Problem1:

1. Build your VAE encoder and decoder model and print the architecture [fig1_1.png] (Please use “print(model)” in PyTorch directly) . Then, train your model on the face dataset and describe implementation details of your model. (Include but not limited to training epochs, learning rate schedule, data augmentation and optimizer) (5%)

```
ENT$ python vae.py
VAE(
  (encoder): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2))
    (3): ReLU()
    (4): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2))
    (5): ReLU()
    (6): Conv2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
    (7): ReLU()
    (8): Flatten()
  )
  (fc1): Linear(in_features=12544, out_features=100, bias=True)
  (fc2): Linear(in_features=12544, out_features=100, bias=True)
  (decoder): Sequential(
    (0): UnFlatten()
    (1): ConvTranspose2d(100, 512, kernel_size=(5, 5), stride=(2, 2))
    (2): ReLU()
    (3): ConvTranspose2d(512, 512, kernel_size=(5, 5), stride=(2, 2))
    (4): ReLU()
    (5): ConvTranspose2d(512, 64, kernel_size=(6, 6), stride=(2, 2))
    (6): ReLU()
    (7): ConvTranspose2d(64, 3, kernel_size=(6, 6), stride=(2, 2))
    (8): Sigmoid()
  )
)
```

fig1_1.png

lr 我設定其為 0.0002,
epochs 我設定其為 50 epoch,
latent vector 如同上面為 100 維。

2. Please plot the learning curve (reconstruction loss and KL divergence) of your model. [fig1_2.png] (5%)

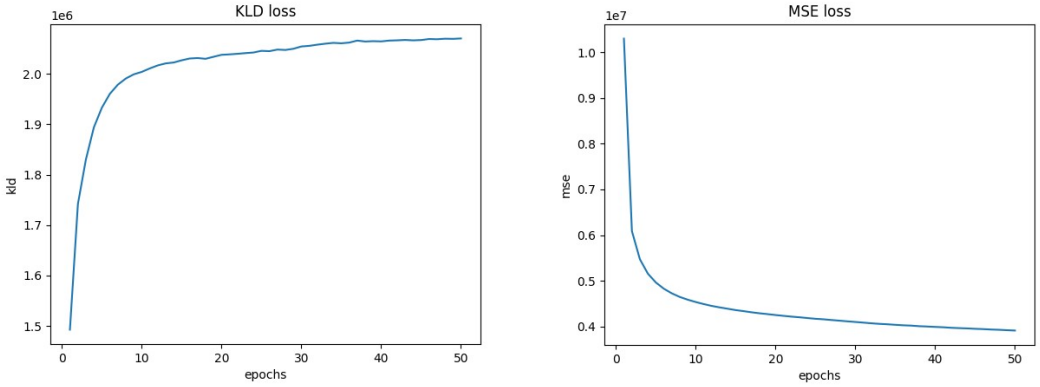


fig1_2.jpg

3. Please random choose 10 testing images and get the reconstructed images from your VAE model. Plot 10 testing images and their reconstructed results (reconstructed images and MSE) from your model. [Table 1_3] (5%)

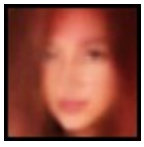
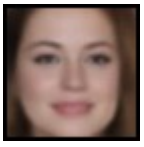
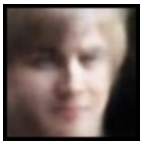
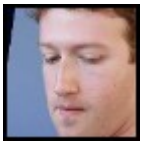
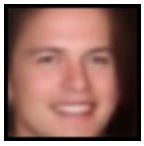
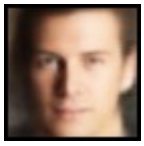
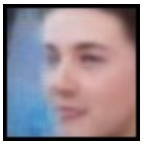
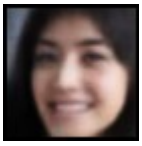
Testing Image					
Recon. Image					
MSE	84.8958	71.7633	56.6538	131.6129	97.2129
Testing Image					
Recon. Image					
MSE	75.5695	80.1066	86.4974	64.3851	132.7158

Table 1_3

4. Now, we utilize decoder in VAE model to randomly generate images by sampling latent vectors from an Normal distribution. Please Plot 32 random generated images from your model. [fig1_4.jpg] (5%)

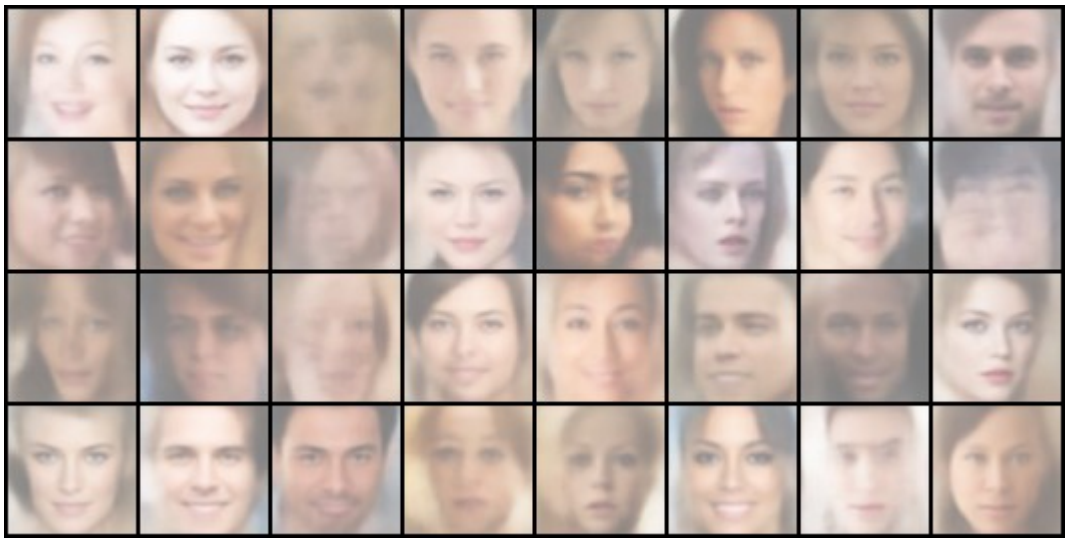


fig1_4.jpg

5. To analyze the latent space of your VAE model, please visualize the latent space by mapping the latent vectors of the test images to 2D space (by tSNE) and color them with respect to an attribute (e.g., gender and hair color) of your choice. (5%) (An example is shown below.)

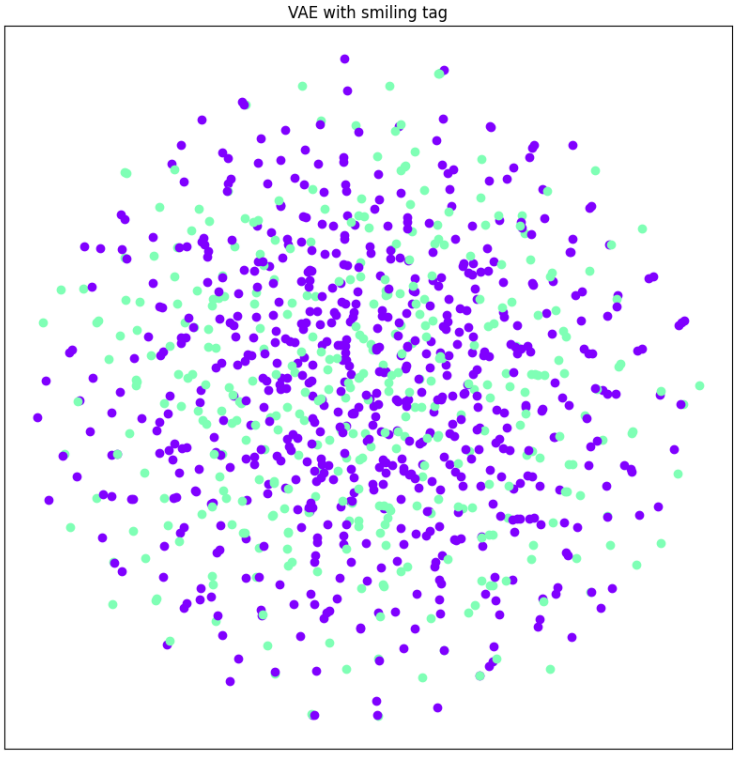


fig1_5.jpg

6. Discuss what you've observed and learned from implementing VAE. (5%)

生成照片時，latent vector 因為是高斯分佈的關係而變得可解釋，也使其能生成照片，但在 tsne 的過程，可能因為微笑本來就是個較困難分辨的特徵，加上 latent vector 有 100 維，導致兩者數據並沒有明顯的分群。二結果也因無對抗（adversarial）關係，導致照片較為模糊。

Problem 2:

1. Build your generator and discriminator in GAN model and print the architecture [fig2_1.png] (Please use “print(model)” in PyTorch directly) . Then, train your model on the face dataset and describe implementation details of your model. (Include but not limited to training epochs, learning rate schedule, data augmentation and optimizer) (5%)

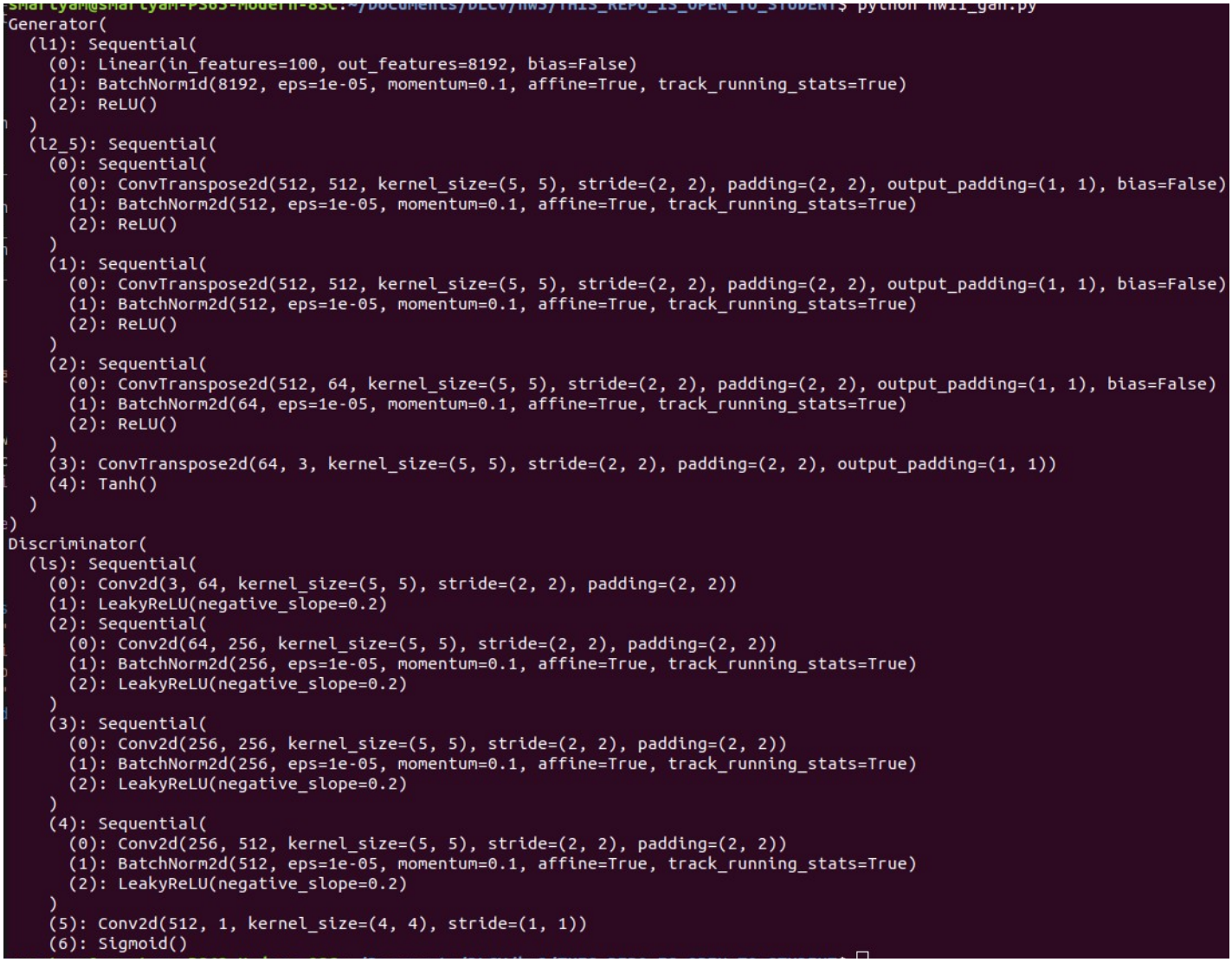


fig2_1.png

我將均值和標準差分別設置為[0.5、0.5、0.5]和[0.5、0.5、0.5]進行歸一化，以適應 tanh 的輸入範圍。（列表中的每個 channel 值對應於 R，G 和 B 的均值和標準差）。

對於數據擴充，我僅將 torchvision.transforms.RandomHorizontalFlip（p = 0.5）應用於以給定概率 p = 0.5 隨機水平翻轉給定圖像。

2. Now, we can use the Generator to randomly generate images. Please samples 32 noise vectors from Normal distribution and input them into your Generator. Plot 32 random images generated from your model. [fig2_2.jpg] (5%)

fig2_2.jpg

3. Discuss what you've observed and learned from implementing GAN. (5%)

我發現 GAN 實際上收斂非常快，如上圖所示，經過訓練 25 個 epoch 後，結果很快就收斂了。和我還發現，儘管 GAN 的作者聲稱 D（G（z））將收斂到 0.5，結果顯示 D（G（z））在整個訓練過程中保持浮動，而不是收斂到某個常數。

4. Compare the difference between image generated by VAE and GAN, discuss what you've observed. (5%)

我發現到 VAE 的人臉輪廓還原較好，彼此之間相差較小，基本上都有符合人臉的輪廓，但是解析度卻變低。而 GAN 得到的結果，雖然解析度較高且細緻的部份較多，但彼此之間相差很大，且扭曲及逋明變形的狀況較多。由此細緻度也可發現，GAN 的發展性明顯高於 VAE。

Problem 3: DANN

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

• Use source images and labels for training

2. Compute the accuracy on target domain, while the model is trained on source and target domain. (domain adaptation) (3+7%)

• Use source images and labels + target images for training

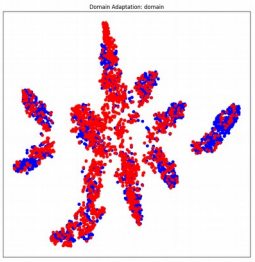
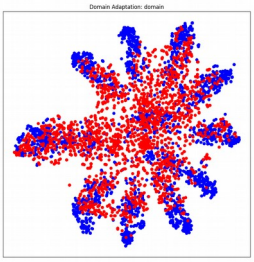
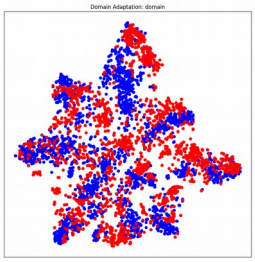
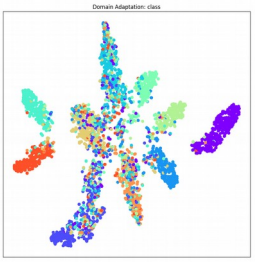
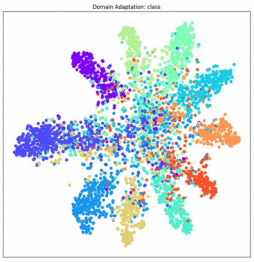
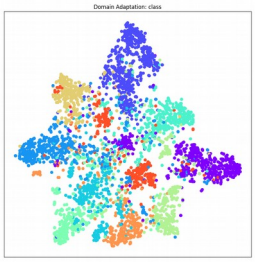
3. Compute the accuracy on target domain, while the model is trained on target domain only. (upper bound) (3%)

• Use target images and labels for training

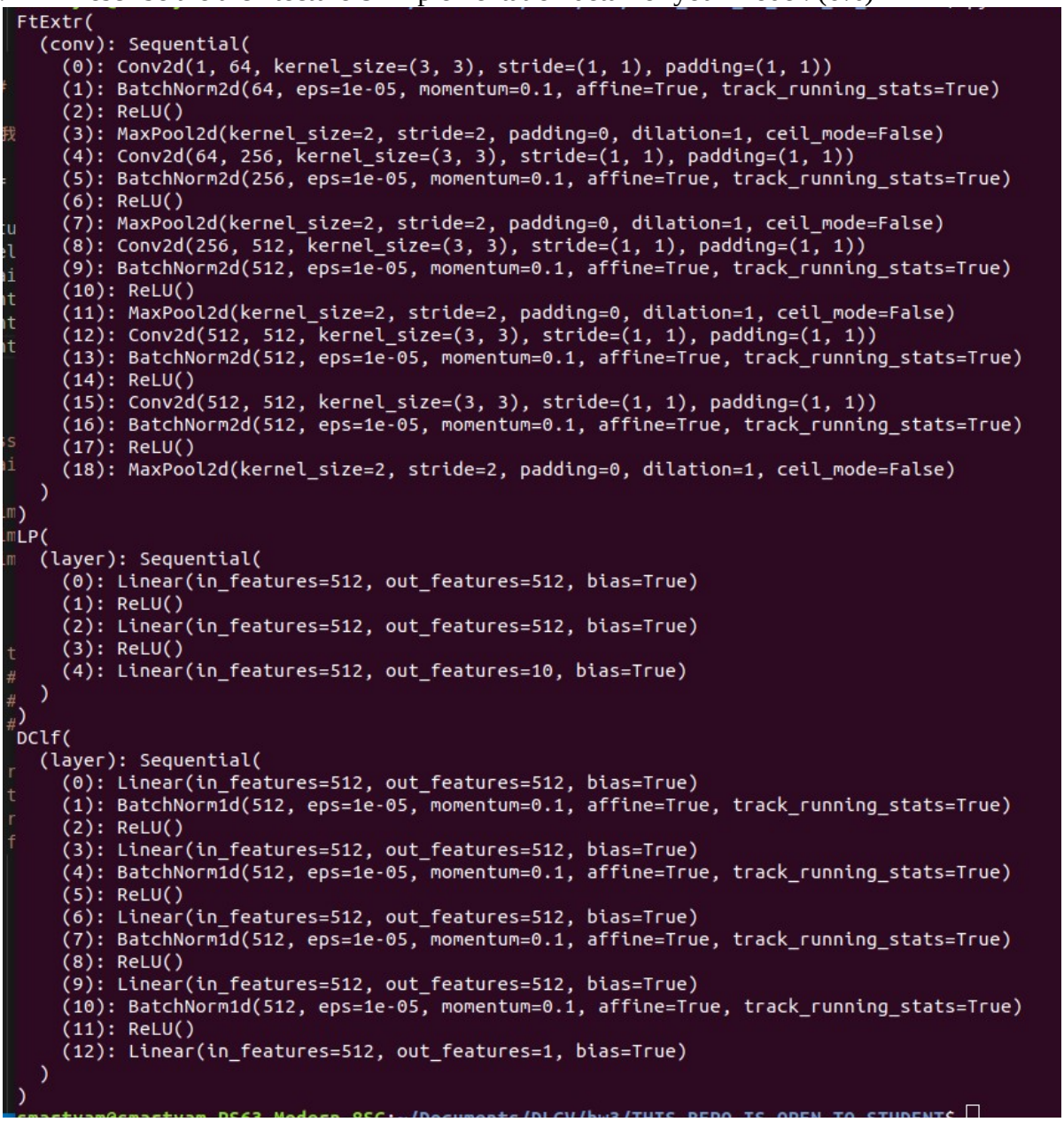
	USPS → MNIST-M	MNIST-M → SVHN	SVHN → USPS
Trained on source	0.1494	0.4061	0.5107
Adaptation (DANN/Improved)	0.460	0.476	0.465
Trained on target	0.8410	0.8675	0.9562

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). (6%)

• Note that you need to plot the figures of all the three scenarios, so you would need to plot 6 figures in total in this sub-problem.

	USPS → MNIST-M	MNIST-M → SVHN	SVHN → USPS
different domains (source/target)			
different digit classes 0-9			

5. Describe the architecture & implementation detail of your model. (6%)



```
FtExtr(
  (conv): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (16): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (17): ReLU()
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
)
LP(
  (layer): Sequential(
    (0): Linear(in_features=512, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
DClf(
  (layer): Sequential(
    (0): Linear(in_features=512, out_features=512, bias=True)
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=512, out_features=512, bias=True)
    (4): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=512, bias=True)
    (7): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Linear(in_features=512, out_features=512, bias=True)
    (10): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
    (12): Linear(in_features=512, out_features=1, bias=True)
  )
)
```

Feature extractor 特徵提取器的功能包括兩部分，提取後續網絡完成任務所需要的特徵，將源域樣本和目標域樣本進行映射和混合。

Label predictor 利用 Feature extractor 提取的信息對樣本進行分類。

Domain classifier 判斷 Feature extractor 提取的信息來自源域還是目標域。

Learning rate 設定為 1e-4，每種組合 train 150 epochs 。

class criterion 使用 CrossEntropyLoss()，domain_criterion 使用 BCEWithLogitsLoss()。

Optimizer 使用 Adam。

6. Discuss what you've observed and learned from implementing DANN. (7%)

DANN 雖然時間過去了的比較久（2016 年發表），但是這種利用對抗來進行域適配的思想卻被大家廣泛使用。它更像是 transfer learning 的一個框架，後面不斷有人向裡面添加各種具有定向功能的網絡來完成專門的任務。不過對於 DANN 這種網絡而言，訓練難度會比較大，因為他只有進行 domain 的 adapt，並沒有保證 classification 的結果需要兩個 domain 相同，而且很難從單源域拓展到多源域，當然，在，這也是我們後面需要解決的問題。

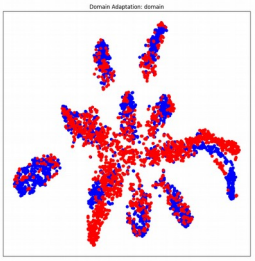
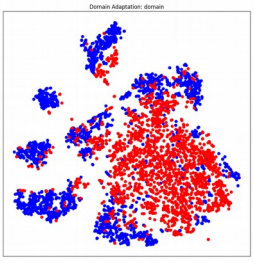
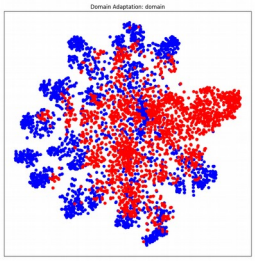
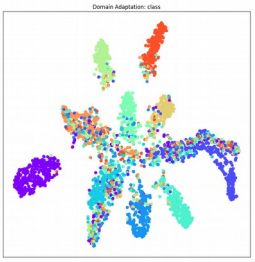
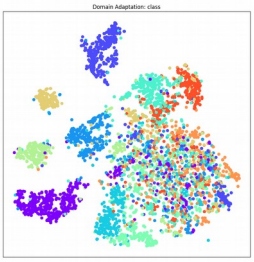
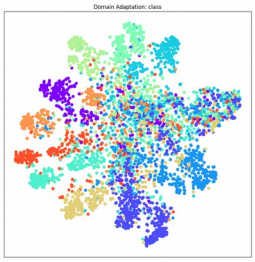
Problem 4: Improved UDA model

1. Compute the accuracy on target domain, while the model is trained on source and target domain. (domain adaptation) (6+10%)

	USPS → MNIST-M	MNIST-M → SVHN	SVHN → USPS
Adaptation (DANN/Improved)	0.3141	0.4940	0.4823

2. Visualize the the latent space by mapping the testing images to 2D space (with t-SNE) and use different colors to indicate data of (a) different digits classes 0-9 and (b) different domains (source/target). (6%)

• Note that you need to plot the figures of all the three scenarios, so you would need to plot 6 figures in total in this sub-problem.

	USPS → MNIST-M	MNIST-M → SVHN	SVHN → USPS
different domains (source/target)			
different digit classes 0-9			

3. Describe the architecture & implementation detail of your model. (6%)

```
>>> Source Encoder <<<
LeNetEncoder(
  (encoder): Sequential(
    (0): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU()
    (3): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
    (4): Dropout2d(p=0.5, inplace=False)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): ReLU()
  )
  (fc1): Linear(in_features=800, out_features=500, bias=True)
)
>>> Source Classifier <<<
LeNetClassifier(
  (fc2): Linear(in_features=500, out_features=10, bias=True)
)
>>> Target Encoder <<<
LeNetEncoder(
  (encoder): Sequential(
    (0): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU()
    (3): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
    (4): Dropout2d(p=0.5, inplace=False)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): ReLU()
  )
  (fc1): Linear(in_features=800, out_features=500, bias=True)
)
>>> Critic <<<
Discriminator(
  (layer): Sequential(
    (0): Linear(in_features=500, out_features=500, bias=True)
    (1): ReLU()
    (2): Linear(in_features=500, out_features=500, bias=True)
    (3): ReLU()
    (4): Linear(in_features=500, out_features=2, bias=True)
    (5): LogSoftmax(dim=None)
  )
)
```

我們先 pretrained 使用 label 的 source image 的 source CNN encoder。接下來，adaptation 通過學習 target encoder 來進行對抗，以至於 discrimimator 看到 encoded source 和 targetexamples 都無法可靠地預測其 domain label。在 testing 過程中，target image 通過 target encoder mapping 到共享特徵空間並按 source classifier 分辨。source encoder train 52 個 epoch，target encoder train 150 個 epoch。lr 皆為 0.0001。

4. Discuss what you've observed and learned from implementing your improved UDA model. (7%)

從 accuracy 上可發現，adaptation 的結果似乎有受到侷限，從比較困難 case adapt 到比較簡單 case 似乎學習效果還不錯，但如果相反過來，則效果會變的很差，比如 usps 到 mnistm，即便 train 較多的 epoch 及調整 lr，跟疊深 model 似乎都沒有比較好的效果，而且 adaptation tsne 出來的結果似乎耶沒有 dann 的好。