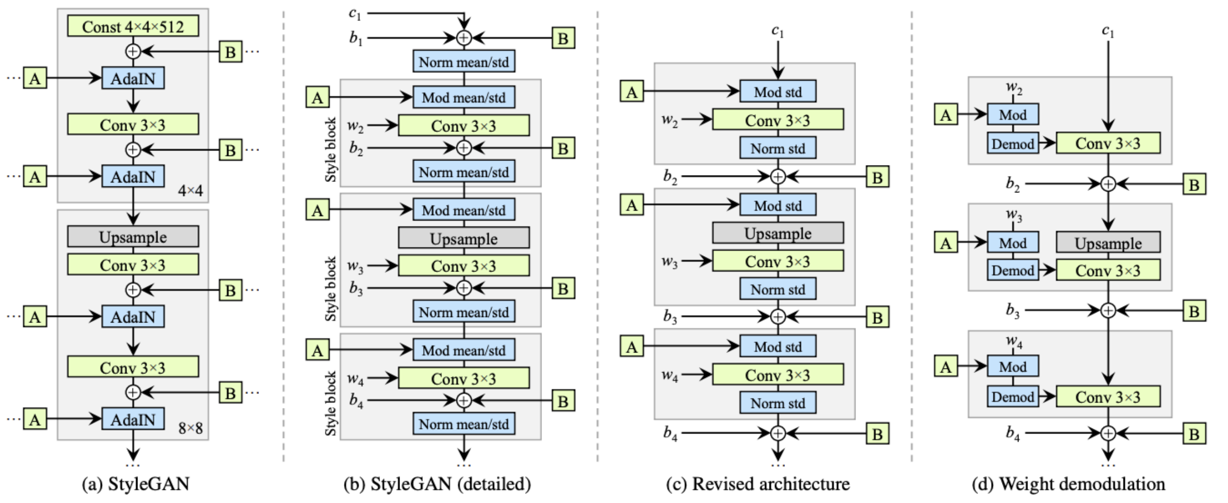
DLCV HW2

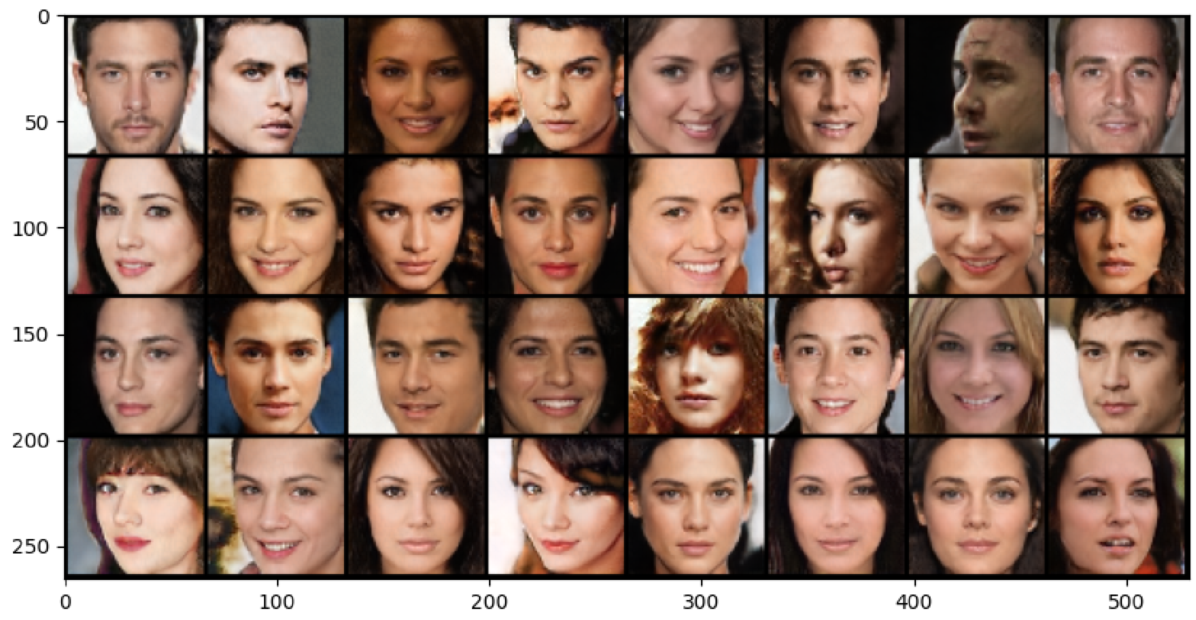
P1

訂閱

1. Model Architecture



2. Show the first 32 images in your report.(5%)



3. Evaluate your 1000 generated images by implementing two metrics:

FID Score: 27.9
IS Score: 2.13

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

GAN is really hard to train, thankfully, TA has lowered the baselines.

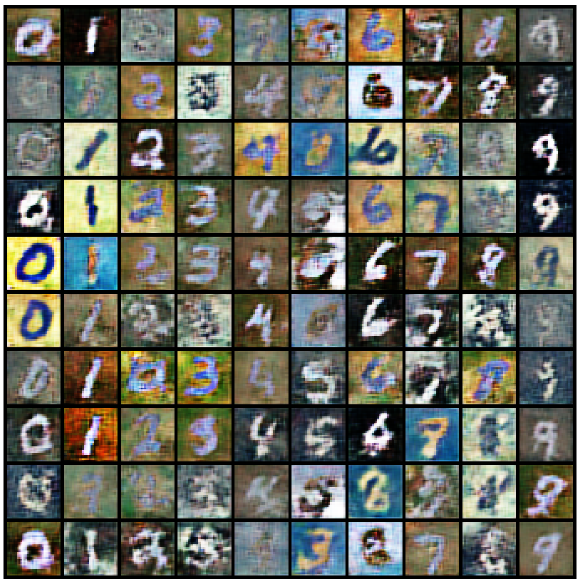
P2

1. ACGAN model architecture. (10%)

```
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)
)
)
```

5.Show 10 images for each digit (0-9).(5%)

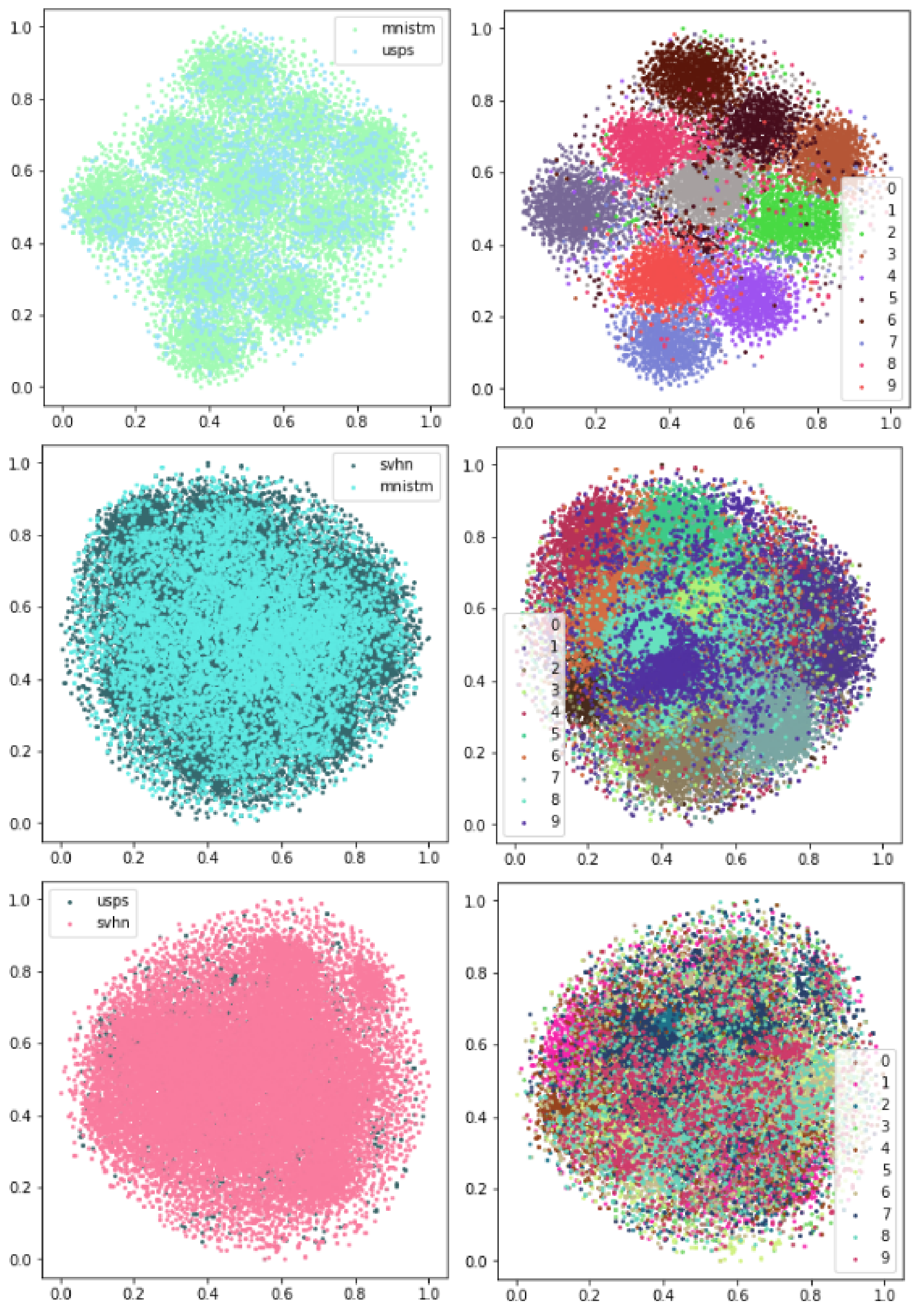


P3

123. Baseline

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Trained on source	64.72%	51.34%	20.4%
Adaptation(DANN/Improved)	86.8%	62%	28.45%
Trained on target	97.56%	97.62%	91.47%

4. TSNE



5. Describe the implementation details of your model and discuss what you’ve observed and learned from implementing DANN. (7%)

The task from USPS to SVHN (adaption) is difficult to train, for USPS are black and white images, but SVHN are images with colors. Therefore, data augmentation is crucial in this task.
The task from MNIST-M to USPS can see better results that models training on source, for MNIST-M dataset are images with colors and it contains more information than USPS does.

Bonus

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Adaptation(Original)	82%	50%	25%
Adaptation(Improved)	84.75%	51.8%	34.36%

Note:
These are results on the training iterations=30000, but the results on problem 3-2 are on the training iterations=50000, so the performance may not be better than the results of problem 3-2. Actually, on my test, the models which are fixed on iterations=30000, the adaptions' accuracy of problem 3-2 are 82%, 50%, 25%, respectively,problem 3-2.Obviously, the modified results are better than those of probelm 3-2 which are fixed on iterations=30000.

Implementation:

1. Loss function 為 \(L = L_{Task} + αL_{recon} + βL_{different} + γL_{similarity}\)，選擇 α = 0.015， β = 0.075，γ = 0.25; Optimizer 使用 Adam(lr=1e-3, betas=(0.9, 0.999))，batch size=32。
2. USPS→SVHN在原本的data上比較難train，導致accuracy很低，需要加上 colorjitter，accuracy 才會比較高

Reference:

1. https://github.com/lucidrains/stylegan2-pytorch (https://github.com/lucidrains/stylegan2-pytorch)
2. https://github.com/eriklindernoren/PyTorch-GAN (https://github.com/eriklindernoren/PyTorch-GAN)
3. https://github.com/kai860115/DLCV2020-FALL/tree/main/hw3 (https://github.com/kai860115/DLCV2020-FALL/tree/main/hw3)

Collaborator:

b07502071 陳志臻 b08902134 曾揚哲