Zero-Reference Low-Light Enhancement via Physical Quadruple Priors (Supplementary Material)

Wenjing Wang	Huan Yang	Jianlong Fu	Jiaying Liu *
Peking University	01.AI	Microsoft Research Asia	Peking University

Contents

*Corresponding author. This work was supported in part by the National Natural Science Foundation of China under Grant 62332010, and in part by the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

1. More Experimental Results

1.1. Computational Complexity Comparison

We present the computational complexity (multiple-accumulate operations, MACs), network parameters, and processing time for input images of resolution $512 \times 512 \times 3$, coupled with the average low-light enhancement performance on LOL [18, 21] and MIT [1] in Tab. 1. These images are processed using a Tesla M40 with an Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz.

Both our full and lightweight versions exhibit superior performance in enhancing low-light conditions compared to previous methods. Moreover, our lightweight version stands out by requiring fewer computational resources than most state-of-the-art methods.

Table 1. Benchmarking results for low-light enhancement. Among unsupervised methods, we highlight the top-ranking scores in red, the second in blue, the third in green. Additionally, we denote the training set used by each model. "LOL+" indicates a fusion of LOL and other datasets.

Datasets		Train Set	Average on LOL [18, 21] and MIT [1]				Computational Complexity		
Metrics			PSNR↑	SSIM↑	LPIPS↓	LOE↓	MACs	Params	Time (s)
Supervised	Retinex-Net [18]	LOL	14.24	0.545	0.396	0.295	87.28 G	555.21 k	0.0931
	KinD++ [26]	LOL	16.20	0.656	0.365	0.232	258.62 G	8.27 M	0.6935
	KinD [25]	LOL	17.46	0.785	0.162	0.210	159.56 G	8.02 M	0.0964
	URetinex-Net [19]	LOL	17.51	0.794	0.143	0.216	229.0 G	340.11 k	0.2477
	Retinexformer [2]	MIT	18.98	0.666	0.214	0.221	68.39 G	1.61 M	0.3096
	Retinexformer [2]	LOL	21.18	0.785	0.170	0.240	68.39 G	1.61 M	0.3096
	DiffLL [6]	LOL+	22.17	0.795	0.173	0.233	21.88 G	22.08 M	0.3399
Unsupervised	RUAS [13]	FACE	10.03	0.411	0.528	0.345	950.8 M	3.438 k	0.0189
	RUAS [13]	LOL	10.31	0.431	0.487	0.364	950.8 M	3.438 k	0.0189
	RUAS [13]	MIT	11.58	0.536	0.324	0.282	950.8 M	3.438 k	0.0189
	SCI [15]	LOL+	12.40	0.553	0.336	0.238	95.16 M	0.258 k	0.0018
	SCI [15]	FACE	13.88	0.614	0.297	0.251	95.16 M	0.258 k	0.0018
	ZeroDCE++ [10]	own data	14.68	0.426	0.297	0.404	20.37 M	10.56 k	0.0023
	PairLIE [3]	LOL+	15.12	0.708	0.254	0.251	89.99 G	341.77 k	0.0562
	ExCNet [24]	test images	15.25	0.587	0.288	0.246	-	8.27 M	14.615
	ZeroDCE [4]	own data	15.59	0.649	0.258	0.243	20.92 G	79.42 k	0.0155
	CLIP-LIT [11]	own data	15.91	0.653	0.265	0.257	73.04 G	278.79 k	0.0543
	EnlightenGAN [7]	own data	15.93	0.719	0.253	0.245	65.88 G	8.64 M	0.0295
	SCI [15]	MIT	16.29	0.595	0.252	0.226	95.16 M	0.258 k	0.0018
	NeRCo [20]	LSRW [5]	18.50	0.744	0.237	0.262	822.8 G	23.30 M	0.9490
	Ours Lightweight	COCO [12]+LOL	19.30	0.784	0.233	0.224	9.83 G	327.36 k	0.0238
	Ours Full	COCO [12]	19.41	0.797	0.182	0.234	-	1.313 B	9.9616

1.2. Comparative Analysis: SCI with Varied Hyper-parameters and Training Data

We present low-light enhancement results of SCI [15] trained on different datasets: LOL [18], MIT [1], and FACE [22]. The training loss of SCI is given by $\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{f}} + \beta\mathcal{L}_{\text{s}}$; more details can be found in [15]. We also demonstrate results under various α values with $\beta = 1$.

As shown in Fig. 1, all versions of SCI struggle to enhance the extremely dark image in the first row. In the unbalanced case in the second row, all SCI versions either overly enhance the sky or fail to enhance the tree. In the daytime scenario in the third row, SCI-MIT, SCI-LOL, SCI-FACE, SCI-MIT $\alpha = 10$, and SCI-MIT $\alpha = 5$ lead to overexposure, while SCI-MIT $\alpha = 2$ maintains the input image almost unchanged. In contrast, our method achieves the best overall visual quality across all three scenarios with varying illumination levels.

Input SCI-MIT SCI-LOL SCI-FACE

SCI-MIT α=2 SCI-MIT α=5 SCI-MIT α=10 Ours

Figure 1. Comparison to SCI with varied hyper-parameters and training data.

1.3. Showcases of Using Different Priors in Our Framework

We display results of replacing our prior with alternative representations: (1) Naive HS channels in the HSV color space. (2) CIConv [9], a trainable prior similar to our W. (3) The reflectance estimated by Retinex-based PairLIE [3] In Fig. 2, HS channels cause noticeable artifacts, while CIConv results in color errors. Reflectance in PairLIE introduces a yellow color bias. Our model delivers the best visual quality.

Input | HS channels | CIConv | Reflectance in PairLIE | Ours

Figure 2. Comparison to our framework combined with other kinds of priors.

1.4. Application: Bypass Decoder for Colorization

We employ a ControlNet-based colorization model[1]. As shown in Fig. 3, our decoder excels in restoring fine details of the clock while preserving the resulting color distribution. This experiment demonstrates the efficacy of our decoder beyond our zero-reference low-light enhancement framework.

Test Image | SD | Consistency | Ours

Figure 3. Effects of different decoders for colorization.

[1] https://github.com/lllyasviel/ControlNet-v1-1-nightly/pull/26

1.5. More Showcases

We show more subjective comparison results on LOL, MIT-Adobe FiveK and the unpaired set in Figs. 4-9. Our model demonstrates superior capability in avoiding both underexposure and overexposure, effectively suppressing noise while avoiding artifacts such as black borders.

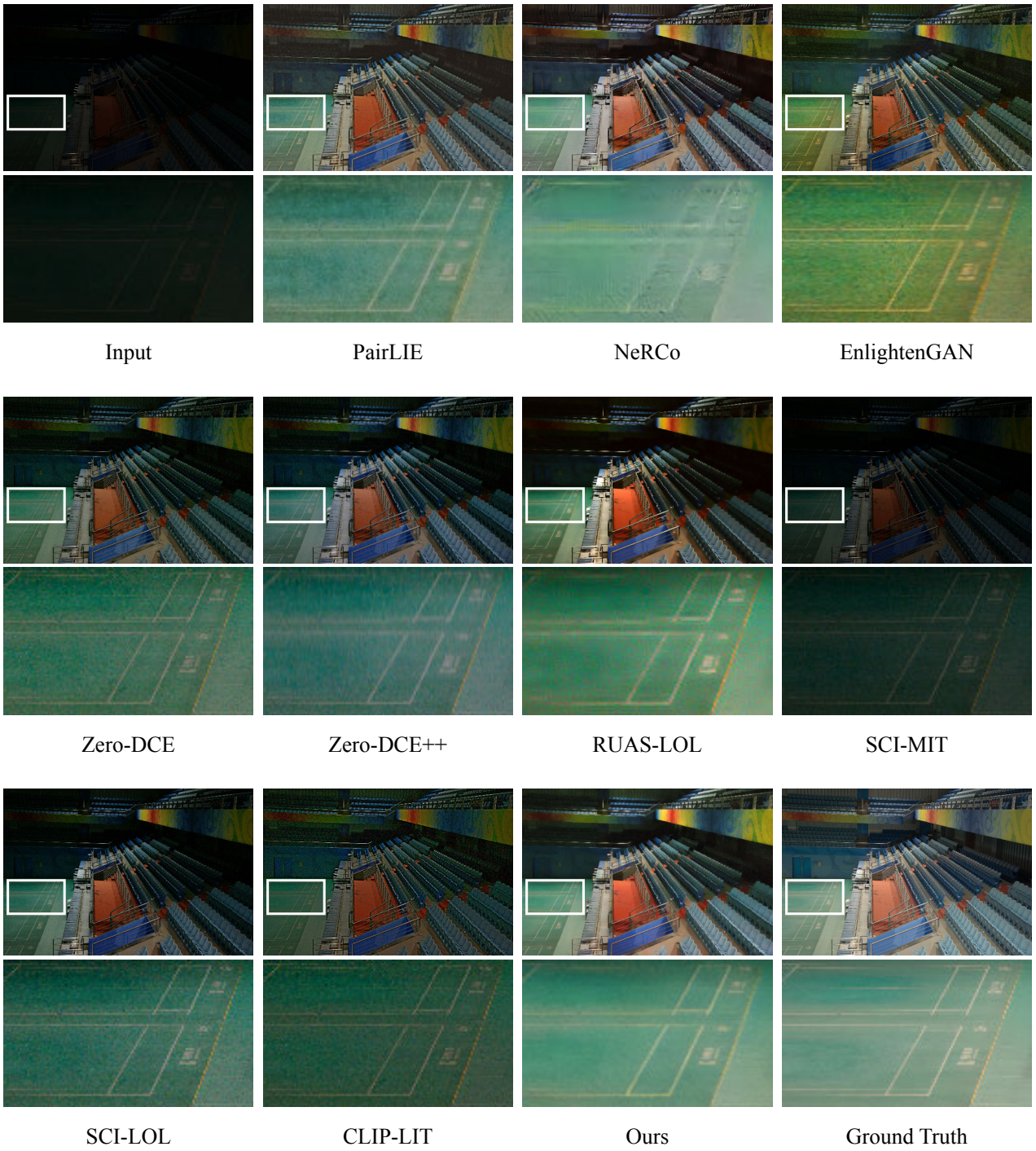


Figure 4. Low-light enhancement results on the LOL dataset. Our model is better at suppressing noise as well as improving the illumination.

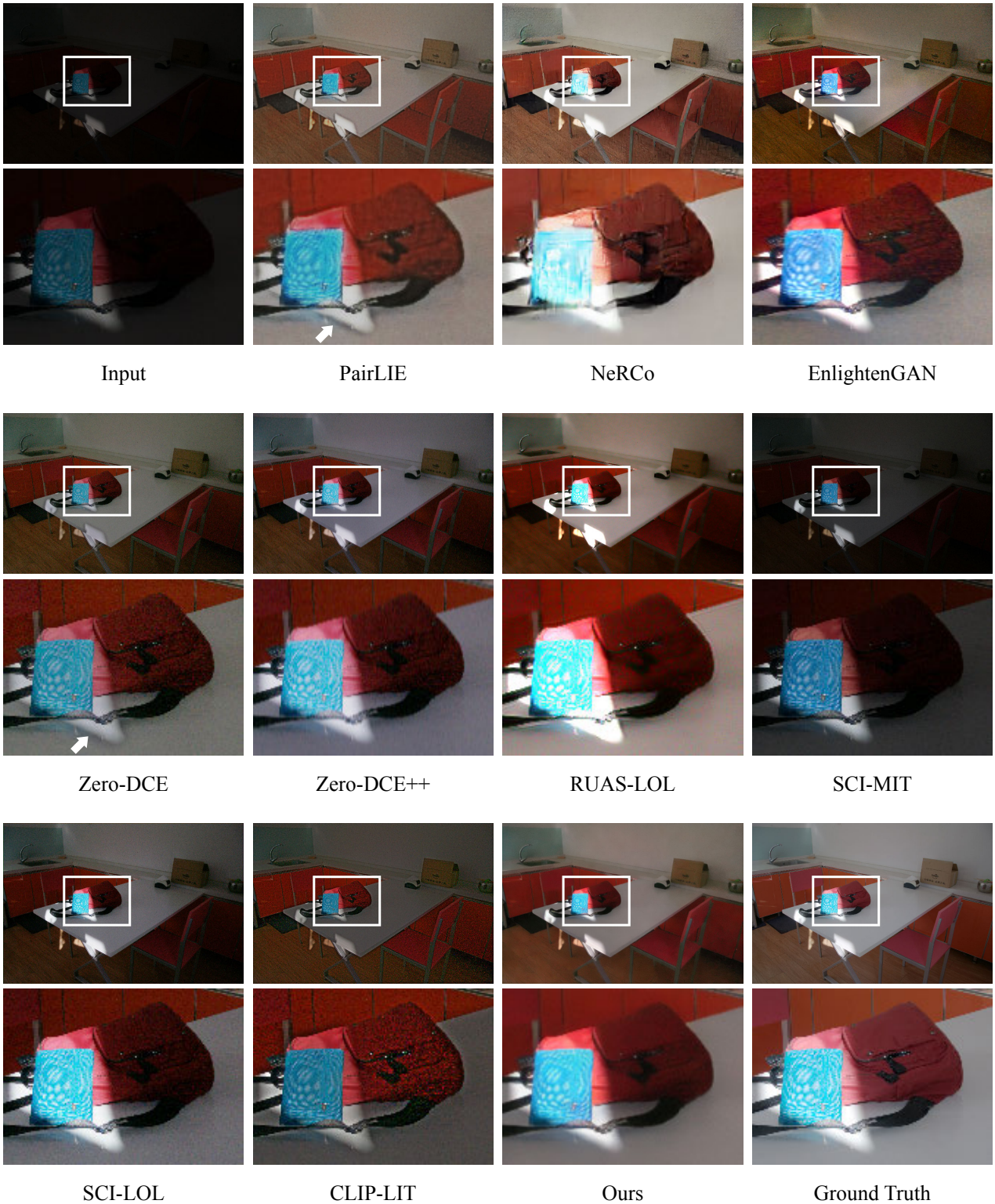
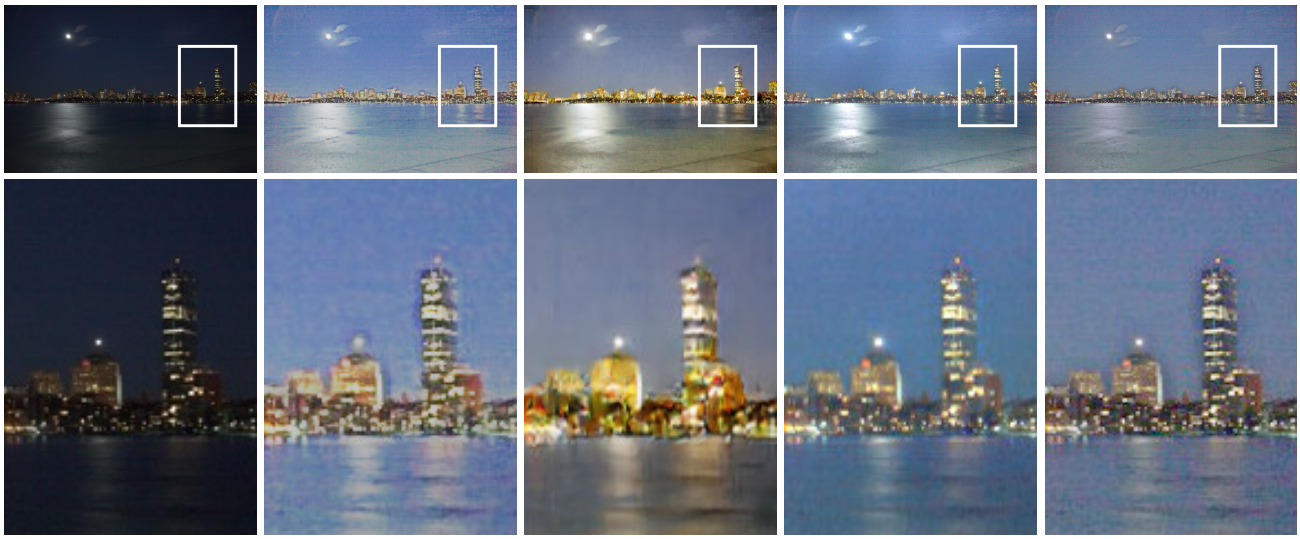
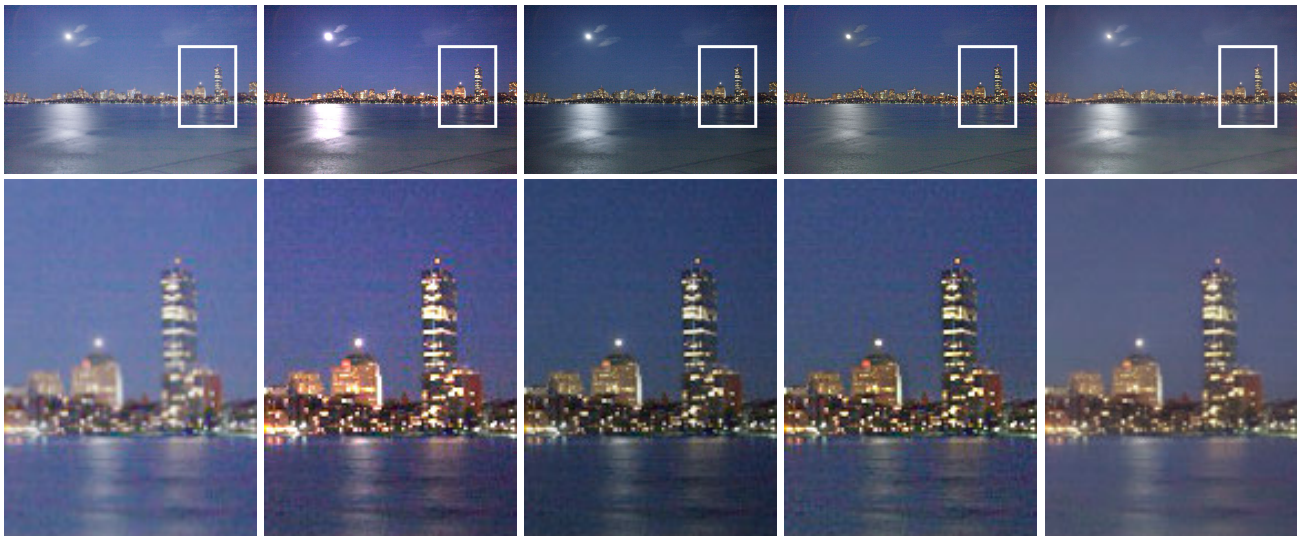


Figure 5. Low-light enhancement results on the LOL dataset about non-uniform illuminaiton. PairLIE [3] and Zero-DCE [4] generate weird black edges. EnlightenGAN [7] and CLIP-LIT [11] increase noises and artifacts.

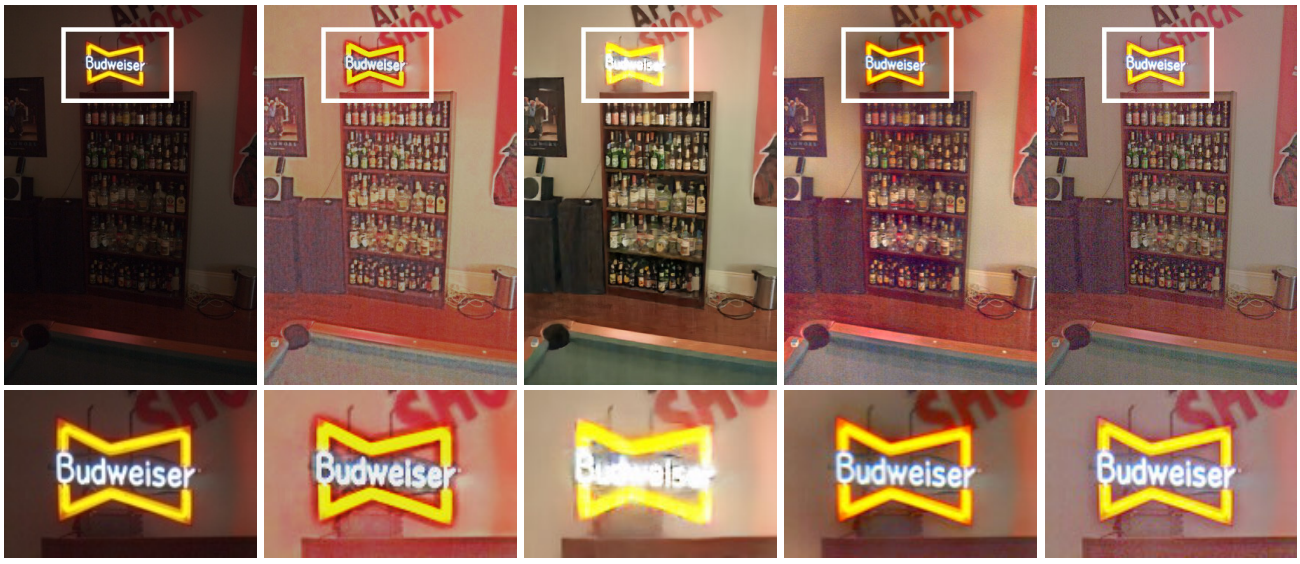


Input PairLIE NeRCo EnlightenGAN Zero-DCE



Zero-DCE++ RUAS-MIT SCI-MIT CLIP-LIT Ours

Figure 6. Low-light enhancement results on the MIT-Adobe FiveK dataset. PairLIE [3], EnlightenGAN [7], Zero-DCE [4], Zero-DCE++[10], RUAS[13] trained on MIT, SCI [15] trained on MIT, and CLIP-LIT [11] exhibit noise. NeRCo [20] compromises the structure of the building, whereas our model successfully removes noise while preserving the building's structure.



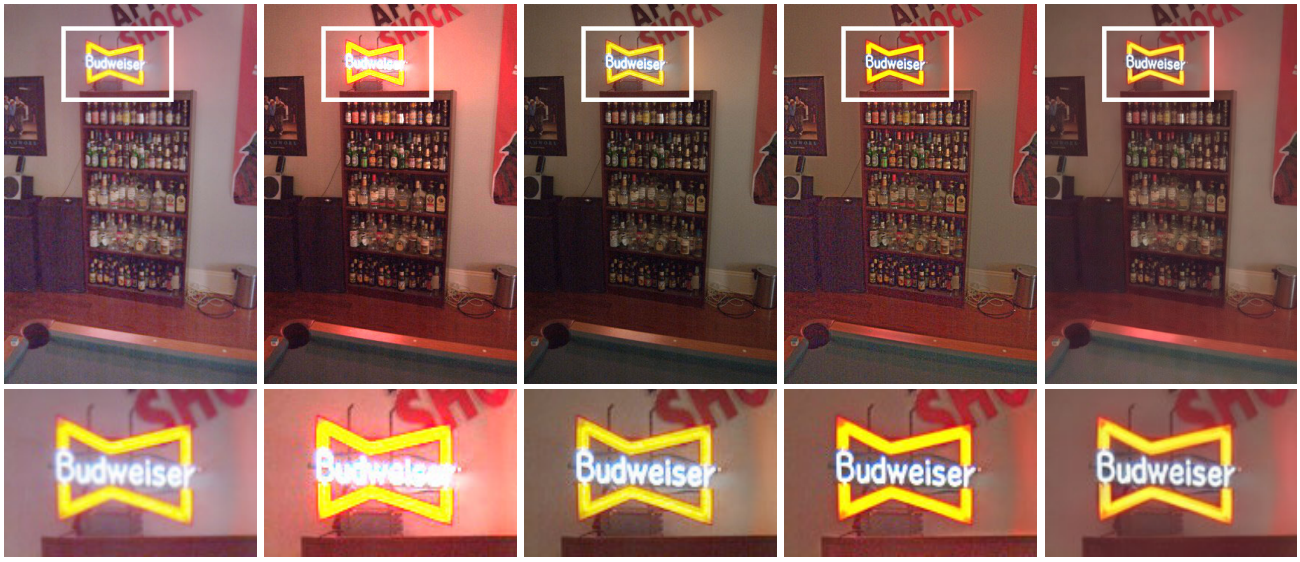
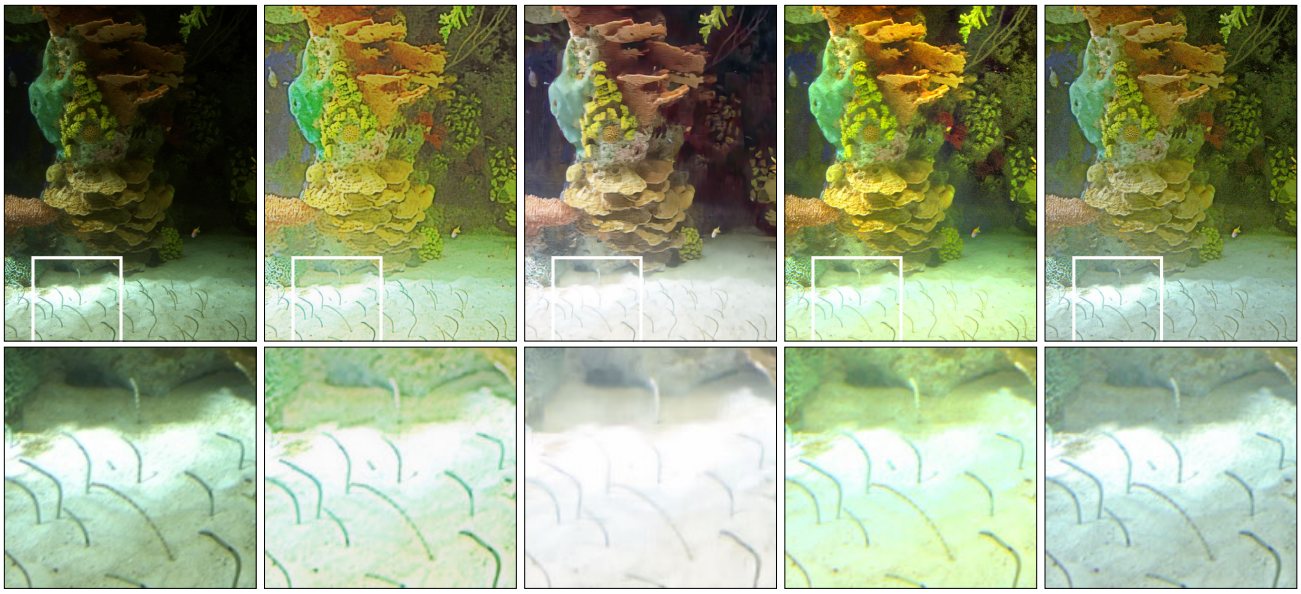


Figure 7. Low-light enhancement results on the MIT-Adobe FiveK dataset about light sources. NeRCo [20] and RUAS [13] trained on MIT suffer from over-exposure.

Figure 8. Low-light enhancement results on the unpaired datasets about pure dark area. PairLIE [3], Zero-DCE [4], and CLIP-LIT [11] cannot suppress noise.



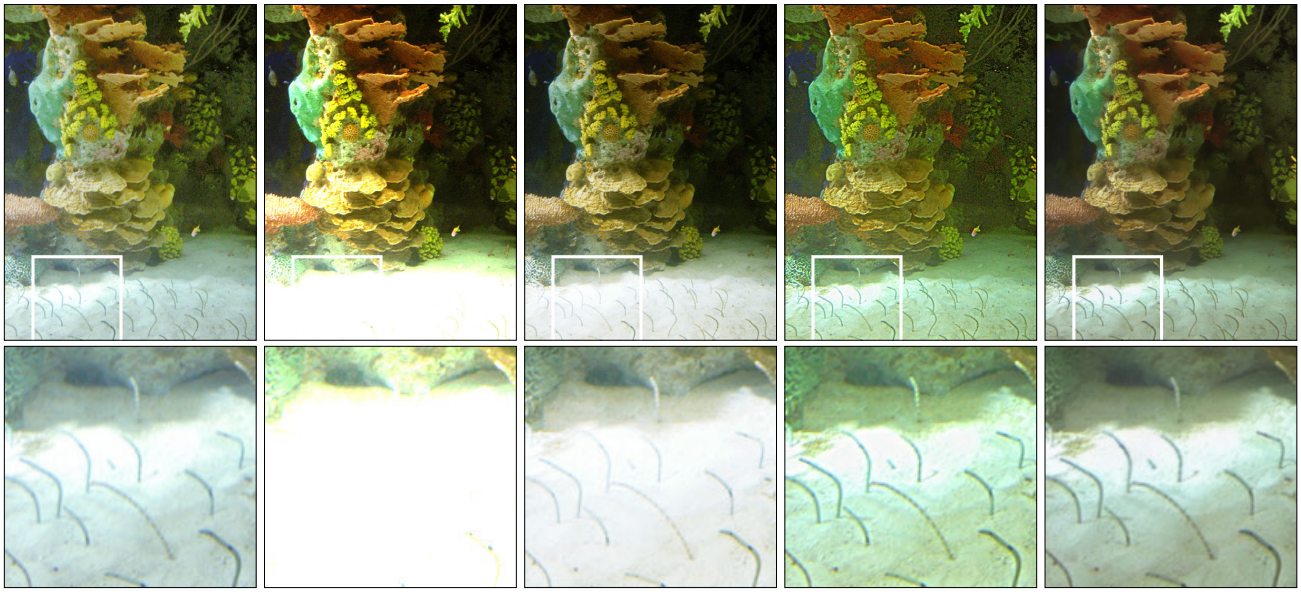
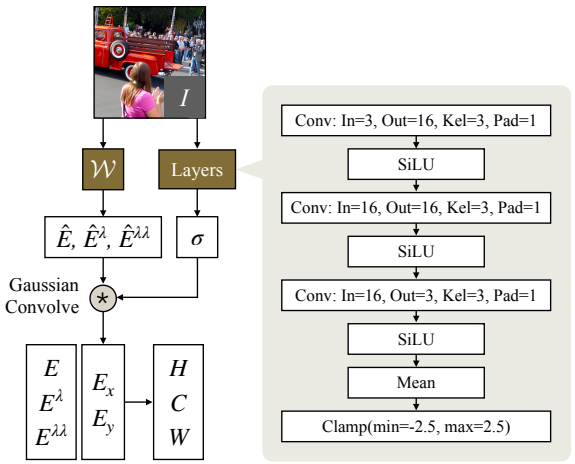


Figure 9. Low-light enhancement results on the unpaired datasets. NeRCo [20], RUAS [13] and SCI [15] suffer from over-exposure.

2. Implementation Details

2.1. Our Framework Design

Prior-to-Image. The architecture of our framework is shown in Fig.10(a) and Fig.11. When computing the physical quadruple prior from the input image, we introduce Gaussian noise with variance σ, where σ ranges within $[0, 0.15]$. Additionally, we incorporate $\pm 0.25\times$ signal-dependent Poisson noise. We further provide the code for computing O in Fig. 10(b). Our O can distinguish between different RGB orders and situations where two or three channels have the same values.



(a) Detailed architecture in Physical Quadruple Prior

```python
def O(batch):
    max_RGB = torch.argmax(batch, dim=1)
    min_RGB = torch.argmin(batch, dim=1)

    batch_ = torch.flip(batch, dims=(1,))

    max_RGB_ = 2 - torch.argmax(batch_, dim=1)
    min_RGB_ = 2 - torch.argmin(batch_, dim=1)

    RGB_order = torch.zeros(batch.shape, device=batch.device, dtype=batch.dtype)
    RGB_order = RGB_order.scatter_(1, max_RGB.unsqueeze(1), 0.5, reduce='add')
    RGB_order = RGB_order.scatter_(1, max_RGB_.unsqueeze(1), 0.5, reduce='add')
    RGB_order = RGB_order.scatter_(1, min_RGB.unsqueeze(1), -0.5, reduce='add')
    RGB_order = RGB_order.scatter_(1, min_RGB_.unsqueeze(1), -0.5, reduce='add')
    return RGB_order
```

(b) Code for computing O

Figure 10. The detailed network for computing H, C, and W, as well as the PyTorch [16] code for computing O.

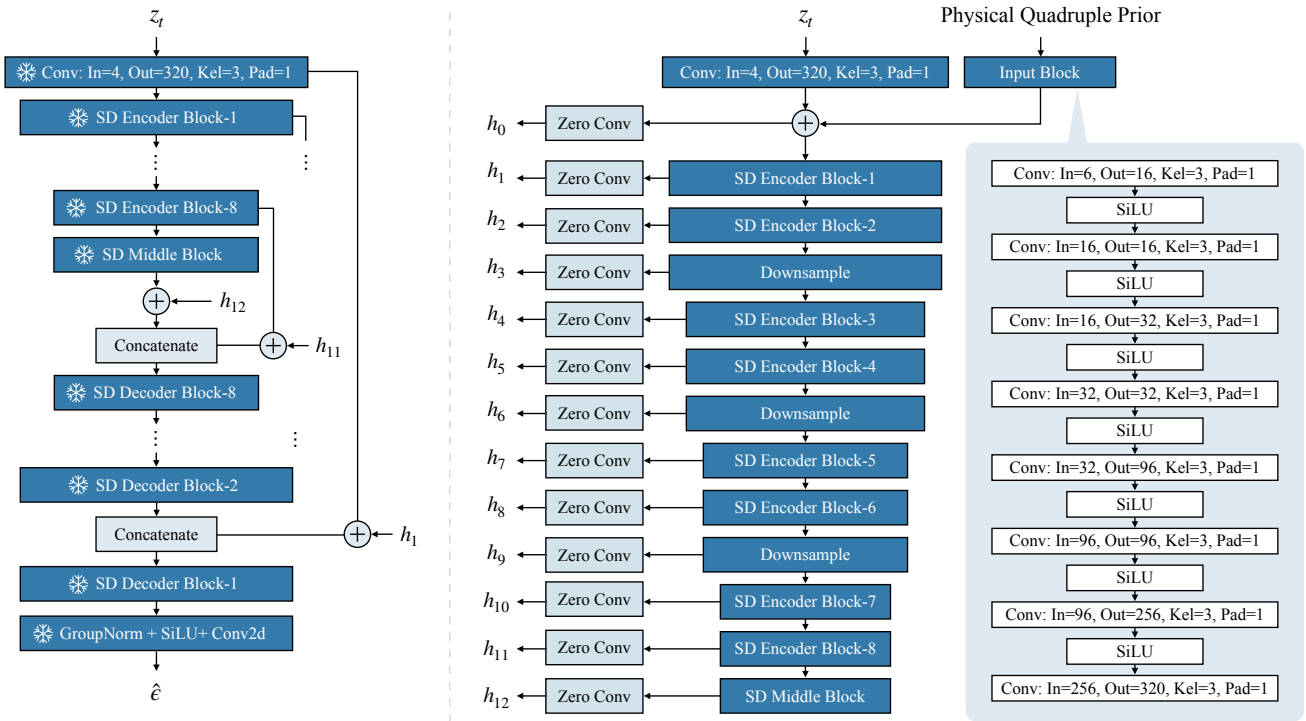


Figure 11. The detailed architecture of our prior-to-image framework.

We train it on both the COCO-2017 train set and an unlabeled set, totaling 241,690 images. Using a minibatch size of 8, we conduct training for approximately 140,000 steps, equivalent to around 5 epochs. Our learning rate is set at 1e-4, and we utilize the ADAM optimizer [8]. The GPUs used are Tesla M40 with 24GB memory, while CPUs are Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz. To fit the substantial model within limited GPU memory, we implement float16 precision and employ DeepSpeed [17]. Training is conducted across two Tesla M40 GPUs, requiring approximately 40GB of GPU memory in total.

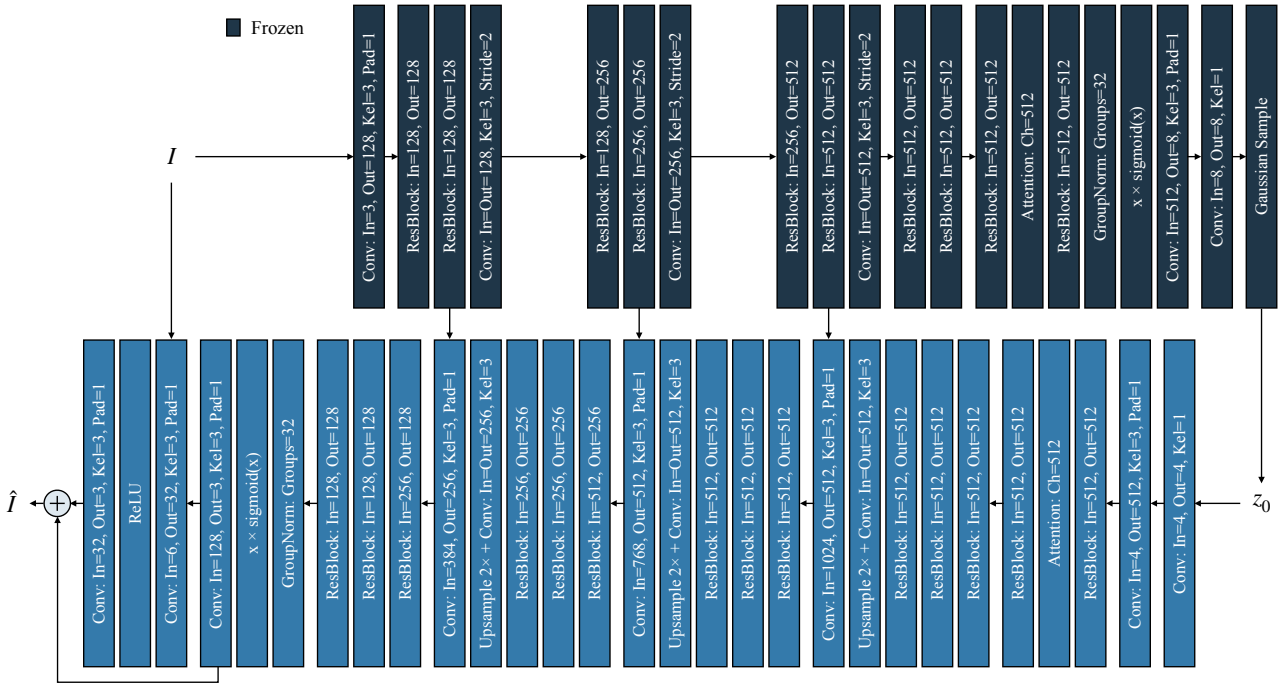


Figure 12. The detailed architecture for training our bypass decoder.

```python
def noise_1(image_tensor)
    noise_poisson = torch.poisson(image_tensor)
    noise_poisson_sign = torch.randint(low=0, high=2, size=image_tensor.shape) * 2 - 1
    noise_poisson = noise_poisson * noise_poisson_sign * random.uniform(0, 0.25)
    noise_gaussian = torch.randn(image_tensor.shape)
    noise_gaussian = noise_gaussian * random.uniform(0, 0.15)
    image_noise_tensor = image_tensor + noise_gaussian + noise_poisson
    return torch.clamp(image_noise_tensor, 0, 1)

def noise_2(image_tensor)
    noise_gaussian = torch.randn(image_tensor.shape)
    noise_gaussian = noise_gaussian * random.uniform(0, 0.3)
    image_noise_tensor = image_tensor + transforms.functional.gaussian_blur(noise_gaussian, kernel_size=15)
    return torch.clamp(image_noise_tensor, 0, 1)

def noise_3(image_tensor)
    noise_gaussian = torch.randn((image_tensor.shape[0], image_tensor.shape[1]//64, image_tensor.shape[2]//64))
    noise_gaussian = noise_gaussian * random.uniform(0, 0.1)
    noise_gaussian = noise_gaussian.unsqueeze(0)
    noise_gaussian = torch.nn.functional.interpolate(noise_gaussian, scale_factor=64, mode='bicubic')[0]
    image_noise_tensor = image_tensor + noise_gaussian
    return torch.clamp(image_noise_tensor, 0, 1)
```

Figure 13. The code for applying noise to I to generate $\hat{I}$.

During inference, we utilize the multistep DPM-Solver++ data prediction model [14] with the order set to 3. This involves sequentially employing one step of DPM-Solver-1, one step of multistep DPM-Solver-2, and eight steps of multistep DPM-Solver-3. Using DPM-Solver++ instead of DDIM decreases the sampling steps from 50 to 10.

Bypass Decoder. The detailed architecture of our bypass decoder is illustrated in Fig. 12. The encoding-related layers are frozen. For generating $\hat{I}$, we use a combination of three kinds of noises. The code for applying each kind of noise (noise_1, noise_2, noise_3) is provided in Fig. 13.

For training the bypass decoder, the model undergoes fine-tuning for 7,000 steps, employing a mini-batch size of 6, operating at a resolution of 256. The learning rate is set to 1e-4, utilizing the ADAM optimizer[8].

Lightweight Version. Even with DPM-Solver++, 10 steps are still cumbersome. In the pursuit of practicality, our framework can create a more lightweight version.

We construct a lightweight U-net consisting of residual blocks and integrate transformer blocks from Restormer [23] at the bottleneck. The detailed architecture is illustrated in Fig. 14. To improve the adaptability of the attention to various resolutions, we further insert positional embedding into the Multi-DConv Head Transposed Self-Attention. For a detailed implementation of the Multi-DConv Head Transposed Self-Attention, please refer to [23].

To train this U-net, we generate 1.7k input-synthetic data pairs using our full framework. Among these samples, 1.2k are from COCO [12], and 485 are from the LOL v1 train set [18]. These samples are split, with 1.3k for training and the rest for validation. The training objective is solely the L1 loss between the prediction of the lightweight U-net and that of the full framework. Our learning rate is set at 1e-4, and we utilize the ADAM optimizer [8]. The training lasts for 200 epochs. The GPU utilized is also a Tesla M40 with 24GB memory, and the CPUs are also Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz.

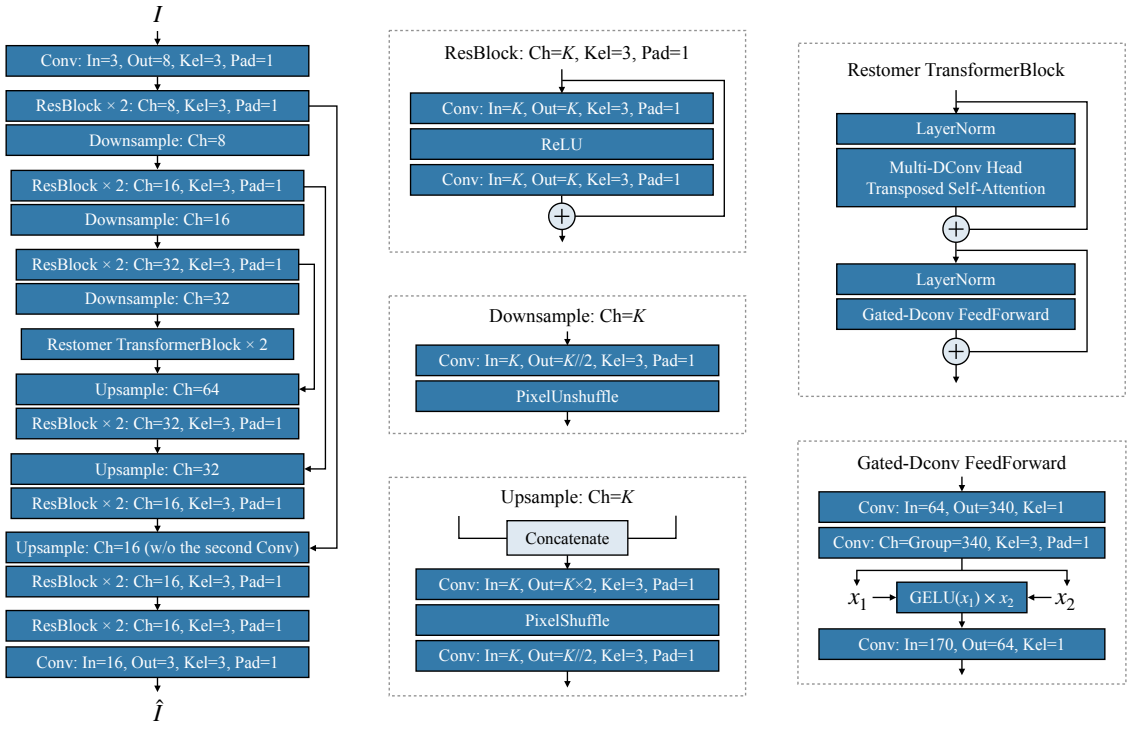


Figure 14. The detailed architecture of our lightweight U-net.

2.2. Compared Methods

For compared methods, we use the official codes and trained models released by the authors, which are listed in Tab. 2.

Table 2. Code sources of compared methods.

Method	Link
RetinexNet [18]	https://github.com/weichen582/RetinexNet
KinD [25]	https://github.com/zhangyhuaee/KinD
KinD++ [26]	https://github.com/zhangyhuaee/KinD_plus
URetinexNet [19]	https://github.com/AndersonYong/URetinex-Net
Retinexformer [2]	https://github.com/caiyuanhao1998/Retinexformer
DiffLL [6]	https://github.com/JianghaiSCU/Diffusion-Low-Light
ExCNet [24]	https://zhanglijun95.github.io/ExCNet/
EnlightenGAN [7]	https://github.com/VITA-Group/EnlightenGAN
PairLIE [3]	https://github.com/zhenqifu/PairLIE
NeRCo [20]	https://github.com/Ysz2022/NeRCo
CLIP-LIT [11]	https://github.com/ZhexinLiang/CLIP-LIT
Zero-DCE [4]	https://github.com/Li-Chongyi/Zero-DCE
Zero-DCE++ [10]	https://github.com/Li-Chongyi/Zero-DCE_extension
RUAS [13]	https://github.com/KarelZhang/RUAS
SCI [15]	https://github.com/vis-opt-group/SCI

References

[1] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011.

[2] Yuanhao Cai, Hao Bian, Jing Lin, Haoqian Wang, Radu Timofte, and Yulun Zhang. Retinexformer: One-stage retinex-based transformer for low-light image enhancement. In *ICCV*, 2023.

[3] Zhenqi Fu, Yan Yang, Xiaotong Tu, Yue Huang, Xinghao Ding, and Kai-Kuang Ma. Learning a simple low-light image enhancer from paired low-light instances. In *CVPR*, 2023.

[4] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *CVPR*, 2020.

[5] Jiang Hai, Zhu Xuan, Ren Yang, Yutong Hao, Fengzhu Zou, Fang Lin, and Songchen Han. R2RNet: Low-light image enhancement via real-low to real-normal network. *Journal of Visual Communication and Image Representation*, 90:103712, 2023.

[6] Hai Jiang, Ao Luo, Songchen Han, Haoqiang Fan, and Shuaicheng Liu. Low-light image enhancement with wavelet-based diffusion models. In *Siggraph Asia*, 2023.

[7] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. EnlightenGAN: Deep light enhancement without paired supervision. *IEEE TIP*, 30:2340–2349, 2021.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[9] Attila Lengyel, Sourav Garg, Michael Milford, and Jan C. van Gemert. Zero-shot domain adaptation with a physics prior. In *ICCV*, 2021.

[10] Chongyi Li, Chunle Guo, and Chen Change Loy. Learning to enhance low-light image via zero-reference deep curve estimation. *IEEE TPAMI*, 44(8):4225–4238, 2021.

[11] Zhexin Liang, Chongyi Li, Shangchen Zhou, Ruicheng Feng, and Chen Change Loy. Iterative prompt learning for unsupervised backlit image enhancement. In *ICCV*, 2023.

[12] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[13] Risheng Liu, Long Ma, Jiaao Zhang, Xin Fan, and Zhongxuan Luo. Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement. In *CVPR*, 2021.

[14] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv*, 2022.

[15] Long Ma, Tengyu Ma, Risheng Liu, Xin Fan, and Zhongxuan Luo. Toward fast, flexible, and robust low-light image enhancement. In *CVPR*, 2022.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[17] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD*, 2020.

[18] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *BMVC*, 2018.

[19] Wenhui Wu, Jian Weng, Pingping Zhang, Xu Wang, Wenhan Yang, and Jianmin Jiang. Uretinex-net: Retinex-based deep unfolding network for low-light image enhancement. In *CVPR*, 2022.

[20] Shuzhou Yang, Moxuan Ding, Yanmin Wu, Zihan Li, and Jian Zhang. Implicit neural representation for cooperative low-light image enhancement. In *ICCV*, 2023.

[21] Wenhan Yang, Shiqi Wang, Yuming Fang, Yue Wang, and Jiaying Liu. From fidelity to perceptual quality: A semi-supervised approach for low-light image enhancement. In *CVPR*, 2020.

[22] Wenhan Yang, Ye Yuan, Wenqi Ren, Jiaying Liu, Walter J Scheirer, Zhangyang Wang, Taiheng Zhang, Qiaoyong Zhong, Di Xie, Shiliang Pu, et al. Advancing image understanding in poor visibility environments: A collective benchmark study. *IEEE TIP*, 29: 5737–5752, 2020.

[23] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022.

[24] Lin Zhang, Lijun Zhang, Xinyu Liu, Ying Shen, Shaoming Zhang, and Shengjie Zhao. Zero-shot restoration of back-lit images using deep internal learning. In *ACM MM*, 2019.

[25] Yonghua Zhang, Jiawan Zhang, and Xiaojie Guo. Kindling the darkness: A practical low-light image enhancer. In *ACM MM*, 2019.

[26] Yonghua Zhang, Xiaojie Guo, Jiayi Ma, Wei Liu, and Jiawan Zhang. Beyond brightening low-light images. *IJCV*, 129(4):1013–1037, 2021.