Theoretical Justification

我這一次實作的是第一個 example，也就是 Accelerating DDPM with DIP-based Initial Priors，實作的步驟如下

1. 先將 dip 設置好並進行訓練
2. 以訓練好的 dip 跑一次 mnist 的資料集並儲存結果下來
3. 設定 ddpm 的模型
4. 調整訓練過程，以前面 dip 的輸出來作訓練
5. 訓練結束，進行測試並儲存結果

而這樣會有效的原因，我認為是由於 dip 本身有辦法將圖片重要的部分特化下來，也就是說，在單獨訓練 ddpm 時，相對不重要的內容，像是一些背景的雜點，可以在前面被 dip 先處理掉，然後這樣作預期可以帶來兩種好處，第一個是能夠把重點突出，所以訓練效果會比單獨 ddpm 處理時還要好，第二個是少了這些多餘的內容後，ddpm 也可以更快速的抓到重點特徵，讓訓練速度可以加快，也可以加大訓練量讓他能適應更多種不同的情況。

但是這樣的做法仍然有一些限制，首先，由於要做兩個 model 的合併，需要整合許多部份，像是輸出跟 ddpm 的輸入就會需要妥善處理，否則無法運作，其次，由於訓練兩種模型，使得複雜度提高，如果沒訓練成功，需要調整的超參數內容，或是出問題的點，會較單獨 ddpm 時更難以確定，另外，ddpm 的訓練會十分仰賴 dip 的能力，若 dip 沒有訓練好，可能反而會使得 ddpm 訓練效果較單獨時更差。

Experimental Verification

與 dip 結合的 ddpm 和單獨的 ddpm 相比有諸多地方都有所改進，下列將一一列出:

1. 訓練速度

訓練速度上，我首先用了跟原版 ddpm 一樣的 3000 筆資料跑 30 個 epoch，-而速度改進相當多，於是我把訓練的資料量，調整為 mnist 完整的 60000 筆資料，而同樣都是 30 個 epoch 的情況下，兩者訓練速度分別如下

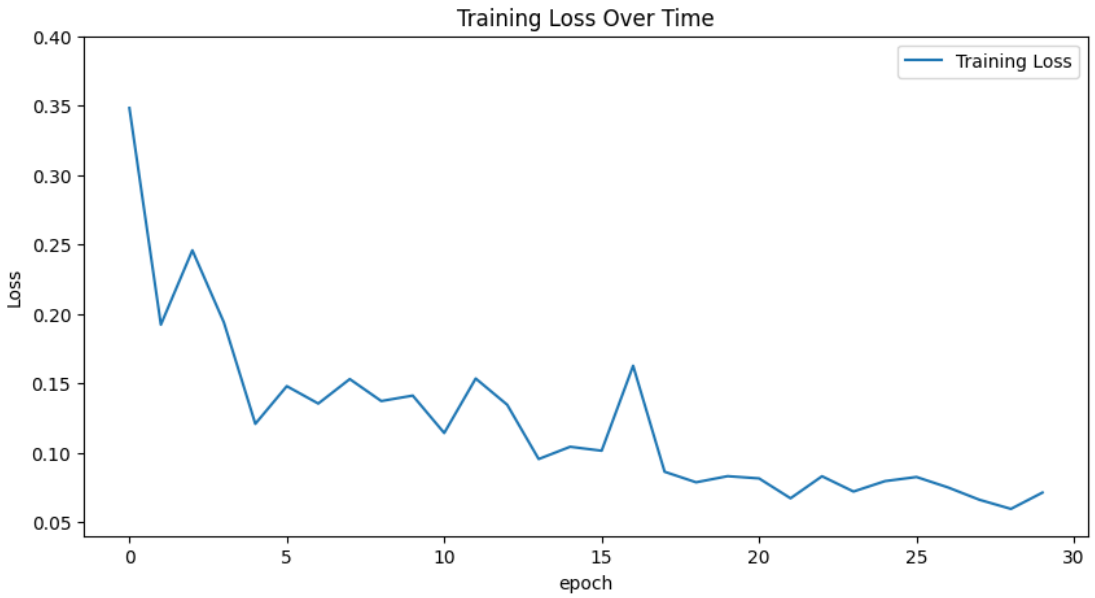
✓ 20 分鐘 52 秒 這是單獨 ddpm 的情況

✓ 11 分鐘 20 秒 這是與 dip 結合後的情況

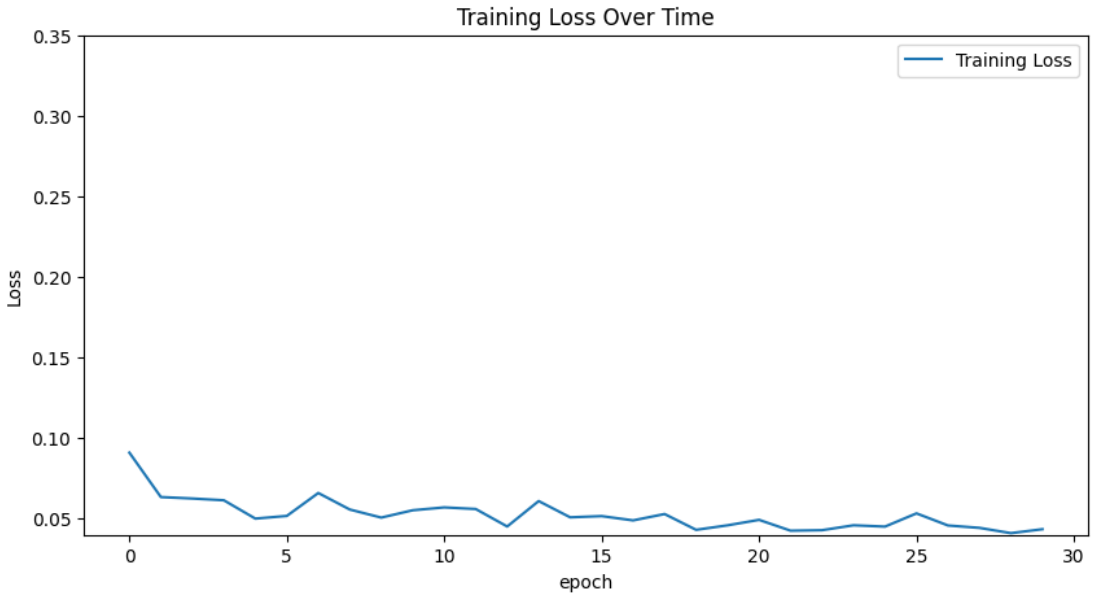
可看出訓練速度上，會有相當大的改進，而我後來將訓練的 epoch 增加到 50 個，兩者的時間才到差不多的長度

2. loss

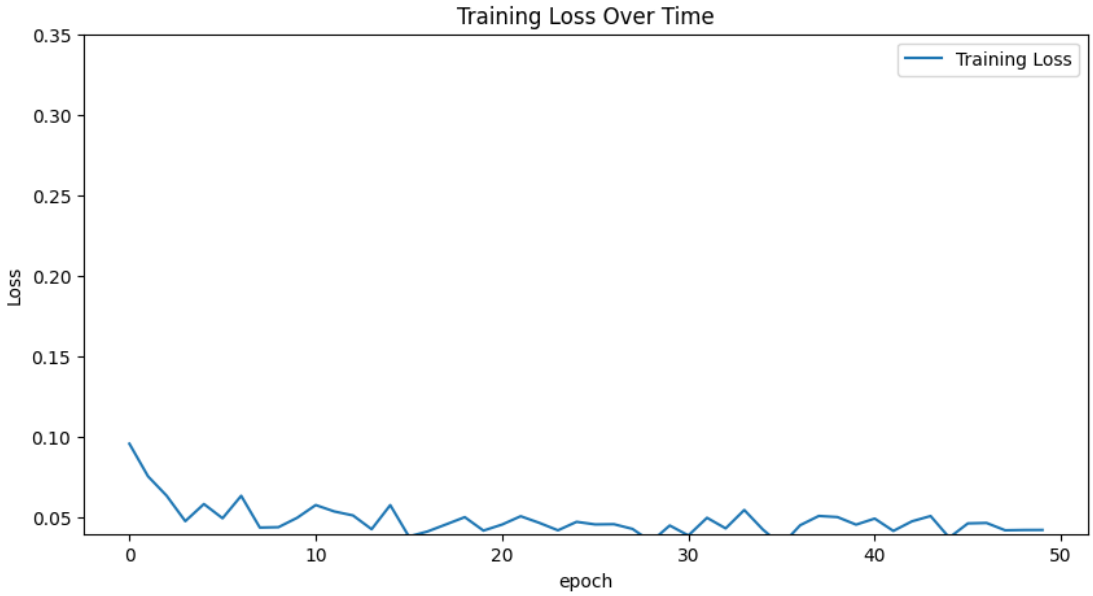
單獨 ddpm:



合成後:



合成後 50 epoch

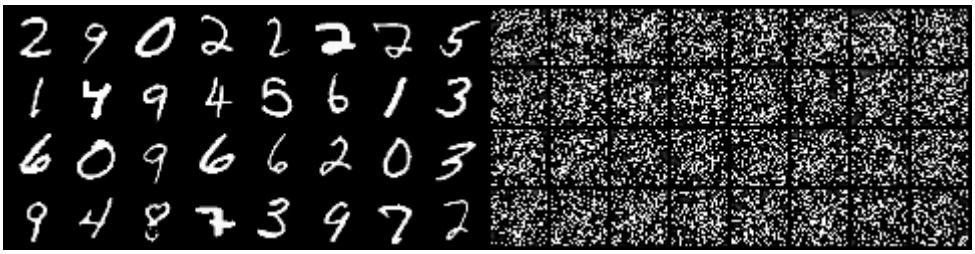


在上面可看到，單獨 ddpm 時的 loss 從一開始就較高，且 loss 收斂的速度，以及最後的 loss 表現也都較差，單獨 ddpm 時，loss 一開始是 0.35 左右，而合成後的，則是 0.1，且最後的結果，單獨 ddpm 的落在 0.07 附近，而合成的則是 0.05

3. 生成結果

單獨 ddpm:

目標/結果:



評估分數:

PSNR: 14.823285102844238 dB
SSIM: 0.35987159609794617

由圖片部分來看，若放大後可以略為看出輪廓，但相當不明顯，而評估的分數也可看出，不管是哪一種，都與真正的圖片有著明顯的差異

合成後 30 epoch:

目標/結果:

評估分數:

PSNR: 18.058216094970703 dB
SSIM: 0.41578084230422974

圖片的部分，輪廓有更明顯，但仍然沒有到非常清楚，而評估分數的部分，雖沒有到非常高，但已經較原本的多了一些

合成後 50epoch:

目標/圖片:

評估分數:

PSNR: 18.39859390258789 dB
SSIM: 0.41747990250587463

圖片的部分，已經可以看出較明顯的輪廓，而評估分數的部分， 可看出與 30epoch 時

雖然三者分數都沒有到非常高，但已經可以看得出來，混合後效果確實比混合前還要好

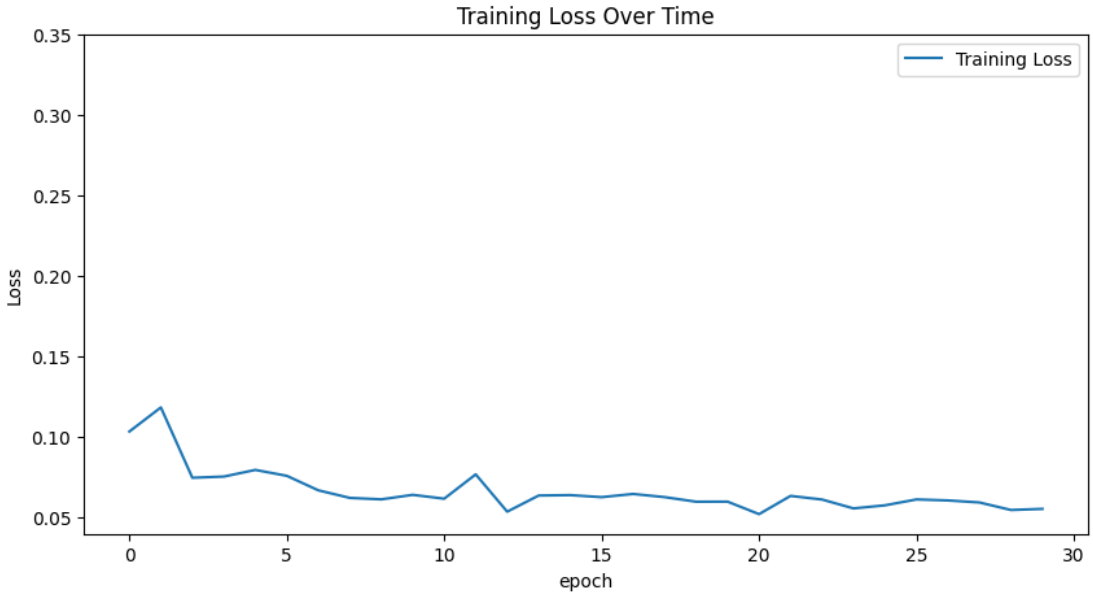
由上述可看出，混合後，不僅是訓練的速度提高，訓練的結果也會更好。

Ablation Studies and Analysis

均以 30 epoch，合成後的 model 為標準，在前面我們已經看過了增加 epoch 的結果，而之後，我也對 time schedule，以及 learning rate 都分別做了調整和測試，原本的參數為 learning_rate = 0.0001，timesteps = 1000

Step 500:

Loss:



生成圖片

目標/結果:

評估分數:

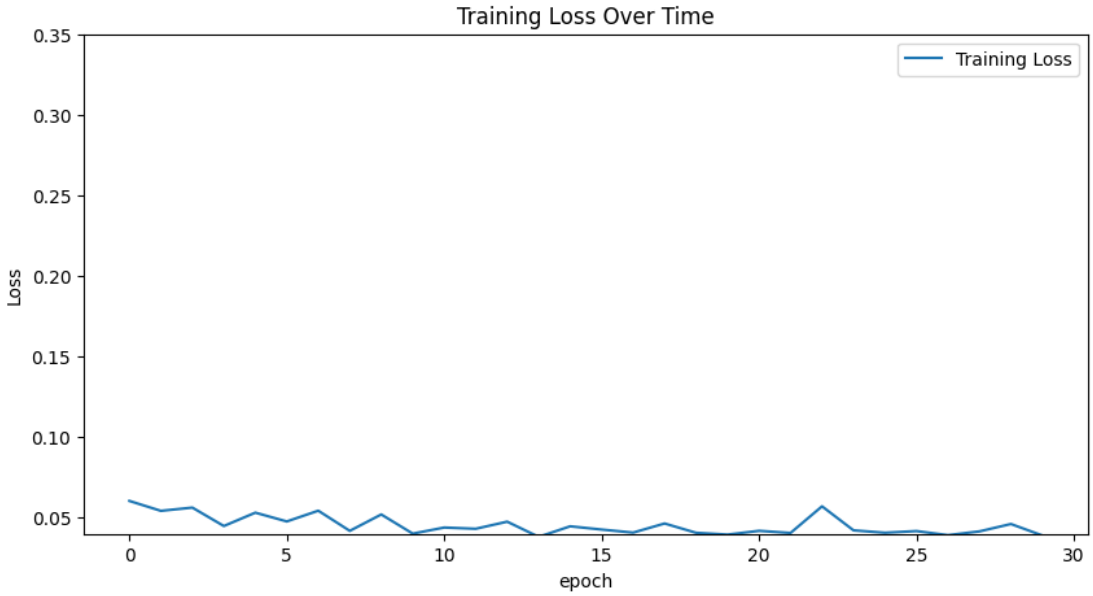
```
PSNR: 16.675128936767578 dB
SSIM: 0.3580429255962372
```

由上面可看出，若 step 設的太少對訓練會有一定程度的影響，不僅 loss 較高，評估分數也低了不少，而這部份我認為是由於 step 就是在訓練過程中增加

以及去除噪聲的步驟數，而若經過較多步數，模型的穩定度以及品質都會有所提高，因此減少到 500 才會讓訓練結果變差，但是還有一點可以提的是，訓練的時間也會受 step 影響，雖然在這次只大概差了 1 分鐘，但這代表，step 不是越高越好，若需要把時間考量進去，則要選擇適中的 step

Learning rate 0.001:

Loss:



生成圖片

目標/結果:

評估分數:

```
PSNR: 18.58850860595703 dB
SSIM: 0.4348922073841095
```

在把 learning rate 調高之後，可以看到 loss 明顯的更低了，開頭就只有約

0.06，而最後更是到了 0.039，而輸出結果的部分，分數甚至比 50 個 epoch 時還要高，對於 learning rate 的特性前面幾個 project 都有進行過分析了這邊先不再贅述，而 0.001 顯然是較 0.0001 更適合這次 model 的學習率

Github link

https://github.com/gino1203/gai-project4