GAI HW 4

F74091093 陳宥橋

- ## Background:
  - dip：參考作者的 github
  - ddpm：參考別人所寫的 example
  - dip 和 ddpm 都是使用 Unet 當其模型，但兩著模型有差異。在最後有附上兩個模型的細節
  - 資料集選擇：MINST，且只用黑白
  - 環境：Colab T4
  - 補充：
    - 會選擇使用 MINST 是因為主要是以 colab 來做訓練，如果選大一點的數據或著圖片，會導致無法 train 完就耗盡資源。
    - 由於 ddpm 生成的圖片不會每次都很好看，所以以下數據有稍微挑選過。
  - Github link : https://github.com/joec1368/ddpm-dip-experiment
- ## Theoretical Justification:
  - 方法：
    - Dip 可以去做 denoise ，把圖片的噪點去除；ddpm 可以從純噪圖變成一張圖片。所以藉由 dip 去除不適合的噪點，讓 ddpm 可以加速生成圖片。
    - 利用 train 好的 ddpm ，在生成的時候，先讓 noise 圖通過 2 次 dip ，產生比較好的 noise 圖，然後讓 ddpm 把圖片完善，希望以此方式，在品質不改變的情況下，去改善 ddpm 生成圖片的速度。
    - ddpm 以及 dip ，都會先以 minst 去做訓練。
  - 可能的優缺點：
    - 優點：
      - 生成圖片的速度比只用 ddpm 還快且畫質不影響太多
      - 相較於 dip ，這個方法可以應用在更多的圖像上，不會受限於 dip 的一張圖，且是可以讓 ddpm 自己生成完全新的圖
    - 缺點：
      - 應用場景不多，因為 dip 還是有先看過類似的圖片，才讓其應用在 ddpm 前面。故 dip 改善的方式，會向他曾經看過的圖片的方向去做改善。故生成的圖片會有所侷限。

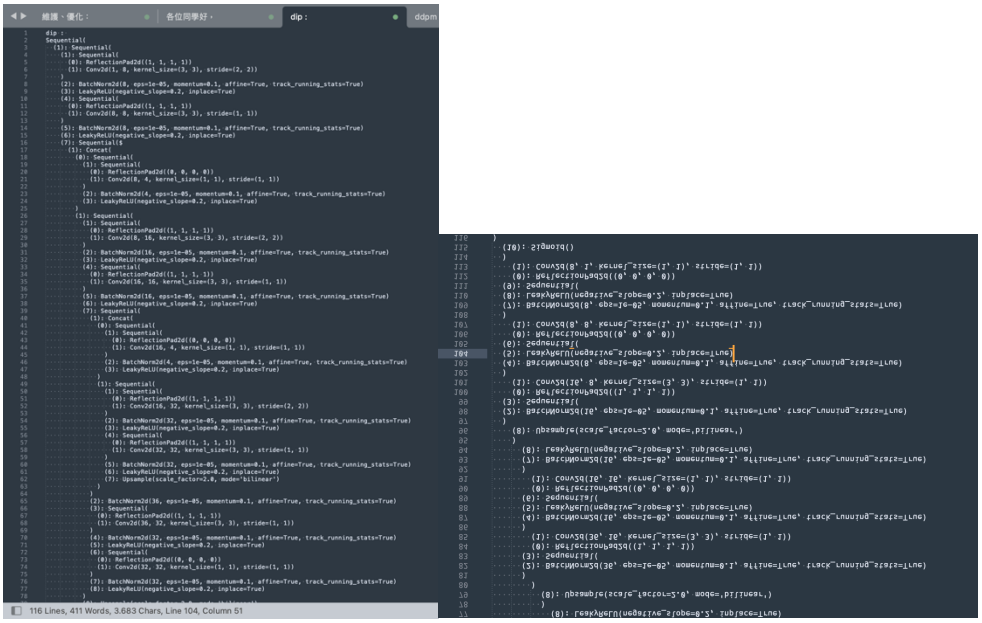
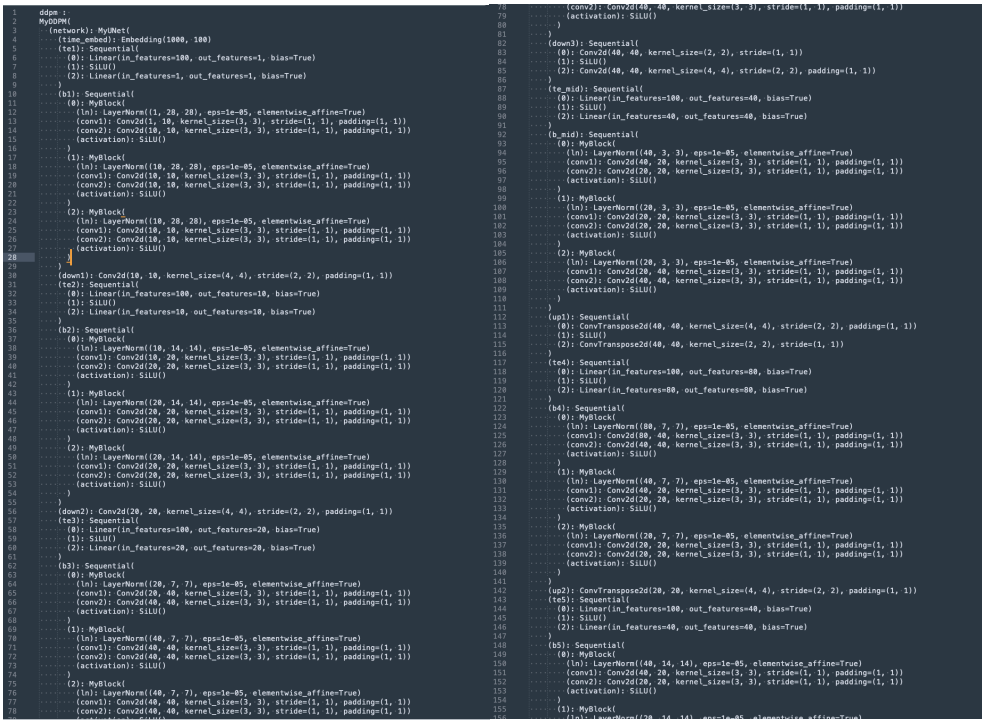
        - 如果單以生成一張圖片來說，會比 dip 久。但 dip 本質上沒有做生成，而是去做修復。

- # Experimental Verification :
    - 下表顯示 ddpm 和 dip + ddpm 生成不同步驟的圖片，所花的時間，可以看到步數越少，時間基本上越短。而因為我們跑的模型和 dataset 都很小，所以 step 上升導致時間上升的差距不明顯。但基本上還是可以看到該趨勢。因為我們想要做的是生成一張新的圖片，故 dip 做不到。

BACKFORWARD	TIMES	100	200	300	400	500	600	700	800	900	1000
DDPM		0.644	1.246	1.846	2.487	3.152	4.570	4.311	5.400	6.053	6.450
DIP + DDPM		0.743	1.252	2.743	2.639	3.106	3.680	5.233	4.948	6.431	6.148

    - 生成品質比較：
        - 我們以兩個 step 當作比較，分別是 200 和 500 的時候：
            - 200:
                - dip + ddpm: 
                - dip along: 
                - 可以發現，在相同的 step 底下，有經過 dip 的會更快讓 noise 全部消失
            - 500:
                - dip + ddpm: 
                - dip along: 
        - 綜上所述，在相同 step 下，ddpm+dip 會有比較好的品質；相反的 純 ddpm 就會需要比較多的 step 才能得到相同的品質。而根據前面的圖片可知，當比較多 step 時，會需要比較多的時間。故可以得出，當有加入 dip 的時候，會比較快形成圖片，且可以達到很好的品質。

- Ablation Studeies and Analysis:
  - 每次跑不同的 dip 次數：
    - 從左至右分別是 dip 在 ddpm 前面跑的次數，也就是圖片會先經過幾次 dip 才送入 ddpm，分別是 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50]
    - Ddpm step = 100:
      - 
    - Ddpm step = 500:
      - 
    - Ddpm step = 900:
      - 
    - 可以看到，在任意 step 數量時，如果 dip 次數是介於 1~3 的話，會有比較好的表現，所以我們以 dip 次數為 2 為主。
    - 推論原因為，因為 dip 多次後的噪圖，不會是 ddpm 喜歡的，故 dip 多次後的結果不怎麼好。
  - ddpm 不同 epoch 數:
    - dip 都是 3 次、ddpm step 設定為 500，由左至右分別是 [5,10,15,20]
      - 
      - 可以看到，在 epoch 是 20 時，效果最好，故我們選擇 epoch 20 當作這次的 epoch 數量。
      - 可以知道，訓練 ddpm 的時候，epoch 數量還是要高。
  - 把 dip 放進 ddpm 的 training 過程中，看成果會不會比較好：
    - Step = 1000, dip 次數 = 2, epoch = 20
      - 
    - 可以看到，再跑完以後，甚至連 denoise 都還沒結束故可知，把 dip 當作 ddpm 訓練的一部分是不可行的方式
    - 推論原因是，因為 ddpm 每次生成出來的圖，在下一個 step 的時候，都還是會再經過 dip 。而因為 dip 對該圖不熟，所以可能會在 denoise 的時候，把重要資訊 denoise 掉，因此成果越來越差。
  - 綜合以上三點所述，所以我們把 dip 次數地定為 2 ， epoch 定為 20 以及不在 ddpm training 過程中放入 dip。

- 模型細節：
  - Dip :
    - 
  - Ddpm :
    - 

```
      (1): MyBlock(
        (ln): LayerNorm((20, 14, 14), eps=1e-05, elementwise_affine=True)
        (conv1): Conv2d(20, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (activation): SiLU()
      )
      (2): MyBlock(
        (ln): LayerNorm((10, 14, 14), eps=1e-05, elementwise_affine=True)
        (conv1): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (activation): SiLU()
      )
    )
    (up3): ConvTranspose2d(10, 10, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (te_out): Sequential(
      (0): Linear(in_features=100, out_features=20, bias=True)
      (1): SiLU()
      (2): Linear(in_features=20, out_features=20, bias=True)
    )
    (b_out): Sequential(
      (0): MyBlock(
        (ln): LayerNorm((20, 28, 28), eps=1e-05, elementwise_affine=True)
        (conv1): Conv2d(20, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (activation): SiLU()
      )
      (1): MyBlock(
        (ln): LayerNorm((10, 28, 28), eps=1e-05, elementwise_affine=True)
        (conv1): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (activation): SiLU()
      )
      (2): MyBlock(
        (ln): LayerNorm((10, 28, 28), eps=1e-05, elementwise_affine=True)
        (conv1): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (activation): SiLU()
      )
    )
    (conv_out): Conv2d(10, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
)
```