DLCV hw2

B07902054 資工四 林子權

Problem 1

Problem 1-1

Generator model structure

```
Generator(
  (fn): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=Fals
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_s
    (2): ReLU(inplace=True)
  )
  (conv): Sequential(
    (0): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), paddi
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): ReLU(inplace=True)
      )
    )
    (1): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), paddi
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): ReLU(inplace=True)
      )
    )
    (2): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), paddin
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
        (2): ReLU(inplace=True)
      )
    )
    (3): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1,
    (4): Tanh()
  )
)
```

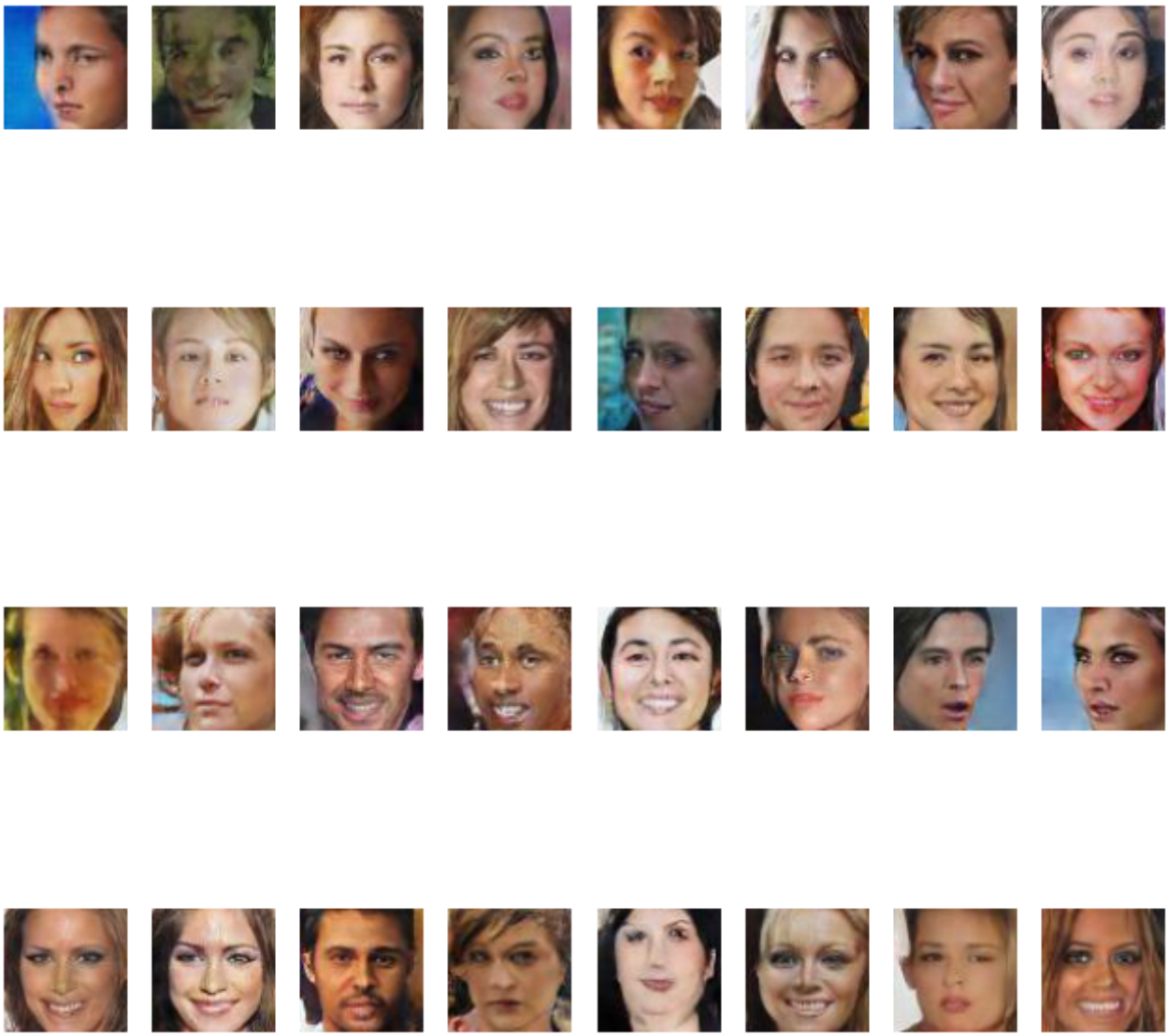
Discriminator model structure

```
Discriminator(
  (conv): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2, inplace=True)
      )
    )
    (3): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2, inplace=True)
      )
    )
    (4): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2, inplace=True)
      )
    )
    (5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (6): Sigmoid()
  )
)
```

Implement detail

```
Training epochs: 300
Learning rate: 2e-4
Batch size: 256
Data augmentation: Normalize with 0.5 mean and 0.5 std
Optimizer: Adam( betas=(0.5, 0.99 ))
Noice vector dimension: 100
```

Problem 1-2



Problem 1-3, 1-4

FID: 29.728547122317565
IS: 2.1595662596859047

Problem 1-5

這次是我第一次自己親自去訓練GAN-based的模型，發現GAN這個模型其實沒有想像中還好訓練，模型架構差一點點都容易有比較大的影響，像是我把所有fully connected layer拿掉換成convolution layer之後，模型收斂的狀況就好很多，這在其他類型的訓練方式上感覺比較不容易觀察到那麼大的影響。

Problem 2

Problem 2-1

Generator model structure

```
Generator(
  (fn): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=Fals
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_s
    (2): ReLU()
  )
  (conv): Sequential(
    (0): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), paddi
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): ReLU()
      )
    )
    (1): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), paddi
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): ReLU()
      )
    )
    (2): Upsampling(
      (uconv): Sequential(
        (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), paddin
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
        (2): ReLU()
      )
    )
    (3): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1,
    (4): Tanh()
  )
  (cls_emb): Embedding(10, 100)
)
```

Discriminator model structure

```
Discriminator(
  (conv): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=
    (1): LeakyReLU(negative_slope=0.2)
    (2): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (3): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (4): Downsampling(
      (dconv): Sequential(
        (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runni
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (5): Sigmoid()
  )
  (real_cls): Sequential(
    (0): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
  (num_cls): Sequential(
    (0): Conv2d(512, 10, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): LogSoftmax(dim=1)
  )
)
```

Implement details

Input class label

Generator的inputs會有2個：random noise vector和class label。Class label會先透過一個embedding lookup table轉換成和random noise vector一樣的維度，然後和noise做inner product，當作後面deep convolution network的輸入。

Hyperparameters

```
Training epochs: 15
Learning rate: 2e-4
Batch size: 256
Data augmentation: Normalize with 0.5 mean and 0.5 std
Optimizer: Adam( betas=(0.5, 0.99))
Noice vector dimension: 100
Class embedding dimension: 100
```

Problem 2-2, Problem 2-3, Problem 2-4

Accuracy: 0.8920

Problem 2-5

Problem 3

Problem 3-1, Problem 3-2, Problem 3-3

Description

- Problem 3-1和Problem 3-3加起來合計六個實驗都用完全一樣的hyperparmeters做training：

```
Training epochs: 10
Learning rate: 2e-4
Batch size: 1024
Data augmentation: Normalize with 0.5 mean and 0.5 std
Optimizer: Adam
```

- Problem 3-2的前兩個scenarios我使用相同的架構，第三個scenarios我使用了和前兩個不同的架構，實做細節請見Problem 3-5

Result

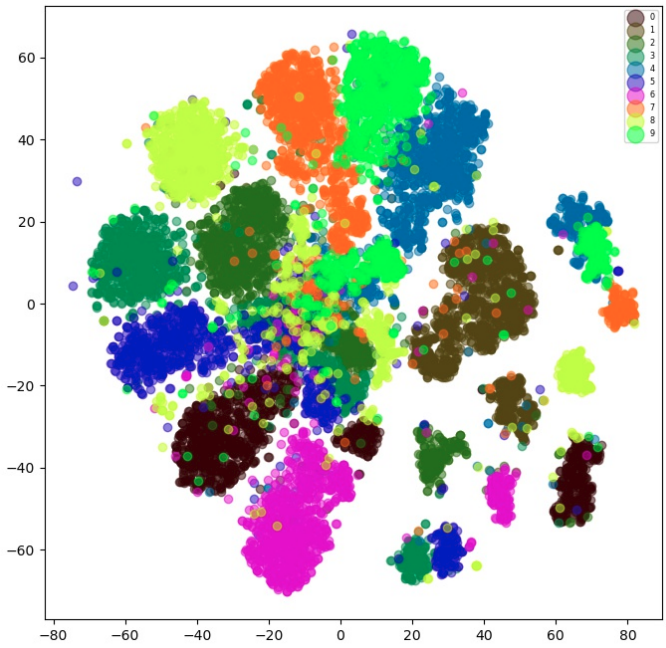
	MNIST-M → USPS	**SVHN → MNIST-M**	**USPS → SVHN**
Trained on source	0.7245766925811768	0.4148	0.16574157774448395
Adaptation	0.8241155954160438	0.4419	0.28607098955132143
Trained on target	0.9417102336883545	0.9715	0.8848407864570618

Problem 3-4

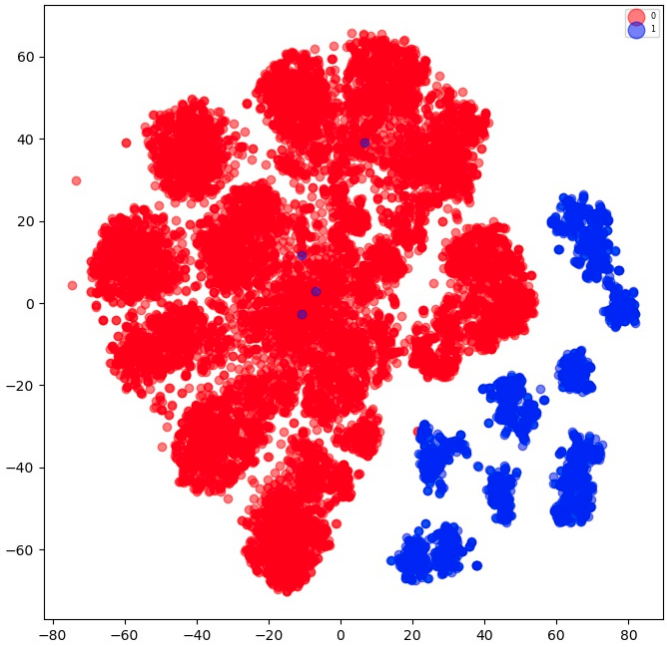
In the visualization result of domain TSNE, label 0 (red spot) represents source domain, label 1 (blue spot) represents target domain.

MNISTM > USPS

Class TSNE

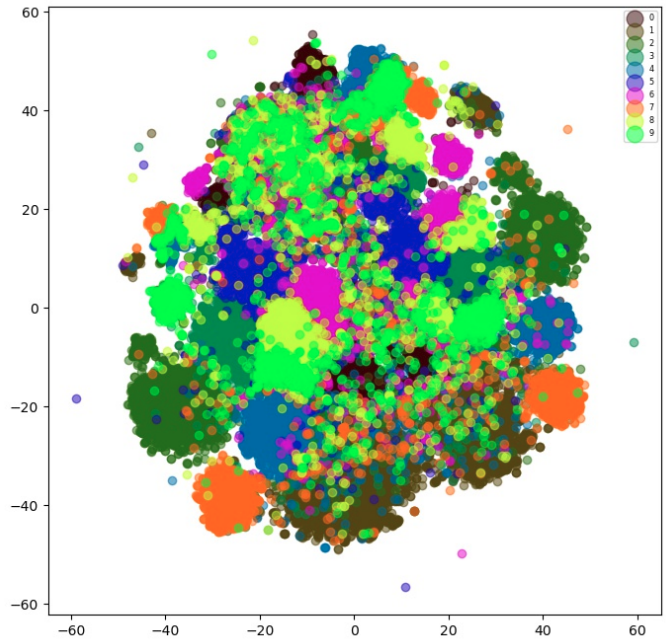


Domain TSNE

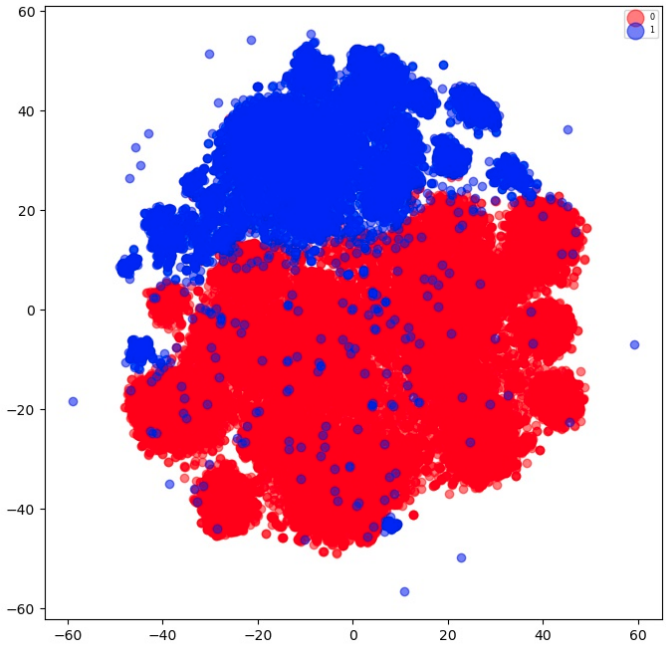


USPS > SVHN

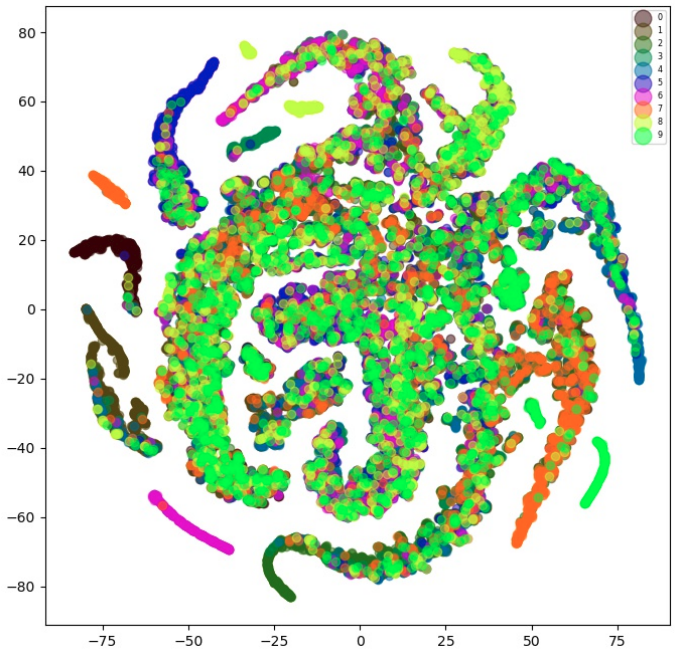
Class TSNE



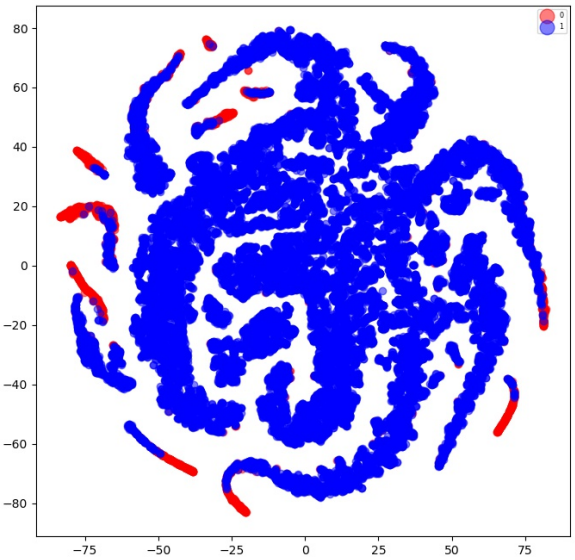
Domain TSNE



Class TSNE



Domain TSNE



Problem 3-5

Implementation detail

MNISTM > USPS and SVHN > MNISTM

- Using two simple convolution blocks as my feature extractor.
- Using fully connected layer as my domain classifier and number classifier.
- Apply gradient reverse layer at the input of domain classifier.
- Training hyperparameter:
  - lr = 1e-4

    - n_epoch = 50
    - batch_size = 64
    - latent vector dimension = 810

USPS > SVHN

- Using non-pretrained ResNet34-like model and three ConvTranspose layer as my feature extractor.
- Using fully connected layer as my domain classifier and number classifier.
- Apply gradient reverse layer at the input of domain classifier.
- Training hyperparameter:
    - lr = 1e-4
    - n_epoch = 100
    - batch_size = 64
    - latent vector dimension = 800

What I have observed and learned

Gradient reverse layer在實做上也是蠻不好訓練的感覺，尤其是第2跟第3個scenarios，都只剛好壓過baseline一點點而已。畢竟也算是一種GAN-based的method，訓練起來比較困難也可以想像。有時候甚至performance沒辦法很輕易的超過直接train在source domain然後inference在target domain的狀況。但這次的作業也很高興可以學會怎麼implement gradient reverse layer，算是學到了一個之前不曾在pytorch看過跟碰過的東西。

Reference

[1] training loop的coding structure參考了這個連結的implementation Hung-yi Lee ML hw11 (https://colab.research.google.com/drive/1JYY_HHtVSSOLixZfLwkxiWTRdPHJCS2t#scrollTo=UZ6d0_cr8R26)