Deep Learning for Computer Vision - Homework 2

資工碩一 R10922005 李澤諺

November 23, 2021

Problem 1: GAN

以下爲我於本次作業中所實作的 DCGAN 架構：

```
DCGAN(
  (generator): DCGAN_generator(
    (linear): Sequential(
      (0): Linear(in_features=100, out_features=8192, bias=False)
      (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (deconvolution): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Dropout2d(p=0.2, inplace=False)
      (4): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Dropout2d(p=0.2, inplace=False)
      (8): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (9): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (10): ReLU()
      (11): Dropout2d(p=0.2, inplace=False)
      (12): ConvTranspose2d(64, 3, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
      (13): Tanh()
    )
  )
  (discriminator): DCGAN_discriminator(
    (convolution): Sequential(
      (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): LeakyReLU(negative_slope=0.2)
      (2): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (4): LeakyReLU(negative_slope=0.2)
      (5): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): LeakyReLU(negative_slope=0.2)
      (8): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (10): LeakyReLU(negative_slope=0.2)
      (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
      (12): Sigmoid()
    )
  )
)
```

其中，我將所有的 convolutional layer 和 transposed convolutional layer 的 weight 以 mean 爲 0 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，並將所有的 batch normalization layer 的 weight 以 mean 爲 1 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，而 batch normalization layer 的 bias 則皆 initialize 爲 0。接著，我將 training dataset 中所有的圖片其 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。最後，我使用了 Adam 作爲 optimizer，並使用 binary cross entropy 作爲 loss function，以此訓練 DCGAN，其中 input noise vector 的 dimension 爲 100，batch size 爲 64，generator 和 discriminator 的 learning rate 皆爲 0.0001，β_1 和 β_2 分別爲 0.5 和 0.999，訓練了 300 個 epoch，以此得到最後的 model。以下爲我訓練出來的 DCGAN 所生成的 32 張人臉圖片：

以下爲我訓練出來的 DCGAN 所得到的 FID 和 IS：

FID	IS
28.642903384558053	2.070996595831873

我有試過使用其它的 model architecture，例如 WGAN、WGAN-GP、SNGAN、SAGAN 等等，發現較爲複雜的 model 其收斂所需的 epoch 數比較少，訓練過程也較爲穩定，但不知道是不是因爲我的訓練方式有誤，例如使用了不適當的 optimizer、loss function、hyper-parameter 等等，所以有時會訓練不起來。此外，助教所提供的 tip，例如在 model 的各個 layer 中加入 noise、加入 dropout、觀察 generator 和 discriminator 的 performance 是否平衡等等，這些 tip 對我在訓練 DCGAN 時有莫大的幫助，讓我的 DCGAN 能產生更好的結果。

Problem 2: ACGAN

以下爲我於本次作業中所實作的 ACGAN 架構：

```
ACGAN(
  (generator): ACGAN_generator(
    (linear): Sequential(
      (0): Linear(in_features=110, out_features=8192, bias=False)
      (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (deconvolution): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Dropout2d(p=0.2, inplace=False)
      (4): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Dropout2d(p=0.2, inplace=False)
      (8): ConvTranspose2d(128, 3, kernel_size=(4, 4), stride=(2, 2), padding=(3, 3), bias=False)
      (9): Tanh()
    )
  )
  (discriminator): ACGAN_discriminator(
    (convolution): Sequential(
      (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
      (3): Dropout2d(p=0.2, inplace=False)
      (4): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): LeakyReLU(negative_slope=0.2)
      (7): Dropout2d(p=0.2, inplace=False)
      (8): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (10): LeakyReLU(negative_slope=0.2)
      (11): Dropout2d(p=0.2, inplace=False)
      (12): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (13): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (14): LeakyReLU(negative_slope=0.2)
      (15): Dropout2d(p=0.2, inplace=False)
      (16): Conv2d(512, 1024, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    )
    (discriminator): Sequential(
      (0): Linear(in_features=1024, out_features=1, bias=True)
      (1): Sigmoid()
    )
    (classifier): Linear(in_features=1024, out_features=10, bias=True)
  )
)
```

其中，我將所有的 convolutional layer 和 transposed convolutional layer 的 weight 以 mean 爲 0 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，並將所有的 batch normalization layer 的 weight 以 mean 爲 1 且 standard deviation 爲 0.02 的 normal distribution 進行 initialization，而 batch normalization layer 的 bias 則皆 initialize 爲 0。接著，我將 training dataset 中所有的圖片其 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。最後，我使用了 Adam 作爲 optimizer，並使用 cross entropy 作爲 loss function，以此訓練 ACGAN，其中 input noise vector 的 dimension 爲 100，並且其會和 label 所轉成的 one-hot vector 做 concatenate 得到 dimension 爲 110 的 vector，以此作爲 ACGAN 的 input，而 batch size 爲 64，learning rate 皆爲 0.0001，β_1 和 β_2 分別爲 0.5 和 0.999，訓練了 300 個 epoch，以此得到最後的 model。

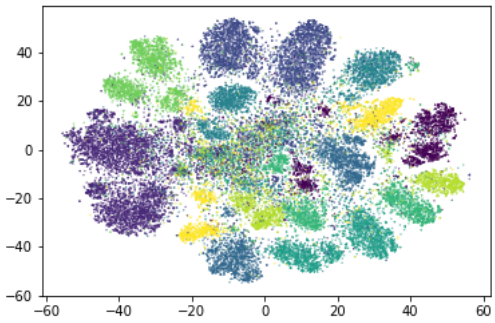
我訓練出來的 ACGAN 所產生的 1000 張數字圖片，經由助教所提供的 classifier 進行辨識之後，所得到的 accuracy 爲 0.922，而以下爲我訓練出來的 ACGAN 所生成的數字圖片：

Problem 3: DaNN

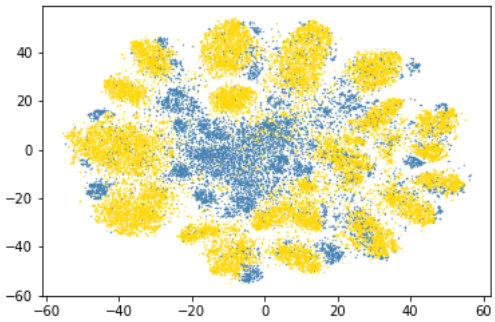
以下爲我訓練出來的 DaNN 所得到的 accuracy：

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Trained on source	90.818%	46.980%	25.161%
Adaptation	91.779%	49.520%	32.195%
Trained on target	95.815%	98.820%	95.594%

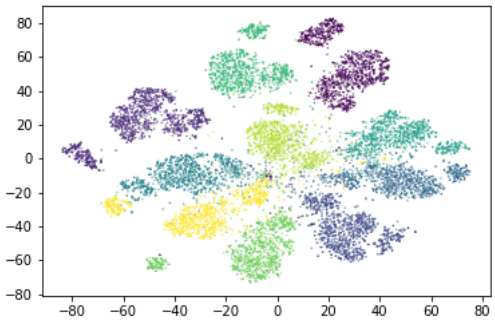
以下爲 DaNN 將圖片轉爲 feature vector 並經由 t-SNE 降到 2 維之後所得到的圖片:



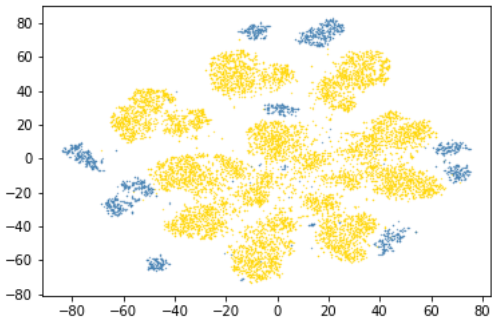
SVHN 和 MNIST-M (顏色按照 digit 區分)



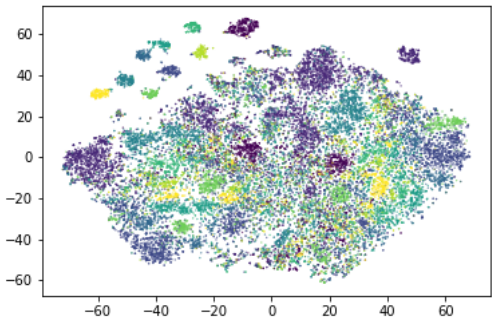
SVHN 和 MNIST-M (顏色按照 domain 區分)



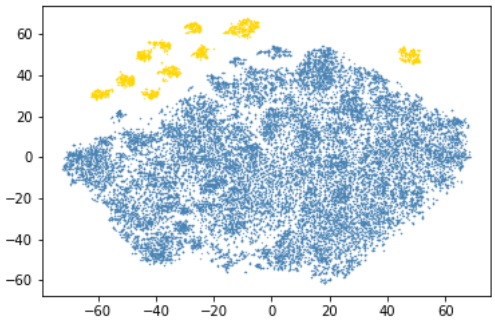
MNIST-M 和 USPS (顏色按照 digit 區分)



MNIST-M 和 USPS (顏色按照 domain 區分)



USPS 和 SVHN (顏色按照 digit 區分)



USPS 和 SVHN (顏色按照 domain 區分)

以下爲我實作的 DaNN 架構:

```
DaNN(
  (feature_extractor): FeatureExtractor(
    (convolution): 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, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (5): BatchNorm2d(128, 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(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (9): BatchNorm2d(256, 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(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()
      (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
  )
  (label_predictor): LabelPredictor(
    (linear): 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): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=512, out_features=512, bias=True)
      (5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Dropout(p=0.5, inplace=False)
      (8): Linear(in_features=512, out_features=10, bias=True)
    )
  )
  (domain_classifier): DomainClassifier(
    (linear): 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): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=512, out_features=512, bias=True)
      (5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Dropout(p=0.5, inplace=False)
      (8): Linear(in_features=512, out_features=1, bias=True)
      (9): Sigmoid()
    )
  )
)
```

我將 source dataset 和 target dataset 中所有的圖片都經過 RandomAffine 和 ColorJitter 進行 data augmentation，並將圖片中 pixel 的值 normalize 到 -1 和 1 之間，以此進行 data preprocessing。此外，我使用了 Adam 作爲 optimizer，並使用 cross entropy 作爲 loss function，以此訓練 DaNN，其中 learning rate 皆爲 0.0001，batch size 皆爲 64，訓練了 15 個 epoch，以此得到最後的 model。此外，當 source dataset 爲 USPS 時，我會將 source dataset 和 target dataset 中的圖片皆轉爲 grayscale，並將 target dataset 中的圖片經過 Otsu thresholding，以此讓 source dataset 和 target dataset 中的圖片外觀更爲接近，因爲在訓練過程中，我發現如果 source dataset 和 target dataset 的圖片外觀如果相差過多，DaNN 仍然無法學到兩個 dataset 共有的 feature，以此達到 classification。

Bonus: Improved UDA model

我實作了 MCD [ref]，以下爲其架構:

```
MCD(
  (generator): Generator(
    (conv1): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(64, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (fc1): Linear(in_features=6272, out_features=3072, bias=True)
    (bn4): BatchNorm1d(3072, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (classifier_1): Classifier(
    (fc1): Linear(in_features=3072, out_features=2048, bias=True)
    (bn1): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (fc2): Linear(in_features=2048, out_features=10, bias=True)
  )
  (classifier_2): Classifier(
    (fc1): Linear(in_features=3072, out_features=2048, bias=True)
    (bn1): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (fc2): Linear(in_features=2048, out_features=10, bias=True)
  )
)
```

以下爲我訓練出來的 MCD 所得到的 accuracy：

	MNIST-M → USPS	SVHN → MNIST-M	USPS → SVHN
Original model	91.779%	49.520%	32.195%
Improved model	94.818%	65.370%	32.971%

我將 source dataset 和 target dataset 中所有的圖片都經過 RandomAffine 和 ColorJitter 進行 data augmentation，並將圖片中 pixel 的值 normalize 到 −1 和 1 之間，以此進行 data preprocessing。此外，我使用了 Adam 作爲 optimizer，並使用 cross entropy 作爲 loss function，以此訓練 MCD，其中 learning rate 皆爲 0.00002，weight decay 爲 0.0005，batch size 皆爲 128，訓練了 100 個 epoch，以此得到最後的 model。此外，當 source dataset 爲 USPS 時，我會將 source dataset 和 target dataset 中的圖片皆轉爲 grayscale，並將 target dataset 中的圖片經過 Otsu thresholding，以此讓 source dataset 和 target dataset 中的圖片外觀更爲接近，因爲在訓練過程中，我發現如果 source dataset 和 target dataset 的圖片外觀如果相差過多，MCD 仍然無法學到兩個 dataset 共有的 feature，以此達到 classification。