Low-Light Image Enhancement with Normalizing Flow

Anonymous AAAI submission

Paper ID 1489

6 Detailed architecture

6.1 The architecture of the conditional encoder

For the design of conditional encoder g, we follow the core idea of residual connection (RRDB) (Wang et al. 2018) to capture more information from low-light images. More specifically, the backbone of our conditional encoder is composed by 24 Residual-in-Residual Dense Blocks (RRDB) (Wang et al. 2018) and the details are shown in Table 1.

To capture a richer representation of low-light images, we concatenate the shallow layer features $z_{shallow} = [z_5, z_7, z_9, z_{11}]$ to persevere more information from the input image, where [] represent the concatenation operation. Then $z_{shallow}$ is interpolated to the same size with $(z_4 + z_{29})$, (z_{31}), and z_{34}, respectively. Finally, $[z_{shallow}, z_4 + z_{29}]$, $[z_{shallow} \downarrow_2, z_{31}]$, and $[z_{shallow} \downarrow_4, z_{34}]$ are used as the conditional features for each level of the invertible network, where $\downarrow$ represents the interpolating operation.

6.2 The architecture of the invertible network

The invertible network we proposed is composed of three levels. For each level, it is composed by a squeeze layer and 12 flow-steps. For each flow step, it consists of a conditional affine coupling layer, an affine injector layer, a 1×1 Convolutional layer, and an Actnorm layer. A brief introduction of each component is illustrated as follows:

Squeeze: To capture the features at different scales, the squeeze layer is proposed by (Kingma and Dhariwal 2018). More specifically, the feature map with shape $C \times H \times W$ is reshaped to $4C \times H//2 \times W//2$ to increase the receptive field of the network.

Invertible 1 $\times$ 1 Convolution: The function and operation are the same with the vanilla convolution layer with the kernel size 1, i.e., $z_{ij}^{n+1} = W z_{ij}^{n} + b$ where $z_i j$ is the feature vector at coordinate (i, j). The reason why the kernel size is limited to 1 is that its determinant can be efficiently calculated by (Kingma and Dhariwal 2018) comparing with others.

Conditional Affine Coupling: Different from the affine coupling layer used in unconditional generative task (Dinh, Krueger, and Bengio 2014; Kingma and Dhariwal 2018), the low light enhancement task requires that we need to introduce the conditional feature $g(x_l)$ into the invertible network. To this end, we utilize the extended version of the affine coupling layer from (Lugmayr et al. 2020) which takes the feature from the encoder that we elaborate previously as an extra input so that the connection between low light images and normally exposed images will be established. The operation of the conditional affine coupling layer is as flows:

$$h_A^{i+1} = h_A^{i+1}, \quad h_B^{i+1} = \exp\left(\theta_s^i([h_A^i, z_i]) \cdot h_B^i\right) + \theta_b^i([h_a^i, z_i]) \tag{1}$$

where z_i is the conditional feature from the encoder with the compatible size, $theta_s^i$ and $theta_b^i$ are two networks that predict the **s**cale and **b**ias respectively.

Affine Injector: To build a stronger connection between the conditional feature and the high light image x_{gt}, affine injector proposed by (Lugmayr et al. 2020) is used which only take the conditional feature from the low light image as input defined as follows:

$$h^{i+1} = \exp\left(\theta_s^i(z_i)\right) \cdot h^i + \theta_b^i(z_i). \tag{2}$$

Actnorm: Similar to batch normalization, actnorm is an activation normalization that performs an affine transformation of the activations using a scale and bias parameter per channel by (Kingma and Dhariwal 2018).

The networks θ_s^i and θ_b^i used in conditional affine coupling layer and affine injector layer are composed by two shared convolutional layers with 64 hidden channels with ReLU and a convolutional layer to predict the scale and bias respectively.

7 Experiment environment

We will release the code after the paper is accepted. Parts of the experiments are conducted on a Windows workstation with Ryzen 5900X, 64GB RAM, and an Nvidia RTX 3090. Others are conducted on a Linux server with Intel(R) Xeon(R) Silver 4210R CPU, 256GB RAM, and RTX 2080-TIs. For the software, PyTorch 1.9 is used. We do not find a significant difference of trained models under two different environments.

The experiments are repeated 3 times using LOL dataset, and we report the mean value and standard deviation (std) in Table 2. As we can see, our method achieves stable performance on all metrics especially in terms of SSIM and LPIPS. The results demonstrate that our method can stably

#	input	output	Layer
1	$[x_l, h(x_l), C(x_l), N(x_l)]$	z_1	Conv2D(in=12,out=64,kernel_size=3,stride=1,padding=1)
2	z_1	z_2	LeakyRelu(negative_slope=0.2)
3	z_2	z_3	Conv2D(in=64,out=64,kernel_size=3,stride=1,padding=1)
4	z_3	z_4	MaxPool2D(kernel_size=2, padding=0, dilation=1)
5	z_4	z_5	RRDB block(nf=64, gc=32)
...	...	...	...
28	z_{27}	z_{28}	RRDB block(nf=64, gc=32)
29	z_{28}	z_{29}	Conv2D(in=64,out=64,kernel_size=3,stride=1,padding=1)
30	$z_4 + z_{29}$	z_{30}	DownSampling(scale_factor=0.5)
31	z_{30}	z_{31}	Conv2D(in=64,out=64,kernel_size=3,stride=1,padding=1)
32	z_{31}	z_{32}	LeakyRelu(negative_slope=0.2)
30	z_{32}	z_{33}	DownSampling(scale_factor=0.5)
31	z_{33}	z_{34}	Conv2D(in=64,out=64,kernel_size=3,stride=1,padding=1)
30	$z_4 + z_{29}$	z_{35}	UpSampling(scale_factor=2)
35	z_{35}	$g(x_l)$	Conv2D(in=64,out=3,kernel_size=3,stride=1,padding=1)

Table 1: The architecture of the conditional encoder.

preserve structural information with high-frequency details and improve the human perception quality.

	PSNR	SSIM	LPIPS
LLFlow	25.19 ± 0.22	0.93 ± 0.0021	0.11 ± 0.0029

Table 2: The mean value and std of our repeated experiments on LOL dataset.

8 Extra visualization results

More visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL and LOL datasets are placed after the reference.

References

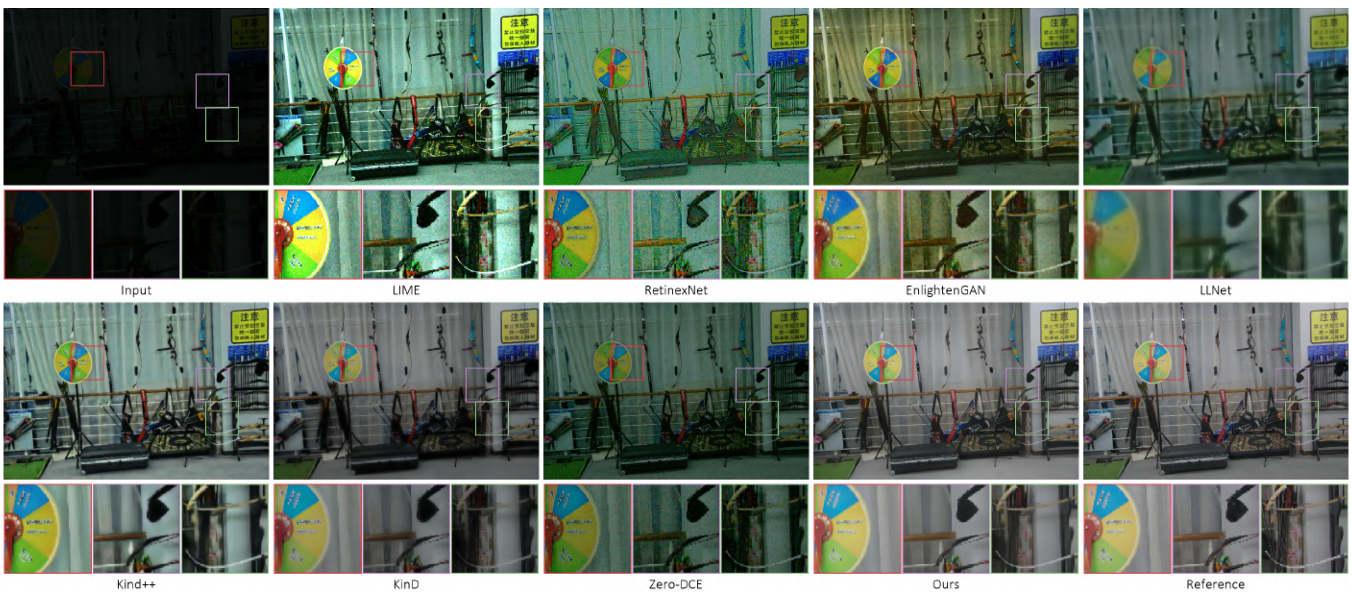
Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

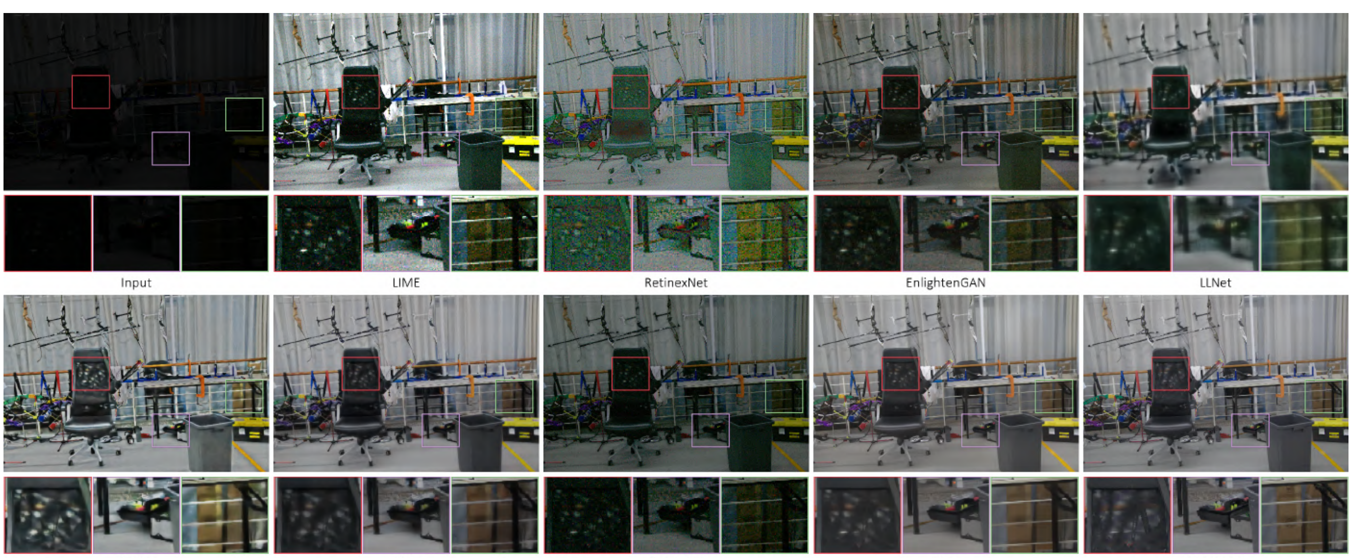
Lugmayr, A.; Danelljan, M.; Van Gool, L.; and Timofte, R. 2020. Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, 715–732. Springer.

Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Qiao, Y.; and Change Loy, C. 2018. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

(a)



(b)



(c)

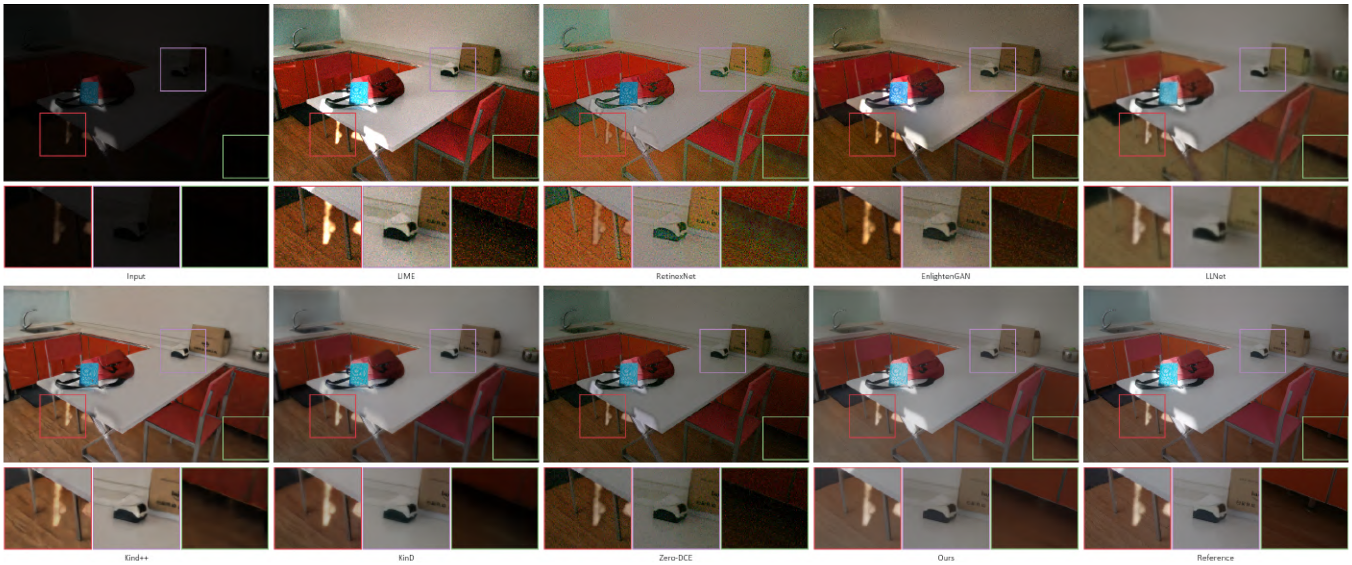
Figure 1: Visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL dataset.

(a)

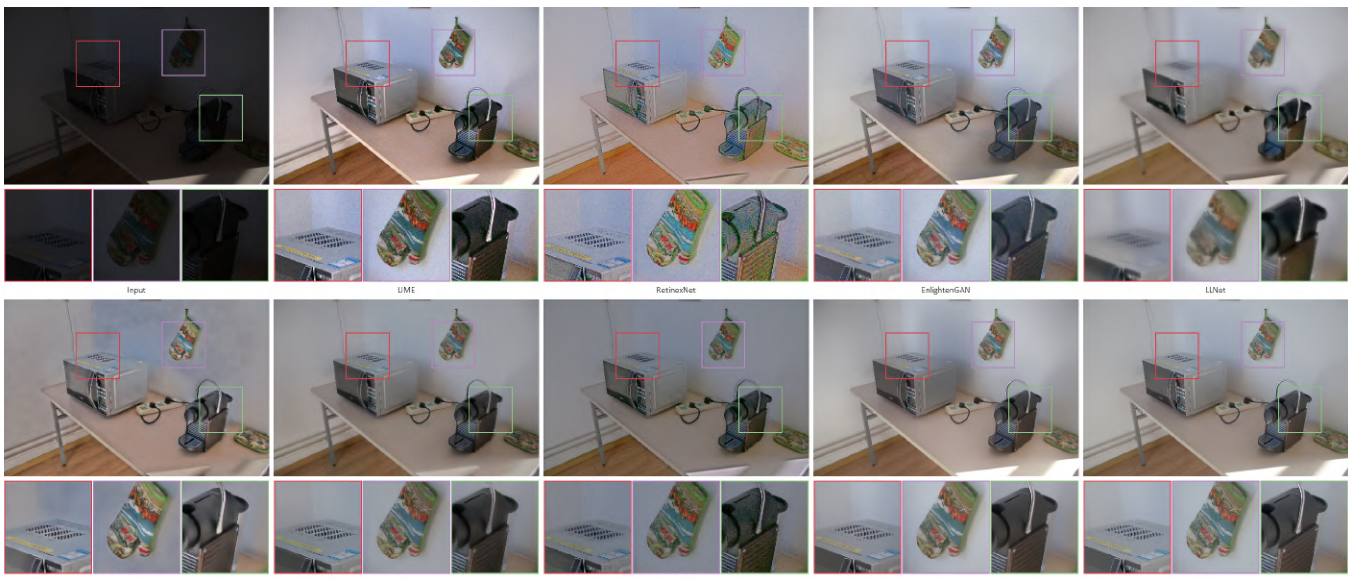
(b)

(c)

Figure 2: Visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL dataset.



(a)



(b)

(c)

Figure 3: Visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL dataset.

(a)

(b)

(c)

Figure 4: Visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL dataset.

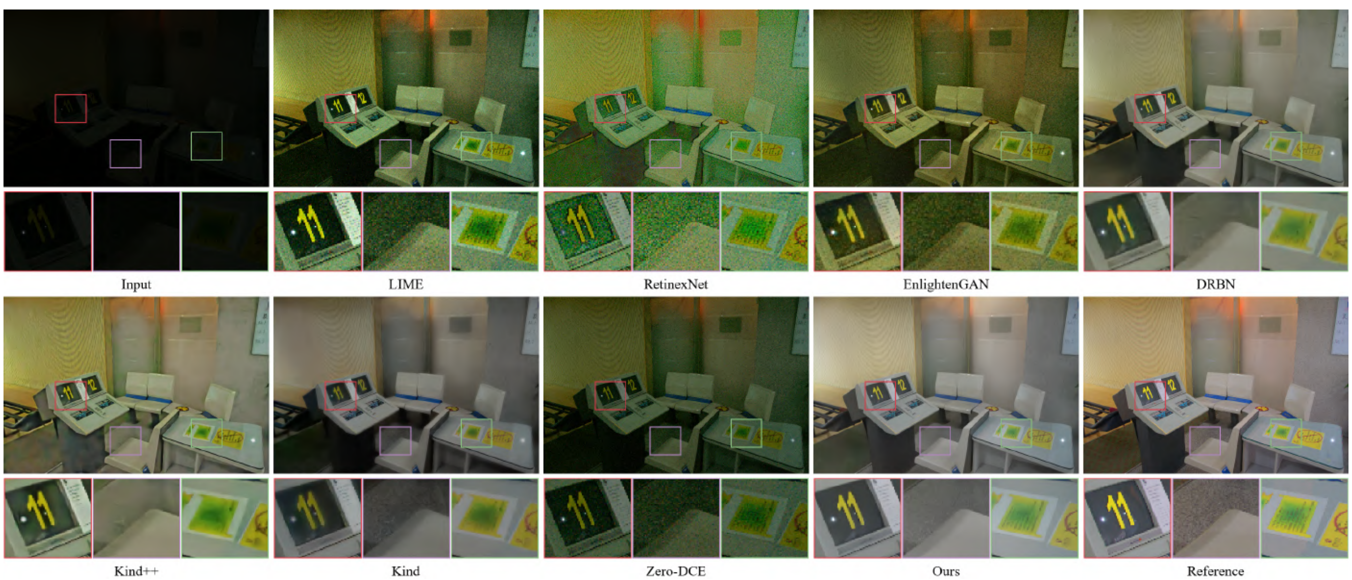
(a)

(b)

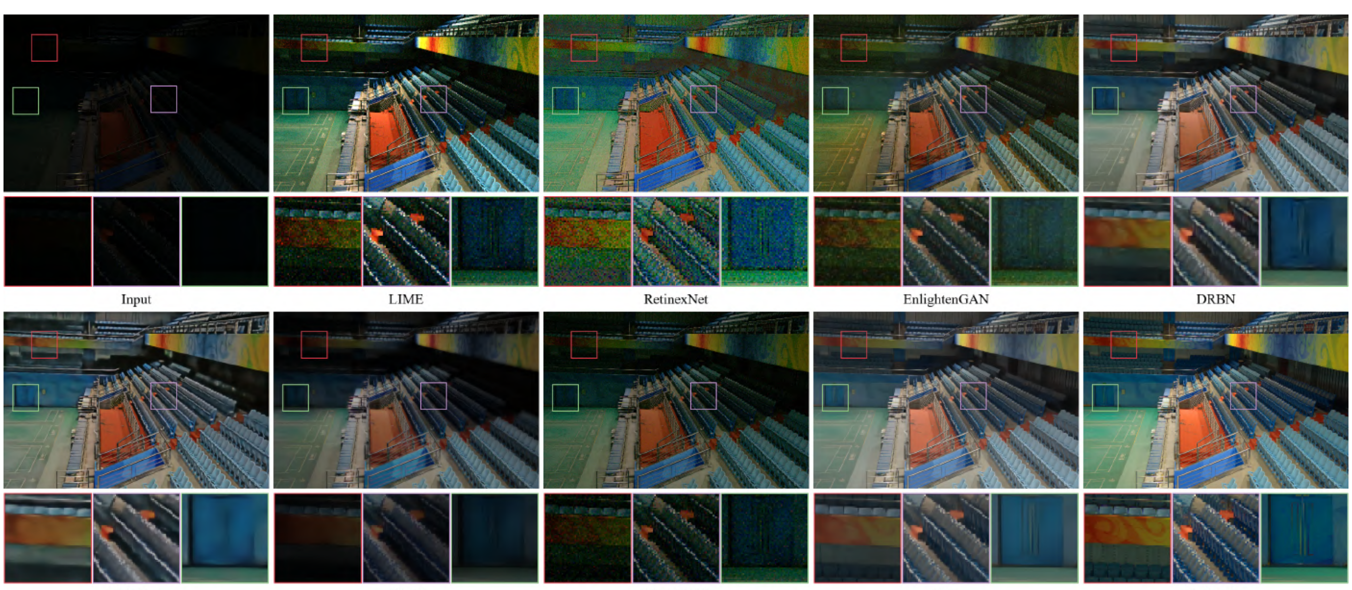
(c)

Figure 5: Visual comparison with state-of-the-art low-light image enhancement methods on VE-LOL dataset.

(a)



(b)



(c)

Figure 6: Visual comparison with state-of-the-art low-light image enhancement methods on LOL dataset.

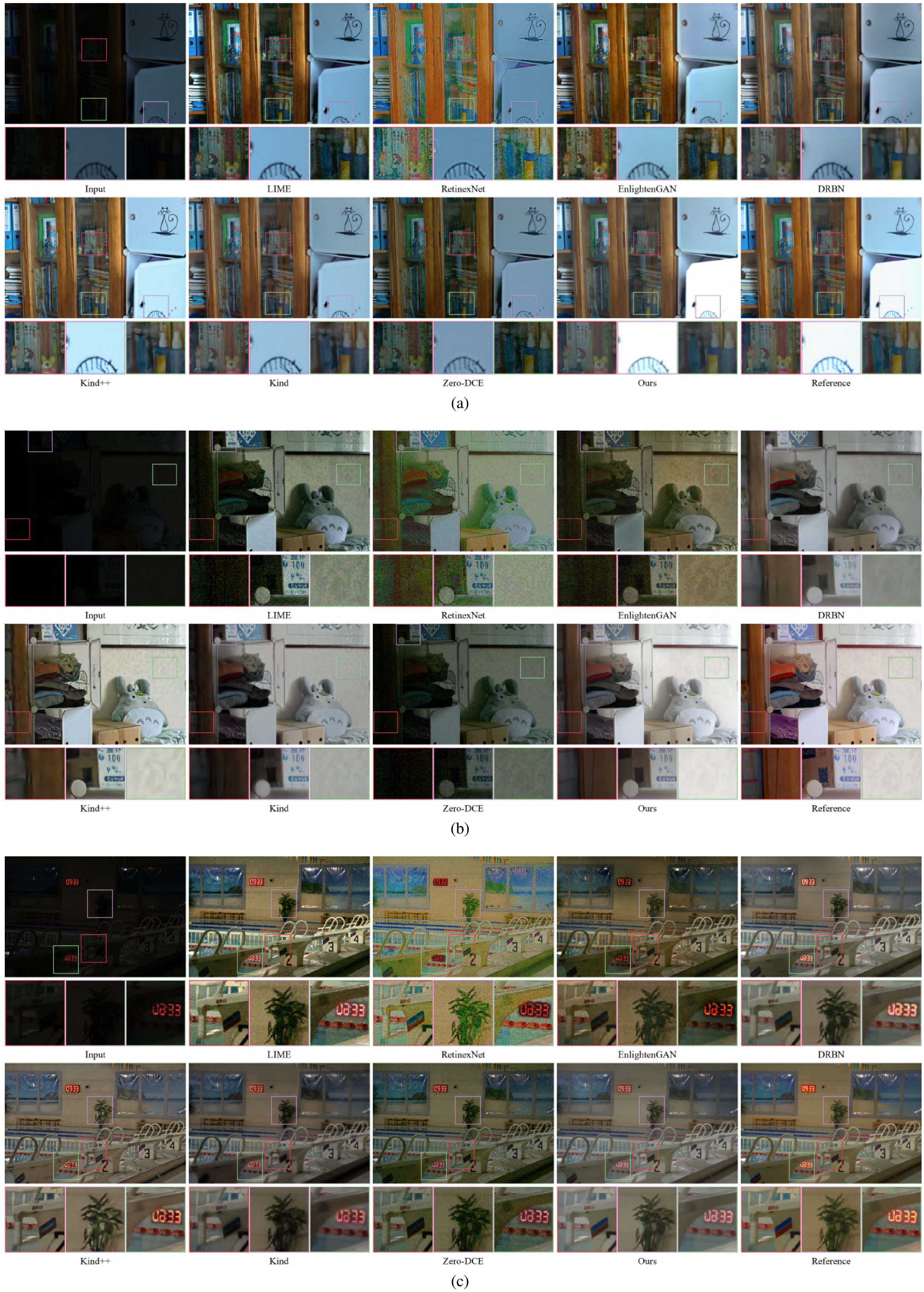


Figure 7: Visual comparison with state-of-the-art low-light image enhancement methods on LOL dataset.