Name: 李華健 Dep.: 電機三 Student ID:B05901119

Reference:

1. GAN https://github.com/eriklindernoren/PyTorch-GAN
2. DANN https://github.com/fungtion/DANN
3. ADDA https://github.com/corenel/pytorch-adda

## Problem 1 GAN:

1. Describe the architecture & implementation details of your model

Implementation Details:

Optimizer: Adam, Learning Rate: 2e-5, Betas(0.5, 0.999)
Epochs: 200
Batch_size: 64
Loss Function: BCELoss()

```
Generator(z): z.shape = (100)
  ConvTranspose2d(100, 512, kernel_size=4)
  BatchNorm2d(512)
  ReLU(),
  ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(256),
  ReLU(),
  ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(128),
  ReLU(),
  ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(64),
  ReLU(),
  ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
  Tanh()
```

```
Discriminator(image):
    Conv2d(3, 64, kernel_size=4, stride=2, padding=1, bias=False),
    LeakyReLU(0.2),
    Conv2d(64, 128, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(128),
    LeakyReLU(0.2),
    Conv2d(128, 256, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(256),
    LeakyReLU(0.2),
    Conv2d(256, 512, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(512),
    LeakyReLU(0.2),
    Conv2d(512, 1, kernel_size=4, stride=1, padding=0, bias=False),
    Sigmoid()
```

2. Plot 32 random images generated from your model. [fig1_2.jpg]

3. Discuss what you're observed and learned from implementing GAN.

- For my result, about 25% images of 32 are blurred. I think my model preforms bad…

- The images come from the Generator being more and more clear in every epoch. But the model sometimes divergence when I set a higher learning rate.

## Problem 2 ACGAN:

1. Describe the architecture & implementation details of your model

```
Generator(z):
    Linear(102, 512 * 4 * 4) -> Reshape(512, 4, 4)
    BatchNorm2d(512)
    ReLU()
    ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(256),
    ReLU(),
    ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(128),
    ReLU(),
    ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
    BatchNorm2d(64),
    ReLU(),
    ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
    Tanh()
```

```
Discriminator(image):
    Conv2d(3, 64, kernel_size=5, stride=2, padding=2),
    LeakyReLU(0.2),
    Conv2d(64, 128, kernel_size=5, stride=2, padding=2),
    BatchNorm2d(128),
    LeakyReLU(0.2),
    Conv2d(128, 256, kernel_size=5, stride=2, padding=2),
    BatchNorm2d(256),
    LeakyReLU(0.2),
    Conv2d(256, 512, kernel_size=5, stride=2, padding=2),
    BatchNorm2d(512),
    LeakyReLU(0.2),
```

```
Classifier_adversarial():
    Linear(512 * 4 * 4, 1)
    Sigmoid()
```

```
Classifier_axuiliary():
    Linear(512 * 4 * 4, 2)
    Softmax()
```

Implementation Details:
Optimizer: Adam, Learning Rate: 2e-5, Betas(0.5, 0.999)
Epochs: 200
Batch_size: 64
Loss Function: CrossEntropyLoss()

2. Plot 10 random pairs of generated images from your model, where each pair should be generated from the same random vector input but with opposite attribute. This is to demonstrate your model's ability to disentangle features of interest. [fig2_2.jpg]

3. Discuss what you've observed and learned from implementing ACGAN

    - Also, The images come from the Generator being more and more clear in every epoch. But the model sometimes divergence when I set a higher learning rate.

    - When the model is divergence, the generator loss and discriminator loss is very very high (>10 at divergent and normally <1). Check the learning rate setting and retrain from 0.

## Problem 3 DANN:

1. Accuracy

| | usps -> mnistm | mnistm -> svhn | svhn -> usps |
|---|---|---|---|
| Source Only | 35.19% | 43.67% | 67.53% |
| DANN | 35.80% | 49.95% | 67.16% |
| Target Only | 98.13% | 93.49% | 96.97% |

The accuracy of 'source only' is chosen from the maximum validation accuracy in training epochs, not converge accuracy. DANN learns the predict model with a **more stable** way and **higher accuracy**.

2. 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
   b) diff domains

3. Describe the architecture & implementation detail of your model

```
Feature_Extractor(z):
    Conv2d(3, 64, kernel_size=5, padding=2),
    BatchNorm2d(64),
    ReLU(),
    Dropout2d(0.5),
    MaxPool2d(kernel_size=2, stride=2),
    Conv2d(64, 64, kernel_size=5, padding=2),
    BatchNorm2d(128),
    ReLU()
    Dropout2d(0.5),
    MaxPool2d(kernel_size=2, stride=2),
    Conv2d(64, 128, kernel_size=5, padding=2),
    BatchNorm2d(128),
    ReLU()
    Dropout2d(0.5),
```

```
Class_Classifier(feature)
    Linear(128 * 7 * 7, 4096),
    BatchNorm(4096)
    ReLU(),
    Dropout(0.5),
    Linear(4096, 2048),
    ReLU(inplace=True),
    Dropout(0.5),
    Linear(2048, 10),
```

```
Domain_Classfier(feature)
    Linear(128 * 7 * 7, 1024),
    BatchNorm(1024)
    ReLU(0.2),
    Dropout(0.5)

    Linear(1024, 1024),
    BatchNorm(1024)
    ReLU(0.2),
    Dropout(0.5)

    Linear(1024, 2),
```

Implementation Details:

Optimizer: Adam, Learning Rate: 2e-5, Betas(0.5, 0.999)
Epochs: 200
Batch_size: 64
Loss Function: CrossEntropyLoss()

4. Discuss what you've observed and learned from implementing DANN.
   - Seems I haven't trained converge enough in source domain training test. In DANN training, it's assumed that the distribution of source domain and target domain is different. So a model is fitted to one domain may hurt the accuracy for another domain (the accuracy and loss usually oscillate when training on source domain)

   - To improve the accuracy in target domain, DANN choose to use an shared feature extractor but it seems decrease the performance in source domain a little bit. By adding the domain classifier, the prediction is higher and more stable in target domain.

## Problem 4 Improved UDA model:

1. Accuracy

| | usps -> mnistm | mnistm -> svhn | svhn -> usps |
|---|---|---|---|
| Source Only | 35.19% | 43.67% | 67.53% |
| DANN | N/A | N/A | N/A |
| Target Only | 98.13% | 93.49% | 96.97% |

2. 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
   b) diff domains

3. Describe the architecture & implementation detail of your model

```
Generator(z):
  Linear(102, 512 * 4 * 4),
  Reshape(512, 4, 4)
  BatchNorm2d(512)
  ReLU()
  ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(256),
  ReLU(),
  ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(128),
  ReLU(),
  ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(64),
  ReLU(),
  ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
  Tanh()
```


```
Generator(z):
  Linear(102, 512 * 4 * 4),
  Reshape(512, 4, 4)
  BatchNorm2d(512)
  ReLU()
  ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(256),
  ReLU(),
  ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(128),
  ReLU(),
  ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(64),
  ReLU(),
  ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
  Tanh()
```

```
Generator(z):
  Linear(102, 512 * 4 * 4),
  Reshape(512, 4, 4)
  BatchNorm2d(512)
  ReLU()
  ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(256),
  ReLU(),
  ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(128),
  ReLU(),
  ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
  BatchNorm2d(64),
  ReLU(),
  ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
  Tanh()
```

4. Discuss what you've observed and learned from implementing your improved UDA model.
   - I have tried ADDA as my improved model but it performs bad.
   - ADDA train 2 encoder, 1 for source and 1 for target domain. The main idea of ADDA is matching the feature from 2 domains as the same distribution.
   - DANN with fine tune and more neurons, more layers is submitted.