Report

Problem 1

1. Please print the model architecture of method A and B.
    - difference
        - model: DCGAN -> SN-GAN spectral_normalization (nn.utils.spectral_norm)
        - optimizer: Adam -> AdamW
        - label smoothing
        - image noise
        - learning rate
    - model A
        - generator

```
DCGAN_Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 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()
  )
)

----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         819,200
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
              ReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
```

```
              ReLU-6          [-1, 256, 8, 8]               0
   ConvTranspose2d-7        [-1, 128, 16, 16]         524,288
       BatchNorm2d-8        [-1, 128, 16, 16]             256
              ReLU-9        [-1, 128, 16, 16]               0
  ConvTranspose2d-10         [-1, 64, 32, 32]         131,072
      BatchNorm2d-11         [-1, 64, 32, 32]             128
             ReLU-12         [-1, 64, 32, 32]               0
  ConvTranspose2d-13          [-1, 3, 64, 64]           3,072
             Tanh-14          [-1, 3, 64, 64]               0
================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 3.00
Params size (MB): 13.64
Estimated Total Size (MB): 16.64
----------------------------------------------------------------
```

- discriminator

```
DCGAN_Discriminator(
              (feature_extract): 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()
              )
            )
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3,072
         LeakyReLU-2           [-1, 64, 32, 32]               0
            Conv2d-3          [-1, 128, 16, 16]         131,072
       BatchNorm2d-4          [-1, 128, 16, 16]             256
         LeakyReLU-5          [-1, 128, 16, 16]               0
```

```
          Conv2d-6            [-1, 256, 8, 8]         524,288
     BatchNorm2d-7            [-1, 256, 8, 8]             512
       LeakyReLU-8            [-1, 256, 8, 8]               0
          Conv2d-9            [-1, 512, 4, 4]       2,097,152
    BatchNorm2d-10            [-1, 512, 4, 4]           1,024
      LeakyReLU-11            [-1, 512, 4, 4]               0
         Conv2d-12              [-1, 1, 1, 1]           8,192
        Sigmoid-13              [-1, 1, 1, 1]               0
================================================================
Total params: 2,765,568
Trainable params: 2,765,568
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.31
Params size (MB): 10.55
Estimated Total Size (MB): 12.91
----------------------------------------------------------------
```

- model B
  - generator

```
My_Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 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()
  )
)

----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         819,200
```

```
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
              ReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
              ReLU-6            [-1, 256, 8, 8]               0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524,288
       BatchNorm2d-8          [-1, 128, 16, 16]             256
              ReLU-9          [-1, 128, 16, 16]               0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131,072
      BatchNorm2d-11           [-1, 64, 32, 32]             128
             ReLU-12           [-1, 64, 32, 32]               0
  ConvTranspose2d-13            [-1, 3, 64, 64]           3,072
             Tanh-14            [-1, 3, 64, 64]               0
================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 3.00
Params size (MB): 13.64
Estimated Total Size (MB): 16.64
----------------------------------------------------------------
```

- discriminator

```
My_Discriminator(
  (feature_extract): 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()
  )
)
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3,072
```

```
        LeakyReLU-2           [-1, 64, 32, 32]               0
            Conv2d-3          [-1, 128, 16, 16]        131,072
       BatchNorm2d-4          [-1, 128, 16, 16]            256
         LeakyReLU-5          [-1, 128, 16, 16]              0
            Conv2d-6            [-1, 256, 8, 8]        524,288
       BatchNorm2d-7            [-1, 256, 8, 8]            512
         LeakyReLU-8            [-1, 256, 8, 8]              0
            Conv2d-9            [-1, 512, 4, 4]      2,097,152
      BatchNorm2d-10            [-1, 512, 4, 4]          1,024
        LeakyReLU-11            [-1, 512, 4, 4]              0
           Conv2d-12              [-1, 1, 1, 1]          8,192
          Sigmoid-13              [-1, 1, 1, 1]              0
================================================================
Total params: 2,765,568
Trainable params: 2,765,568
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.31
Params size (MB): 10.55
Estimated Total Size (MB): 12.91
----------------------------------------------------------------
```

2. Please show the first 32 generated images of both method A and B then discuss the difference between method A and B.
    - A

- B

- difference
  - A整體比較白
  - B整體臉的占比比較大
  - A的四周白色或黑色區塊比較多也比較大
  - A的第二排比較多崩壞照，B沒什麼崩壞照

3. Please discuss what you've observed and learned from implementing GAN.
   - model 越深不一定越好
   - GAN不是和把圖片變太多
   - 網路上說D可以train比較多step，learning rate也適合比較大，但似乎是通常是D太弱，而本次作業感覺是D偏強，所以那些training tricks反而比較沒有幫助。
   - D(x) 和 D(G(z))越接近0.5越好，但太接近0.5就不太會繼續train了，而D(x)太接近0.9也不太會繼續train
   - 偶爾把real label和fake label交換對GAN的幫助不顯著

Problem 2

1. Please print your model architecture and describe your implementation details.
   - Unet

```
UNet(
  (time_embedding): TimeEmbedding(
    (time_embedding): Sequential(
      (0): Embedding(200, 128)
      (1): Linear(in_features=128, out_features=512, bias=True)
      (2): Swish()
      (3): Linear(in_features=512, out_features=512, bias=True)
    )
  )
  (cond_embedding): ConditionalEmbedding(
    (conditional_embedding): Sequential(
      (0): Embedding(11, 128, padding_idx=0)
      (1): Linear(in_features=128, out_features=512, bias=True)
      (2): Swish()
      (3): Linear(in_features=512, out_features=512, bias=True)
    )
  )
```

```
  (head): Conv2d(3, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
  (downblocks): ModuleList(
    (0): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 128, eps=1e-05, affine=True)
        (proj_q): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (1): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
```

```
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 128, eps=1e-05, affine=True)
        (proj_q): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (2): DownSample(
      (conv_1): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
      (conv_2): Conv2d(128, 128, kernel_size=(5, 5), stride=(2, 2),
padding=(2, 2))
    )
    (3): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 256, eps=1e-05, affine=True)
        (proj_q): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (4): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
```

```
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 256, eps=1e-05, affine=True)
        (proj_q): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (5): DownSample(
      (conv_1): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
      (conv_2): Conv2d(256, 256, kernel_size=(5, 5), stride=(2, 2),
padding=(2, 2))
    )
    (6): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 256, eps=1e-05, affine=True)
        (proj_q): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
```

```
        (proj_k): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (7): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 256, eps=1e-05, affine=True)
        (proj_q): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
      )
    )
  )
  (middleblocks): ModuleList(
    (0): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
```

```
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): AttentionBlock(
        (group_norm): GroupNorm(32, 256, eps=1e-05, affine=True)
        (proj_q): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_k): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj_v): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (proj): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (1): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Identity()
      (attention): Identity()
    )
  )
  (upblocks): ModuleList(
    (0): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 512, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
```

```
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (1): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 512, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (2): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 512, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
```

```
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (3): UpSample(
      (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (convtrans): ConvTranspose2d(256, 256, kernel_size=(5, 5), stride=
(2, 2), padding=(2, 2), output_padding=(1, 1))
    )
    (4): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 512, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (5): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 512, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
```

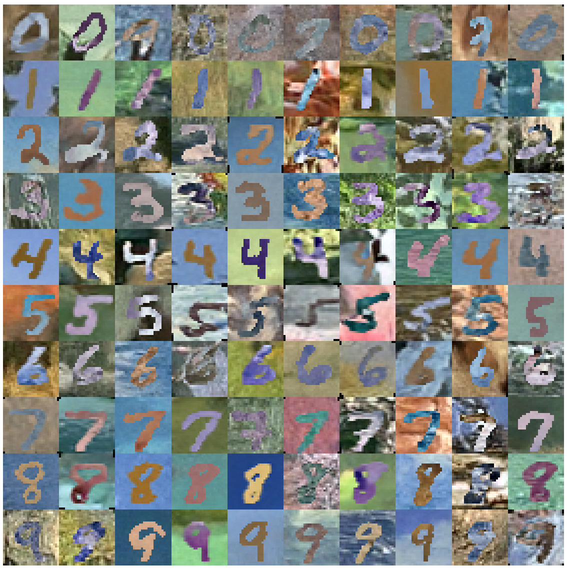
```
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (6): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 384, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=256, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (7): UpSample(
      (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (convtrans): ConvTranspose2d(256, 256, kernel_size=(5, 5), stride=
(2, 2), padding=(2, 2), output_padding=(1, 1))
    )
    (8): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 384, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(384, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
```

```
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (9): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (shortcut): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
      (attention): Identity()
    )
    (10): ResidualBlock(
      (block1): Sequential(
        (0): GroupNorm(32, 256, eps=1e-05, affine=True)
        (1): Swish()
        (2): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
      (time_embedding_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (cond_proj): Sequential(
        (0): Swish()
        (1): Linear(in_features=512, out_features=128, bias=True)
      )
      (block2): Sequential(
        (0): GroupNorm(32, 128, eps=1e-05, affine=True)
        (1): Swish()
        (2): Dropout(p=0.15, inplace=False)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      )
```

```
        (shortcut): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
        (attention): Identity()
      )
    )
    (tail): Sequential(
      (0): GroupNorm(32, 128, eps=1e-05, affine=True)
      (1): Swish()
      (2): Conv2d(128, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    )
)
```

- train process
  - 把time和label(condition)進行embedding
  - 每個時間點都會加上隨機但亮在固定範圍的雜訊
  - image forward Unet到最後會生成image size的latent
  - latent和常態分布取loss，希望學出常態分布
  - 從latent生成圖片時會去看統計量去加noise

2. Please show 10 generated images for each digit (0-9) in your report. You can put all 100 outputs in one image with columns indicating different noise inputs and rows indicating different digits.



3. Visualize total six images in the reverse process of the first “0” in your grid in (2) with different time steps.
   - T: 0 40 80 120 160 200
     

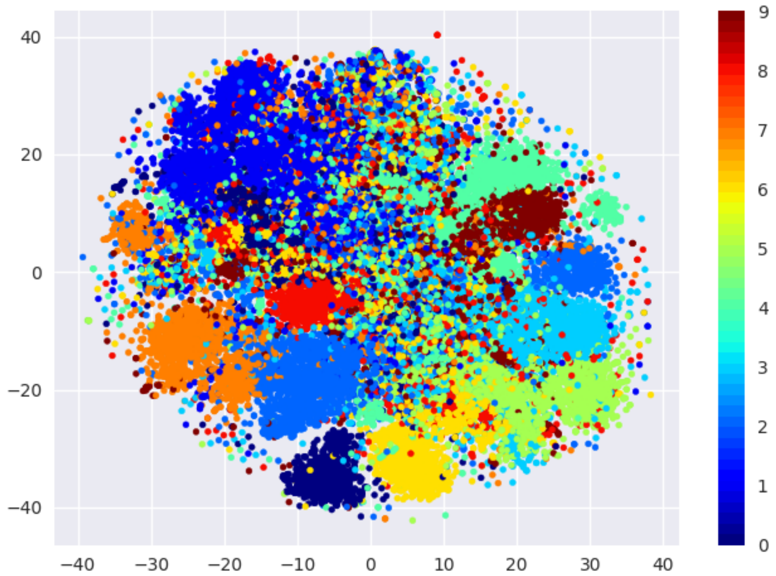
4. Please discuss what you’ve observed and learned from implementing conditional diffusion model.
   - 因為想讓不同時間點同時訓練，tensor會擴增時間維度，時間越久所需空間也就越大
   - 統計值是看各個時間點各種condition的整組，所以batch之中可能會互相影響到，也就代表batch size也變得重要
   - 慢慢加noise是一種幫助generate的手段，而能不能train好主要跟model比較相關

Problem 3

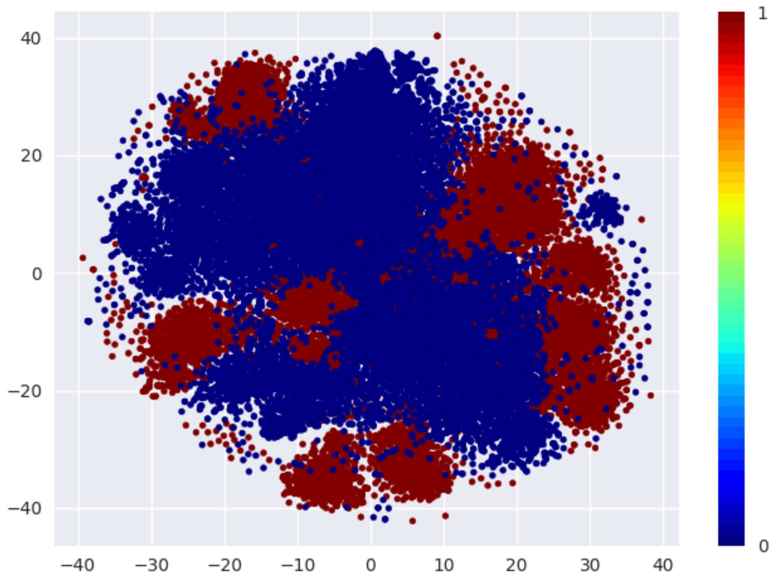
1. Please create and fill the table with the following format in your report

	MNIST-M → SVHN	MNIST-M → USPS
Trained on source	0.42852646818153206	0.719758064516129
Adaptation (DANN)	0.4484798892175993	0.7681451612903226
Trained on target	0.8558569899918173	0.8487903225806451

2. Please visualize the latent space of DANN by mapping the validation images to 2D space with t-SNE. For each scenario, you need to plot two figures which are colored by digit class (0-9) and by domain, respectively.
    - Note that you need to plot the figures of both 2 scenarios, so 4 figures in total.
    - SVHN
        - by class

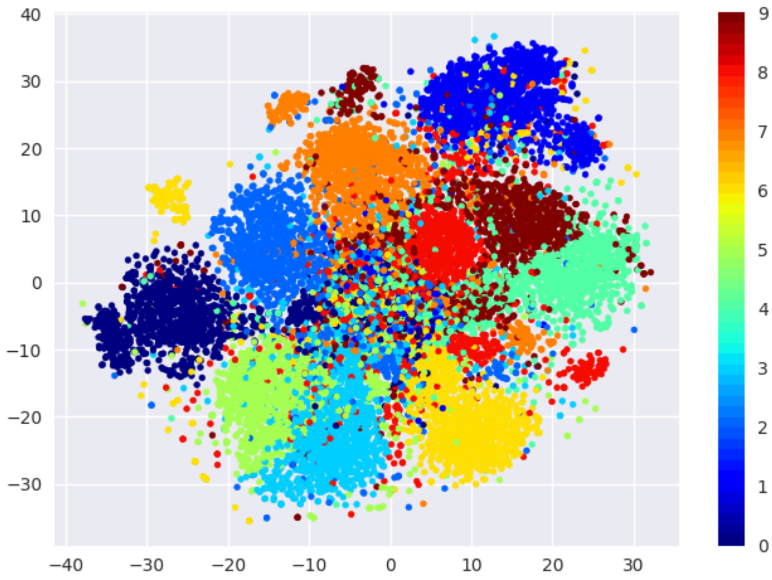


        - by domain

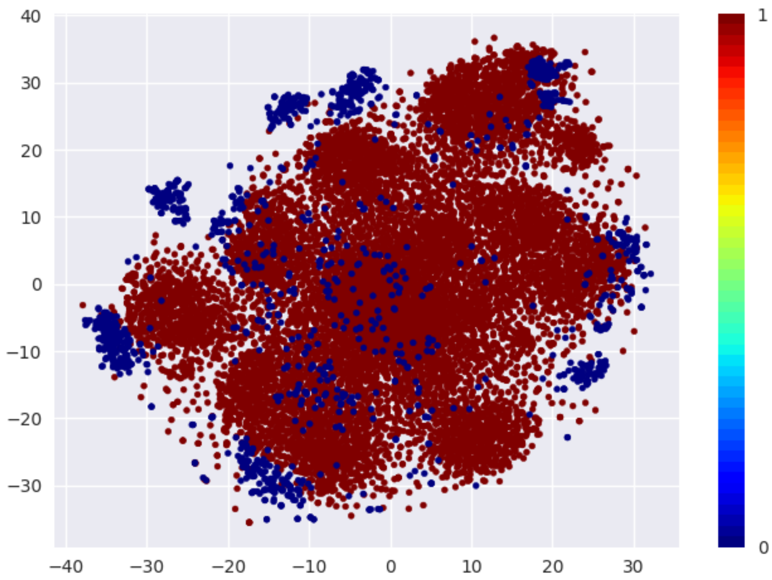


    - USPS

- by class



- by domain



3. Please describe the implementation details of your model and discuss what you've observed and learned from implementing DANN.

```
DANN_model(
  (feature_extract): 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): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (4): Conv2d(64, 48, kernel_size=(5, 5), stride=(1, 1))
    (5): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (6): Dropout2d(p=0.3, inplace=True)
    (7): ReLU(inplace=True)
```

```
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (class_classifier): Sequential(
    (0): Dropout(p=0.3, inplace=False)
    (1): Linear(in_features=768, out_features=192, bias=True)
    (2): BatchNorm1d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): Mish(inplace=True)
    (4): Dropout(p=0.3, inplace=False)
    (5): Linear(in_features=192, out_features=128, bias=True)
    (6): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (7): Mish(inplace=True)
    (8): Linear(in_features=128, out_features=10, bias=True)
  )
  (domain_classifier): Sequential(
    (0): Dropout(p=0.3, inplace=False)
    (1): Linear(in_features=768, out_features=128, bias=True)
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): Mish(inplace=True)
    (4): Linear(in_features=128, out_features=2, bias=True)
  )
)
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 24, 24]           4,864
       BatchNorm2d-2           [-1, 64, 24, 24]             128
              ReLU-3           [-1, 64, 24, 24]               0
         MaxPool2d-4           [-1, 64, 12, 12]               0
            Conv2d-5             [-1, 48, 8, 8]          76,848
       BatchNorm2d-6             [-1, 48, 8, 8]              96
         Dropout2d-7             [-1, 48, 8, 8]               0
              ReLU-8             [-1, 48, 8, 8]               0
         MaxPool2d-9             [-1, 48, 4, 4]               0
          Dropout-10                  [-1, 768]               0
           Linear-11                  [-1, 192]         147,648
      BatchNorm1d-12                  [-1, 192]             384
             Mish-13                  [-1, 192]               0
          Dropout-14                  [-1, 192]               0
           Linear-15                  [-1, 128]          24,704
      BatchNorm1d-16                  [-1, 128]             256
             Mish-17                  [-1, 128]               0
           Linear-18                   [-1, 10]           1,290
          Dropout-19                  [-1, 768]               0
           Linear-20                  [-1, 128]          98,432
      BatchNorm1d-21                  [-1, 128]             256
             Mish-22                  [-1, 128]               0
           Linear-23                    [-1, 2]             258
================================================================
Total params: 355,164
Trainable params: 355,164
Non-trainable params: 0
```

```
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.04
Params size (MB): 1.35
Estimated Total Size (MB): 2.40
----------------------------------------------------------------
```

- detail
  - 用convolution作為feature extraction，取的latent後，分別有class跟domain的classifier。訓練domain時要把gradient反轉，用alpha作為反轉的開關
  - alpha和現在的訓練進度有關，依據當前在總共要訓練的資料是中的比例來調整
  - model 主要還是在學class classifier，domain classifie會讓class classifier不會學到太好
  - 認為DANN幫助有限
- MNISTM -> SVHN
  - class_classifier不宜太難
  - regularization大比較合適
  - 難train是因為task難很多而且差異domain很大(數字數量)，所以model太複雜容易over fitting
- MNISTM -> USPS
  - class_classifier深比較好
  - 很快就overfitting