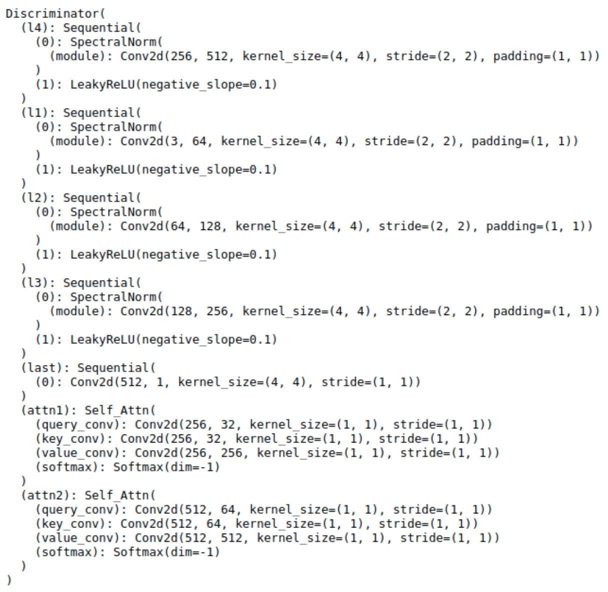
r09922050 陳學韋 資訊工程研究所 (no collaborators)

Problem1-GAN

- [Problem 1-1] Build your generator and discriminator from scratch and show your model architecture in your report. Then, train your model on the face dataset and describe implementation details.
  Ans: 在本次作業的 GAN 實作，我是使用 Self-Attention-GAN 這個架構，也有 generator 和 discriminator 這兩個模型架構，如下圖。與其他 GAN 不同的地方在於多了 attention 的機制。在訓練參數的部分，這兩個架構都是使用 Adam optimizer，在 learning rate 的設定為 generator 是 0.0001，而 discriminator 是 0.0002，batch size 設定 64，並訓練 200 個 epoch。



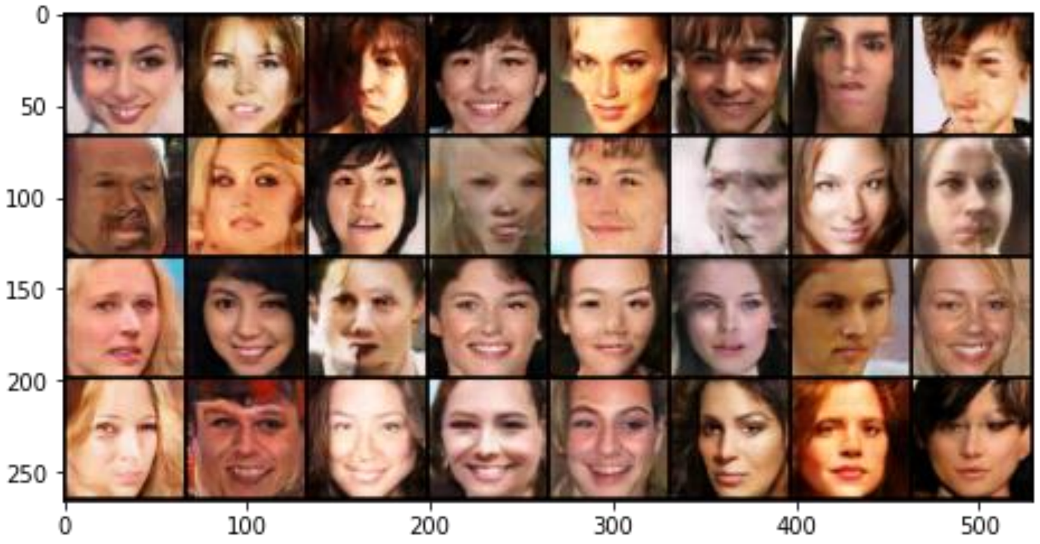
```
Generator(
  (l4): Sequential(
    (0): SpectralNorm(
      (module): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l1): Sequential(
    (0): SpectralNorm(
      (module): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1))
    )
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l2): Sequential(
    (0): SpectralNorm(
      (module): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l3): Sequential(
    (0): SpectralNorm(
      (module): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (last): Sequential(
    (0): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): Tanh()
  )
  (attn1): Self_Attn(
    (query_conv): Conv2d(128, 16, kernel_size=(1, 1), stride=(1, 1))
    (key_conv): Conv2d(128, 16, kernel_size=(1, 1), stride=(1, 1))
    (value_conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
    (softmax): Softmax(dim=-1)
  )
  (attn2): Self_Attn(
    (query_conv): Conv2d(64, 8, kernel_size=(1, 1), stride=(1, 1))
    (key_conv): Conv2d(64, 8, kernel_size=(1, 1), stride=(1, 1))
    (value_conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
    (softmax): Softmax(dim=-1)
  )
)
```



```
Discriminator(
  (l4): Sequential(
    (0): SpectralNorm(
      (module): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): LeakyReLU(negative_slope=0.1)
  )
  (l1): Sequential(
    (0): SpectralNorm(
      (module): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): LeakyReLU(negative_slope=0.1)
  )
  (l2): Sequential(
    (0): SpectralNorm(
      (module): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): LeakyReLU(negative_slope=0.1)
  )
  (l3): Sequential(
    (0): SpectralNorm(
      (module): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    )
    (1): LeakyReLU(negative_slope=0.1)
  )
  (last): Sequential(
    (0): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
  )
  (attn1): Self_Attn(
    (query_conv): Conv2d(256, 32, kernel_size=(1, 1), stride=(1, 1))
    (key_conv): Conv2d(256, 32, kernel_size=(1, 1), stride=(1, 1))
    (value_conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (softmax): Softmax(dim=-1)
  )
  (attn2): Self_Attn(
    (query_conv): Conv2d(512, 64, kernel_size=(1, 1), stride=(1, 1))
    (key_conv): Conv2d(512, 64, kernel_size=(1, 1), stride=(1, 1))
    (value_conv): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
    (softmax): Softmax(dim=-1)
  )
)
```

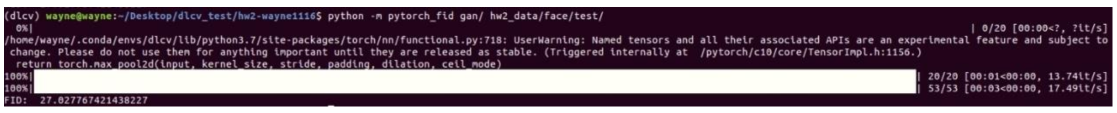
- [Problem 1-2] Please samples 1000 noise vectors from Normal distribution and input them into your Generator. Save the 1000 generated images in the assigned folder path for evaluation, and show the first 32 images in your report.

  Ans: 顯示 1000 張影像的前 32 張如下圖:



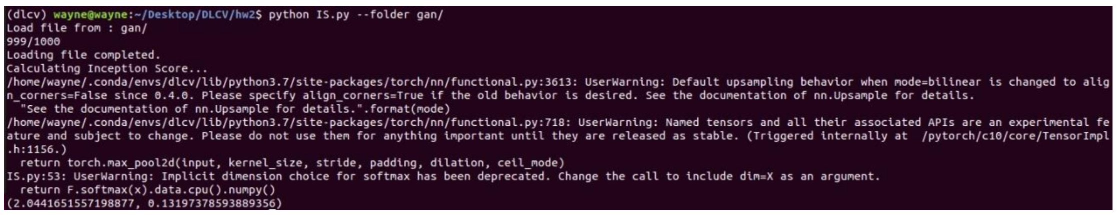
- [Problem 1-3, 1-4] Evaluate your 1000 generated images by implementing two metrics: Fréchet inception distance (FID) and Inception score (IS).

  - Fréchet inception distance (FID) : **27.0277**



```
(dlcv) wayne@wayne:~/Desktop/dlcv_test/hw2-wayne1116$ python -m pytorch_fid gan/ hw2_data/face/test/
  0%|                                                                                  | 0/20 [00:00<?, ?it/s]
/home/wayne/.conda/envs/dlcv/lib/python3.7/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental feature and subject to
 change. Please do not use them for anything important until they are released as stable. (Triggered internally at  /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
100%|                                                                                  | 20/20 [00:01<00:00, 13.74it/s]
100%|                                                                                  | 53/53 [00:03<00:00, 17.49it/s]
FID:  27.027767421438227
```

  - Inception score (IS): **2.0441**



```
(dlcv) wayne@wayne:~/Desktop/DLCV/hw2$ python IS.py --folder gan/
Load file from : gan/
999/1000
Loading file completed.
Calculating Inception Score...
/home/wayne/.conda/envs/dlcv/lib/python3.7/site-packages/torch/nn/functional.py:3613: UserWarning: Default upsampling behavior when mode=bilinear is changed to alig
n_corners=False since 0.4.0. Please specify align_corners=True if the old behavior is desired. See the documentation of nn.Upsample for details.
  "See the documentation of nn.Upsample for details.".format(mode)
/home/wayne/.conda/envs/dlcv/lib/python3.7/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental fe
ature and subject to change. Please do not use them for anything important until they are released as stable. (Triggered internally at  /pytorch/c10/core/TensorImpl
.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
IS.py:53: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x).data.cpu().numpy()
(2.0441651557198877, 0.13197378593889356)
```

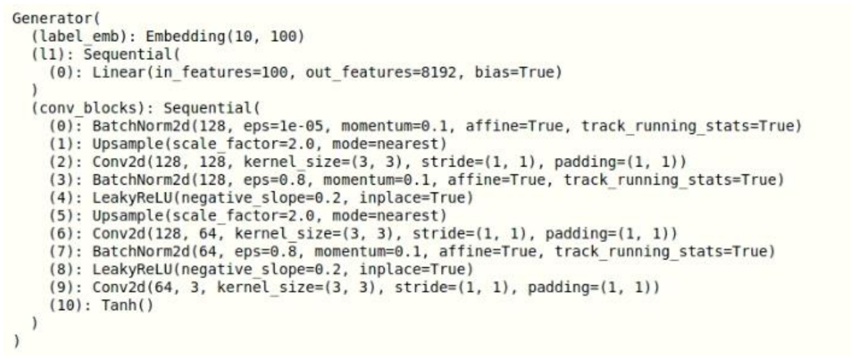
- [Problem 1-5] Discuss what you've observed and learned from implementing GAN.

  Ans: 在對 GAN 進行訓練時，不像之前的 CNN 分類模型 cross entropy loss 越小代表模型是真的有在進步，看當下的 generator loss 或是 discriminator loss 很難知道現在的 generator 和 discriminator 是否表現很好，通常會從分布中隨機挑選，接著透過 generator 產生出對應的影像資訊來評估目前的模型效能，或是使用 Fréchet inception distance 或是 Inception score 等指標。另外，要達到好的 GAN 模型，訓練時間需要很久。

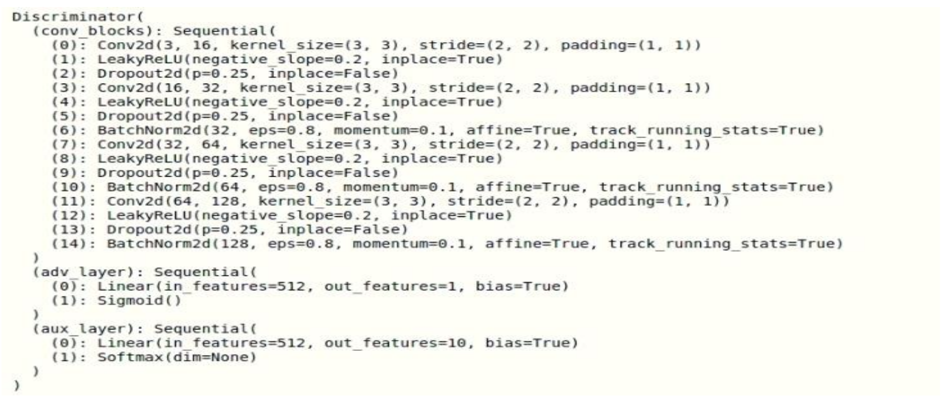
Problem2-ACGAN

- [Problem 2-1] Build your ACGAN model from scratch and show your model architecture in your report. Then, train your model on the mnistm dataset and describe implementation details.

  Ans: 以下為 ACGAN 裡的 generator 和 discriminator 的架構。在訓練參數的部分，這兩個模型都是使用 Adam optimizer，learning rate 皆設為 0.0002，並訓練 100 個 epoch。在模型訓練的時候，不同於一般的 GAN，除了從 normal distribution 選出的 vectors 外，另外會在每個 vector 加上類別數字的資訊(先轉換成 embedding 形式再用 torch.mul 與 vector 作資訊結合)，而且在 discriminator 中除了判別是否是真實數字影像外,也會對影像作類別數字的分類。



```
Generator(
  (label_emb): Embedding(10, 100)
  (l1): Sequential(
    (0): Linear(in_features=100, out_features=8192, bias=True)
  )
  (conv_blocks): Sequential(
    (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (1): Upsample(scale_factor=2.0, mode=nearest)
    (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Upsample(scale_factor=2.0, mode=nearest)
    (6): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (8): LeakyReLU(negative_slope=0.2, inplace=True)
    (9): Conv2d(64, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): Tanh()
  )
)
```



```
Discriminator(
  (conv_blocks): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Dropout2d(p=0.25, inplace=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Dropout2d(p=0.25, inplace=False)
    (6): BatchNorm2d(32, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (7): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (8): LeakyReLU(negative_slope=0.2, inplace=True)
    (9): Dropout2d(p=0.25, inplace=False)
    (10): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (11): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (12): LeakyReLU(negative_slope=0.2, inplace=True)
    (13): Dropout2d(p=0.25, inplace=False)
    (14): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
  (adv_layer): Sequential(
    (0): Linear(in_features=512, out_features=1, bias=True)
    (1): Sigmoid()
  )
  (aux_layer): Sequential(
    (0): Linear(in_features=512, out_features=10, bias=True)
    (1): Softmax(dim=None)
  )
)
```

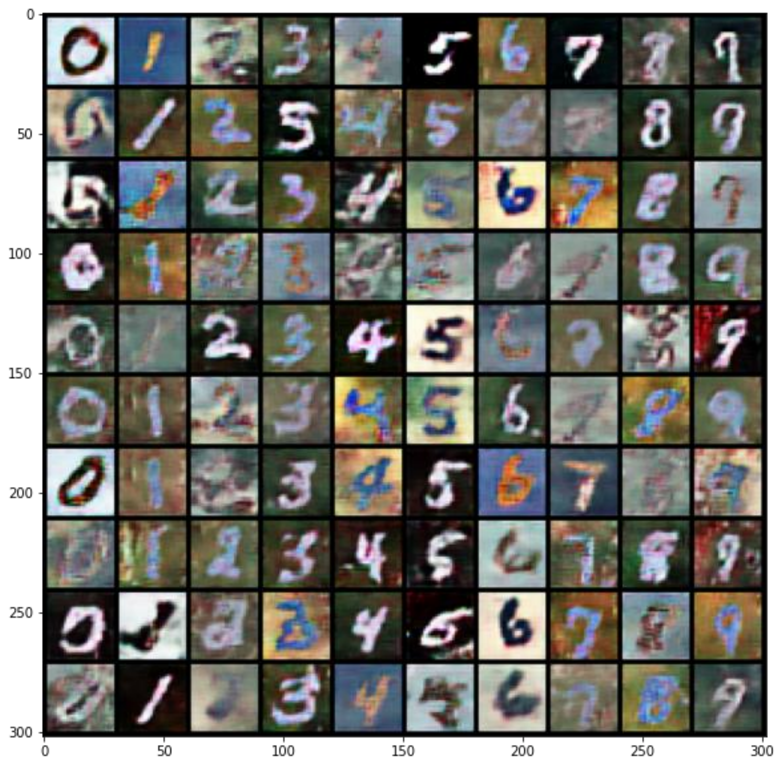
- [Problem 2-2, 2-3, 2-4] Sample random noise and generate 100 conditional images for each digit (0-9). Save total 1000 outputs in the assigned folder path for further evaluation. We will evaluate your generated output by the classification accuracy with a pretrained digit classifier, and we have provided the model architecture [digit_classifer.py] and the weight [Classifier.pth] for you to test.

  Ans: 透過 generator 產生的 1000 張數字影像，在 Classifier.pth 這個模型得出的準確率為 **95.5%**

- [Problem 2-5] 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.

Ans: column 由左至右為 0-9，row 由上而下為不同 noise 的輸入。



Problem3-DANN

- [Problem 3-1, 3-2, 3-3] Implement DANN on digit datasets (USPS, MNIST-M and SVHN) and consider the following 3 scenarios:

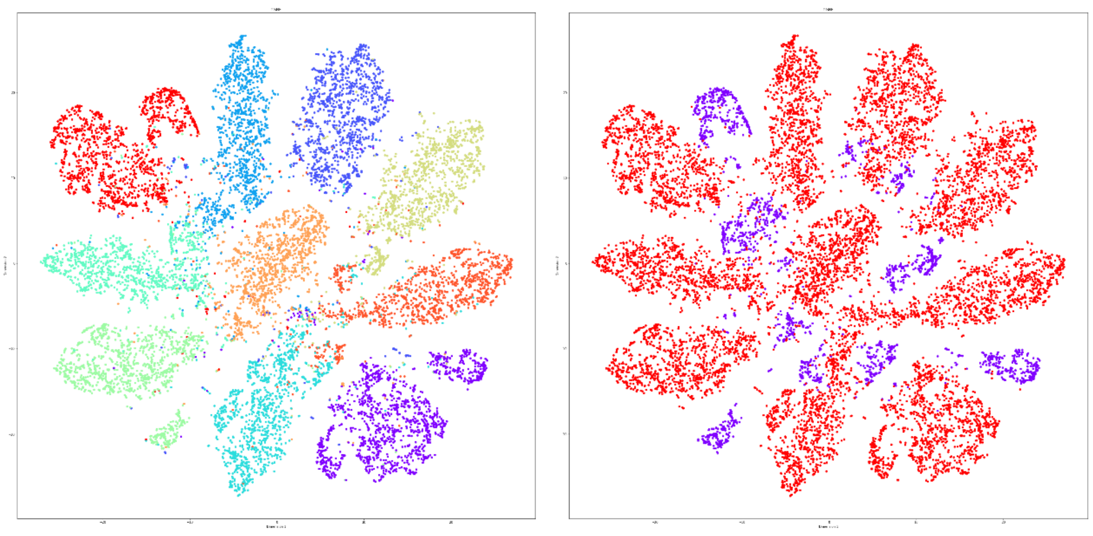
Ans: 以下表格為在不同 dataset 下的 domain adaption:

	MNIST-M→USPS	SVHN→MNIST-M	USPS→SVHN
Trained on source	51.121	40.35	12.16
Adaptation (DANN/Improved)	**78.525 / 6.525**	**73.45 / 31.45**	**28.346 / 0.346**
Trained on target	96.11	97.37	91.864

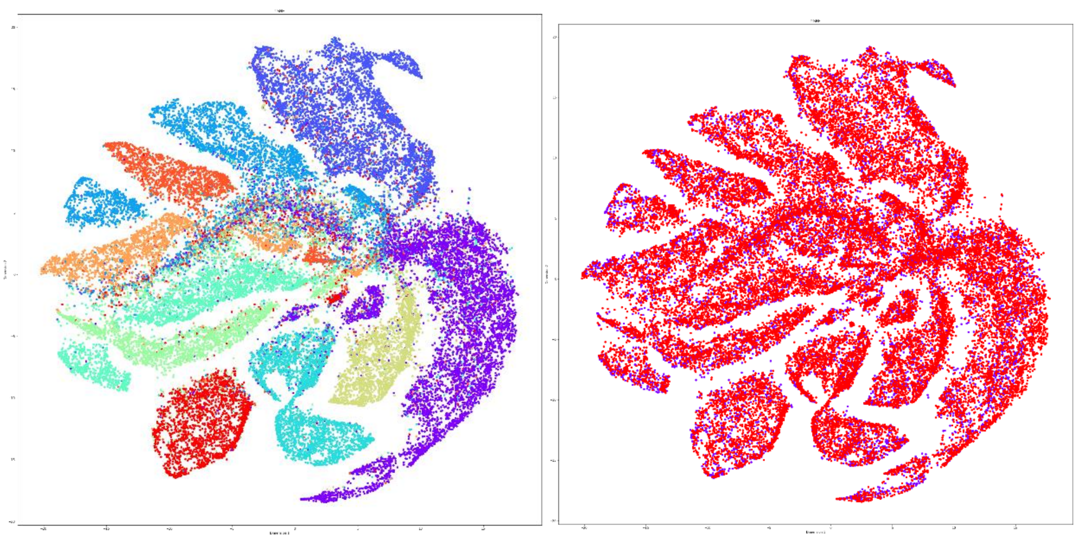
- [Problem 3-4] Visualize the latent space by mapping the testing images to 2D space with t-SNE and use different colors to indicate data of (a) different digit classes 0-9 and (b) different domains (source/target).

Ans: 以下為每個 domain adaptation 中，source 和 target testing dataset 全部資料的不同數字類別(左)和不同 domain(右)的 t-sne 圖示:

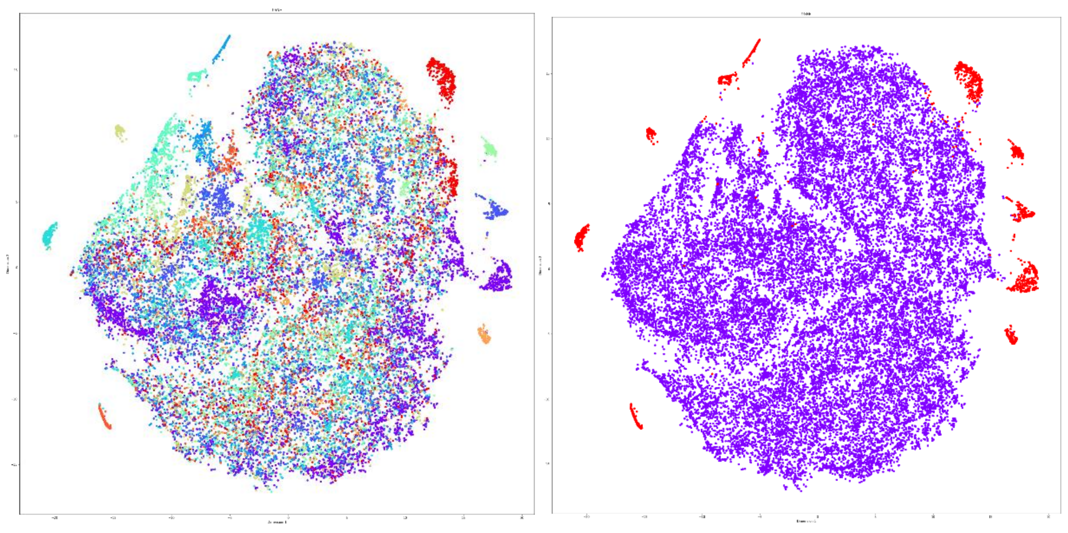
- MNIST-M→USPS



- SVHN→MNIST-M



- USPS→SVHN



- [Problem 3-5] Briefly describe the implementation details of your model and discuss what you’ve observed and learned from implementing DANN.

  Ans: 在本次作業的 DANN 模型中，有三個主要部分: feature extractor、label predictor 和 domain classifier，在 feature extractor 中是將 source domain 和 target domain 映射至 feature space 上，再同時送入至 label predictor 和 domain classifier 作 source domain 的類別分類和不同 domain 間的分類。在訓練 DANN 時，觀察到幾種現象，一是在只使用 target domain 的 image 資訊，很難知道模型訓練的時間越長，target domain 的準確率會逐漸提高的保證，二是在訓練 epoch 設定，會根據 target domain 和 source domain 而不同，很難有一個標準。

Reference

- t-SNE:https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
- Open source github : https://github.com/heykeetae/Self-Attention-GAN
- Open source github: https://github.com/christiancosgrove/pytorch-spectral-normalization-gan
- Open source github: https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/acgan/acgan.py
- Machine Learning 2021 Spring, Heng-Jui Chang @ NTUEE: https://github.com/ga642381/ML2021-Spring/blob/main/HW11/HW11_EN.ipynb
- Domain-Adversarial Training of Neural Networks: https://arxiv.org/pdf/1505.07818.pdf