DLCV HW 3

R08945011　張祐祥

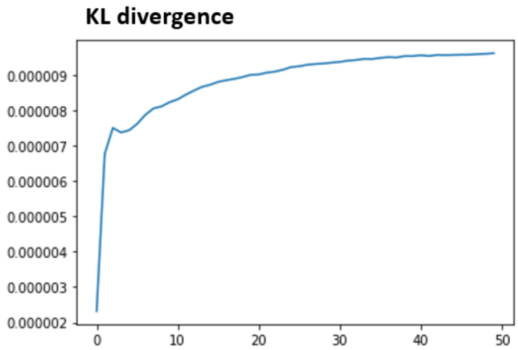
Problem1

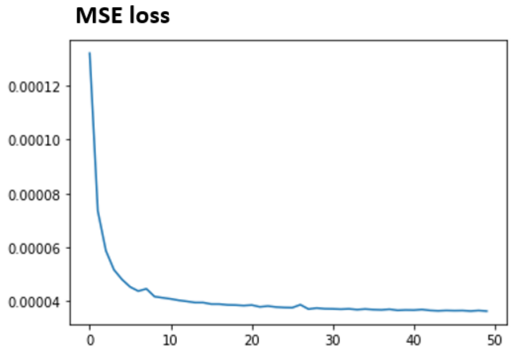
1. model

```
VAE(
  (encoder): Encoder(
    (conv1): Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (relu1): ReLU()
    (conv2): Conv2d(32, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (relu2): ReLU()
    (conv3): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (relu3): ReLU()
    (conv4): Conv2d(256, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (relu4): ReLU()
    (fc_mu): Linear(in_features=1024, out_features=256, bias=True)
    (fc_logvar): Linear(in_features=1024, out_features=256, bias=True)
  )
  (decoder): Decoder(
    (main): Sequential(
      (0): ConvTranspose2d(256, 512, kernel_size=(4, 4), stride=(1, 1))
      (1): ReLU(inplace=True)
      (2): ConvTranspose2d(512, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (3): ReLU(inplace=True)
      (4): ConvTranspose2d(128, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (5): ReLU(inplace=True)
      (6): ConvTranspose2d(32, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (7): ReLU(inplace=True)
      (8): ConvTranspose2d(16, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (9): Sigmoid()
    )
  )
)
```

- Training epoch : 50
- Learning rate schedule : 初始為 0.001 之後每 15 個 epoch 乘上 0.6
- Data augmentation : 水平垂直翻轉
- Optimizer : Adam

2. learning curve





3. Reconstruct testing image

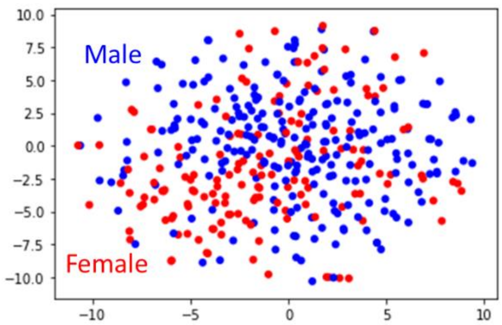
Testing image

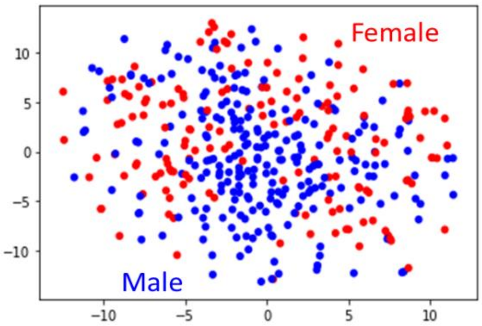
Reconstructed image

MSE							
0.148	0.107	0.096	0.082	0.068	0.081	0.092	0.085
0.077	0.092						

4. Randomly generate images

5. tSNE visualize





6.

VAE 在訓練時收斂速度非常快，但其重建結果較難達到很好的還原效果，還原出來的影像共同特徵較多，也較為模糊，而這樣的結果也可以在進行 tSNE 可視化時看到，VAE 模型可能因為對於特徵提去能力較差，因此難以良好區分不同類別。

Problem2

1.

```
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(64, 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(
    (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()
    )
)
```

- Training epoch : 100
- Learning rate schedule : 初始為 0.001 之後每 15 個 epoch 乘上 0.6
- Data augmentation : 水平垂直翻轉、標準化、亮度對比隨機調整
- Optimizer : Adam

2.Randomly generate images

3.
在 GAN 訓練過程中因為難以用數值及時驗證訓練情形，必須將各階段訓練結果生成圖像。此外在 generator 與 discriminator 分別訓練的過程中要將另一項輸出先進行 detach 避免更新權重時同時更新到另一項。而訓練時放入的雜訊總類中，隨機雜訊訓練結果要比高斯雜訊來的好。最後生成的圖像可以看到會受到一開始訓練資料的變換影響，像是有對比亮度調整、圖片旋轉等。

4.
GAN 在訓練中加入了 discriminator 因此相較於 VAE 訓練過程複雜許多，但也使得 GAN 的訓練結過更佳，在圖片生成的結果中就可看到，VAE 的結果較為模糊，不同影像間共同點也比較多，而 GAN 生成的圖片影像輪廓與差異較為分明。

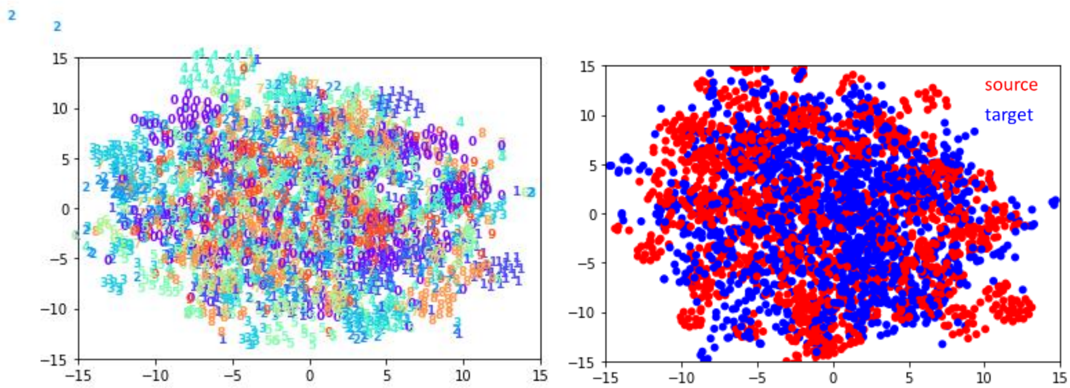
Problem3

1~3.

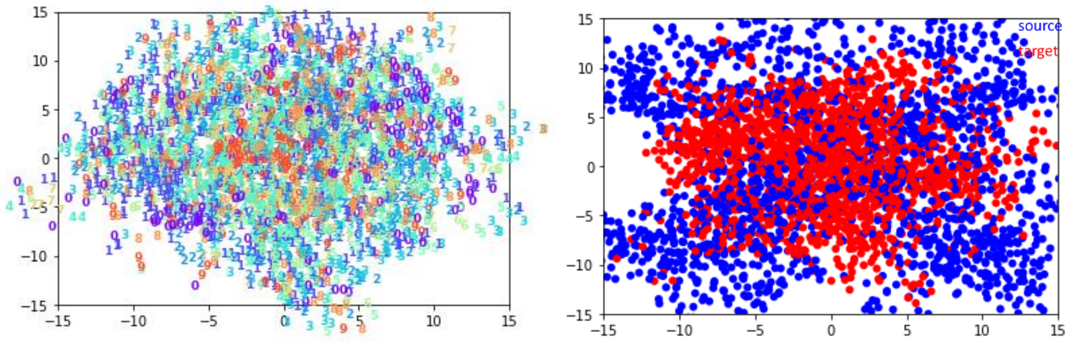
	USPS → MNIST-M	MNIST-M→SVHN	SVHN→USPS
Trained on source	13.67%	19.99%	40.51%
DANN	28.73%	48.2%	51.67%
Trained on target	91.28%	91.49%	96.91%

4.

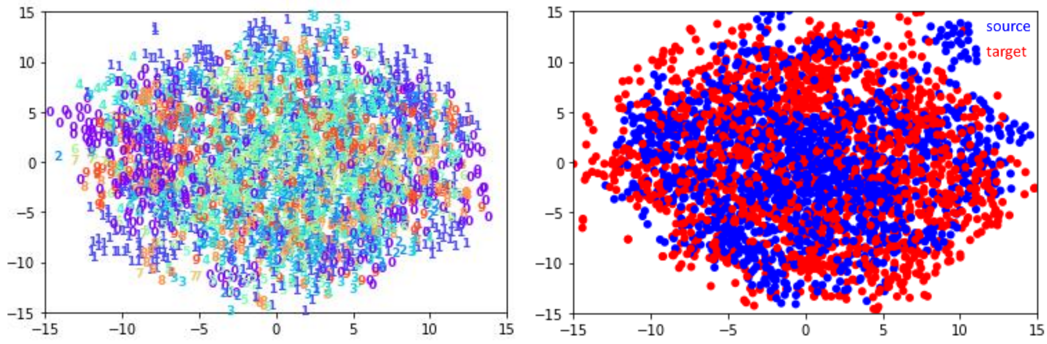
USPS → MNIST-M



MNIST-M→SVHN



SVHN→USPS



5.

```
FeatureExtractor(
  (conv): Sequential(
    (0): Conv2d(3, 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(inplace=True)
    (3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 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(inplace=True)
    (15): AdaptiveAvgPool2d(output_size=(1, 1))
```

```
  )
) Classifier(
  (linear1): Linear(in_features=512, out_features=256, bias=True)
  (relu1): ReLU(inplace=True)
  (linear2): Linear(in_features=256, out_features=10, bias=True)
) Discriminator(
  (layer): Sequential(
    (0): Linear(in_features=512, out_features=256, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): LeakyReLU(negative_slope=0.2)
    (4): Linear(in_features=128, out_features=1, bias=True)
    (5): Sigmoid()
  )
)
```

- Training epoch : 50
- Learning rate schedule : 0.001
- Data augmentation : 標準化{ transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))}
- Optimizer : 皆為 Adam
- 三組資料集都以相同模型進行訓練

6.

在訓練之前需要先將影像轉為彩色，若都轉為黑白彩色影像會損失資訊使得無法訓練，而在上述結果中以黑白資料作為 Source 來訓練彩色資料時，結果會較差。

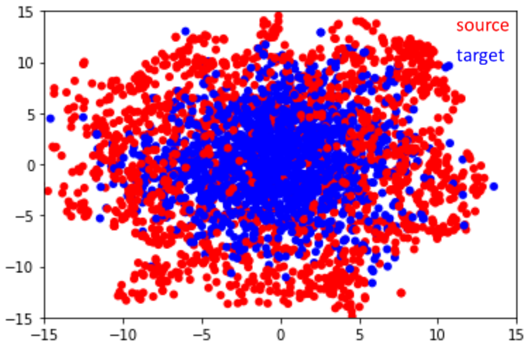
Problem4

1.

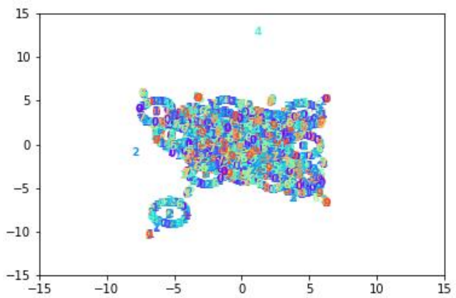
Accuracy on target	USPS → MNIST-M	MNIST-M→SVHN	SVHN→USPS
Improved(??)	10.07	15.938%	13.154

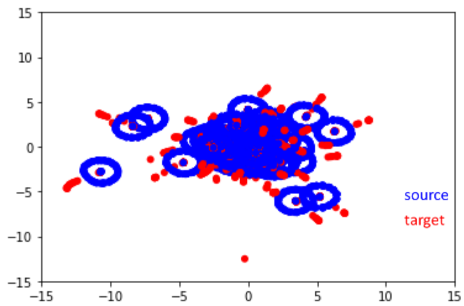
2.

USPS → MNIST-M

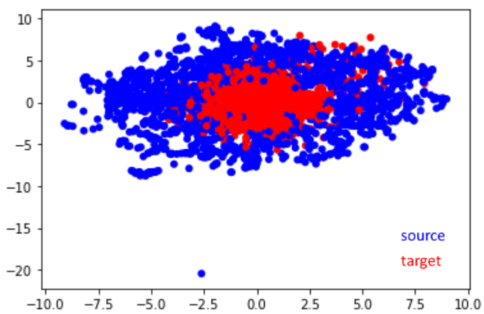


MNIST-M→SVHN





SVHN→USPS



3.

```
LeNetEncoder2(
  (encoder): Sequential(
    (0): Conv2d(3, 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)
)
LeNetClassifier(
```

```
    (fc2): Linear(in_features=500, out_features=10, bias=True)
)
Discriminator(
    (layer): Sequential(
        (0): Linear(in_features=500, out_features=200, bias=True)
        (1): ReLU()
        (2): Linear(in_features=200, out_features=200, bias=True)
        (3): ReLU()
        (4): Linear(in_features=200, out_features=2, bias=True)
        (5): LogSoftmax(dim=None)
    )
)
```

此題使用方法為 Adversarial Discriminative Domain Adaptation，且三個資料集也是用同樣模型進行訓練

- Training epoch : discriminator :20 ; classifier : 20
- Learning rate schedule : 0.005
- Optimizer :　皆為 Adam

4.

本題利用 ADDA 來做為不同的 UDA 訓練方式，其主要概念為先利用 source domain 訓練自身的分類器，再將訓練好的 source model 結果與 target domain 一起訓練 discriminator，與 target domain 自身的特徵提取網路，同時在訓練過程中變換 target domain 分類來混淆 discriminator 提升訓練效果，而這樣過程相較於 DANN，ADDA 能夠使得不同來源得資料能有不同的特徵提取能力，同時又能去除兩者的種類差異，來達到更好的自身分類效果，但本題無充裕時間作答，因此訓練效果十分不佳 :(.

參考資料

1. https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/variational_autoencoder.ipynb#scrollTo=mZaVrj0hX1ry
2. https://github.com/atinghosh/VAE-pytorch/blob/master/VAE_celeba.py
3. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
4. https://github.com/Yangyangii/DANN-pytorch/blob/master/DANN.ipynb
5. https://github.com/corenel/pytorch-adda