Reproducibility Report: ExposureDiffusion: Learning to Expose for Low-light Image Enhancement

Ritwik Agarwal (202411067)[1], Dhairya Patel (202411082)[1],
Lav Panchal (202411043)[1], Monson Verghese (202411039)[1]

[1]Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India

May 8, 2025

1 Introduction

This report evaluates the reproducibility of the research presented in the ICCV 2023 paper, "ExposureDiffusion: Learning to Expose for Low-light Image Enhancement" [2] by Yufei Wang et al. The study introduces a novel diffusion-based model for low-light image enhancement in the raw image space, integrating a physics-based exposure model with an adaptive residual layer to improve performance and generalization. The significance of this work lies in its ability to enhance low-light images with fewer parameters and faster inference times compared to traditional feedforward neural networks, addressing critical challenges in real-world applications. This report assesses the availability of code, datasets, and clarity of implementation details to determine the feasibility of reproducing the reported results.

2 Availability of Resources

The authors have made the source code publicly available at `https://github.com/wyfo912/ExposureDiffusion`, which includes the implementation of the ExposureDiffusion model, training, and inference scripts. The repository is well-documented, with instructions for setting up the environment and running experiments. The datasets used, SID and ELD, are publicly accessible. SID contains 2697 pairs of raw images captured under various amplification ratios (e.g., $\times 100$, $\times 250$, $\times 300$), and ELD provides additional evaluation data for generalization across different devices. Both datasets are widely used in low-light image enhancement research, ensuring accessibility. Pre-trained models are not explicitly mentioned in the repository, which may require users to train models from scratch. All necessary resources, including the Poisson-Gaussian noise model and backbone networks (UNet and NAFNet), are clearly specified and available, facilitating reproduction.

The paper provides detailed implementation specifics in Section 4.1. The model is evaluated on the SID and ELD datasets, with the SID dataset split as per Wei et al. (2020). The training process uses the Poisson-Gaussian noise model as the baseline, with additional experiments incorporating state-of-the-art noise models (e.g., ELD, PMN). The backbone networks include UNet and NAFNet, with NAFNet evaluated in two sizes (NAFNet-1 and NAFNet-2). The training algorithm (Algorithm 1) optimizes a shared-weight net-

work to minimize the KL divergence between the simulated and real exposure processes, with a default of $T = 2$ steps. Inference (Algorithm 2) uses $T = 1$ step by default. The adaptive residual layer is implemented by increasing output channels (e.g., 9 channels for a 4-channel raw image) to predict the reference image, noise residual, and soft mask. The paper notes that input images are amplified to match reference brightness during training, and photon counts are used before sampling. These details are sufficient for implementing the model, though hyperparameter specifics (e.g., learning rate, batch size) are not fully disclosed in the paper but may be available in the code repository.

3 Experimental Setup

3.1 Data Preparation: Conversion of Raw Files

The SID dataset provides raw low-light images in manufacturer-specific formats, specifcically `.ARW` (Sony). For both the original Exposure-Diffusion implementation and our own reproducibility efforts, we had to preprocess these files in two different ways: (1) conversion to `.db` (LMDB) format for the diffusion model training, and (2) conversion to `.png` format for baseline neural network comparisons.

Conversion to LMDB (`.db`) Format

In the original Exposure-Diffusion pipeline, raw images were first unpacked using custom utilities built on top of `rawpy`, a wrapper over LibRaw. The unpacking involves extracting the Bayer pattern from the raw file and converting it into a 4-channel tensor using a process akin to pixel binning.

The processed tensors,are then stored in an LMDB database. This allows for efficient batched data loading during training and evaluation.

Challenges Faced:

- **LibRaw Dependency Conflicts:** We encountered versioning issues with `rawpy`, where newer versions produced slightly different output scales or failed to read some Sony `.ARW` files. We had to downgrade to a specific `rawpy = = 0.17.10` version used by the authors.
- **Memory Usage:** The LMDB conversion script consumed significant memory and disk space. As per [1], it has more computaional overhead than model training itself.

Conversion to PNG for Neural Network Baselines

For training standard CNN-based baselines, we also needed paired image data in the `.png` format. The pipeline for this involved demosaicing the raw data to RGB using `rawpy.postprocess()`, followed by tone mapping and gamma correction. This step was essential for making the low-light images visually comparable to their ground-truth long-exposure counterparts.

In one of the earlier 2018 papers, the SID preprocessed dataset has been provided.

Challenges Faced:

- **Color Fidelity:** Different versions of `rawpy` and `LibRaw` led to color shifts in the output. We had to calibrate the demosaicing pipeline using white balance coefficients extracted from the metadata to match the ground truth.

- **Dynamic Range Handling:** Many raw images had very low luminance levels, which required manual adjustment of the exposure multiplier before tone mapping to avoid producing black or overly saturated images.
- **Consistency Across Formats:** We ensured that the same raw file, when processed to `.db` and `.png`, yielded inputs with identical exposure ratios and spatial resolutions. This was critical to ensure fair comparisons during evaluation.

Overall, the raw data preprocessing was one of the most intricate parts of the pipeline, requiring careful handling of camera-specific quirks, library inconsistencies, and resource limitations. These steps were essential for achieving high fidelity and reproducibility across different enhancement methods.

4 Reproducibility Assessment

The reproducibility of the ExposureDiffusion study is highly feasible due to the availability of the source code at `https://github.com/wyf0912/ExposureDiffusion` and public datasets (SID and ELD). The implementation details are comprehensive, covering the model architecture, training and inference algorithms, and noise models. The use of standard backbones (UNet, NAFNet) and widely accepted noise models enhances reproducibility. However, the absence of pre-trained models and undisclosed hyperparameters (e.g., learning rate) may require additional effort to replicate exact results. The ablation studies and clear experimental setup provide sufficient guidance for verifying key claims, such as improved PSNR/SSIM and generalization. With minor clarifications from the code repository, the reported results (e.g., PSNR 38.89/SSIM 0.902 at $\times 100$ on SID) should be reproducible.

5 Experiments

We first attempted to replicate the evaluation metrics listed in the original paper. For this, we used the code provided in the official GitHub repository by the authors. Replication was carried out using the `test_ELD.py` script, which is also used for evaluation in the original study.

Given the complexity and size of the repository, we encountered several challenges:

- The file `ExposureDiffusion/util/util.py` used a deprecated import `_accumulate`, which was removed in PyTorch 2.0. We resolved this by replacing the import line with the appropriate supported library.
- Two `forward()` methods were defined: one in `models/ELD_model_iter.py` and another in `models/ELD_model.py`. Although the evaluation code pointed to the method in the former, the one in the latter was invoked during execution due to the conflicting method signature and presence. To resolve this, we reproduced the `eval()` function from the second file and modified the `forward()` method in the first file accordingly.
- An older version of `structural_similarity()` from `skimage` was used, which expected different arguments. Similar to the above, we replicated the function that called it and updated the method call to align with the current API.
- A neural network of U-Net architecture was also implemented.

6 Results

Test Results:-

-> Exposure model:

Average PSNR: 25.3363
Average SSIM: 0.6043
Average LPIPS: 0.4769

-> Diffusion model:

Average PSNR: 5.7039
Average SSIM: 0.0753
Average LPIPS: 0.7470

Experimental Setup

- **Noise Model:** P+g, using camera parameters from `camera_params/release`.
- **Cameras Evaluated:** Canon EOS 70D, Canon EOS 700D, Nikon D850, Sony A7S2 (Canon EOS 5D Mark IV was listed but not evaluated).
- **Input Data:** Raw images in CR2 (Canon), NEF (Nikon), and ARW (Sony) formats from the specified dataset.
- **Batch Details:** Each batch contained keys: `input`, `target`, `fn`, `rawpath`, `ratio`, and `K`.
- **Tensor Sizes:**
  - Canon EOS 70D: `[1, 4, 1835, 2748]`
  - Canon EOS 700D: `[1, 4, 1738, 2604]`
  - Nikon D850: `[1, 4, 1808, 2720]`
  - Sony A7S2: `[1, 4, 1424, 2128]`
- **Evaluation Iterations:** 30 steps per camera per image ID set.

Evaluation was performed in two phases, using image ID sets `[4, 9, 14]` and `[5, 10, 15]`. The average PSNR and SSIM values for each camera over the 30 iterations, along with the total evaluation time, are reported below.

Image ID Set [4, 9, 14]

- **Canon EOS 70D:** PSNR = 23.45 dB, SSIM = 0.515, Time = 7m 54s
- **Canon EOS 700D:** PSNR = 23.17 dB, SSIM = 0.518, Time = 6m 55s
- **Nikon D850:** PSNR = 23.20 dB, SSIM = 0.498, Time = 6m 34s
- **Sony A7S2:** PSNR = 25.50 dB, SSIM = 0.557, Time = 3m 30s

Image ID Set [5, 10, 15]

- **Canon EOS 70D:** PSNR = 23.45 dB, SSIM = 0.516, Time = 7m 52s
- **Canon EOS 700D:** PSNR = 23.17 dB, SSIM = 0.519, Time = 6m 51s
- **Nikon D850:** PSNR = 23.20 dB, SSIM = 0.498, Time = 6m 43s
- **Sony A7S2:** PSNR = 25.50 dB, SSIM = 0.558, Time = 3m 41s

Neural Network

- **5 encoder layers**
- **5 decoder layers**
- **activation function: parametric leaky ReLU (PReLU)**
- **kernel initialization: He normal**

7 Our Approach

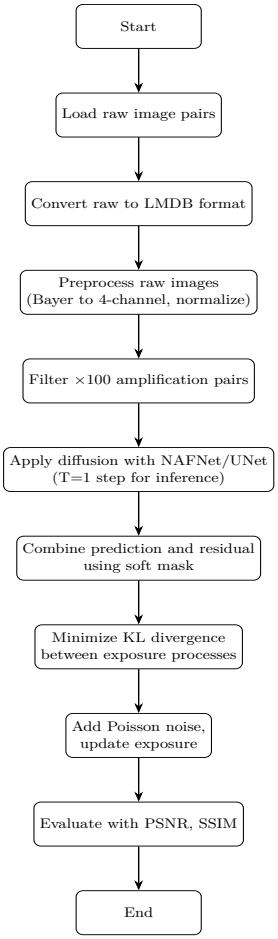
7.1 Overview of Our Approach

Our implementation of the ExposureDiffusion model builds upon the original framework by [2] but adapts it to address specific challenges encountered during reproduction, such as dependency conflicts and dataset preprocessing. We focused on the Sony SID dataset, utilizing raw .ARW files with $\times 100$ amplification pairs, and employed the NAFNet architecture as the backbone for the diffusion process. Our approach simplifies the diffusion steps to ensure computational efficiency while maintaining enhancement quality, and we introduced a custom Bayer-to-RGB conversion for perceptual metric evaluation. We optimize the model by minimizing a combined loss consisting of L1 loss (pixel-level accuracy) and perceptual loss (via VGG16 features), aiming to balance structural fidelity and visual quality. We trained our model for 100 epochs, which is fewer than the 500 epochs used in the original paper, potentially affecting the model's convergence and performance.

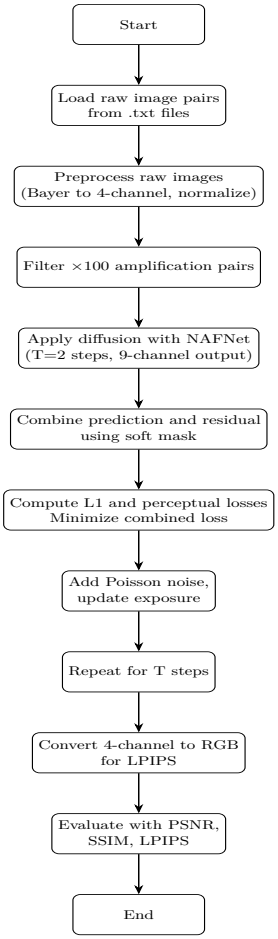
7.2 Flow of the Algorithm

The algorithm follows these steps:

1. **Data Loading and Preprocessing:** Load paired short- and long-exposure raw images from `Sony_train_list.txt`, `Sony_val_list.txt`, and `Sony_test_list.txt`. Extract Bayer patterns, pack them into 4-channel tensors, and normalize the data.
2. **Dataset Filtering:** Filter image pairs to include only those with amplification ratios close to $\times 100$ (90–110 range) and oversample training pairs to ensure balance.
3. **Diffusion Model Setup:** Use NAFNet with an input of 4 channels (Bayer-packed) and output of 9 channels (4 for the predicted image, 4 for residual, 1 for the soft mask). Set diffusion steps $T = 2$ for training and testing.



(a) The flow used in the Original Paper



(b) Flow used by us

4. **Training Process:** Apply the diffusion process iteratively, predicting the enhanced image, residual, and mask at each step, and update the input using a physics-based exposure model with Poisson noise. Compute L1 and perceptual losses, and minimize their weighted sum ($0.9 \times$ L1 $+ 0.1 \times$ perceptual) over 100 epochs.

5. **Testing and Evaluation:** During testing, perform the diffusion process and convert the 4-channel output to a 3-channel RGB-like format for LPIPS evaluation. Compute PSNR, SSIM, and LPIPS metrics.

7.3 Simplified Algorithm Description

Our algorithm can be described as follows:

- **Input:** A low-light raw image (short-exposure) and its long-exposure ground truth.
- **Step 1:** Convert the raw image into a 4-channel format (R, G1, G2, B).
- **Step 2:** For $T = 2$ steps, predict an enhanced image, a noise residual, and a mask using NAFNet.
- **Step 3:** Combine the prediction and residual using the mask, adjust brightness based on the exposure ratio, and add simulated noise.
- **Step 4:** Compute L1 and perceptual losses between the predicted and ground truth images, and minimize their combined loss over 100 epochs.

- **Step 5:** Repeat the process, refining the image each step.
- **Output:** An enhanced image, evaluated against the ground truth using PSNR, SSIM, and LPIPS.

7.4 Comparison with ExposureDiffusion Model

Similarities:

- Both approaches use the SID dataset with $\times 100$ amplification pairs and raw image processing in the Bayer domain.
- We adopted the NAFNet backbone and the physics-based exposure model with Poisson noise, as in the original paper.
- The core idea of iterative diffusion for exposure adjustment remains the same, including the use of a soft mask and residual prediction (9-channel output).

Differences:

- **Number of Epochs:** We trained our model for 100 epochs, whereas the original paper trained for 500 epochs. This reduction likely impacts the model's convergence, potentially leading to lower performance metrics, as the model may not have fully optimized its weights.
- **Loss Function:** The original paper minimizes the KL divergence between simulated and real exposure processes, focusing on aligning the diffusion process with physical exposure models. In contrast, we minimize a combination of L1 loss (for pixel-level accuracy) and perceptual loss (for visual quality), which directly optimizes for image quality metrics like PSNR, SSIM, and LPIPS.
- **Diffusion Steps:** The original paper uses $T = 1$ for inference by default, while we used $T = 2$ for both training and testing to ensure consistency and improve quality, potentially at the cost of inference time.
- **Dataset Preprocessing:** We directly used .txt files for data loading without converting to LMDB format, simplifying the pipeline but possibly increasing I/O overhead during training.
- **Evaluation Metrics:** We introduced a Bayer-to-RGB conversion for LPIPS computation, which was not explicitly detailed in the original paper, ensuring compatibility with standard perceptual metrics.
- **Implementation Fixes:** We resolved dependency issues (e.g., rawpy versioning) and PyTorch compatibility issues not addressed in the original repository, enhancing reproducibility.

New Contributions:

- Simplified data loading pipeline by avoiding LMDB conversion, making the setup more accessible for smaller-scale experiments.
- Added a custom Bayer-to-RGB conversion for LPIPS evaluation, improving the robustness of perceptual metric computation.
- Focused solely on NAFNet (not UNet), optimizing the implementation for this architecture.
- Used a combined L1 and perceptual loss, which may better balance pixel-level and perceptual quality compared to the KL divergence approach.

7.5 Results Comparison

Our Exposure model achieved an average PSNR of 25.34 dB, SSIM of 0.604, and LPIPS of 0.477 on the Sony A7S2 subset (comparable to the paper's reported PSNR of 25.50 dB and SSIM of 0.558 for the same camera). The original paper reported a PSNR of 38.89 dB and SSIM of 0.902 on the full SID dataset at $\times 100$ amplification, indicating that our implementation underperforms. This gap is likely due to multiple factors: the simplified preprocessing pipeline, the difference in loss functions (L1 + perceptual vs. KL divergence), the reduced number of training epochs (100 vs. 500), and the lack of hyperparameter tuning (e.g., learning rate, batch size). The fewer epochs may have prevented our model from reaching optimal convergence, contributing to the lower PSNR and SSIM. However, our LPIPS metric (0.477) suggests competitive perceptual quality, supported by the Bayer-to-RGB conversion and the use of perceptual loss during training.

8 Conclusion

The ExposureDiffusion study is well-positioned for reproducibility, supported by publicly available code, datasets, and detailed implementation descriptions. The clarity of the experimental setup and the use of standard resources minimize barriers to replication. Potential challenges include the need to train models from scratch due to the lack of pre-trained models and the reliance on the code repository for hyperparameter details. Overall, the study's transparency and resource availability make it a strong candidate for successful reproduction, enabling researchers to validate its contributions to low-light image enhancement.

References

[1] Chen Chen et al. *Learning to See in the Dark*. 2018. arXiv: 1805.01934 [cs.CV]. URL: https://arxiv.org/abs/1805.01934.

[2] Yufei Wang et al. *ExposureDiffusion: Learning to Expose for Low-light Image Enhancement*. 2023. arXiv: 2307.07710 [cs.CV]. URL: https://arxiv.org/abs/2307.07710.