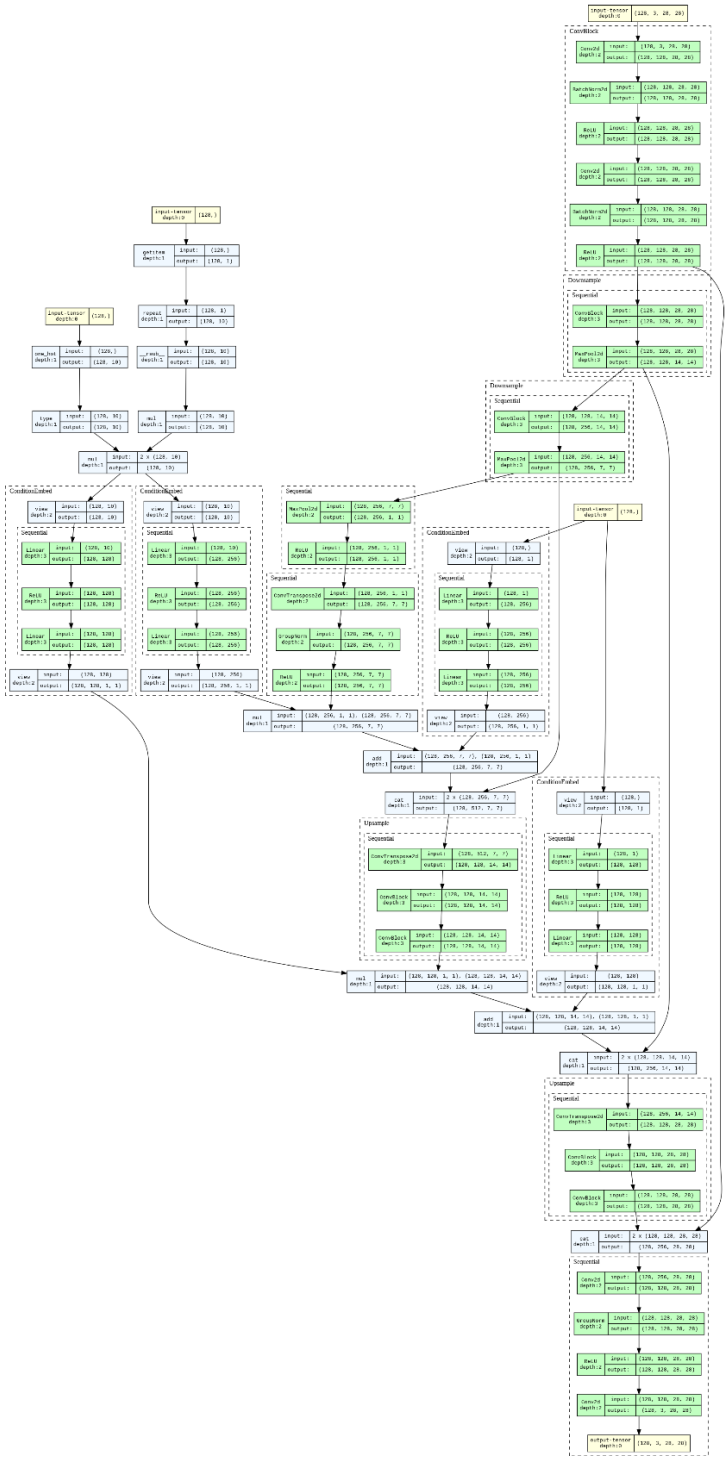
DLCV HW2 Report

R11921078 李鎮宇

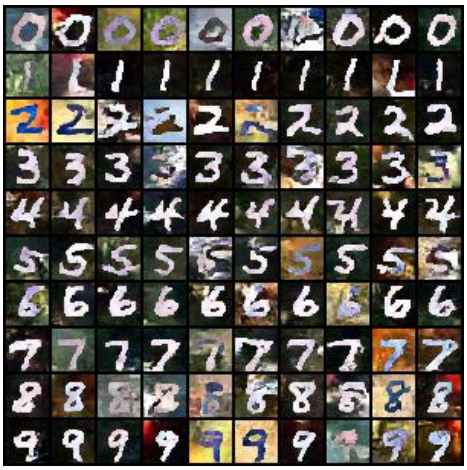
Problem 1: Conditional Diffusion Models

1. Follow the Github Example to draw your model architecture and describe your implementation details.



參考[1]中 Unet 模型架構的設計以及實作 Classifier-Free Diffusion Guidance(CFDG)。Unet 主要分成 Downsample、Conditional Embedding 和 Upsample。Downsample 的部分透過 Conv2d 和 Maxpool2d 降維；Conditional Embedding 對 time 和 label 的資訊進行 embedding，透過 Linear layer 轉換成對應大小的向量；Upsample 則是利用 ConvTranspose2d 和 Conv2d 進行升維，並且只有在 Upsample 的過程中加入 time 和 label 的 embedding 讓模型學習，以 label * tensor + time 的方式讓向量帶有 condition。整體架構中我沒有使用 github 中定義的 Residual，而是單純使用 Convolution，而 activation function 我都改用 ReLU，希望能簡化並穩定計算的過程。CFDG 主要訓練 conditional 和 unconditional 兩種情況，讓 diffusion model 後續生成時效果更好，做法是在原本的訓練過程中對每一批要訓練的影像，隨機地讓它沒有 label 的資訊，就能完成 unconditional 的訓練，在 sample 的部分，分配不同的比重(guide weight)給 conditional 和 unconditional 生成影像。

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=500 t=600 t=700 t=800 t=900 t=1000

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

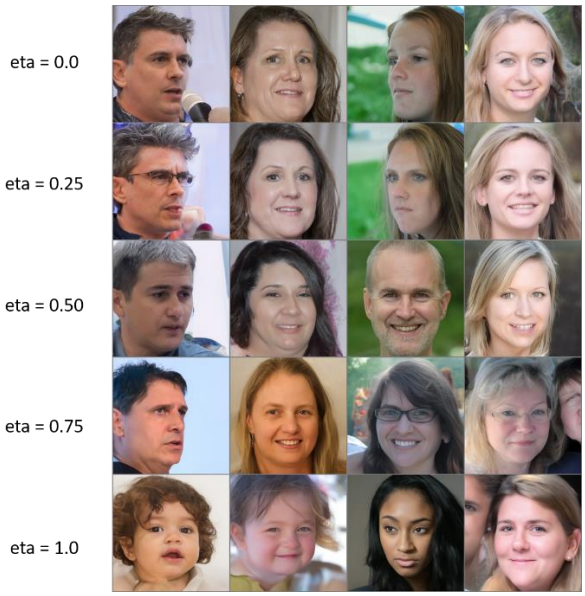
   在實作過程中慢慢了解 diffusion model 學習雜訊的過程，而且可以透過 Unet 這種架構相對簡單的模型完成也覺得很不可思議。另外一開始其實不太確定 condition 的資訊要用什麼方式加諸在影像的 tensor 上，後來也在實作的過程中了解要先將所有資訊都轉成 embedding 的方式，然後加 condition 的方式有的使用相加、相乘或是 tensor 的 concat，而 concat 我自己在實作時覺得比較麻煩，需要考慮在每一層 convolution 前將 input channel 加 1。

   另外為了知道若沒有使用 CFDG，模型生成的效果是否良好，我另外跑了沒有使用 CFDG 的訓練和測試。同樣生成 1000 張影像，使用 CFDG 的準確度可高達 0.995，而沒有使用的僅 0.781，由此可見 condition 和 uncondition 的聯合訓練可以讓模型生成的更好。

Problem 2: DDIM

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

   當 eta=1.0 時，也就是 DDPM，可以看到影像在脖子的部分沒有生成的很好，隨著 eta 逐漸變小，越接近 DDIM 生成的人像品質也有逐漸變好。而切不同的 noise 也讓生成的效果有所差異，比如 03.pt 的在 DDPM 時生成的人像品質就比 00.pt 和 01.pt 來的好，這樣讓它在後續不同 eta 的情況下都能生成出相對不錯的品質。



2. 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 result:

- linear result:

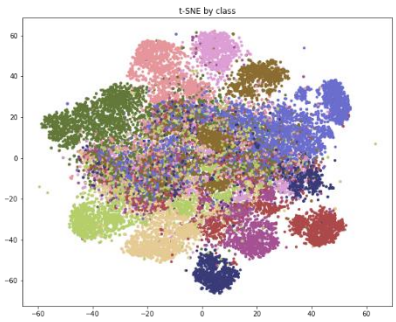
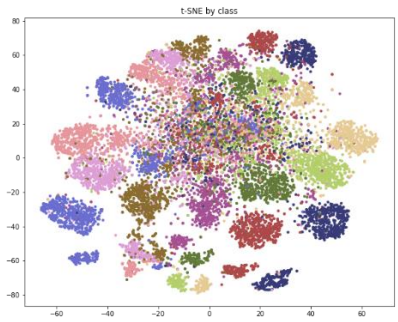
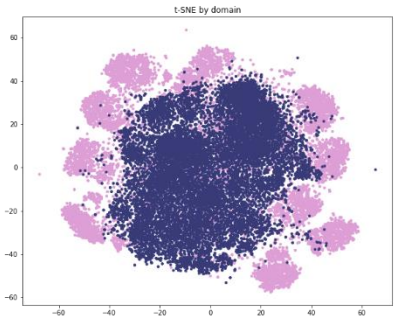
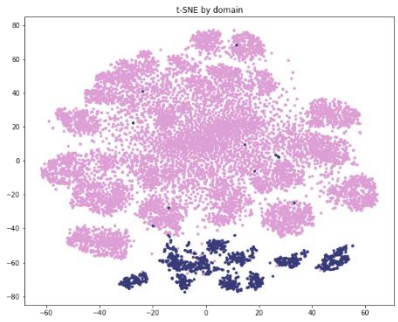
從結果可知 slerp 比 linear 的效果好很多，因為 slerp 產生的插植沒有像 linear 變化那麼大。透過 alpha 的變化可以逐步看出結合兩個 noise 所生成的影像，在 alpha={0.4, 0.5, 0.6}的這幾個結果中能看出有融合左右兩側的人臉風格。

Problem 3: DANN

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

	MNIST-M → SVHN	MNIST-M → USPS
Trained on source	0.2750	0.8079
Adaptation (DANN)	0.4651	0.8272
Trained on target	0.9369	0.9757

2. 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.

MNIST-M → SVHN (by class)	MNIST-M → USPS (by class)
	
MNIST-M → SVHN (by domain)	MNIST-M → USPS (by domain)
	

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

參考[3]的架構設計三個主要模型：Feature Extractor(F)、Label Predictor(L)和 Domain Classifier(D)。在 F 的設計中，考量影像不會太複雜且解析度不高，因此只使用四層 convolution 做特徵提取；L 和 D 的架構基本上一樣，以 linear 層為主，在 L 中需判斷 digit class，因此最後輸出 channel 為 10，而 D 需分辨影像為 source 或是 target domain，因此最後輸出為 channel 為 1，並加上一層 sigmoid function 將數值壓在(0, 1)。loss function 選用 cross entropy 評估 L，binary cross entropy 評估 D；optimizer 皆使用 Adam，learning rate 設為 0.0001，batch size 為 128，並在訓練過程中加入 scheduler 調整 learning rate。

DANN 的訓練開始前，先檢查和處理預計使用的資料，training set 包含 source domain 的 image 和對應的 label，以及 target domain 的 image，validation set 我有使用過 source domain 也有使用過 target domain，發現最後模型的表現差異不大。針對 training set 進行 data augmentation，在 mnistm→svhn 中，隨機旋轉，調整影像的 brightness、contrast、saturation，邊緣的銳化，高斯模糊等，而在 mnistm→usps 中，影像皆轉換成灰階影像處理，也套用上述的 augmentation 再做一些調整。

訓練的過程中，source 和 target 的影像都先經過 F 提取特徵，得到 feature 的 latent space 後，送往 D 進行訓練，在此我將 source domain 和 target domain 分別標為 1 和 0。接著 source 影像和 label 送往 L 進行訓練，兩個分類器最後會得到兩個 loss(domain loss、label loss)，接著要做 back propagation，因為要讓 domain classifier 分不清 source 和 target，因此 loss 的部分我使用以下公式：$\text{total loss} = \text{label loss} - 0.31 \times \text{domain loss}$，讓 domain loss 無法往好的方向更新，0.31 是我從 DANN 的 paper 中看到作者們實驗得到的參數。

Reference:

[1] Conditional Diffusion Model: https://github.com/TeaPearce/Conditional_Diffusion_MNIST/tree/main

[2] DDIM: https://zhuanlan.zhihu.com/p/565698027

[3] DANN: https://github.com/Yangyangii/DANN-pytorch/tree/master