Supplementary Material: Binarized Diffusion Model for Image Super-Resolution

Zheng Chen[1], **Haotong Qin**[2*], **Yong Guo**[3], **Xiongfei Su**[4],
Xin Yuan[4], **Linghe Kong**[1], **Yulun Zhang**[1*]
[1]Shanghai Jiao Tong University, [2]ETH Zürich,
[3]Max Planck Institute for Informatics, [4]Westlake University

1 Diffusion Model for Image Super-Resolution

We apply the diffusion model, conditioned on the LR image $\mathbf{y} \in \mathbb{R}^{h \times w \times 3}$ ($h \times w$ is the LR resolution), to realize image super-resolution (SR) [8]. The DM includes forward and reverse processes. Given the HR/LR pair ($\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, $\mathbf{y}$), where $H \times W$ is the HR resolution, it is defined as follows.

Forward Process. In this process, Gaussian noise is added to the HR image $\mathbf{x}$ over T iterations. For consistency, we set $\mathbf{x}_0 = \mathbf{x}$. Then, the process is formulated as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}), \tag{1}$$

where $t \in \{1, 2, \ldots, T\}$ represents the timestep index, and $\beta_t \in (0, 1)$ is the hyperparameter that controls the noise level. When $T \to \infty$, $\mathbf{x}_T$ approximates the Gaussian distribution [2].

Reverse Process. This process aims to estimate the posterior distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ through a learned conditional distribution $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{y})$. Note that the distribution p_θ also condition on the LR image $\mathbf{y}$ (bicubic to HR resolution) to constrain the generation scope. The function is defined as:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{y}) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_\theta(\mathbf{x}_t, \mathbf{y}, t), \tilde{\beta}_t \mathbf{I}), \tag{2}$$

where the mean $\tilde{\boldsymbol{\mu}}_\theta(\mathbf{x}_t, \mathbf{y}, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{y}, t) \right)$ and variance $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t$ are derived by reparameterization trick [2]; and $\alpha = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. The noise estimation network $\boldsymbol{\epsilon}_\theta$, as Fig. 2 (main paper), employs the UNet structure, which is the mainstream choice of DMs [7, 1].

Training Strategy. To train the noise estimation network $\boldsymbol{\epsilon}_\theta$, we employ the training objective following previous methods [8, 3]. It is defined as follows:

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{y}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), t} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \mathbf{y}, t \right) \right\| \right]. \tag{3}$$

Inference Strategy. We start from a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, *i.e.*, $\mathbf{x}_T = \boldsymbol{\epsilon}$. Then, we execute the reverse process (Eq. (2)) T times. Finally, we can obtain the HR image $\mathbf{x}$ ($\mathbf{x}_0$).

2 Inference Time Comparison

Method	Params (M)	OPs (G)	Simulated Time (s)
SR3 [8]	55.41	176.41	55.37
BI-DiffSR	4.58	36.67	13.00

Table 1: Comparison of simulated running time of one step inference on the output size is $3 \times 256 \times 256$.

Inference Frame	OPs (G)	Time (ms)
Caffe (FP)	1	313.85
daBNN (BI)	1	354.56

Table 2: The relationship between OPs and running time according to daBNN [10].

General GPU inference libraries do not support binarized modules, requiring specific hardware. Therefore, we are currently unable to measure running times in practice. Here, we refer to previous methods [5] and use the inference frame, daBNN [10], to estimate the running time. According to daBNN, on the ARM64 CPU, there is a correlation between OPs and running speed, detailed in Tab. 2. The Caffe architecture is used for full-precision (FP) modules, while daBNN is used for binarized (BI) modules. We present simulated running times in Tab. 1. We can find that our method has a much shorter running time compared to the full-precision method.

---

*Corresponding authors: Haotong Qin, qinhaotong@gmail.com; Yulun Zhang, yulun100@gmail.com

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3 More Visualizations and Analyses

In this section, we provide more visualizations and analyses corresponding to the contents in Figs. 4, 6, and 7 of the main paper. **First**, we offer more visualizations of changes in convolution modules. **Next**, we display the distributions of different (input and output) features within the skip connections. **Finally**, we present the weight distributions for TaR and TaA.
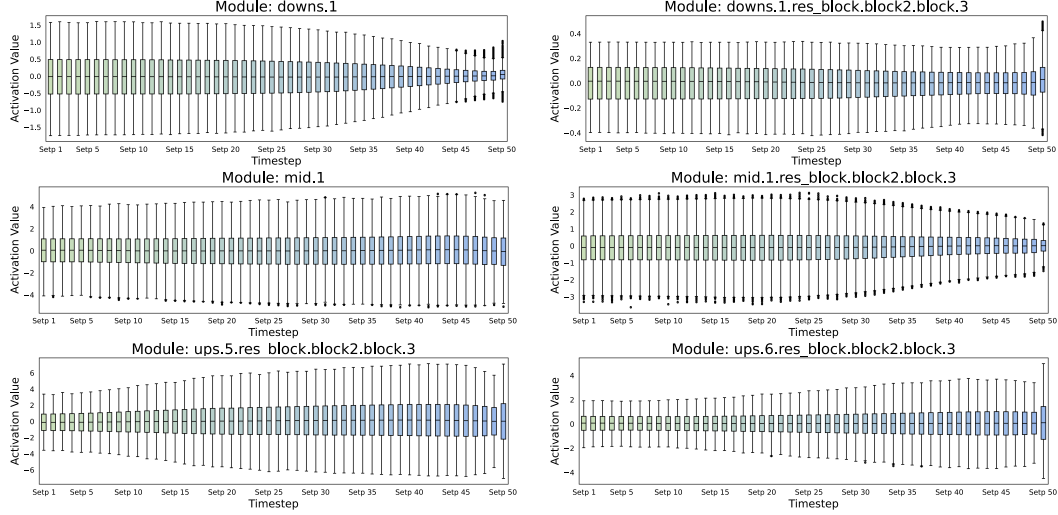
Figure 1: More visualizations of variations in the activation distribution across 50 timesteps (DDIM sampler). We select multiple convolution modules at different positions. It can be observed that in many modules, the activation distribution difference is significant across different times.

3.1 Activation Changes across Timestep.

We provide more instances of the output activation distributions from convolution modules located in the encoder (downs), bottleneck (mid), and decoder (ups) in Fig. 1. It is evident that in many convolutions, there are significant differences in activation distributions across different timesteps. Additionally, the trends of these changes vary across different modules, manifesting in multiple ways, such as initially broad then narrow, or initially narrow then broad. Meanwhile, we also find that changes within adjacent timesteps tend to be relatively stable.

Furthermore, we also observe that in some modules, the ranges of activation values are similar. This may indicate that at these specific modules, the network is required to extract more consistent feature information. Overall, the network activation distributions vary across timestep. This variability increases the difficulty of learning representations for binarized models, thus affecting model performance. Consequently, we propose the timestep-aware redistribution (TaR) and timestep-aware activation function (TaA) to enhance the binarized modules.

3.2 Activation Distribution in the Skip Connection.

We visualize the distribution of different activation values in the skip connection in Fig. 2, including: two input features (Input 1(2), $\mathbf{x}_1$, $\mathbf{x}_2$), the result of addition fusion (Sum, $\mathbf{x}_1 + \mathbf{x}_2$), and the shuffled features obtained through channel shuffle (Fusion 1(2), $\mathbf{x}_1^{sh}$, $\mathbf{x}_2^{sh}$). We find that in the shallow decoder layers (near the bottleneck), since the modules connected by the skip connection are close, the range of feature distributions is similar. Direct addition can achieve a certain degree of feature fusion. As shown in the first row, the distribution of the Sum shows some differences from the two input features.

However, as the number of decoder modules increases, the difference in value distribution between the two inputs of the skip connection significantly increases. The distribution resulting from direct addition closely resembles that of the wider distribution input. This is what we refer to as a "**mask**", where the activation distribution of a smaller range is overshadowed. This can lead to substantial information loss, hindering performance. Our experiments in Tab. 1b demonstrate this effect (Model Sum and Model Split). Conversely, after channel shuffle, the distribution curve of the fused features

2
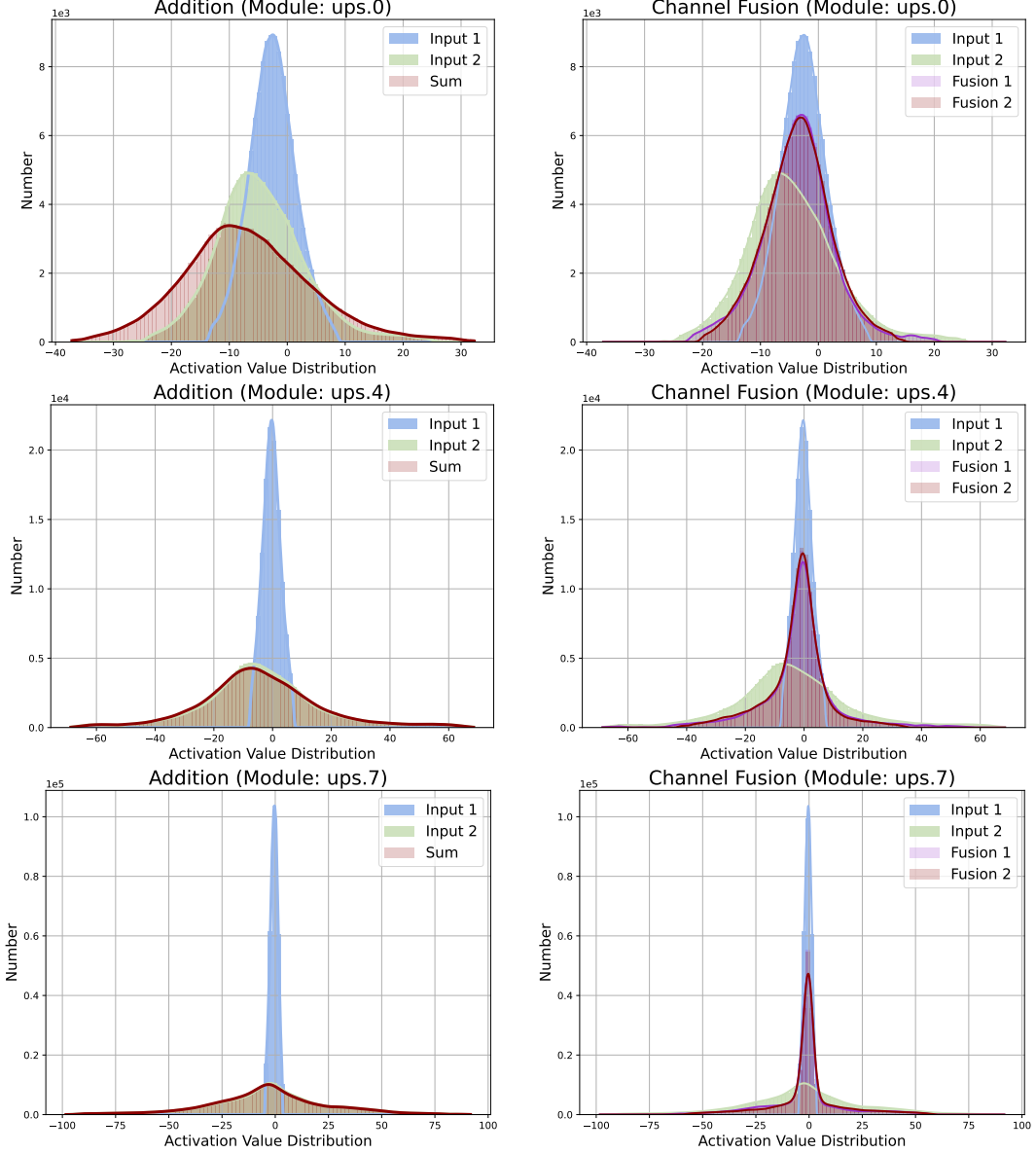
Figure 2: More activation distribution in the skip connection. In each row, we provide multiple features from the same location: input features (Input 1(2), $\mathbf{x}_1$, $\mathbf{x}_2$), the addition result (Sum, $\mathbf{x}_1 + \mathbf{x}_2$), and the shuffled features from channel shuffle (Fusion 1(2), $\mathbf{x}_1^{sh}$, $\mathbf{x}_2^{sh}$). We observe that as the number of decoder layers increases, the differences between input features increase, which heightens the complexity of fusion. Meanwhile, the proposed channel shuffle can effectively fuse them.

differs from that of either input feature. It represents their combined result, proving the effectiveness of our proposed method. This is consistent with the results of ablation experiments.

3.3 Weight Statistics of TaR and TaA.

We present more statistical results of the weights for TaR and TaA. TaR includes five learnable biases $\mathbf{b}^i \in \mathbb{R}^C$ ($i \in \{1, 2, \ldots, 5\}$), while TaA comprises five RPReLU functions. Each RPReLU contains three learnable weights. Given an input $\mathbf{x}_{in} \in \mathbb{R}^{H \times W \times C}$, it is defined as follows:

$$\text{RPReLU}(x_{out}) = \begin{cases} x_{out} - \gamma + \zeta, & x_{out} > \gamma \\ \beta \cdot (x_{out} - \gamma) + \zeta, & x_{out} \leq \gamma \end{cases}, \quad \forall x_{out} \in \mathbf{x}_{out}, \tag{4}$$

where $\mathbf{x}_{out} \in \mathbb{R}^{H \times W}$ is the i-th channel feature of the output feature $\mathbf{x}_{out} \in \mathbb{R}^{H \times W \times C}$, $i \in \{1, 2, \ldots, C\}$; $\gamma, \zeta, \beta \in \mathbb{R}^{\hat{C}}$ are learnable parameters for distribution moving.

3
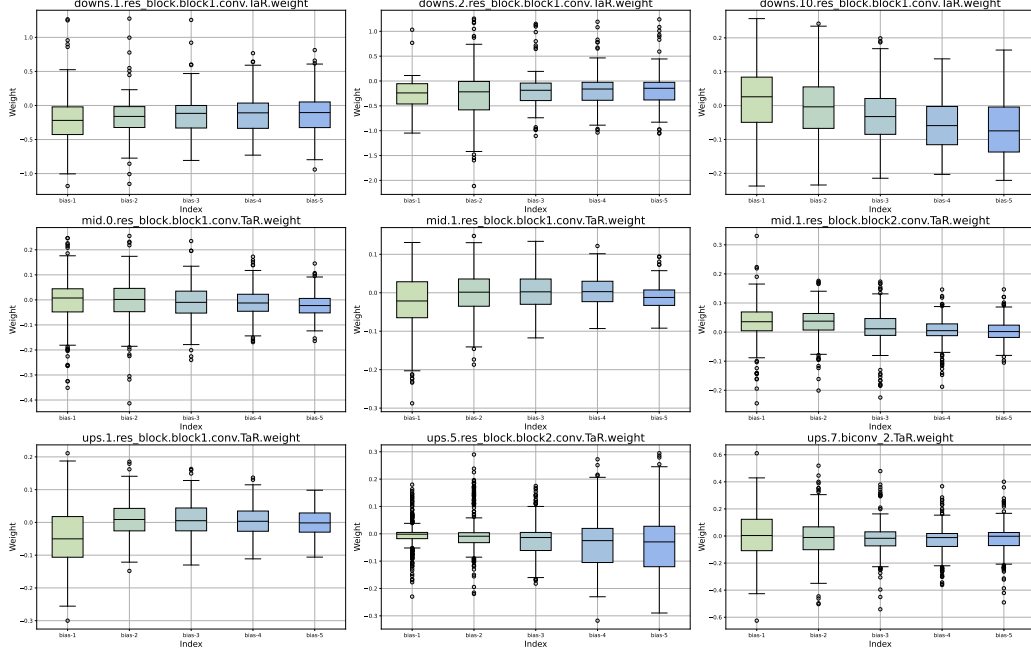
Figure 3: Weights of learnable biases $\mathbf{b}^i$ ($i \in \{1, \ldots, 5\}$) in TaR. Each subplot represents one module.
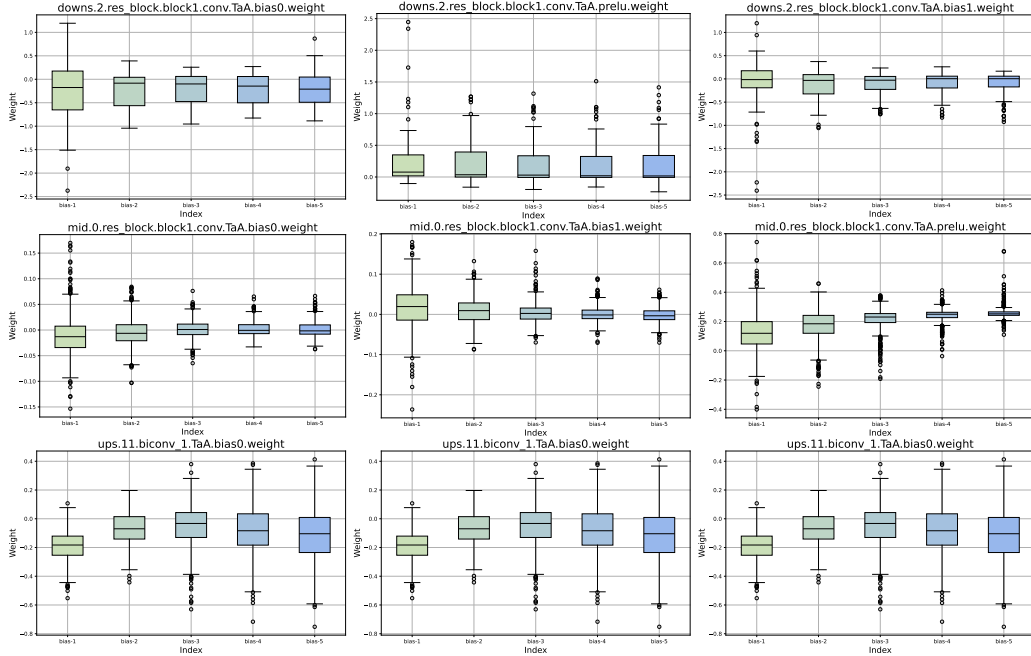
Figure 4: Weights of learnable biases in TaA. Each row denotes three weights of the same RPReLU.

In Fig. 3, each subplot represents the weight distribution of the five biases of one TaR. We can observe that, there are substantial differences among the biases, indicating that TaR effectively adapts to different activation distributions across various timesteps. In Fig. 4, each row represents the distribution of the three weights in one TaA. Variations in weight distribution can also be observed. Moreover, in some modules (*e.g.*, the first row), we find that the greatest differences occur in the first bias (acting on the input), with subsequent differences being smaller. This might be because the first weight adjustment reduces the variance in distribution.

4

4 More Visual Comparisons

We provide more visual comparisons on Urban100 and Manga109, in Figs. 5 and 6. Compared to other binarization methods, our approach can restore more accurate and perceptually pleasing results. In contrast, the comparison methods exhibit artifacts or blurriness, and some are even inapplicable (*e.g.*, IRNet [5]) in some challenge cases. Additionally, compared to the full-precision (FP) model SR3 [8], our method exhibits similar restoration results. These results serve as the supplement to the visual comparisons in the main paper and demonstrate the effectiveness of our method.

5 Explanations for Checklist

5.1 Limitations

In this paper, we implemented a binarized diffusion model for image SR. Considering multi-timestep in diffusion, we designed TaR and TaA to adapt to different timesteps. While these improve performance, they also introduced additional parameters and increased training time. Furthermore, we observe that activation changes over time are not uniform across all modules, hence, the same selection strategy for all modules may not be the optimal choice. We will explore more efficient timestep-aware designs in the future work.

5.2 Broader Impacts

The application of image SR is extensive, playing a critical role in fields such as photography and medical imaging. The usage of DMs in image SR is also becoming increasingly widespread. Lightening diffusion models to facilitate their use on a broader range of platforms have practical value. Therefore, we believe our proposed method is timely and effective, benefiting both academia and industry. Meanwhile, we think that our method has no potential negative societal impact.

References

[1] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *TPAMI*, 2023. 1

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[3] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 2022. 1

[4] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020. 6, 7

[5] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *CVPR*, 2020. 1, 5, 6, 7

[6] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 6, 7

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[8] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *TPAMI*, 2022. 1, 5, 6, 7

[9] Bin Xia, Yulun Zhang, Yitong Wang, Yapeng Tian, Wenming Yang, Radu Timofte, and Luc Van Gool. Basic binary convolution unit for binarized image restoration network. In *ICLR*, 2022. 6, 7

[10] Jianhao Zhang, Yingwei Pan, Ting Yao, He Zhao, and Tao Mei. dabnn: A super fast inference framework for binary neural networks on arm devices. In *ACM MM*, 2019. 1

5
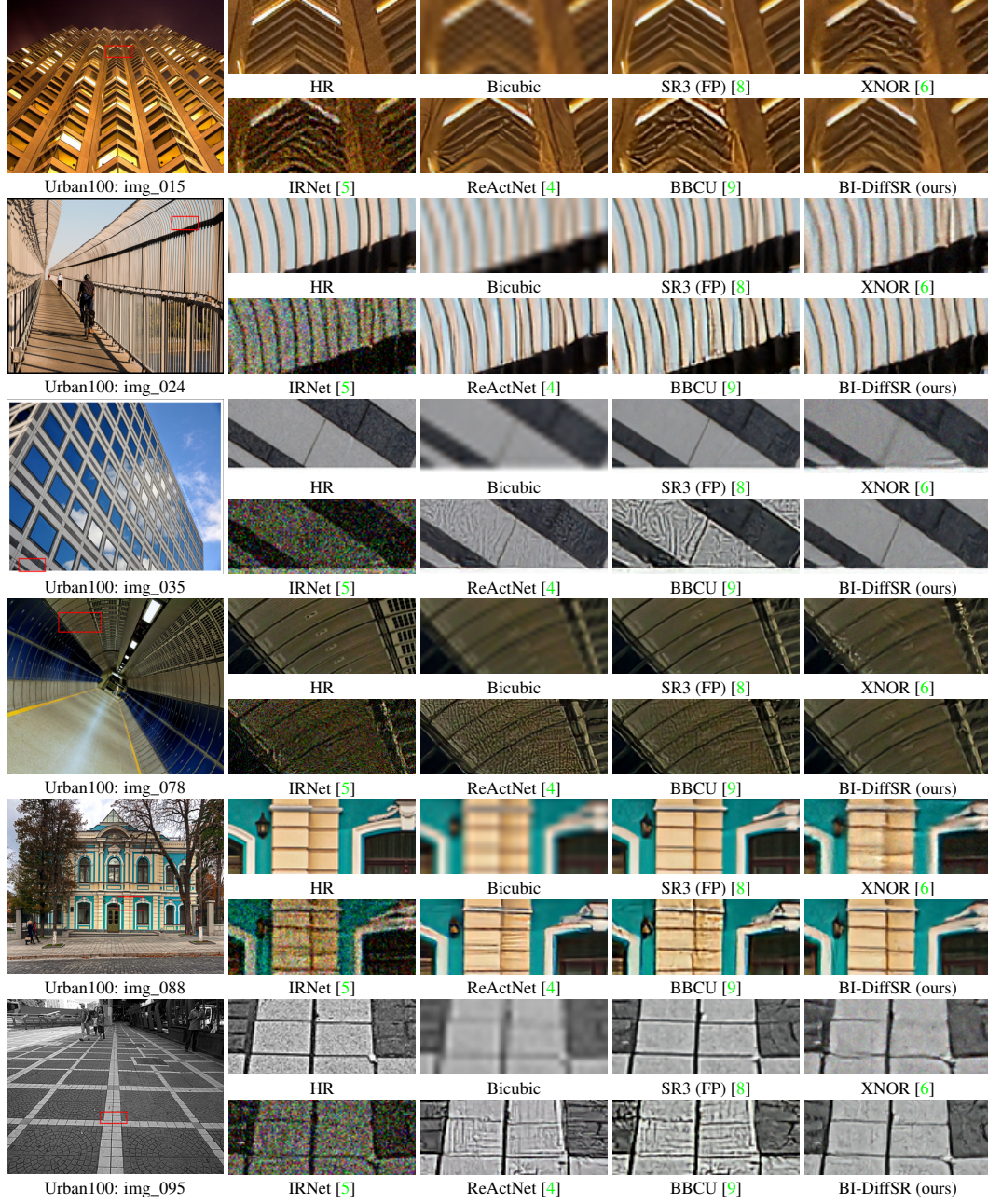
Figure 5: Visual comparison (×4) in some challenge cases.

6

Figure 6: Visual comparison (×4) in some challenge cases.

7