F74104032_資訊 114 彭昇哲

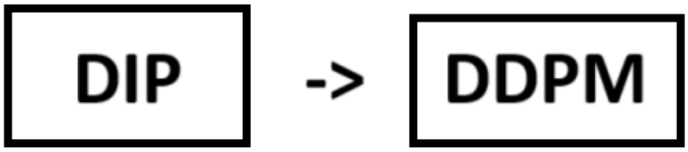
我有使用開源的專案來協助完成個作業 SOURCE:

https://github.com/MarceloGennari/diffusion_mnist/blob/main/main.py

Theoretical Justification

使用 DIP 來強化 DDPM 的效率。

架構圖如下



我的想法是參考說明文件的 solution1，因為 DIP 的存在，或許 DDPM 不需要徹底從頭開始將訊號去除。也就是說原先的 DDPM，需要考慮 0~999 的 time schedule，但是在 DIP 預處理後可能只需要考慮 0~500 的 schedule，也就是從 500 步開始繼續將雜訊去除。

Benefit:

減少 DDPM 的訓練成本、DDPM 生成速度提高

Limitation:

在開始前我認為這個想法有巨大缺陷。因為 DDPM 是一種 Diffusion model，原先就是設計成一種生成式模型，允許輸入隨機的參數，產生隨機的結果。

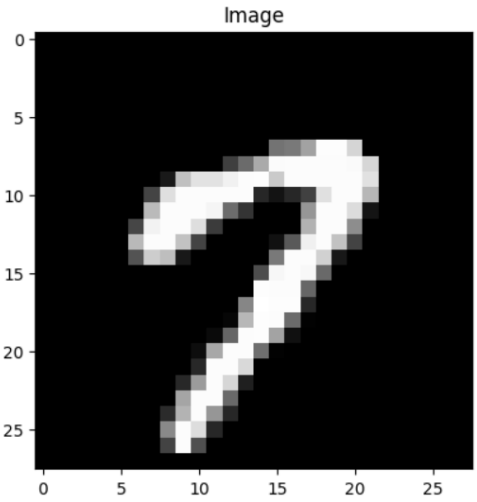
然而 DIP 的固有結構會限制 DDPM 的輸入，因為 DIP 限制輸入為一個固定的 noise，計算 Loss 也只跟同一張圖片，跟 DDPM 的隨機性相比，它更像一個一對一的函數。

只用一張圖讓 diffusion model 進行 Denoising 存在消除極限，模型消除掉一部分雜訊之後就無法再繼續消除。因為不存在其他的對照組，所以模型無法識別全部的雜訊，或著說模型無法徹底理解雜訊(備註 1)。

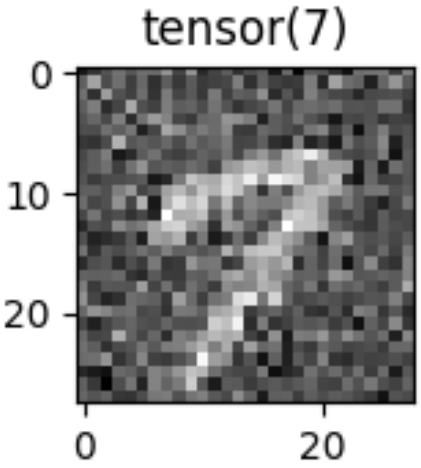
備註 1:我使用 DIP 的概念讓 DDPM 只針對一張 MNIST 的手寫圖進行訓練，diffusion time schedule 為 1000 個 steps，共 5000 個 step，(5000 是因為 loss 很難繼續降低)，每次的流程為從 0~999 挑一個數進行 diffusion forward，將 diffused image 餵進 diffusion model，將輸出跟真正的 noise 計算誤差(MSE loss)。訓練結束後隨機輸入一個 noise 給 diffusion model 進行 inverse process，我給 DDPM 設定了 1000 個 time step，因此 inverse 從 time step 999 開始直到 time step 為 0 結束。

其結果如下:

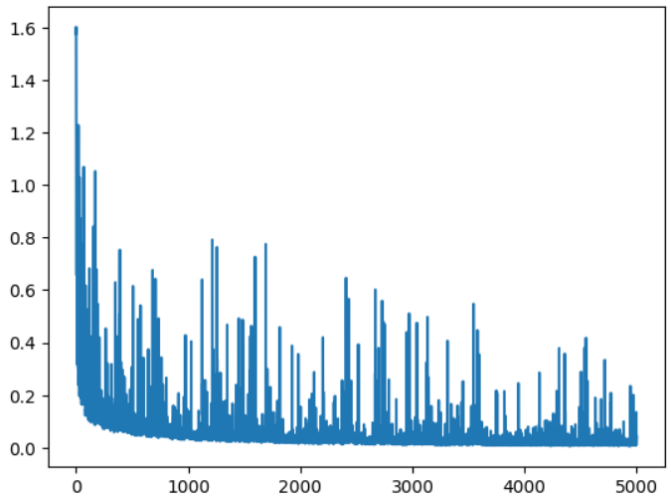
Input image:



reconstructed image:



Loss regression:



從結果上來看或許 DDPM 不適合進行 denoising。

但是如果不要求徹底 Denoising 那就是另一回事了

Potential compared using DDPM alone:

因為一部分的處理由 DIP 承擔，我認為 DDPM 能夠更好的專注於預測剩下的 noise(需要考慮的 time step 較少)，因此具有訓練成本較低，輸出的 image 比只有 DDPM(備註 2)更清晰。

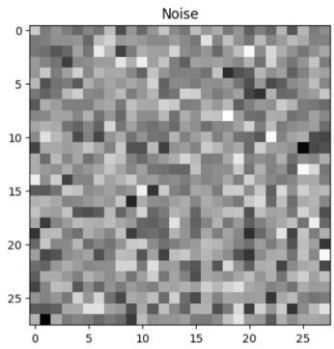
備註 2:這裡的 DDPM 是同樣跟 DIP 串 DDPM 一樣只針對一張 noised image 訓練出來的，我認為同樣限定一張圖片比較好比較。

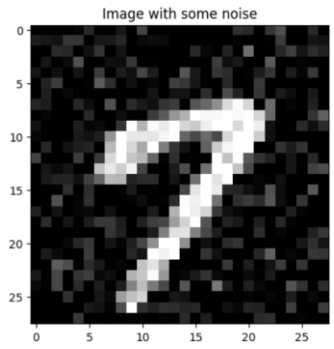
Part2:

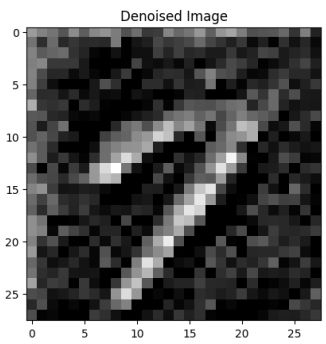
Experiment verification:

我先訓練了一個 DIP，訓練 2000 個 epoch 結果如下:

Input noise: target image: output of DIP:







程式碼保有在 DIP_sample.ipynb 中

接著我準備兩個 DDPM 一個串 DIP，一個就只有 DDPM

DDPM 串 DIP 訓練 3000 個 epoch

只有 DDPM 的部分則訓練 5000 個 epoch

任務是

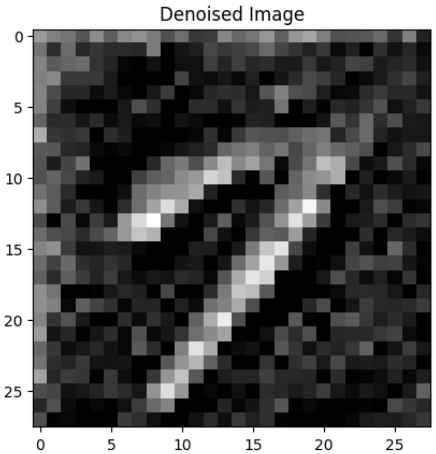
DDPM 串 DIP 只根據圖一進行訓練

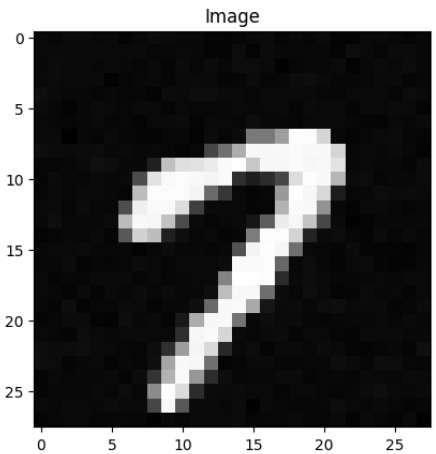
只有 DDPM 只根據圖二進行訓練

兩者只在 time step 從 0~500 計算 loss

最後輸入一段 noise 從 time step 500 進行 diffusion inverse process

圖一: output of DIP 圖二: clean image from MNIST





備註三:這裡的訓練流程跟備註一相同，只不過 time step 限

定在 0~500，但是 diffusion process 的 time step 定義域依然維持在 1000，只不過是實際上在訓練、驗證時 time step 只會出現 0~500 而已

程式碼保有在 Main_DIP_concat_DDPM.ipynb、

Main_DDPM_sole.ipynb、

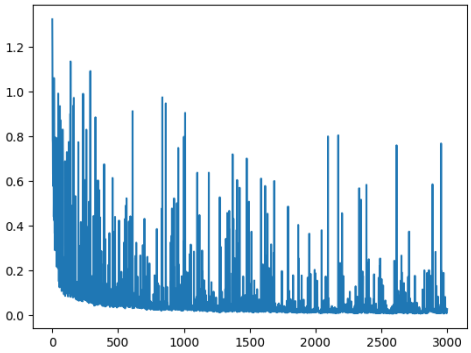
Inference_DIP_concat_DDPM.ipynb、

Inference_DDPM_sole.ipynb 中

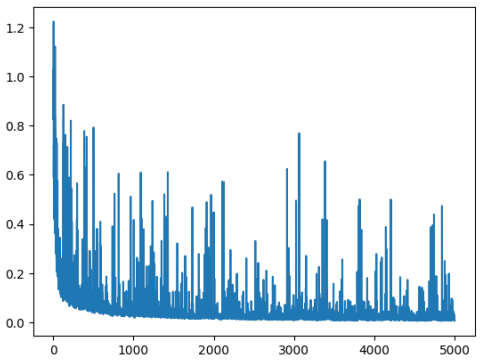
實驗結果

Loss regression:

DIP 串 DDPM



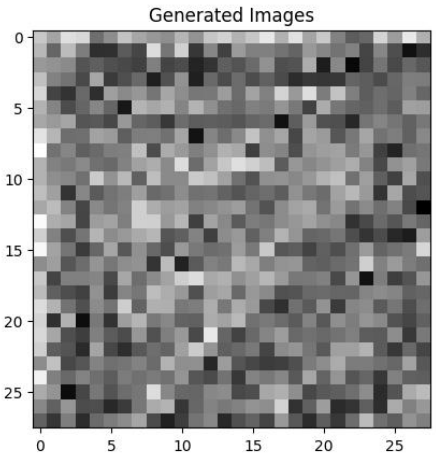
sole DDPM



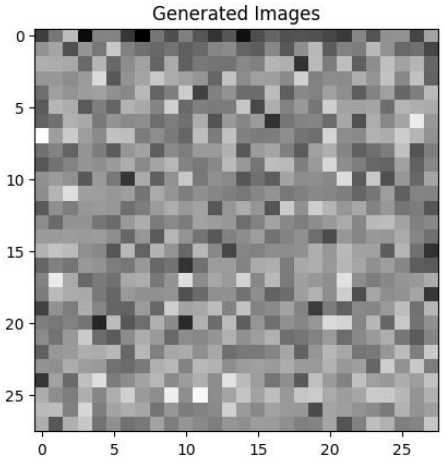
Loss 具有相似的趨勢

分別輸入隨機的 noise:

圖三:DIP 串 DDPM



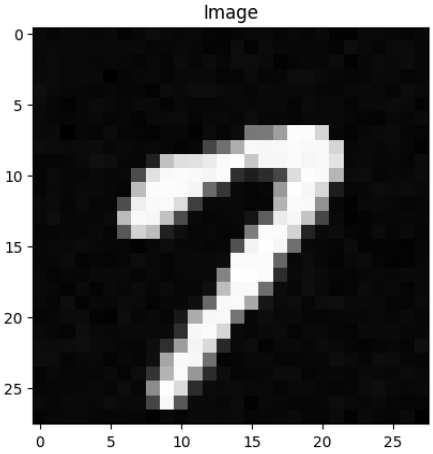
圖四:sole DDPM



可以看出是左邊，也就是 DIP 串 DDPM 效果比較好，勉強能看出”7”的輪廓

最後我使用 PSNR 讓圖三、圖四分別跟圖二進行計算

圖二: clean image from MNIST



Result(PSNR):

DIP 串 DDPM: 8.108132618880658

Sole DDPM: 1.3431491336455483

數值越大代表越好。

備註:PSNR 數值低我認為跟 dataset 的特性有關、MNIST 配色為黑白。

從結果上來看，串上 DIP 能在一定程度上讓 DDPM 表現更好，訓練成本更低。

(訓練 Epoch 更少)

Part 3: Ablation Studies and Analysis

我認為比較重要的超參數是 time schedule 的長度，因此我將 schedule 從 1000 調整成 501。備註四

備註四，寫到這才發現如果要從 time = 500 進行 diffusion inverse process 需要的 time schedule 長度為 501，前面是因為定義長度為 1000 因此從 500 開始 inverse 不會有問題。

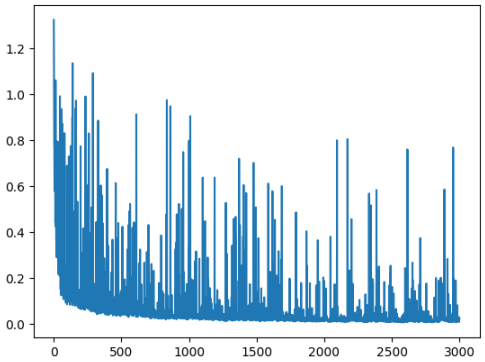
程式碼保有在 Main_schedule_modified.ipynb、Inference_modified_version1.ipynb、Inference_modified_version2.ipynb 中

在這裡出現了一個有趣的結果

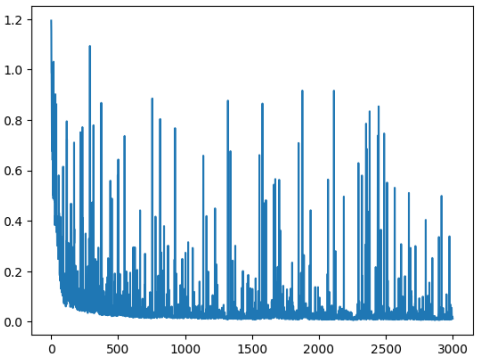
我訓練出了兩個模型:

Regression process

Model1:

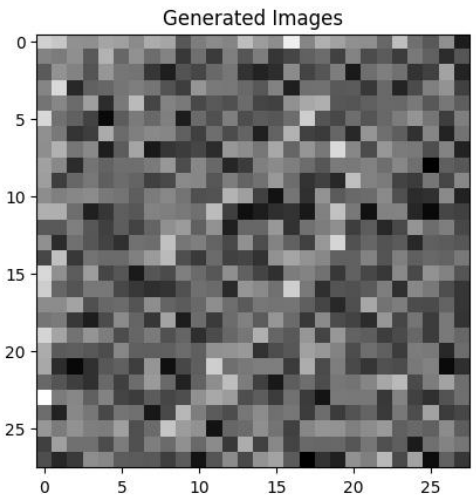


Model2:

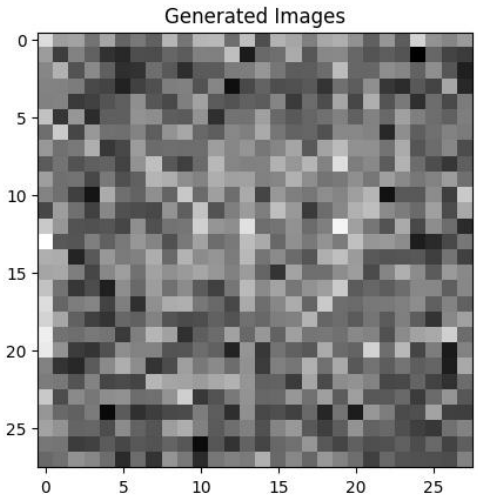


可以看出 model2 的 loss regression 更加不穩定

Output of model1:



output of model2:



PSNR value:

Model1：7.7997909067912

Model2：6.070183400575599

而 model2 的 output 品質也比不上 model1

而 model1、model2 都是基於 diffusion process time schedule 長度定義為 501 的情況下，使用跟 Part 2 中 DIP 串 DDPM 相同的 DIP，經過同一個演算程序訓練出來的，然而兩者卻具有顯著差異。

這種情況在 time schedule 為 1000 時沒有發生過，然而現在調整成 501 確有這個問題，我認為原因在於 schedule 長度變短導致的。

考慮這其中的物理意義

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

從 forwarding process 中，我們知道成功預測 Forward t 次之後的準確率和乘積 1~t 個定義在 forwarding process 中的 variance，還有初始圖片，以及 epsilon 有關。

但是當 schedule 從 1000 刪減成 500 時，一樣是從 t = 500 嘗試逆推 t = 0，但是

X0 的權重會比原來大(具體來說應該會大 2 的 t/2 次方倍)，因為從第 1~第 t 個 variance 的乘積變大，epsilon 的權重變小，變成更需要考慮 X0 而非當下預測的 noise，這不只反直覺邏輯上也沒有道理。

因此若減少 time schedule 的長度會導致無法穩定訓練出 DDPM(體質差別更大)，或者說模型比較不易收斂。