# Homework #2

Deep Learning for Computer Vision

NTU, Fall 2023

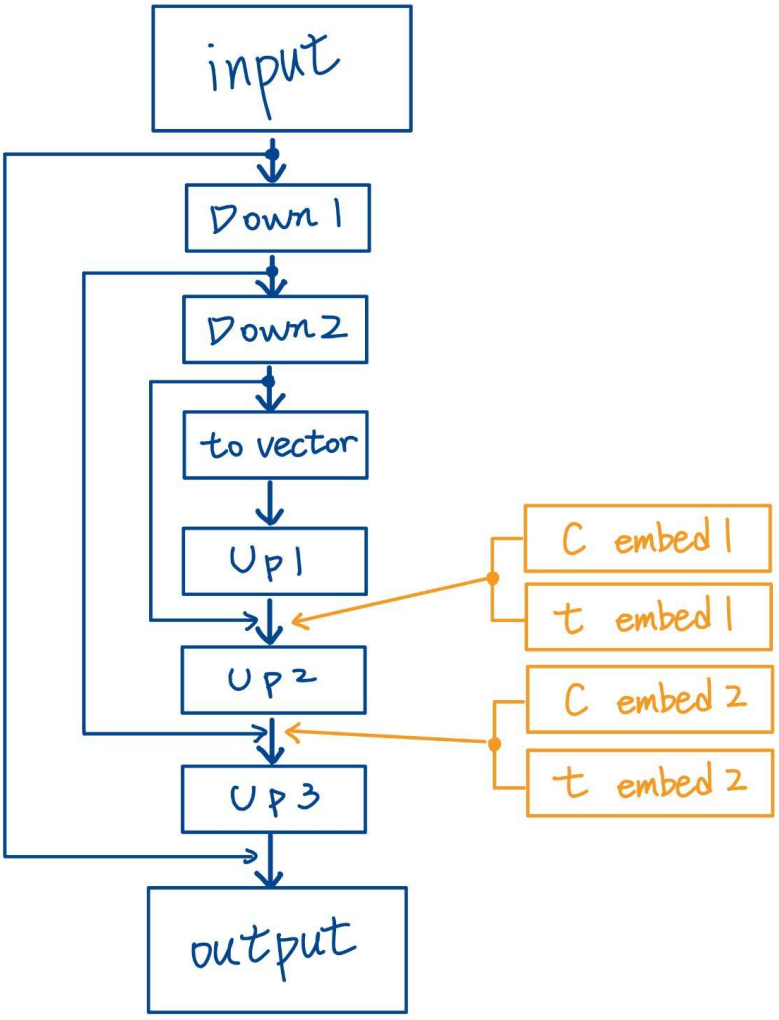
周奕節
r10943131

## Problem 1: Diffusion models (35%)

1. (5%) Follow the Github Example to draw your model architecture and describe your implementation details.

   A. Model architecture:

   The model contains: upsampling, downsampling and embedding fully connected layers.

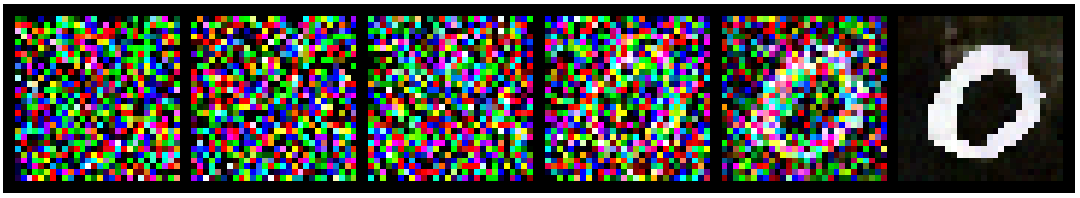



ref: https://github.com/TeaPearce/Conditional_Diffusion_MNIST

B. Implementation details
    a. Time steps = 1000
    b. optimizer: ‘AdamW’, an improved version of Adam that includes weight decay. Learning rate = 1e-4, weight_decay=1e-5
    c. scaler: optimizing memory consumption and accelerating the training process.
    d. torch.autocast: reduce the memory requirements and potentially speed up training, as float16 operations are faster but have a smaller range and less precision compared to float32.

2. (5%) 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. (5%) Visualize total six images in the reverse process of the **first "0"** in your grid in (2) **with different time steps**.



4. (5%) Please discuss what you've observed and learned from implementing conditional diffusion model.

   a. Diffusion model is a cool idea.
   b. Time steps: higher number of time steps may result better quality images, however it also increases the training time.
   c. Although didn't implement GAN, diffusion model seems to be easier to train.

## Problem 2: DDIM (35%)

Public Baseline:

| Metric | Baseline | Result |
|---|---|---|
| MSE ↓ | 20 | 0.3 |

1. (7.5%) Please generate face images of noise **00.pt ~ 03.pt with different eta** in one grid. Report and explain your observation in this experiment.

   The eta is 0.0 on the left and gets larger to value 1.0 on the right. We can observe that as eta increases, the image is more different from the original one, it looks totally different after eta>0.50. Also, the image quality is not so good after eta>0.5, we can tell some of them are fake easily.

eta = 0.0 eta = 0.25 eta = 0.50 eta = 0.75 eta = 1.0

2. (7.5%) Please generate the face images of the interpolation of noise **00.pt ~ 01.pt.** The interpolation formula is **spherical linear interpolation**, which is also known as **slerp**. What will happen if we simply use linear interpolation? Explain and report your observation.

Slerp seems to be a good interpolation since during the transition of two noise every images still maintain there facial features. On the other hand, linear interpolation might not be a good way, it's obvious that all the images with interpolation looks weird.

A. Spherical Linear interpolation (Slerp)

B. Linear interpolation

## Problem 3: DANN (35%)

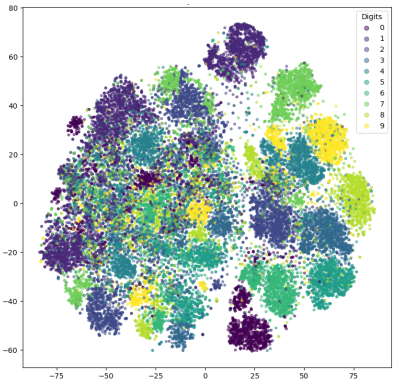
1. (10%) Please create and fill the table with the following format **in your report**:

| | MNIST-M → SVHN | MNIST-M → USPS |
|---|---|---|
| Trained on source | 0.3122 | 0.7332 |
| Adaptation (DANN) | 0.4138 | 0.8575 |
| Trained on target | 0.8629 | 0.9805 |

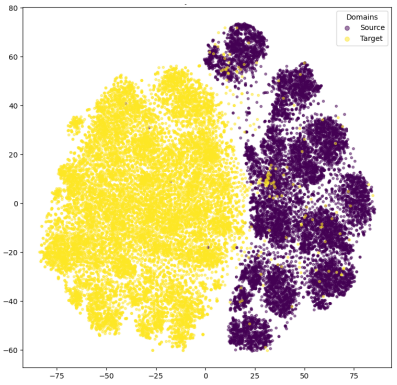
2. (10%) Please visualize the latent space **(output of CNN layers)** of DANN by mapping the ***validation*** images to 2D space with t-SNE. For each scenario, you need to plot two figures which are colored **by digit class (0-9)** and **by domain**, respectively.

A. MNIST-M→SVHN

**by digit class (0-9)**

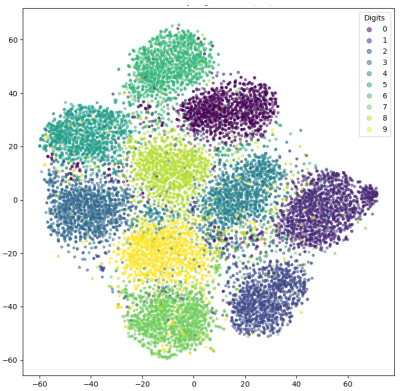



**by domain**

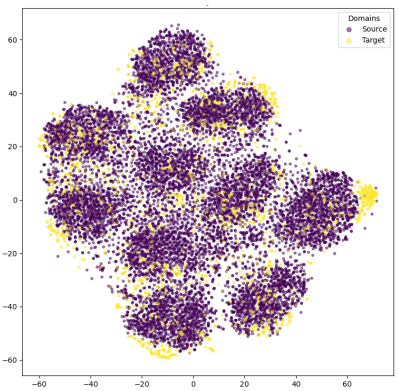



B. MNIST-M→USPS

**by digit class (0-9)**




**by domain**



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

A. Model:

There are three parts in the model: Feature extractor, class classificier and domain classificatier.

a. Feature Extractor

```
Sequential(
    Conv2d(3, 64, kernel_size=5)
    BatchNorm2d(64)
    MaxPool2d(2)
    ReLU()
    Conv2d(64, 64, kernel_size=5)
    BatchNorm2d(64)
    Dropout2d()
    MaxPool2d(2)
    ReLU()
    Conv2d(64, 128, kernel_size=3)
)
```

b. Class Classifier

```
Sequential(
    nn.Linear(512, 100)
    BatchNorm1d(100)
    ReLU()
    Dropout(),
    Linear(100, 100)
    BatchNorm1d(100)
    ReLU()
    Linear(100, 10)
    LogSoftmax(dim=1)
)
```

c. Domain Classifier

```
Sequential(
        Linear(512, 100)
        BatchNorm1d(100)
        ReLU()
        Linear(100, 2)
        LogSoftmax(dim=1)
)
```

B. Implementation details:

a. Loss: Cross entropy loss
b. Optimizer: AdamW, learning rate = 1e-4, weight decay = 1e-5
c. Scheduler: CosineAnnealingWarmRestarts, T_0=2, T_mult=2

C. Observations

a. Train on source v.s. DANN v.s. Train on target

‘Training on source’ is the lower bound and got the lowest accuracy because the model what we trained can adapt to the target we want to anticipate. On the other hand, ‘training on target’ is the upper bound because the model is trained directly on the target domain and thus we can get the highest accuracy. The DANN model uses domain adaptation technique to get results better than the lower bound while train the model on the source, and the goal is to get performances near the upper bound.

b. MNIST-M→SVHN v.s. MNIST-M→USPS

MNIST-M→USPS performs a lot better than MNIST-M→SVHN because images in USPS is more similar to MNIST-M, while images in SVHN are hard to read.

We can also get the result from the visualize-plot. In MNIST-M→SVHN, the digit classes are mixed and the source and target domain are separated. In MNIST-M→USPS, the digit classes are seperated and the source and target domain are mixed.