Pyramid Diffusion Models for Low-light Image Enhancement

Dawei Zhou, Zongxin Yang, Yi Yang
ReLER, CCAI, Zhejiang University
{zdw1999, yangzongxin, yangyics}@zju.edu.cn

Abstract

Recovering noise-covered details from low-light images is challenging, and the results given by previous methods leave room for improvement. Recent diffusion models show realistic and detailed image generation through a sequence of denoising refinements and motivate us to introduce them to low-light image enhancement for recovering realistic details. However, we found two problems when doing this, i.e., 1) diffusion models keep constant resolution in one reverse process, which limits the speed; 2) diffusion models sometimes result in global degradation (e.g., RGB shift). To address the above problems, this paper proposes a Pyramid Diffusion model (PyDiff) for low-light image enhancement. PyDiff uses a novel pyramid diffusion method to perform sampling in a pyramid resolution style (i.e., progressively increasing resolution in one reverse process). Pyramid diffusion makes PyDiff much faster than vanilla diffusion models and introduces no performance degradation. Furthermore, PyDiff uses a global corrector to alleviate the global degradation that may occur in the reverse process, significantly improving the performance and making the training of diffusion models easier with little additional computational consumption. Extensive experiments on popular benchmarks show that PyDiff achieves superior performance and efficiency. Moreover, PyDiff can generalize well to unseen noise and illumination distributions. Code and supplementary materials are available at https://github.com/limuloo/PyDiff.git.

1 Introduction

Low-light images suffer from noise bursts, and recovering ideal normal-light images from them is a long-studied problem. Thanks to the development of deep learning, many effective methods have been proposed. LLNet [Lore et al., 2017] and SID [Chen et al., 2018] show the power of neural networks by training them on lots of paired data. According to the Retinex theory [Land, 1977], KIND [Zhang et al., 2021], and RetinexNet [Wei et al., 2018] decompose the illumination and reflectance map through a well-designed loss. To handle this highly ill-posed problem, LLFLOW [Wang et al., 2022] introduces normalizing flow models [Kingma and Dhariwal, 2018] to low-light image enhancement.

Although the above methods have made significant progress in low-light image enhancement, noise-covered details restored by them can be further enhanced. As shown in Fig. 1(a), previous methods often lead to blurred details and distorted colors. Diffusion models [Ho et al., 2020; Song et al., 2020b] have recently shown their talents in image generation, which can generate more realistic details through a sequence of refinements. Therefore, we introduce diffusion models to low-light image enhancement for better restoring noise-covered details, as shown in Fig. 1(a).

When introducing diffusion models to low-light image enhancement, we found two problems, as demonstrated in Fig. 1(b). 1) The resolution is constant in one reverse process, which limits the speed. 2) Diffusion models result in global degradation.
degradation similar to the RGB shift we analyze in Fig. 5.

To solve these problems, we propose a Pyramid Diffusion model (PyDiff) for low-light image enhancement. As shown in Fig. 1(c), PyDiff uses a novel pyramid diffusion method to sample images in an efficient pyramid resolution style (i.e., progressively increasing resolution in one reverse process). Performing noisier sampling at lower resolution makes the reverse process faster and provides PyDiff with a larger inception field, which benefits global information recovery. Moreover, we analyze the cause of global degradation (Fig. 5) and argue that denoising networks are hard to treat global degradation as part of the noise and correct it during denoising since the reverse process is biased to eliminate Gaussian noise. To alleviate the global degradation that denoising networks can not notice, PyDiff performs sampling with a global corrector. With little additional computational consumption, the global corrector significantly improves the performance and makes the training of diffusion models easier.

We conduct extensive experiments on two popular benchmarks (i.e., LOL [Wei et al., 2018] and LOLV2 [Yang et al., 2021]) to validate the effectiveness and efficiency of PyDiff. Experimental results show that PyDiff achieves superior performance quantitatively and qualitatively under various scenarios. Compared to the previous state-of-the-art (SOTA) method LLFLOW, which also requires iterative refinements, PyDiff significantly outperforms LLFLOW with a speed of nearly 2× faster. In particular, when dealing with unseen noise distributions, PyDiff significantly outperforms other SOTA competitors, e.g., 10 points (SSIM) higher than the second place (NE [Jin et al., 2022]). When handling unseen illumination distributions, PyDiff also gives competitive results, demonstrating our generalization ability further.

Our contributions can be summarized below:

• To the best of our knowledge, we are the first to introduce diffusion models to low-light image enhancement and achieve SOTA. Using a novel pyramid diffusion method, PyDiff is nearly twice as fast as the previous SOTA method LLFLOW.

• We propose a global corrector to alleviate the global degradation that occurs in the reverse process. This significantly improves the performance and makes the training of diffusion models easier with little additional computational consumption.

• Experiments on popular benchmarks show that PyDiff achieves new SOTA performance, and PyDiff can generalize well to unseen noise and illumination distributions.

2 Related Work
2.1 Low-Light Image Enhancement
Low-light image enhancement has been studied for a long time, with numerous deep learning-based approaches proposed. LLNet [Lore et al., 2017] and SID [Chen et al., 2018] collect lots of low/normal-light image pairs to train the network. For getting illumination and reflectance maps [Land, 1977], RetinexNet [Wei et al., 2018], KIND [Zhang et al., 2019], and KIND++ [Zhang et al., 2021] carefully design the loss to train a decomposition network. Enlighten-GAN [Jiang et al., 2021], ZeroDCE [Guo et al., 2020], and NE [Jin et al., 2022] propose effective unsupervised methods which do not require paired data. BREAD [Guo and Hu, 2022] decouples the entanglement of noise and color distortion. Some works [Fan et al., 2022a; Cui et al., 2022; Kim et al., 2021] have brought performance improvements by designing novel and efficient networks. LLFLOW [Wang et al., 2022] models this ill-posed problem via a normalizing flow model [Dinh et al., 2016; Kingma and Dhariwal, 2018]. Although the above methods have made significant progress in low-light image enhancement, noise-covered details restored by them can be further enhanced. This paper introduces diffusion models [He et al., 2020] to low-light image enhancement for better recovering the details.

2.2 Diffusion Models
Diffusion Models [Ho et al., 2020; Song et al., 2020b] present high-quality image synthesis results through a high number of denoising iterations, and various training-free samplers [Song et al., 2020a; Nichol and Dhariwal, 2021; Bao et al., 2022; Lu et al., 2022] have been proposed to achieve comparable results with fewer iterations. To further achieve conditional generation, Guided-Diffusion [Dhariwal and Nichol, 2021] samples with classifier guidance, while our PyDiff concatenates noisy images with source images to guide denoising like some low-level vision methods [Saharia et al., 2022b; Saharia et al., 2022a; Whang et al., 2022].

To generate high-resolution images more efficiently, some works [Saharia et al., 2022b; Ho et al., 2022; Fun et al., 2022b] use multiple diffusion models to achieve cascaded high-resolution image synthesis, while LDM [Rombach et al., 2022] makes reverse processes situated within the image encoder’s latent space. In one reverse process, the above methods perform sampling at a constant resolution style, limiting the speed. In this paper, PyDiff uses a pyramid diffusion method to achieve faster speed and a global corrector to ensure the sample quality for low-light image enhancement.
3 Background: Denoising Diffusion Probabilistic Models

The Denoising Diffusion Probabilistic Model [Ho et al., 2020; Song et al., 2020a] is a latent variable model specified by a T-step Markov chain. It starts with a given data distribution \( x_0 \sim q(x_0) \) and repeatedly adds Gaussian noise according to \( q(x_t|x_{t-1}) \) as follows:

\[
q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I),
\]

where \( \alpha_t \in (0, 1) \), and \( \alpha_t \geq \alpha_{t+1} \). Using the notation \( \bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i \), the marginal distribution \( q(x_t|x_0) \) can be expressed as follows:

\[
q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I) \tag{2}
\]

When \( \sqrt{\bar{\alpha}_t} \) is close to 0, the defined forward process will transform this data distribution into an isotropic Gaussian distribution.

In practical applications, the reverse process of diffusion models is used more often, which converts an isotropic Gaussian distribution to a target data distribution. It is worth mentioning that \( q(x_{t-1}|x_t) \) is hard to estimate while \( q(x_{t-1}|x_0) \) is tractable. We can derive the posterior distribution of \( x_{t-1} \) given \( (x_t, x_0) \) with some algebraic manipulation:

\[
q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \hat{\mu}_t(x_t, x_0), \hat{\beta}_t I), \tag{3}
\]

\[
\hat{\mu}_t(x_t, x_0) := \frac{\sqrt{\alpha_t - 1}\beta_t}{1 - \alpha_t} x_0 + \frac{\sqrt{\alpha_t}(1 - \alpha_{t-1})}{1 - \alpha_t} x_t, \tag{4}
\]

\[
\hat{\beta}_t := \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t, \tag{5}
\]

where \( \beta_t := 1 - \alpha_t \). We have no \( x_0 \) during testing, but we can calculate its approximate value according to Eq. (2):

\[
y_{\theta}(x_t) := \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1 - \alpha_t}e_{\theta}(x_t)), \tag{6}
\]

where \( e_{\theta}(x_t) \) is the predicted noise derived from the denoising network, and \( y_{\theta}(x_t) \) is an approximation of \( x_0 \) calculated according to \( e_{\theta}(x_t) \). Furthermore, we update Eq. (3) as follows:

\[
p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \hat{\mu}_t(x_t, y_{\theta}(x_t)), \hat{\beta}_t I) \tag{7}
\]

When it comes to image-to-image translation, [Saharia et al., 2022a; Saharia et al., 2022b; Choi et al., 2021] make the reverse process conditional on an input signal. Specifically, when we need to translate \( z \) to \( y \) (e.g., low-light image to normal-light image), we update Eq. (6) as follow:

\[
y_{\theta}(x_t, z) := \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1 - \alpha_t}e_{\theta}(x_t, z)) \tag{8}
\]

4 Methods

This section presents PyDiff, an effective and efficient method for low-light image enhancement. First of all, we describe the motivation for designing PyDiff. Secondly, we introduce our proposed pyramid diffusion, which significantly improves the inference speed without any performance degradation. Furthermore, we present our proposed global corrector, which alleviates the global degradation that may occur in the reverse process of the diffusion models. Finally, we describe the training and sampling procedures of PyDiff.
downsampling schedule \( \{s_t\}_{t=0}^{T} \), which means that the ith sampling will be performed at the resolution downsampling with a scale factor \( s_t \). While \( \alpha_t \geq \alpha_{t+1} \) to get bigger and bigger noise, \( s_t \leq s_{t+1} \) to get lower and lower resolution.

**Forward Process.** For the forward process, pyramid diffusion updates the Eq. (1) in diffusion models as follows:

\[
q(x_t|X_{t-1}) := \mathcal{N}(x_t; \sqrt{\alpha_t} (x_{t-1} \downarrow_{s_t/s_{t-1}}), (1 - \alpha_t) I),
\]

where \( \downarrow \) means downsampling with a scale factor of \( r \). The marginal distribution \( q(x_t|X_0) \) can be expressed as follows:

\[
q(x_t|X_0) := \mathcal{N}(x_t; \sqrt{\alpha_t} (x_0 \downarrow_{s_t}), (1 - \alpha_t) I) \tag{10}
\]

**Reverse Process.** In the case of \( s_{t-1} = s_t \), we can derive \( x_{t-1} \) from \( x_t \) according to Eq. (7). However, this is no longer applicable in the case of \( s_{t-1} < s_t \) since differences in resolution. As shown in Fig. 4(a), \( (x_0 \uparrow_r) \uparrow_r \) can serve as \( x_0 \) at noisy sampling (i.e., Larger \( r \) matches noisier sampling), where \( \uparrow_r \) means upsampling with a scale factor of \( r \). Therefore, with well-scheduled \( \{s_t\}_{t=0}^{T} \) and \( \{\alpha_t\}_{t=0}^{T} \), we can take \( y_\theta(x_t) \uparrow_{s_t/s_{t-1}} \), as \( x_0 \downarrow_{s_{t-1}} \). According to Eq. (10), we can further add noise to \( x_0 \downarrow_{s_{t-1}} \) for deriving \( x_{t-1} \). Adding noise such a method leads to inconsistency between \( x_t \) and \( x_{t-1} \). However, this inconsistency has little impact on noisy sampling, which is primarily concerned with recovering global information. To summarize, the posterior distribution of pyramid diffusion can be expressed as follows:

\[
p_\theta(x_{t-1}|x_t) = \begin{cases} 
\mathcal{N}(x_{t-1}; \sqrt{\alpha_{t-1}} \beta_t y_\theta(x_t) + \sqrt{\alpha_{t-1}}(1-\beta_t) x_t, \\
1 - \alpha_{t-1} \alpha_t \beta_t I), & \text{if } s_t = s_{t-1}, \\
\mathcal{N}(x_{t-1}; \sqrt{\alpha_{t-1}} y_\theta(x_t) \uparrow_{s_t/s_{t-1}}, (1 - \alpha_{t-1})(I)), & \text{if } s_t > s_{t-1} \end{cases} \tag{11}
\]

**Position Encoding.** Pyramid diffusion requires one network to process images of multiple resolutions. As the main operator of the denoising network [Ho et al., 2020], convolution kernels cannot perceive the change of resolution. We consider using position encoding to guide the network. For an image \( I \) with a resolution of \( H \times W \), its coordinates are \( X, Y \in \mathbb{R}^{H \times W} \), where \( X_{i,j} = i \) and \( Y_{i,j} = j \). After normalizing \( X \) and \( Y \), we apply sinusoidal positional encoding of them to guide the network. Specifically, the position encoding is expressed as:

\[
\text{pos}(I) = [\sin(X), \cos(X), \sin(Y), \cos(Y)] \tag{12}
\]

Convolution kernels have a constant receptive field. When dealing with images downsampled with a scale factor of \( r \), the range of position encoding perceived by convolution kernels will be correspondingly expanded by \( r \) times, which may tell convolution kernels the change of resolution.

### 4.3 Global Corrector

As shown in Fig. 2, PyDiff uses the global corrector to alleviate global degradation in reverse processes. The global corrector can significantly improve performance with little additional computational consumption. In this section, we introduce the proposed global corrector.

**Global Degradation.** When applying diffusion models to low-light image enhancement, we found them sometimes result in significant global degradation, as shown in Fig. 5(a). This global degradation looks like a shift in the RGB channels, similar to the RGB shift shown in Fig. 3(b).

**Cause of Global Degradation.** Looking back on Eq. (6), we can rewrite it as follows:

\[
y_\theta(x_t) := \frac{1}{\sqrt{\alpha_t}} (x_t - \sqrt{1 - \alpha_t}(\epsilon_t - \delta_t)) \tag{13}
\]

\[
:= x_0 + \frac{\sqrt{1 - \alpha_t}}{\sqrt{\alpha_t}} \delta_t, \tag{14}
\]

where \( \epsilon_t \) is the actual noise in \( x_t \), and \( \delta_t \) is the error between \( \epsilon_t \) and \( \epsilon_\theta(x_t) \). We found that there is a coefficient \( \frac{\sqrt{1 - \alpha_t}}{\sqrt{\alpha_t}} \) in front of the error \( \delta_t \). When \( t \) is relatively large, this coefficient will also be large, as shown in Fig. 4(b). As demonstrated in Fig. 5(b), the original error \( \delta_T \) is small, but the coefficient \( \frac{\sqrt{1 - \alpha_T}}{\sqrt{\alpha_T}} \) enlarges the error and leads to an obvious RGB shift (e.g., a significant gain in the R channel). As shown in Fig. 5(a), the denoising network treats the image under global degradation as usual and only performs its denoising duties, which can not eliminate the global degradation.

**Design of Global Corrector.** We add a global corrector to alleviate the global degradation that denoising networks can not notice. The design of the global corrector needs to meet the following requirements: 1) The global corrector should alleviate global degradation while preserving generated edges and textures. 2) The global corrector is lightweight and fast. Inspired by CSRNet [He et al., 2020], we design an efficient global corrector that performs pixel-independent retouching based on global conditions. Please refer to the supplementary.
In the algorithm 2, the sampling process begins with the input noise schedule \( \alpha \), downsampling schedule \( s \), correction threshold \( \gamma \), denoising network \( \theta \), global corrector \( c \), low-normal-light image pairs \( q(x_{\text{low}}, y) \).

1. **Input**: noise schedule \( \alpha \), downsampling schedule \( s \), correction threshold \( \gamma \), denoising network \( \theta \), global corrector \( c \), low-normal-light image pairs \( q(x_{\text{low}}, y) \).
2. **repeat**
   3. Sample \( (x_{\text{low}}, y) \) \( \sim q(x_{\text{low}}, y) \).
   4. Sample \( t \sim \text{Uniform}(1, ..., T) \).
   5. Sample \( \epsilon \sim \mathcal{N}(0, I) \).
   6. \( x_t = \sqrt{\alpha_t} (y_s \downarrow s_t) + \sqrt{1 - \alpha_t} \epsilon \).
      - \( \uparrow_r \) means upsampling with a scale factor of \( r \).
   7. Take gradient descent step on \( \nabla_\theta \| \epsilon \|_1 \).
   8. if \( \sqrt{\alpha_t} > \gamma \) then
      9. Take gradient descent step on \( \nabla_c \| (y_s \downarrow s_t) \|_1 \).
10. **end if**
11. **end if**
12. **until** converged
13. return \( x_0 \)

Correction Threshold. As shown in Fig. 4(b), the amplification factor \( \sqrt{\alpha} \) will gradually decrease to 0 as \( t \) decreases. When the amplification factor is small enough, it will no longer amplify the error of the denoising network. Therefore, we set a correction threshold \( \gamma \), and the global corrector is needed only when \( \sqrt{\alpha} \) \( > \gamma \). We set \( \gamma = 1 \).

### 4.4 Training and Sampling

To better demonstrate the overall framework of our PyDiff, we omit some details (e.g., position encoding) when describing the algorithm.

**Training.** Algorithm 1 shows the specific training procedure of the proposed PyDiff. The global corrector aims to alleviate the global degradation that the denoising network cannot notice, and it will not impact the denoising network.

**Sampling.** Algorithm 2 shows the specific sampling procedure of the proposed PyDiff. PyDiff can be easily combined with DDIM [Song et al., 2020a] or DDPM+ [Nichol and Dhariwal, 2021] to achieve further speedup.

**Training Loss.** As described in algorithm 1, we use the simple L1 loss to optimize the denoising network and global corrector without additional optimization objectives.

### 5 Experiments

#### 5.1 Setup

**Dataset.** We conduct experiments on LOL [Wei et al., 2018] and LOLV2 [Yang et al., 2021] datasets. The LOLV2 dataset contains two parts, REAL and SYNC. REAL PART has noise distributions not present in the LOL dataset, and SYNC PART has illumination distributions not present in the LOL dataset. For supervised learning methods involved in comparison, we use their pre-trained model only trained on the LOL dataset. Correspondingly, we train PyDiff only on the LOL dataset too. For unsupervised learning methods, we use their released pre-trained models no matter how they train.

**Schedules.** First of all, we set \( T = 2000 \). For the noise schedule, we decrease \( \alpha_t \) linearly from \( \alpha_1 = 0.9999999 \) to \( \alpha_T = 0.99 \). For the downsampling schedule, our default setting is to set \( \{s\}_t \rightarrow \frac{T}{2} = 1 \) and \( \{s\}_t \rightarrow \frac{T}{4} = 2 \), and we will experiment with more schedules in ablation studies.

**Training.** We set the patch size to 192 \( \times \) 288 and the batch size to 16. We use the Adam optimizer with an initial learning rate of \( 1 \times 10^{-4} \) for 320k iterations and halve the learning rate at 50k, 75k, 100k, 150k, and 200k. The optimizer does not use weight decay. We complete training on two NVIDIA GeForce RTX 3090s.
5.2 Comparison With SOTA Methods

LOLV2 REAL PART. Since the test set of LOLV2 REAL PART overlaps with the training set of the LOL dataset, we combine the training set and test set of LOLV2 REAL PART and filter out the overlapping parts with the LOL training set by the ID of the images. For the filtered images, we sort them by ID and select 100 (i.e., the same size as the original test set) images with the smallest ID as the test set of LOLV2 REAL PART. Many of the selected images were taken at ISOs not included in the LOL training set, which is a good test of the model’s ability to deal with unseen noise. Tab. 2 shows the quantitative comparison with other SOTA methods on LOLV2 REAL PART. As PyDiff can deal with unseen

<table>
<thead>
<tr>
<th>Methods</th>
<th>PSNR ↑</th>
<th>SSIM</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIND [Zhang et al., 2019]</td>
<td>18.32</td>
<td>0.822</td>
<td>0.27</td>
</tr>
<tr>
<td>KIND++ [Zhang et al., 2021]</td>
<td>19.44</td>
<td>0.830</td>
<td>0.23</td>
</tr>
<tr>
<td>IAT [Cui et al., 2022]</td>
<td>19.18</td>
<td>0.813</td>
<td>0.29</td>
</tr>
<tr>
<td>Bread [Guo and Hu, 2022]</td>
<td>19.28</td>
<td>0.831</td>
<td>0.24</td>
</tr>
<tr>
<td>HWMNet [Fan et al., 2022a]</td>
<td>18.79</td>
<td>0.817</td>
<td>0.24</td>
</tr>
<tr>
<td>LLFLOW [Wang et al., 2022]</td>
<td>19.15</td>
<td>0.860</td>
<td>0.22</td>
</tr>
<tr>
<td>PyDiff (ours)</td>
<td>19.60</td>
<td>0.878</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 3: Quantitative results on the LOLV2 SYNC PART in terms of PSNR, SSIM, and LPIPS. All methods involved in the comparison were not retrained on the corresponding training set. ↑ (↓) denotes that larger (smaller) values lead to better quality.

EVALUATION. Combined with DDIM [Song et al., 2020a], PyDiff requires only 4 iterations to obtain results comparable to other SOTA methods.

OTHER DETAILS. The reverse process is conditional on low-light images $x_{low}$. low-light images after histogram equalization $\text{hiseq}(x_{low})$, and position encoding $\text{pos}(x_{low})$. We use the method of concatenating to achieve conditional sampling [Saharia et al., 2022b; Saharia et al., 2022a]. During training, we swap the concatenating order of $x_{low}$ and $\text{hiseq}(x_{low})$ with a probability of 0.5. Please refer to the supplementary materials for model configuration and details of the global corrector.

5.2 Comparison With SOTA Methods

LOLV2 Dataset. We first compare PyDiff with SOTA methods on the LOL dataset. The quantitative results are shown in Tab. 1. PyDiff outperforms other methods in all three metrics: PSNR, SSIM [Wang et al., 2004], and LPIPS [Zhang et al., 2018]. Beating second place by 2.1 points on PSNR shows that PyDiff can recover more accurate colors. Surpassing second place by 1 point on SSIM shows that PyDiff accurately preserves more high-frequency details. Exceeding second place by 1 point on LPIPS shows that PyDiff gives more eye-pleasing results. Fig. 6 shows qualitative comparisons with other methods, where PyDiff exhibits the best result.

<table>
<thead>
<tr>
<th>Methods</th>
<th>PSNR ↑</th>
<th>SSIM</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAT [Cui et al., 2022]</td>
<td>19.18</td>
<td>0.813</td>
<td>0.29</td>
</tr>
<tr>
<td>Bread [Guo and Hu, 2022]</td>
<td>19.28</td>
<td>0.831</td>
<td>0.24</td>
</tr>
<tr>
<td>HWMNet [Fan et al., 2022a]</td>
<td>18.79</td>
<td>0.817</td>
<td>0.24</td>
</tr>
<tr>
<td>LLFLOW [Wang et al., 2022]</td>
<td>19.15</td>
<td>0.860</td>
<td>0.22</td>
</tr>
<tr>
<td>PyDiff (ours)</td>
<td>19.60</td>
<td>0.878</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 2: Quantitative results on the LOLV2 REAL PART in terms of PSNR, SSIM, and LPIPS. All methods involved in the comparison were not retrained on the corresponding training set. ↑ (↓) denotes that larger (smaller) values lead to better quality.

Figure 6: Qualitative comparison with state-of-the-art methods on the LOL dataset. It can be seen that IAT [Cui et al., 2022] cannot even restore the correct brightness, and the results generated by KIND++ [Zhang et al., 2021], BREAD [Guo and Hu, 2022], NE [Jin et al., 2022], and HWMNet [Fan et al., 2022a] have apparent artifacts, while the KIND [Zhang et al., 2019] cannot restore colors well. LLFLOW [Wang et al., 2022] gives a not-bad result, but its result can be too smooth, and the colors of some items need to be more accurately restored. PyDiff exhibits the best result, which restores the correct color and preserves the details covered by noise. Please zoom in for the best view.

Figure 7: Qualitative comparison with state-of-the-art methods on the REAL PART of the LOLV2 dataset. Please zoom in for the best view.
Tab. 5 shows that vanilla diffusion models without any performance degradation.

**Robustness to Batch Size.** Tab. 5 shows that vanilla diffusion models without global corrector are very dependent on large batch size, which shows a significant performance drop when the batch size decreases. As our analysis in section 4.3, we argue that this is caused by the amplification factor, which has rigorous requirements for denoising networks. This problem has been significantly improved by adding a global corrector. As shown in Tab. 5, the global corrector enhances the performance of diffusion models under \( bs = 4(8) \) and outperforms the ones without global corrector under \( bs = 8(16) \), which means that the global corrector can make diffusion models more robust to batch size and easier to train.

**Comparison With LLFLOW.** LLFLOW [Wang et al., 2022] used to be first place on the LOL dataset based on the normalizing flow [Dinh et al., 2016; Kingma and Dhariwal, 2018]. Both FLOWs and diffusion models are generative models that require multiple iterations. Therefore, it will be interesting to compare the speed of LLFLOW and PyDiff. According to Tab. 4, PyDiff significantly enhances performance, achieving an 87\% faster speed than LLFLOW.

### Table 5: Ablation study on global corrector. batch stands for batch size, while gc means global corrector.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Metrics</th>
<th>Hyperparameters</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{schedule} )</td>
<td>( \text{pe} )</td>
<td>PSNR↑</td>
<td>SSIM↑</td>
</tr>
<tr>
<td>LLFLOW</td>
<td>-</td>
<td>-</td>
<td>20.70</td>
</tr>
<tr>
<td>PyDiff</td>
<td>1, 1, 1, 1</td>
<td>\text{YES}</td>
<td>22.11</td>
</tr>
<tr>
<td></td>
<td>1, 1, 2, 2</td>
<td>\text{YES}</td>
<td>22.17</td>
</tr>
<tr>
<td></td>
<td>1, 1, 2, 2</td>
<td>NO</td>
<td>21.96</td>
</tr>
<tr>
<td></td>
<td>1, 2, 2, 4</td>
<td>\text{YES}</td>
<td>22.02</td>
</tr>
<tr>
<td></td>
<td>1, 2, 2, 2</td>
<td>\text{YES}</td>
<td>22.15</td>
</tr>
<tr>
<td></td>
<td>1, 2, 4, 8</td>
<td>\text{YES}</td>
<td>22.12</td>
</tr>
</tbody>
</table>

**5.3 Ablation Study**

In this section, we conduct ablation studies on the main components of PyDiff to observe their impact on performance. The score for this section is calculated by combining the performance on the LOL dataset, LOLV2 REAL PART, and LOLV2 SYNC PART, which gives a better indication of the effectiveness of a component. FPS is measured on the LOL dataset (i.e., the resolution is 400 × 600).

**Downsampling Schedules.** In Tab. 4, schedule \([1, 1, 1, 1] \) represents vanilla diffusion models, which sample at a constant resolution. Our default setting, schedule \([1, 1, 2, 2] \), performs noisy sampling at a 1/2 resolution. Schedule \([1, 1, 2, 2] \) is 54\% faster while slightly outperforming the schedule \([1, 1, 1, 1] \). Furthermore, our investigation revealed that faster schedules (e.g., \([1, 1, 2, 4] \), \([1, 2, 2, 2] \), and \([1, 2, 4, 8] \) produce comparable results to the vanilla schedule \([1, 1, 1, 1] \). These findings indicate that noisier sampling can be performed at a lower resolution, while still maintaining high performance.

**Position Encoding.** Tab. 4 shows that the position encoding boosts PSNR and SSIM for PyDiff, which may tell networks about the change of resolution.

**Effectiveness of Global Corrector.** Tab. 5 shows that the global corrector can bring improvements to PyDiff under various settings. Fig. 2 and Fig. 5(a)(c) show that the global corrector effectively alleviates global degradation. Furthermore, we can see from Tab. 5 that the global corrector gives little additional computational consumption to PyDiff.

This paper proposes PyDiff, a diffusion model based method for low-light image enhancement. PyDiff uses a novel pyramid diffusion method, which makes sampling faster than vanilla diffusion models without any performance degradation. Furthermore, PyDiff uses a global corrector to alleviate global degradations that cannot be noticed by the denoising network and significantly improves performance with little additional computational consumption. Experimentally, PyDiff shows superior effectiveness, efficiency, and generalization ability on popular benchmarks. We hope that PyDiff will serve as a strong baseline for low-light image enhancement and that the pyramid diffusion method will facilitate the application of diffusion models in more low-level vision tasks.

### Acknowledgements

This work is supported by the Fundamental Research Funds for the Central Universities (No. 226-2022-00051).
References


[Song et al., 2020b] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through


