# Supplementary Materials

Fengbo Lan
fengbo.lan@connect.polyu.hk
The Hong Kong Polytechnic University
Hong Kong, China

Chang Wen Chen
changwen.chen@polyu.edu.hk
The Hong Kong Polytechnic University
Hong Kong, China

## 1 DETAILS ON EXPERIMENTAL SETUP

This section provides a comprehensive overview of the methodologies used in preparing our dataset, the models employed for flare removal, and the specific training protocols followed. We detail the procedures for collecting and processing data, outline the models used in our experiments, and describe the training details that underpin our research.

### 1.1 Data Preparation

We compiled a dataset comprising 2,200 raw image pairs for scattering flares and 1,100 raw pairs for reflective flares. From the latter, we generated 2,200 pairs, using a center crop to achieve a resolution of $1024 \times 1024$ for training, ensuring comprehensive coverage of the flare-corrupted areas.

For scattering flares, the 2,200 high-resolution raw image pairs were cropped into 30,000 flare-corrupted pairs, each with a resolution of $512 \times 512$, centered around a light source. We refined the light-source detection method originally used in Flare7K [3] and Wu et al. [9], implementing two major improvements. The initial detection algorithm filtered overexposed areas by thresholding the grayscale image and used morphological operations to isolate the light source. However, this method often misclassified the background as the light source, especially in indoor settings with large white areas. To resolve this, we introduced a manual masking scheme to more accurately identify the light source area and defined a maximum area for potential light sources to avoid background misclassification. We also adapted the algorithm to detect multiple light sources, recognizing that most real-world images feature more than one light source. These enhancements are crucial for accurately segmenting light sources in training data, and we aim to further refine these detection techniques in the future.

### 1.2 Models

Following the approaches in Flare7k++[4] and Bracket Flare[5], we employed Uformer and MPRNet to restore images affected by scattering and reflective flares, respectively. Utilizing the pretrained models, which have demonstrated effectiveness in addressing localized scattering flare and in-focus reflective flare issues, we fine-tuned these models on our dataset for enhanced performance evaluation.

### 1.3 Training Details

For scattering flares, we trained the Uformer model with a batch size of 3 (due to GPU memory limits) over 50,000 iterations, processing images of $512 \times 512$ resolution. We enabled Automatic Mixed Precision (AMP) training to optimize resource use. The network was optimized using the Adam optimizer [7], with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We employed a combination of $l_1$ loss and perceptual loss [6] to achieve high-quality restoration. The training began with an initial learning rate of $1 \times 10^{-4}$, which was gradually reduced. This training phase was conducted on two RTX 3090 GPUs and completed in approximately two days.

For reflective flares, we implemented similar settings but adjusted the batch size to 2 and reduced the total iterations to 20,000, with images also processed at a $512 \times 512$ resolution. These sessions were carried out on a single RTX 4090 GPU and took about one day.

Both models were trained and evaluated using RAW2RGB and ISPRGB data to assess their performance comprehensively.

## 2 IMAGE SIGNAL PROCESSING OPERATIONS

In this section, we describe the operations used to simulate the non-invertible processing steps typically found in image signal processing pipelines. We introduce these operations and discuss their specific parameter settings used in our experiments.

### 2.1 Denoise

Denoising in image processing refers to the technique of removing noise from an image, thus enhancing its clarity and quality. Noise can arise from various factors, including sensor imperfections, poor lighting conditions, or high ISO settings in photography. Effective denoising is critical for many applications, such as medical imaging, satellite image analysis, and photography enhancement. Recently, deep learning-based methods have shown significant advancements in denoising, with architectures like NAFNet providing state-of-the-art results. NAFNet [2] uses a nonlinear activation-free network to adaptively refine feature representations, effectively reducing noise while preserving important details and textures.

In terms of denoising model, we use the NAFNet model, which has a depth of 32 and is trained on the Smartphone Image Denoising Dataset (SIDD) [1]. This dataset is crucial as it includes a variety of real noisy images from smartphone cameras, allowing researchers to develop and test algorithms under realistic noise conditions.

### 2.2 Sharpen

To approximate the varied sharpening approaches used by different smartphone manufacturers, we utilize a USM (Unsharp Masking) sharpening operator with a dynamic range of weights. Unsharp Masking (USM) sharpening is a technique used to enhance the sharpness of images by subtracting a blurred version of the image from the original. Specifically, the process involves creating a blurred copy of the original image $I$ using a Gaussian blur $B(I)$ with a standard deviation $\sigma$. The unsharp mask $M$ is then generated by computing:

$$M = I - B(I). \tag{1}$$

The final sharpened image $S$ is obtained by adding a scaled version of this mask back to the original image, given by:

$$S = I + k \cdot M \tag{2}$$

where $k$ is a scaling factor determining the strength of the sharpening. The parameters $\sigma$ and $k$ can be adjusted to control the extent of the blur and the impact of sharpening, respectively. In our experiments, we set $k$ to the range of 0.5 to 2.

## 2.3 Compression

The quality factor in image compression is a key parameter that dictates the trade-off between compression rate and image quality, particularly in lossy compression formats such as JPEG. A higher quality factor results in lesser data loss, preserving more details in the image and yielding larger file sizes, while a lower quality factor increases compression efficiency by allowing more significant data reduction, leading to reduced file sizes but poorer image quality. This factor is typically scaled from 1 to 100 in JPEG implementations, where values closer to 100 indicate minimal compression and maximum quality.

Additionally, DiffJPEG [8], a differentiable approximation of JPEG designed for use in deep learning pipelines, incorporates the traditional JPEG compression mechanics but allows for gradient-based optimization. DiffJPEG is particularly useful in training neural networks that need to be robust to JPEG compression artifacts, enabling a more seamless integration of image compression within machine learning models. It adjusts parameters like the quality factor during the learning process, optimizing for both performance and image quality preservation.

To simulate the mid-level compression, we use a quality factor in the range of 40 to 70 for compressing the image.

## 2.4 Evaluation

For our evaluation, we employ distinct pipelines that sequence image processing operations in various configurations to assess their impact. To thoroughly analyze the effects of these operations, we apply them to RAW2RGB data during both the training and inference stages. This approach allows us to discern how commonly utilized configurations influence the training dynamics and the performance of the neural networks.

# 3 COMPUTATIONAL COST

We detail the computational complexity of the models used in this paper in Table 1. We evaluate the computational demand for handling scattering flare, reflective flare, and noise reduction. Specifically, we calculate the number of parameters, the floating point operations (FLOPs), and the average inference time. The inference time is determined on an RTX 4090 by averaging over 1000 runs. The table reveals that MPRNet has higher GFLOPs compared to the Uformer model, despite having fewer parameters. This discrepancy is due to MPRNet's multi-stage training strategy, which, while increasing FLOPs, allows for better performance control over model size.

**Table 1: Computational cost for the models.**

| Model | GFLOPs | Params. (M) | Inference time (s) |
|---|---|---|---|
| MPRNet (Reflective) | 527.435 | 3.642 | 0.076 |
| Uformer (Scattering) | 150.818 | 20.474 | 0.047 |
| NAFNet (Denoiser) | 59.836 | 29.160 | 0.029 |

# REFERENCES


[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. 2018. A High-quality Denoising Dataset for Smartphone Cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1692–1700.

[2] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. 2022. Simple Baselines for Image Restoration. In *European Conference on Computer Vision*. Springer, 17–33.

[3] Yuekun Dai, Chongyi Li, Shangchen Zhou, Ruicheng Feng, and Chen Change Loy. 2022. Flare7K: A Phenomenological Nighttime Flare Removal Dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[4] Yuekun Dai, Chongyi Li, Shangchen Zhou, Ruicheng Feng, Yihang Luo, and Chen Change Loy. 2023. Flare7k++: Mixing Synthetic and Real Datasets for Nighttime Flare Removal and Beyond. *arXiv preprint arXiv:2306.04236* (2023).

[5] Yuekun Dai, Yihang Luo, Shangchen Zhou, Chongyi Li, and Chen Change Loy. 2023. Nighttime Smartphone Reflective Flare Removal using Optical Center Symmetry Prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 20783–20791.

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-time Style Transfer and Super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 694–711.

[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] Richard Shin and Dawn Song. 2017. Jpeg-resistant Adversarial Images. In *NIPS 2017 workshop on machine learning and computer security*, Vol. 1. 8.

[9] Yicheng Wu, Qiurui He, Tianfan Xue, Rahul Garg, Jiawen Chen, Ashok Veeraraghavan, and Jonathan T Barron. 2021. How to Train Neural Networks for Flare Removal. In *Proceedings of the IEEE International Conference on Computer Vision*. 2239–2247.