黃政維

R09521603

HW2

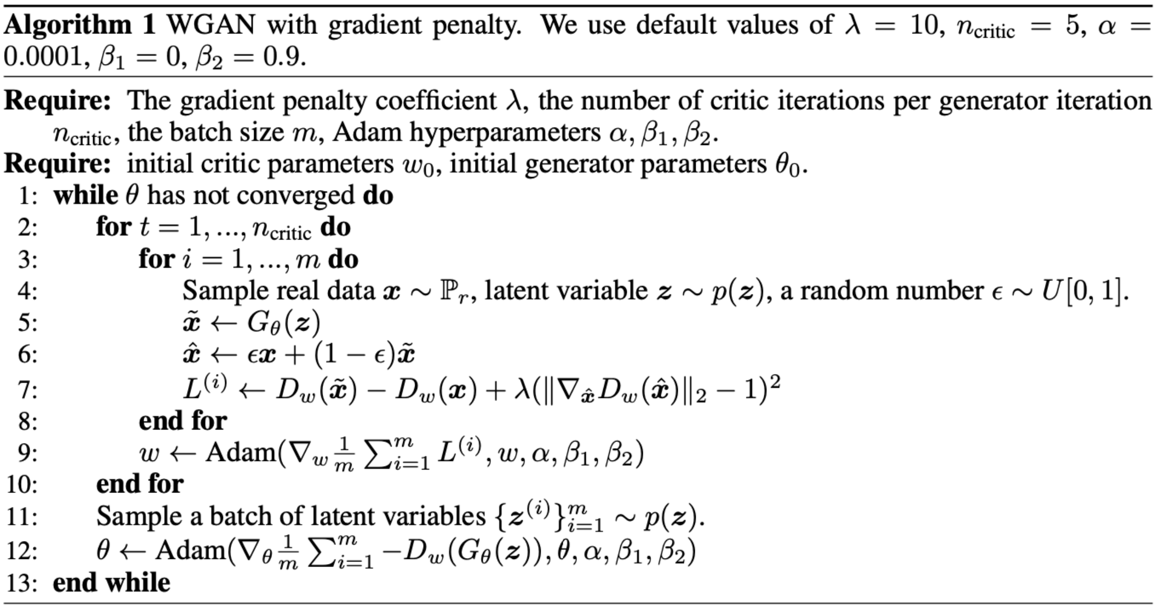
1

1.

```
Generator(
 (l1): Sequential(
  (0): Linear(in_features=128, out_features=8192, bias=False)
  (1): LayerNorm((8192,), eps=1e-05, elementwise_affine=True)
  (2): ReLU()
 )
 (l2_5): Sequential(
  (0): Sequential(
   (0): ConvTranspose2d(128, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
   (1): LayerNorm((512, 4, 4), eps=1e-05, elementwise_affine=True)
   (2): ReLU()
  )
  (1): Sequential(
   (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
   (1): LayerNorm((256, 8, 8), eps=1e-05, elementwise_affine=True)
   (2): ReLU()
  )
  (2): Sequential(
   (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
   (1): LayerNorm((128, 16, 16), eps=1e-05, elementwise_affine=True)
   (2): ReLU()
  )
```

```
    (3): Sequential(
      (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (1): LayerNorm((64, 32, 32), eps=1e-05, elementwise_affine=True)
      (2): ReLU()
    )
    (4): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (5): Tanh()
  )
)
Discriminator(
  (ls): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Sequential(
      (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (1): LayerNorm((128, 16, 16), eps=1e-05, elementwise_affine=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (3): Sequential(
      (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (1): LayerNorm((256, 8, 8), eps=1e-05, elementwise_affine=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (4): Sequential(
      (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (1): LayerNorm((512, 4, 4), eps=1e-05, elementwise_affine=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
  )
)
```

I implement the WGAN-GP, which is same as GAN but modify the output of discriminator (remove the sigmoid) and how to calculate the loss.

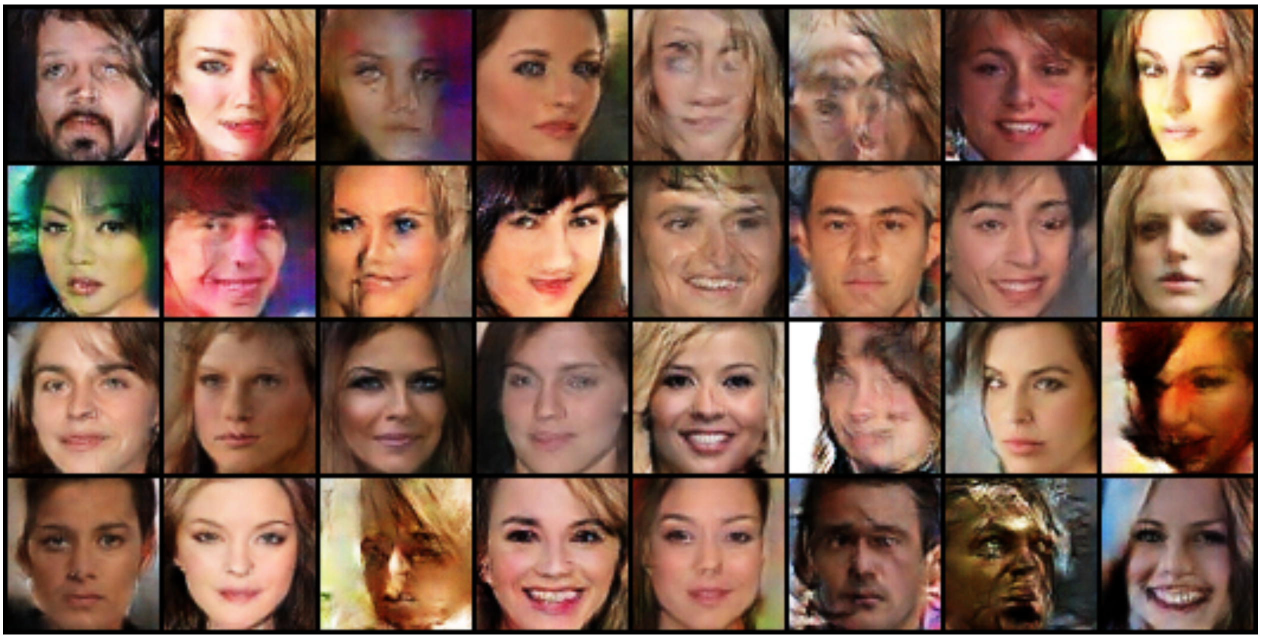


Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
while $\theta$ has not converged do
    for $t = 1, ..., n_{\text{critic}}$ do
        for $i = 1, ..., m$ do
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while
```

I use the default hyperparameter, and add the number of critic to 20 and lower the learning rate to 1e-5 to fine tune the model.

2.



3.

FID : 29.53715015572601

IS : 2.027483335974662

5.

I first use the WGAN to train the model , and i found out i need more epoch to get the output of face compare to simple GAN. When i use WGAN-GP it became more epoch to get good result. But i tried to increase the training batch_size and i will receive more confident result.

2

1.

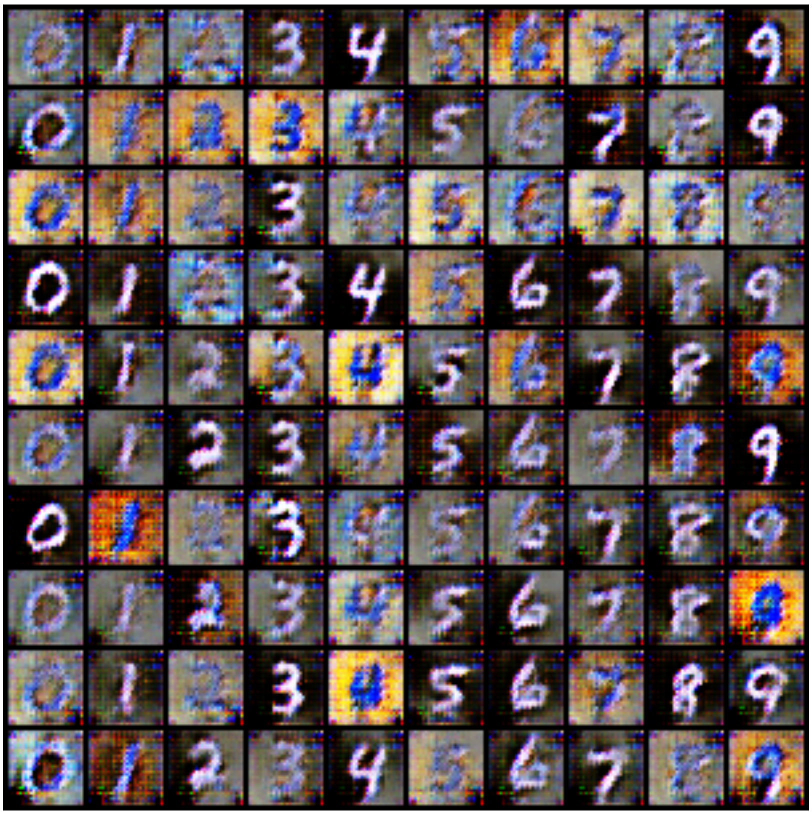
```
ACGenerator(
  (l1): Sequential(
    (0): Linear(in_features=110, out_features=25088, bias=False)
    (1): LayerNorm((25088,), eps=1e-05, elementwise_affine=True)
    (2): ReLU()
  )
  (l2_5): Sequential(
    (0): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (1): LayerNorm((256, 14, 14), eps=1e-05, elementwise_affine=True)
      (2): ReLU()
    )
    (1): ConvTranspose2d(256, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (2): Tanh()
  )
)
ACDiscriminator(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=256, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=10, bias=True)
  (ls_label): Sequential(
    (0): Linear(in_features=64, out_features=10, bias=True)
  )
  (ls_discrem): Sequential(
    (0): Linear(in_features=64, out_features=1, bias=True)
  )
)
```

Basically i use same model as p1. But i first one hot encode the label and concats it with the random feature z. And i replace my discriminator to TA's classifier, but without loading the pertained weight.

2.3.4

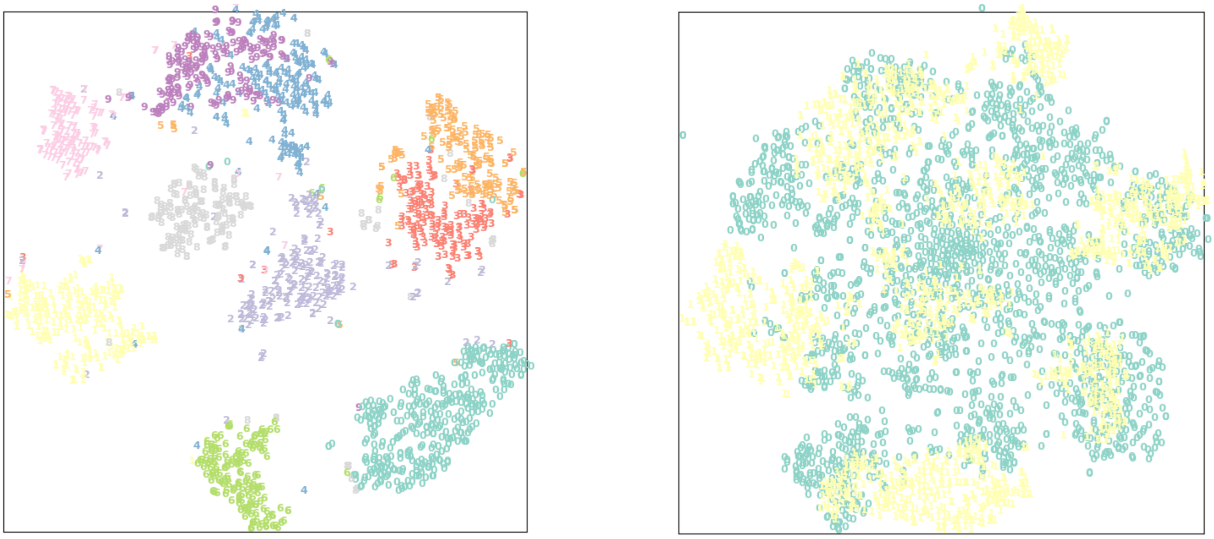
Accuracy : 0.92734

5.

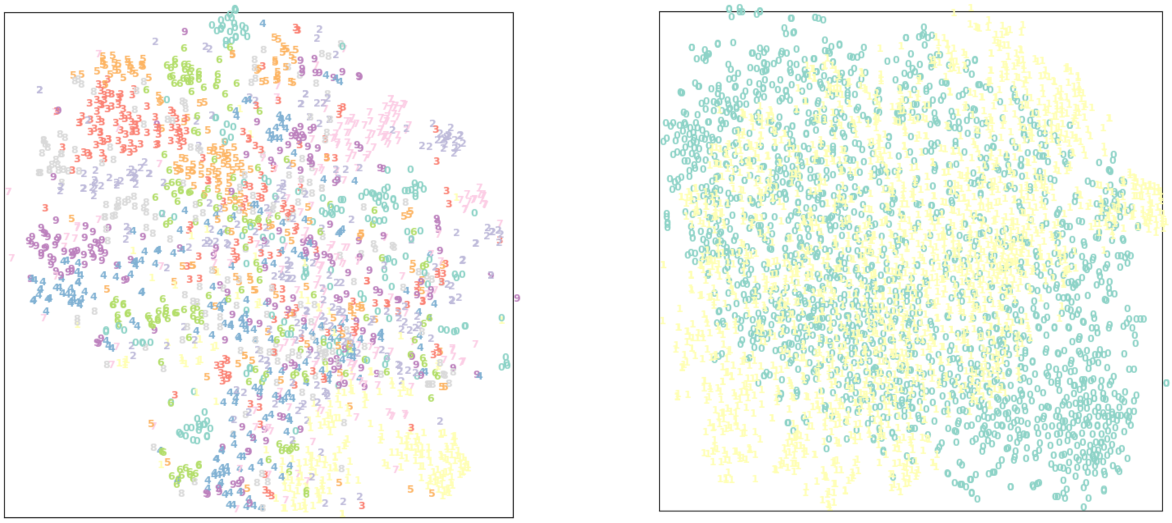


3

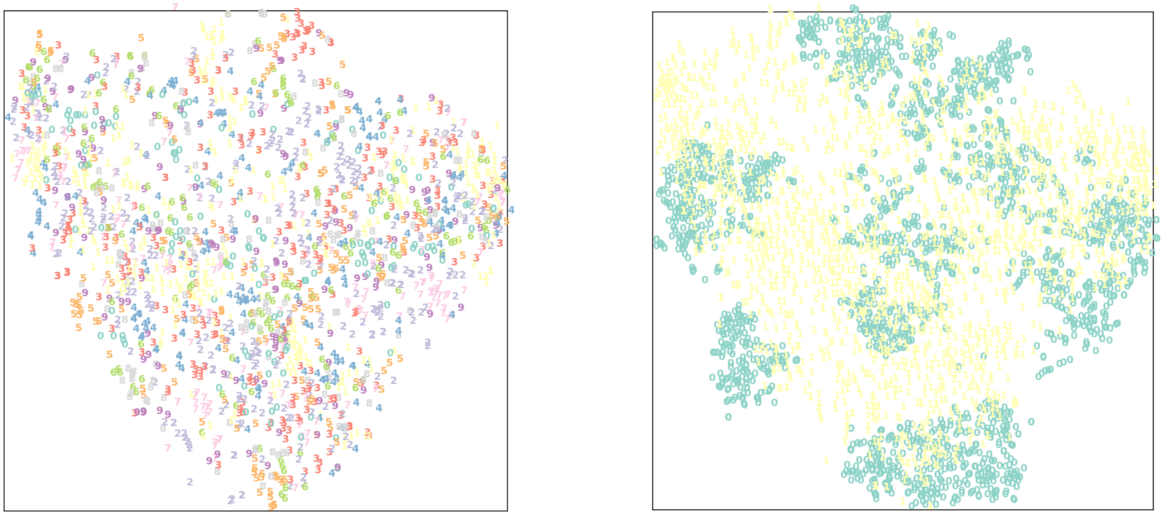
	MNIST-M -> USPS	SVHN -> MNIST-M	USPS -> SVHM
Trained on source	0.78479	0.80576	0.96096
Adaptation (DANN/Improved)	0.41419	0.49648	0.96933
Trained on target	0.19100	0.28008	0.86238



MNIST-M -> USPS



SVHN -> MNIST-M



USPS -> SVHN

Model Implement

```
DANNModel(
 (FeatureExtract): DANNFeatur(
  (conv1): Conv2d(3, 6, kernel_size=(3, 3), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(3, 3), stride=(1, 1))
  (conv3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (convs): Sequential(
   (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(1, 1))
   (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (3): ReLU()
   (4): Conv2d(64, 50, kernel_size=(5, 5), stride=(1, 1))
   (5): Dropout2d(p=0.5, inplace=False)
   (6): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (8): ReLU()
  )
 )
 (classifier): DANNClass(
  (fc2): Linear(in_features=256, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=64, bias=True)
  (fc4): Linear(in_features=64, out_features=10, bias=True)
  (classifier): Sequential(
   (0): Linear(in_features=800, out_features=100, bias=True)
   (1): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): ReLU()
   (3): Dropout2d(p=0.5, inplace=False)
```

```
      (4): Linear(in_features=100, out_features=100, bias=True)
      (5): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): ReLU()
      (7): Linear(in_features=100, out_features=10, bias=True)
    )
  )
  (domainClassifier): DANNDomain(
    (fc1): Linear(in_features=32, out_features=256, bias=True)
    (fc2): Linear(in_features=256, out_features=128, bias=True)
    (fc3): Linear(in_features=128, out_features=64, bias=True)
    (fc4): Linear(in_features=64, out_features=2, bias=True)
    (domainer): Sequential(
      (0): Linear(in_features=800, out_features=100, bias=True)
      (1): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Linear(in_features=100, out_features=2, bias=True)
    )
  )
)
```

I implement the DANN just like professor show in the course. I have implement an inverse layer to inverse the features gradient to minus with dynamic lambda. Don’t know why my tsne is not good as my prediction accuracy.