Generative Models for Visual Signals – Assignment

F4092269 陳冠廷

https://github.com/larrychen20011120/Guided-DIP

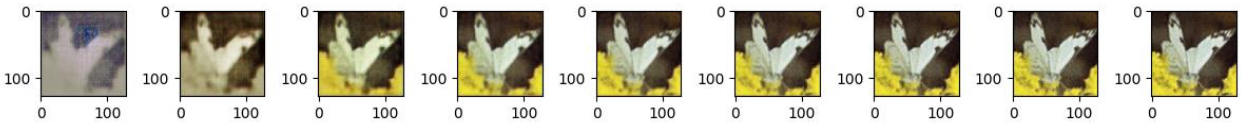
一、 問題定義

Deep Image Prior (DIP) 是一種僅訓練在單一圖片即可修復影像的想法，透過原論文的實驗結果，作者發現給定涵蓋 Noise 的影像時，使用一般的卷積神經網路並利用 MSE Loss 來擬合原圖時，網路會傾向先建構出影像低頻的部分，也就是輪廓、邊框等，隨時間愈久模型就會學習出愈多的影像細節，最後過擬合成原本的 Noise 圖片(圖一)，但是原始的 DIP 方法中無法有效地確定最佳的停止點，大多都是使用 MSE Loss 的變化來確定比較接近原圖的時間點。

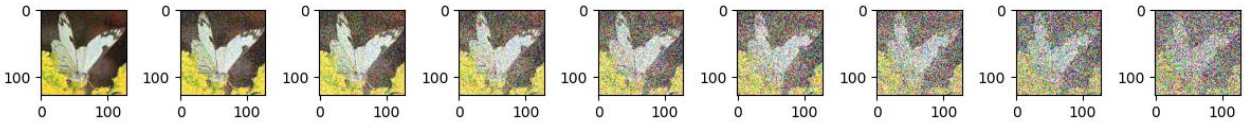
此次作業希望能夠結合 DDPM 的加噪想法來找出比較好的停止方法，或是找出比較早結束的停止點。DDPM 假設加噪T次以後的圖片會趨於常態分佈的 Noise，而且每次加噪過程都會上一次加噪後的圖片以一定比率加上新的高斯 Noise(圖二)，在經過高斯的獨立公式可以得出以下的加噪公式，只需一步即可取出不同時間點的含噪圖片：

$$x_t = \sqrt{\alpha_t} \cdot x_0 + \sqrt{1-\alpha_t} \cdot N(0,1)$$

由於 DIP 在修復圖片時會傾向於修復出低頻的內容，而噪音通常是高頻的資訊，因此使用不同加噪程度的圖片可以產生階層式的圖片特徵，從較少的輪廓到接近原圖的輪廓，有機會讓 DIP 可以循序漸進地學習而不會 overfitting 到 Noise 上。



圖一、DIP 訓練過程的 snapshot，可以看出蝴蝶的輪廓先被重建，接著才是依些較鮮艷的圖片細節。



圖二、DDPM 加噪的過程示意圖，不同時間點的圖像都是原圖與高斯 Noise 的加權平均，愈後期圖片的細節會先被噪音蓋掉，而輪廓仍然明顯。

二、 提出的方法 – DDPM Guided DIP

✓ DDPM Guided DIP：如何結合 DDPM 和 DIP

✧ Time Generator

我設定一個$count$代表要產生多少個 DDPM forward 過程的中間圖片，並在指定的時

間區間中，以等間隔的方式篩選出$count$個 DDPM 的加噪過程圖片。例如：指定區間為[0, 500]取出 6 個中間圖片就會有[0, 100, 200, 300, 400, 500]不同時間的圖片作為階層式的特徵。

✧ Iteration Generator

考慮到 DIP 會優先學習出圖片輪廓，因此 DIP 應該漸進地學習輪廓，由不清晰的輪廓到愈來愈接近原圖，因此我反轉了 forward 過程的圖片，變成由最多 noise 到最少 noise。此外，noise 愈多的圖片可能會使模型容易學習到 noise，因此根據 noise 的多寡我賦予這些圖片訓練次數反向的權重，也就是愈多 noise 的圖片應該訓練愈少次數，計算方法如下：

$$w_i = \frac{e^{-c\cdot\beta_i}}{\sum_{k=1}^{count} e^{-c\cdot\beta_k}}, \qquad i = 1, 2, \dots, count$$

$$n_i = n \cdot w_i, \qquad i = 1, 2, \dots, count$$

其中w_i代表每個 noise 程度的訓練次數比重，其與 noise 程度成反向關係，c 是一個縮放係數使不同 beta 的差異可以更大（實作中我將 linear scheduler 的 c 值設為 250、cosine scheduler 的 c 值為 400），而 n 表示所有 Guided-DIP 應該要訓練的總次數，n_i代表第 i 張噪音圖應該訓練的次數。

有了這些個別的訓練次數和每一張 noise 圖片後，我進一步地修改了 DIP 的訓練流程，使其能夠受到不同時期的圖片引導，並以下方的 pseudo code(圖三)來表示。我沒有修改計算損失函數的方法，而是變動訓練的流程讓不同程度的 noise 圖可以在對的時間被放入模型進行訓練，更具體的流程與架構則如(圖四)所呈現。

Training the original DIP

Input: x_0, n, DIP_θ
Output: *the approximated clean image*

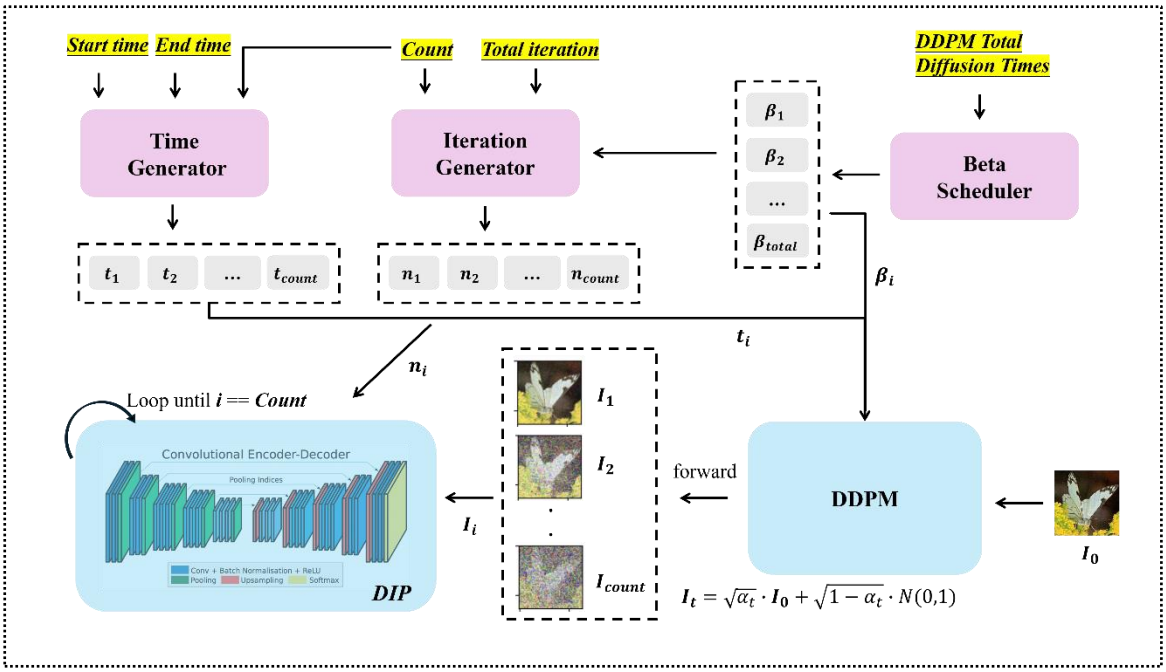
iteration ← *0*
$z \leftarrow N(0, I)$
repeat
  $\hat{y} \leftarrow DIP_\theta(z)$
  $loss = mse(x_0, \hat{y})$
  update θ *by* $loss$ *backprop*
  iteration ← *iteration+1*
until *iteration* $== n$
return $DIP_\theta(z)$

Training DDPM Guided DIP

Input: $x_0, x_1, .., x_{count}, n_0, n_1, \dots, n_{count}, DIP_\theta$
Output: *the approximated clean image*

cnt← *0*
$z \leftarrow N(0, I)$
repeat
  iteration ← *0*
  repeat
    $\hat{y} \leftarrow DIP_\theta(z)$
    $loss = mse(x_c, \hat{y})$
    update θ *by* $loss$ *backprop*
  until *iteration* $== n_c$
  cnt ← *cnt+1*
until $cnt == count$
return $DIP_\theta(z)$

圖三、一般的 DIP 訓練方法與 DDPM-Guided DIP 的訓練方法。



圖四、DDPM-Guided DIP 完整的流程圖與架構。螢光筆標註的部分為可以設定的參數，圖片中的 backbone 則是用 SegNet 為範例。

✓ **Beta Scheduler 的選擇**

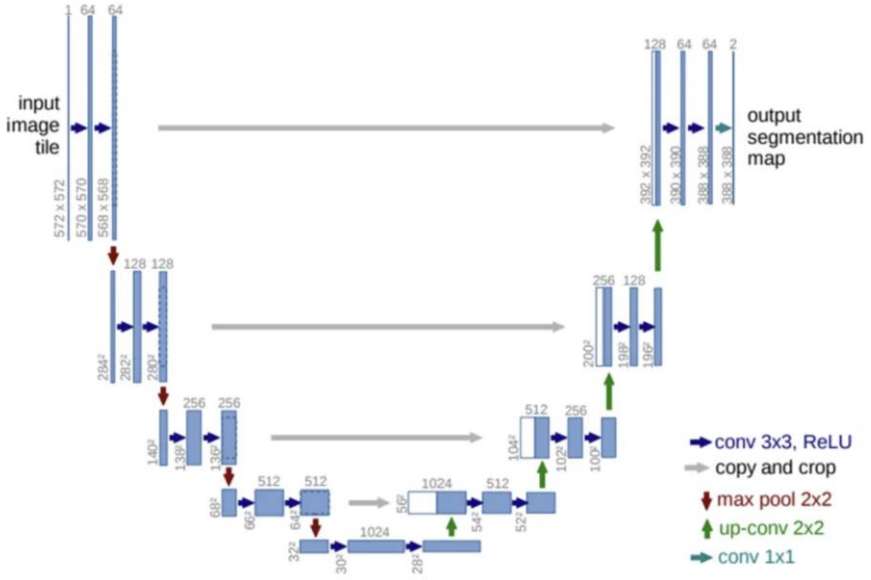
✧ Linear Scheduler

✧ Cosine Scheduler

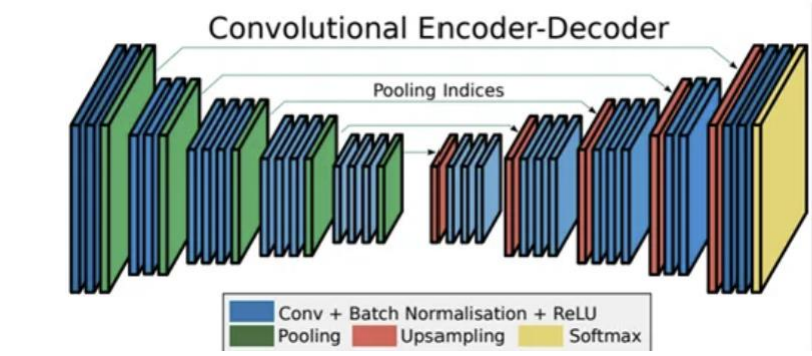
✓ **DIP Backbone 的選擇**

為了確定方法的可行性，並找出最佳的模型架構，除了實驗原始 DIP 中的 Unet 架構外，我也尋找其他適用於 Image Segmentation 的模型當作 DIP 的 backbone model。這次實驗分別採用了以下兩種模型結構：

✧ Unet



✧ SegNet



✓ **評價指標**

✧ PSNR：計算修復影像與原始影像的差異。

✓ **實驗設定**

為了使實驗建立在相同的比較基準，我統一將所有實驗的基本參數設置在下表的實驗設定中，進一步比較其最後的 PSNR 分數與最佳的 PSNR 分數，同時確認最佳點與最終收斂的時間的差距來看出早停點。針對原始 DIP 的方法，我把目標的學習影像設定為 DDPM 加最小噪聲的圖片。

設定類型	實驗設定	設定數值
一般設定	Image Size	(128, 128)
DDPM Forward	Beta Start	0.0001
	Beta End	0.02
	Total Steps	1000
DIP-based Training	Number of Training Steps	2000
	Learning Rate	0.001
	Optimizer	Adam
Guided-DIP	Number of Timestamp	20
	Start Timestamp	500
	End Timestamp	40

在上述相同的設定下，我比較了 DIP 與 Guided-DIP 在不同 backbone 下的差異。除此之外，我也進一步分析不同 DDPM 的 beta scheduler 的影響力。

三、 實驗結果

✓ **綜合比較**

從下表的綜合比較中，我得出了以下的結果結論在影像修復中。在我的實驗配置中，使用原論文的 UNet 並不是最好的 backbone，反而使用 SegNet 可以得到更好的效果。此外我也發現原始的 DIP 方法確實需要提前設定終止時間，像是 SegNet 最好的修復結果出現在第 378 個 step、UNet 則是落在第 1491 個 Step。反觀我提出的整合方式，幾乎不太需要設定最佳停止點，因為最好的結果都落在後半部，只需要跑完整個 Guided-

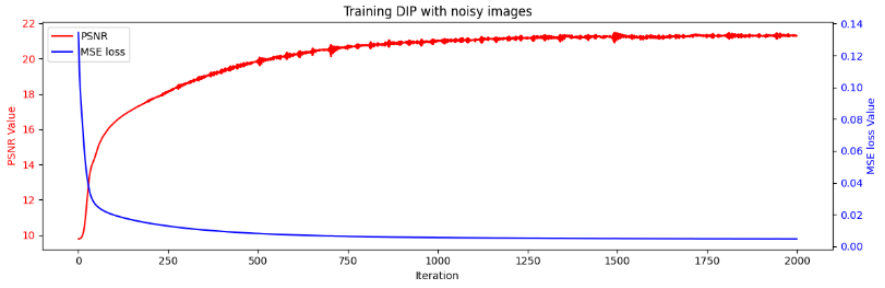
DIP 即可取得不錯的修復結果。另外，值得一提的是 beta scheduler 也可以提升我們修復圖像的品質，在所有實驗中，使用 cosine scheduler 都比使用 linear scheduler 得出更好的 PSNR 分數，而且都比原始的 DIP 還要好。

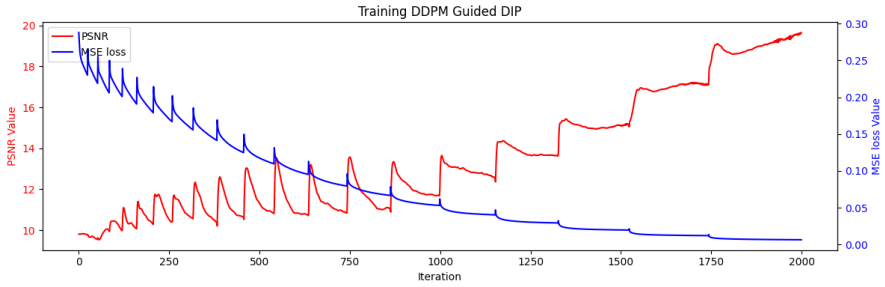
Method	Backbone	Beta scheduler	Final PSNR	Best PSNR	Best time	Final Loss
DIP	UNet		21.29	21.48	1491	0.0047
	SegNet		26.06	26.94	378	**0.0001**
Guided-DIP	UNet	Linear	19.65	19.65	**1999**	0.0064
		Cosine	**21.35**	**23.91**	1857	**0.0022**
	SegNet	Linear	21.26	22.62	1761	0.0022
		Cosine	**26.96**	**26.95**	**1981**	0.0010

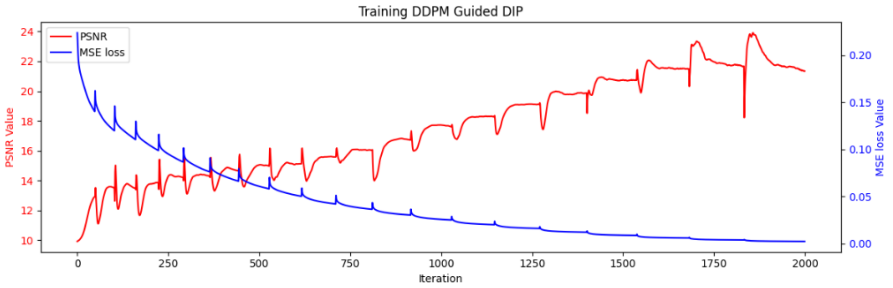
✓ **訓練過程的變動**

✧ 以 UNet 為 backbone

以下三張圖片分別為一般的 DIP、linear scheduled Guided-DIP 和 cosine scheduled Guided-DIP 在訓練過程的 PSNR 與 MSE Loss 的變化過程。可以看到一般的 DIP 很快就可以收斂到很好的分數，但是其後會有小震盪就是在你和高頻資訊而學習到 Noise。然而，我提出的方法會漸進式的學習，所以 loss 和 psnr 都會有明顯的跳動，每一次跳動皆代表新的圖片被賦予為 target image，透過這樣的方式可以引導 DIP 學習出漸進式的資訊，而不會一下子就過擬合到 Noise 上面。

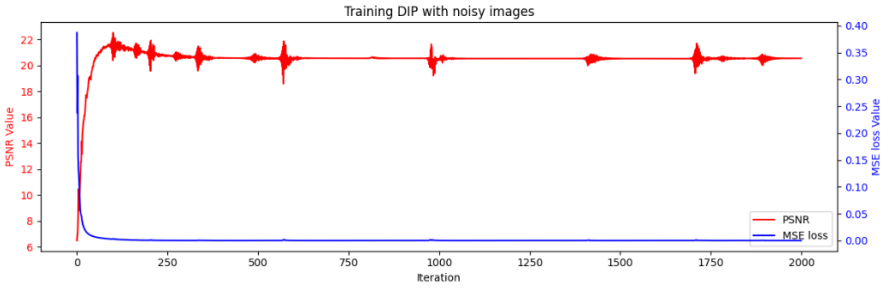


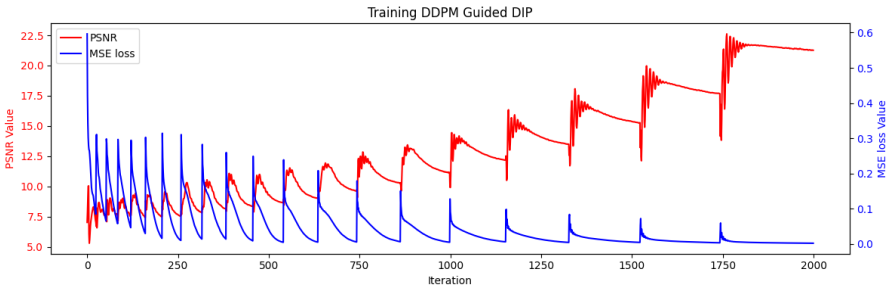


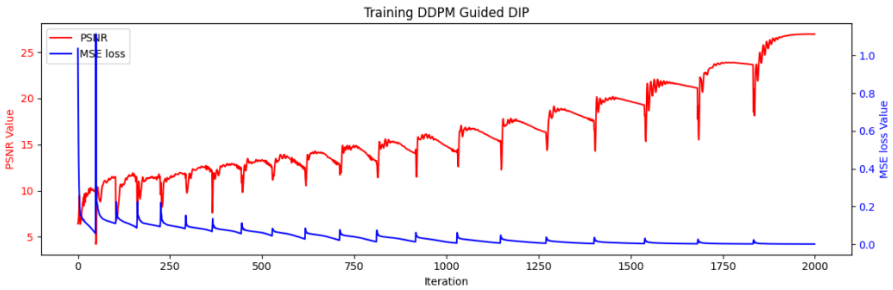


✧ 以 SegNet 為 backbone

下面三張圖與 UNet 也是大同小異，主要的差別在 SegNet 可以更快收斂到結果，因此早停點更為前面。而透過我的方法可以不用設定早停點仍保證後期的修復品質不輸給原始的 DIP。



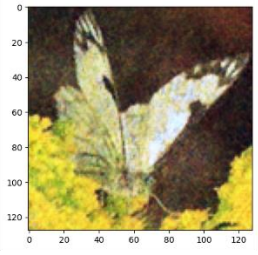




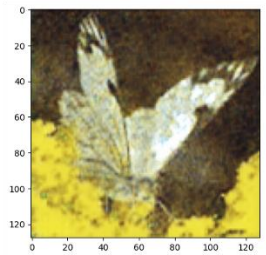
✓ **方法之間的最後修復結果**

下方為各種方法的修復結果，可以明顯看到與原圖仍有非常大的差異。其中 UNet Linear Guided-DIP 在 PSNR 的分數是最低的，從生成的結果可以看到並沒有修復到綠色的草，且整體的背景也偏模糊。此外 UNet 方法普遍生成的圖片也偏暗沉，蝴蝶幾乎都變成灰色的了。整體看來 SegNet 的修復品質比 UNet 好，不論是亮度還是修復的細節(如：草、蝴蝶的紋路、花朵、背景的亮暗)。

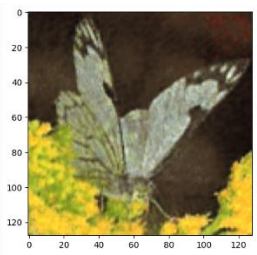
Original Image



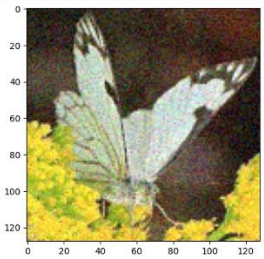
UNet DIP



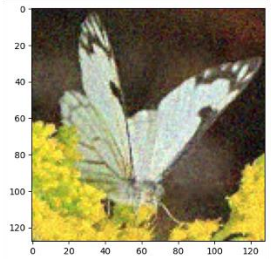
UNet Linear Guided-DIP



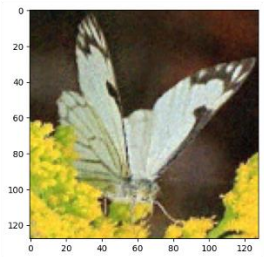
UNet Cosine Guided-DIP



SegNet DIP



SegNet Linear Guided-DIP

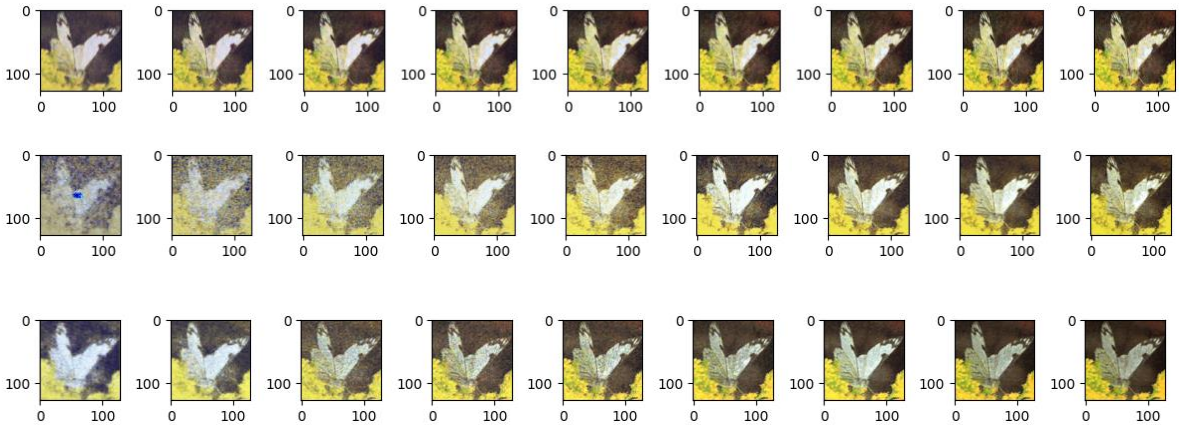


SegNet Cosine Guided-DIP

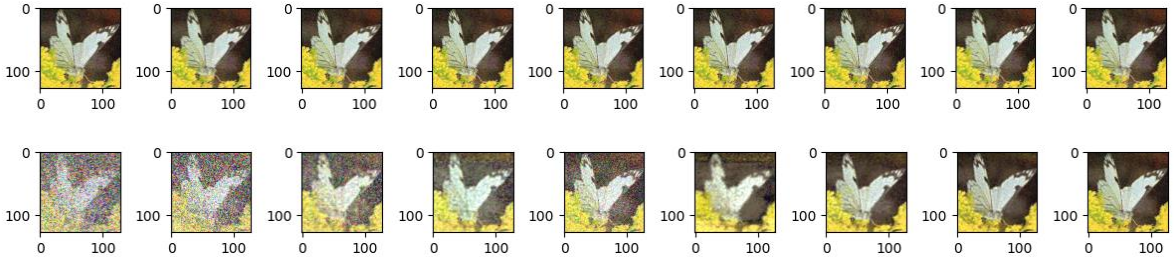
✓ **方法之間的修復過程**

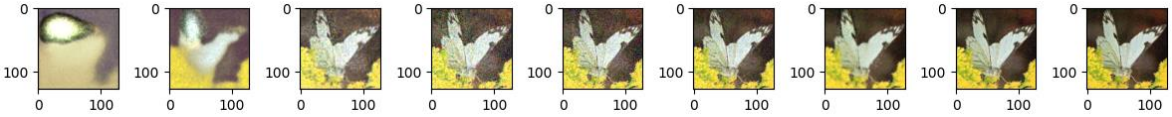
不論是 UNet 還是 SegNet，我提出的方法可以看出修復過程中會漸進式地修復圖像輪廓，而且愈來愈明顯，透過這樣階層式的特徵引導 DIP 學習而不會一下子就過擬合到 Noise 上。透過 cosine scheduler 可以讓前期 noise 的程度不那麼大，而影響到前期 overfitting 至過多的 noise，才能真正實驗漸進式的學習，可以看到 cosine scheduler 前期學習的輪廓比較沒有那麼多噪音。

✧ UNet



✧ SegNet





四、 總結與分析

考量到原始 DIP 在修復影像時會容易 overfitting 到有雜訊的圖片，先前的研究多會設定早停點來確定學習的終止點，但是直接使用 PSNR 來尋找早停點是非常不合理的，因為在影像修復任務中，往往不會有真實的乾淨圖片。除此之外，不一樣的 backbone 模型會有不一樣的早停點，UNet 的早停點會偏後期但 SegNet 則是在前期。有鑑於此，我設計了一種結合 DDPM 的加噪過程來引導 DIP 學習，透過產生過程中的不同程度的 Noise 圖片，並賦予其不一樣的訓練次數，可以漸進式地引導 DIP，同時解決了尋找早停點的需求。

在我們的實驗中，證實了我方法的可行性，甚至在採用適當的 scheduler 後，達到超越 DIP 的表現。值得一提的就是，我的方法不再需要設置早停點，而是執行完所有的訓練流程後，結果自然會收斂到相對好的結果。