Low Light Image Enhancement with Wavelet-Based Diffusion Models

Anushree Chaudhary

1 Introduction

This report documents the implementation details and new ideas explored for the paper titled *Low Light Image Enhancement with Wavelet-Based Diffusion Models*. The model is developed in PyTorch and trained to enhance low-light images using a U-Net based architecture with diffusion processes and wavelet regularization.

2 Dataset Description

The LOLv1 dataset was used for training and evaluation. It comprises 498 paired images, with one image in low-light conditions and the other in normal-light, allowing the model to learn the transformations required to enhance low-light images effectively.

3 Standard Implementation Details

The low-light image enhancement model was trained using the LOLv1 dataset, which consists of 498 real-world low-light and normal-light image pairs. Training was conducted for 500 epochs with a batch size of 8. The Adam optimizer was used with an initial learning rate of 1×10^{-4}, decaying by 0.8 every 50 steps. Validation was performed every 1000 iterations, and checkpoints were saved every 50 epochs. The model utilizes a U-Net architecture with a linear beta schedule during the diffusion process, consisting of 200 timesteps. Epoch times and losses (at every 10 steps) were logged for detailed analysis.

3.1 Hardware Setup

- **CPU:** 13th Gen Intel(R) Core(TM) i7-1335U — Training on 3 data points with 5-10 epochs to check the running of the codebase. It had slow processing.
- **GPU:** NVIDIA GeForce RTX 3090 — The GPU significantly accelerated the training process, enabling efficient training over 500 epochs with regular checkpointing and logging.

4 New Idea Implementation

4.1 LPIPS-Based Perceptual Loss

LPIPS (Learned Perceptual Image Patch Similarity) enhances perceptual quality by comparing deep features rather than relying solely on pixel-level accuracy. This approach allows for the retention of essential textures and fine details, aligning the results more closely with human visual perception.

4.1.1 Implementation

An additional requirement, `lpips==0.1.4` was installed. The LPIPS-based loss was incorporated into the code (in `ddm.py`). It was added to the photo loss component as follows:

$$\text{photo\_loss} = \text{content\_loss} + \text{ssim\_loss} + \lambda_{\text{lpips}} \times \text{lpips\_loss}$$

where λ_{lpips} is a tunable parameter. Experiments were conducted with $\lambda_{\text{lpips}} = 0.3$ and $\lambda_{\text{lpips}} = 0.5$. Tuning λ_{lpips} adjusts the emphasis on perceptual quality in the model's outputs: higher values prioritize visually realistic textures and edges, while lower values focus on pixel-level accuracy. This helps balance perceptual similarity with other objectives, depending on the desired output quality.

4.2 Wavelet Coefficient Regularization

Wavelet coefficient regularization is used to preserve high-frequency components, ensuring sharp, sparse features in the output. This method is beneficial for retaining key edges and textures, which are essential in high-quality image enhancement.

4.2.1 Implementation

An L1 loss for high-frequency components was incorporated into the code (in `ddm.py`) as follows:

$$L_{L1} = \sum_{k=1}^{K} (\|V_{k,\text{high}}\|_1 + \|H_{k,\text{high}}\|_1 + \|D_{k,\text{high}}\|_1)$$

where K is the number of wavelet decomposition levels, and $V_{k,\text{high}}, H_{k,\text{high}}, D_{k,\text{high}}$ represent the vertical, horizontal, and diagonal high-frequency components, respectively, at the k-th wavelet decomposition level.

It was added to the total loss component with a tuning parameter β (taken as 0.000001 to prevent collapse of loss value).

4.3 Cross Attention Layers in High-Frequency Restoration Module

To further enhance the high-frequency restoration, more cross attention layers were added between different high-frequency components (horizontal, diagonal, and vertical). Initially, two cross attention layers were present: one between horizontal and diagonal components, and another between diagonal and vertical components. These were concatenated to form a new diagonal component, which was then concatenated with the existing horizontal and diagonal components as shown in the Figure 1.

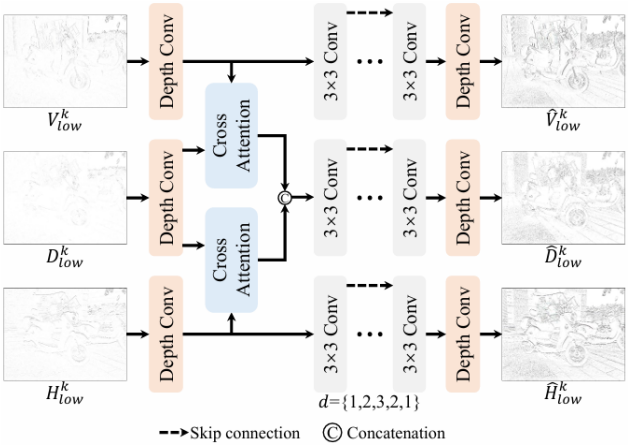


Figure 1: The detailed architecture of the existing high-frequency restoration module. Depth Conv denotes the depth-wise separable convolution.

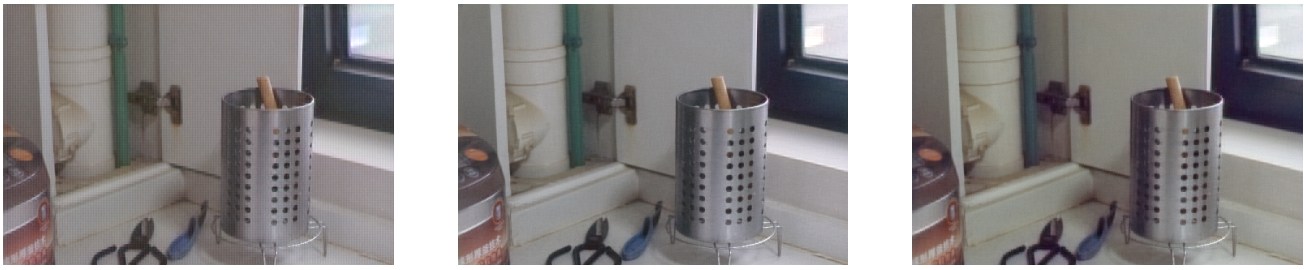
4.3.1 Implementation

Initially, I expanded the structure by adding a cross attention layer between horizontal and vertical components to create new horizontal and vertical components, similar to the way the new diagonal component was made in the existing structure. These new components were concatenated, but the resulting image had a grid-like pattern in the background (most likely high-frequency artifacts).

Another implementation was to add two additional layers of cross attention, further on the new diagonal component and the existing horizontal and vertical components. The final diagonal component was concatenated with the original horizontal and vertical components. Despite these modifications, grid-like artifacts persisted, though they were slightly reduced.

Gaussian blurring was also applied after the cross attention layers to smooth out artifacts, but the grid pattern remained visible upon zooming in, leading to lower PSNR values.

The code changes were made in `mods.py`.



(a) Initial Implementation (b) Second Implementation (c) Gaussian Blurring Addition

Figure 2: Grid-like pattern formation in the images on modification of cross attention structure.

4.4 Results

4.4.1 Loss Plots

The loss plots for the different models are fairly similar. The noise, photo and frequency losses in the training steps are plotted in the Figure 3. It can be seen that the losses are deceasing as the number of iterations increase.

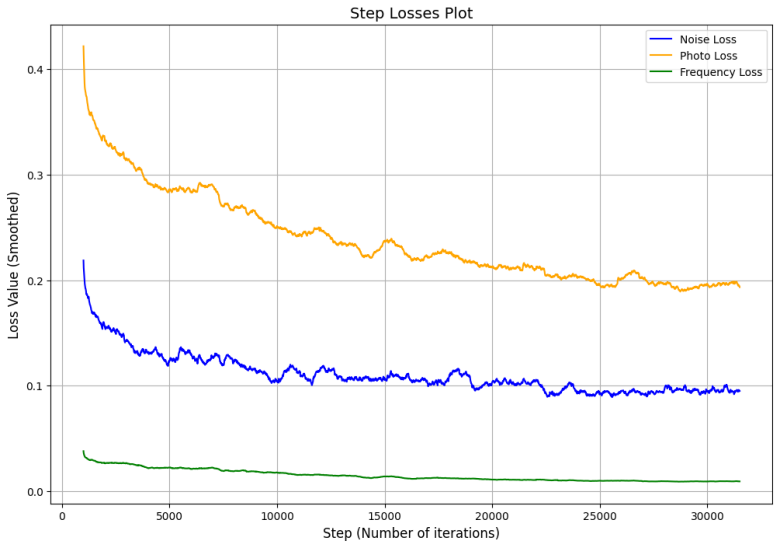


Figure 3: Loss reduction over training steps (smoothed over a rolling window of size = 100)

The wavelet model has an added high frequency loss component which shows a similar trend.

4.4.2 Average Epoch Training Time

Average Epoch Training Time (in seconds)					
Normal Model	**LPIPS** $\lambda_{\text{lpips}} = 0.3$	**LPIPS** $\lambda_{\text{lpips}} = 0.5$	**Wavelet Loss**	**Cross Attention (No Blur)**	**Cross Attention (With Blur)**
28.99	29.57	29.55	31.75	30.05	30.34

Table 1: Average epoch training time (in seconds) across different models

4.4.3 PSNR and SSIM

The PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) metrics were used to compare the outputs of the models at regular checkpoint after every 50 epochs (training data as input) with the corresponding ground truth.

Epoch	**PSNR (in dB)**					
	Normal Model	**LPIPS** $\lambda_{\text{lpips}} = 0.3$	**LPIPS** $\lambda_{\text{lpips}} = 0.5$	**Wavelet Loss**	**Cross Attention (No Blur)**	**Cross Attention (With Blur)**
50	20.58	20.88	20.85	20.59	20.26	19.98
100	18.65	21.68	21.79	21.48	20.89	20.06
150	22.31	22.69	22.65	22.27	21.01	21.47
200	23.31	22.69	22.86	22.74	21.56	22.13
250	23.73	23.45	23.97	24.08	22.60	22.60
300	24.28	24.13	24.04	24.02	22.24	22.81
350	24.23	24.21	24.68	24.57	22.61	22.78
400	24.82	24.97	25.15	24.70	22.83	21.47
450	25.02	25.12	24.60	25.01	23.01	20.56
500	25.45	25.59	**25.63**	24.93	23.32	20.40

Table 2: Average PSNR values (in dB) across different ideas at regular checkpoints

Epoch	**SSIM**					
	Normal Model	**LPIPS** $\lambda_{\text{lpips}} = 0.3$	**LPIPS** $\lambda_{\text{lpips}} = 0.5$	**Wavelet Loss**	**Cross Attention (No Blur)**	**Cross Attention (With Blur)**
50	0.82	0.83	0.83	0.82	0.74	0.61
100	0.79	0.84	0.85	0.84	0.73	0.75
150	0.85	0.85	0.85	0.85	0.81	0.81
200	0.86	0.86	0.86	0.86	0.77	0.84
250	0.86	0.86	0.86	0.86	0.76	0.84
300	0.87	0.87	0.87	0.87	0.68	0.84
350	0.87	0.87	0.87	0.87	0.63	0.73
400	0.87	0.87	0.87	0.87	0.60	0.44
450	0.87	0.87	0.87	0.87	0.55	0.34
500	0.87	0.87	0.87	0.87	0.59	0.32

Table 3: Average SSIM values across different models at regular checkpoints

The model with LPIPS-based perceptual loss shows the best results with an approximate 0.2dB increase in PSNR and similar SSIM and training times (on training over 500 epochs).

Normal Model

(a) Epoch 100

(b) Epoch 300

(c) Epoch 500

LPIPS $\lambda_{\text{lpips}} = 0.3$

(d) Epoch 100

(e) Epoch 300

(f) Epoch 500

LPIPS $\lambda_{\text{lpips}} = 0.5$

(g) Epoch 100

(h) Epoch 300

(i) Epoch 500

Wavelet Loss

(j) Epoch 100

(k) Epoch 300

(l) Epoch 500

Cross-Attention (No Blur)

(m) Epoch 100

(n) Epoch 300

(o) Epoch 500

Cross-Attention (With Blur)

(p) Epoch 100

(q) Epoch 300

(r) Epoch 500

Low-light and Ground Truth

(s) Low-light Image

(t) Ground Truth Image

Figure 4: Comparison of model outputs at epochs 100, 300, and 500 across different models