Appendix

A. Network Architecture

In this paper, we propose PatchScaler, an efficient patch-independent diffusion model for the SISR task. The overall architecture of PatchScaler is illustrated in Fig. 2 of the main paper. To alleviate the computational overhead for high-resolution images, our PatchScaler performs reconstruction in the latent space of a pre-trained variational autoencoder [41], following the previous diffusion models [7, 41, 48]. To achieve adaptive reconstruction of different regions of the image, we introduce a Global Restoration Module (GRM) and a Patch-DiT for PatchScaler. Here, we present the detailed architecture of them.

A.1. Global Restoration Module

To remove the degradations (e.g., noise or distortion artifacts) and capture long-range dependencies, we introduce a Global Restoration Module (GRM) based on a UNet-like architecture, as shown in Fig. 9(a). Specifically, GRM consists of a series of residual blocks and downsampling convolutions followed by a stack of residual blocks with upsampling convolutions and skip connections. Unlike traditional image restoration networks, our GRM has two output branches to generate the confidence map C and the coarse HR feature $\boldsymbol{y}_{HR}$, respectively. C predicts the distance between the coarse HR $\boldsymbol{y}_{HR}$ and ground truth $\boldsymbol{x}_{HR}$, and $Qmap$ is obtained by quantifing C.

The supervision of the confidence map C is:

$$L(\theta) := \|\boldsymbol{y}_{\text{HR}} - \boldsymbol{x}_{\text{HR}}\|_1 + \lambda(C\,\|\boldsymbol{y}_{\text{HR}} - \boldsymbol{x}_{\text{HR}}\|_2^2 - \eta \log(C)), \tag{8}$$

where λ and η are the weights of the losses, and $\log(C)$ represents the logarithm of C. When the regions of coarse HR are far from the ground truth, the term $C\,\|\boldsymbol{y}_{HR} - \boldsymbol{x}_{HR}\|_2^2$ is dominant, encouraging C to decrease. Conversely, the term $-\log(C)$ becomes dominant, encouraging C to approach 1. We emphasize that this constraint has been demonstrated to be effective in the low-level tasks [8, 32, 54]. In this way, the generated C can accurately reflect the reconstruction difficulty of the image.

A.2. Patch-DiT

Since most popular open-source diffusion models [35, 41, 42] produce inferior results in low-resolution patches [69], we propose a Patch-wise Diffusion Transformer, i.e., Patch-DiT, as the backbone of PatchScaler to reconstruct the fine textures from the coarse HR patches. The overall architecture of Patch-DiT is illustrated in Fig. 9(b). We build our Patch-DiT upon DiTs [37] and explore the following architectural modifications: (1) concatenating $\boldsymbol{y}_0$ with the noisy latent $\boldsymbol{y}_\tau$ as the input of Patch-DiT to generate content-consistent results; (2) adding a cross-attention layer in each base block to incorporate the texture prompt; (3) removing the class label embedding and learnable covariance.

Methods	Prompt	ManIQA	CLIPIQA	MUSIQ	Runtime (s)
SUPIR [61]	text	0.5141	0.8122	68.77	113.82
PatchScaler	texture	**0.5442**	**0.8213**	**70.97**	**3.41**

Table 8. Quantitative comparison of SUPIR (w/ text prompt) and PatchScaler (w/ texture prompt) on RealSet110 dataset. The best results are **highlighted**. Noted that the runtime is measured on the $\times$4 (512 $\rightarrow$2048) SR task using an NVIDIA Tesla A100 GPU.

To optimize the patch-level reconstruction process of Patch-DiT, we further propose the texture prompt. First, we introduce a texture classifier to learn high-dimensional semantic representations of texture feature patches offline and achieve texture retrieval for the target patches from RTM. The texture classifier is trained for 100K iterations based on Describable Textures Dataset (DTD) [9]. Second, to transfer the retrieved texture priors into Patch-DiT, we propose a similarity-aware texture encoder, as shown in Fig. 9(b). We copy the first half of the Patch-DiT and trim it as the similarity-aware texture encoder. The structure of the similarity-based texture encoder is much lighter and replaces time embeddings with similarity embeddings. As the encoder is time-step independent, it does not need to be recomputed at each iteration, significantly speeding up inference. To account for time, we introduce dimension-wise scaling parameters based on time embedding in each cross-attention layer.

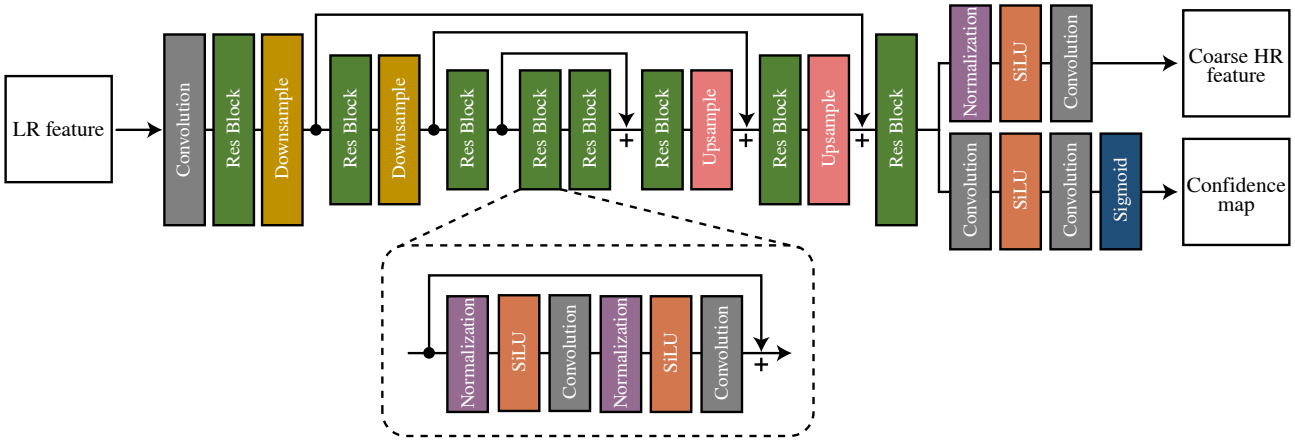
During the training, we optimize the reweighted variational lower bound at the patch level:

$$L(\theta) := \mathbb{E}_{\boldsymbol{x}_0,t,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - f_\theta\left(\boldsymbol{x}_t, t, \boldsymbol{y}_0, tp^k, s^k\right)\right\|_2^2\right], \tag{9}$$

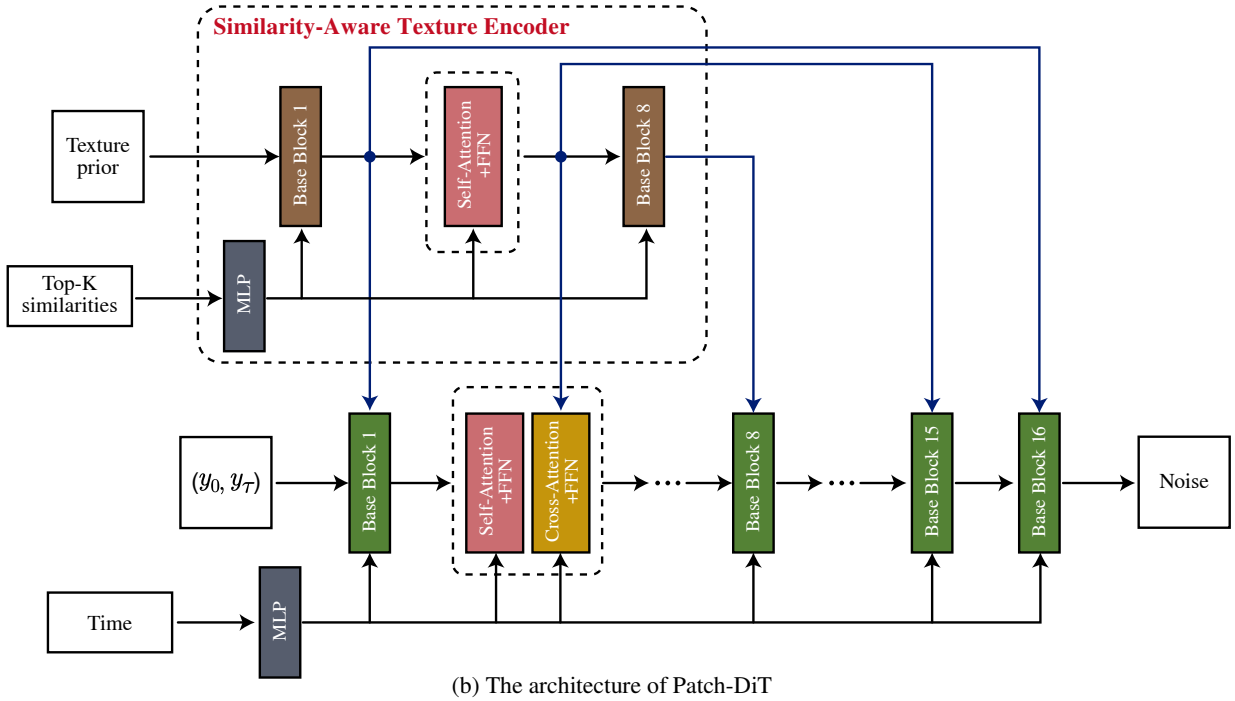
where $\boldsymbol{x}_t$ is sampled as $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\boldsymbol{\epsilon}$, $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{i=0}^{t} \alpha_i$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$, $\boldsymbol{x}_0$ is the patch from the ground truth $\boldsymbol{x}_{HR}$, $\boldsymbol{y}_0$ is the coarse HR patch, tp^k denote the retrieved texture priors, and s^k denote the corresponding top-K similarities.

B. Compare with MLLM-based SR Method

PatchScaler is designed to dynamically accelerate inference by analyzing reconstruction difficulty and handling different image regions discriminatively. In the experiments presented in the main paper, PatchScaler demonstrates superior performance over existing multi-step and one-step diffusion-based super-resolution (SR) methods in both quantitative and qualitative evaluations. Furthermore, we note the emergence of MLLM-based SR methods, such as SUPIR [61], which exhibit powerful generative capabilities through the introduction of dense captions. SUPIR leverages a 2.6-billion-parameter StableDiffusion-XL (SDXL) [38] as its generative prior and a 13-billion-parameter Multi-modal Large Language Model (MLLM) to generate detailed descriptive text.



(a) The architecture of Global Restoration Module (GRM)



(b) The architecture of Patch-DiT

Figure 9. Overview of the proposed PatchScaler. (a) The architecture of Global Restoration Module; (b) The architecture of Patch-DiT.

Although SUPIR has achieved outstanding performance, it often generates details that are not faithful to the LR images. We present four representative examples in Fig. 10, following the default hyperparameter settings of SUPIR. It can be observed that SUPIR produces wrinkles on the human face that do not match the original age attributes of the person. The structure of the landmark building is altered, and the content and meaning of the LOGO is also manipulated, significantly reducing the recognition of these images. In addition, SUPIR also generates unexpected animal fur in the illustration image. We analyze that this issue is due to excessive guidance from text prompts and the misalignment between images and text prompts in diffusion models. In contrast, PatchScaler, leveraging its texture prompt, produces results with higher visual quality and objective consistency. Overall, our method is more suitable for restoring the user's significant images that are faithful to the LR inputs.

We further perform a quantitative evaluation on the RealSet110 dataset, as shown in Tab. 8. Thanks to the proposed patch-independent diffusion pipeline that integrates Patch-adaptive Group Sampling (PGS) and texture prompt, our PatchScaler consistently outperforms SUPIR on all non-reference metrics. In addition, due to the large number of model parameters, SUPIR exhibits significantly slower inference speed, taking more than 30× longer than PatchScaler. This limits its applicability to a few creative scenarios that

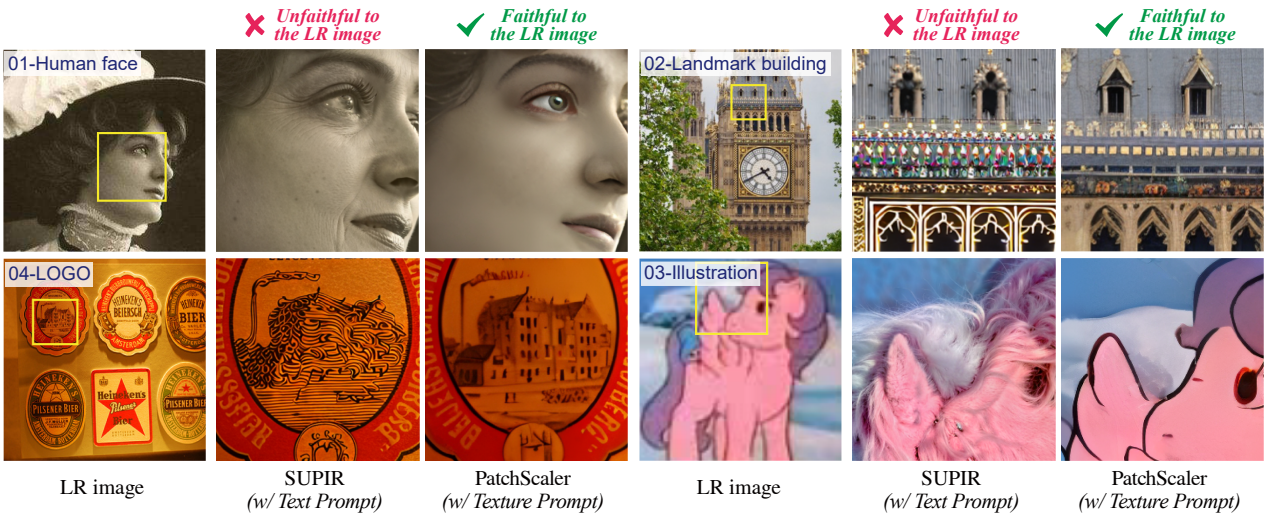


Figure 10. Comparison between SUPIR (w/ text prompt) and PatchScaler (w/ texture prompt).

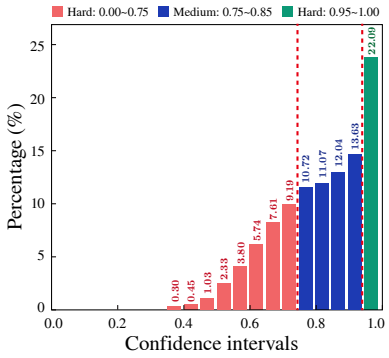


Figure 11. Distribution statistics of the number of patches of patches with different confidence intervals. Note that the statistics are based on the RealSet110 dataset.

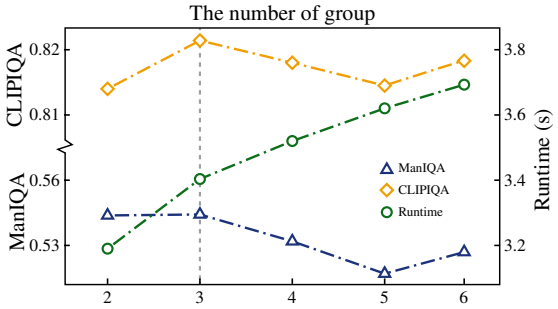


Figure 12. Performance of different numbers of groups.

are not sensitive to inference speed and consistency. Unlike SUPIR, our PatchScaler dynamically accelerates the inference process, significantly improving computational efficiency without compromising reconstruction consistency, which is of broad practical value in real-world applications.

C. More Quantitative Analysis

C.1. Distribution Visualization

In this work, we classify the patches into different groups based on the quantified confidence map $Qmap$. To analyze the distribution of patches within different confidence intervals, we count the number of patches in each interval on the RealSet110 dataset, as shown in Fig. 11. We conduct ablation studies under different confidence thresholds in Sec. 4.3.1 of the main paper, and PatchScaler achieves a trade-off between model performance and runtime with γ_1=0.95 and γ_2=0.75.

C.2. Divide Qmap into Different Number Groups

We provide an intuitive experiment analysis (metrics against runtime) of PatchScaler under different numbers of groups (ranging from 2 to 6) on the RealSet110 dataset in Fig. 12. Our method achieves a balance between quality and efficiency when dividing the Qmap into three groups. Increasing the number of groups not only fails to enhance performance, as it introduces more randomness into the model, but also increases inference complexity. Note that the runtimes are evaluated using an NVIDIA Tesla A100 GPU.

C.3. Runtime and Memory Usage

The GRM in PatchScaler is responsible for generating a coarse HR feature, while the Patch-DiT refines it at the patch level. As shown in Tab. 9, we report the runtime and memory usage of each component. Note that they are evaluated on the 512→2048 task and memory usage records the peak value during execution.

Modules	Autoencoder	GRM	Patch-DiT
Runtime (s)	1.04	0.01	2.36
Memory usage (G)	9.45	5.52	5.66

Table 9. Analysis of runtime and memory usage of each component of PatchScaler.

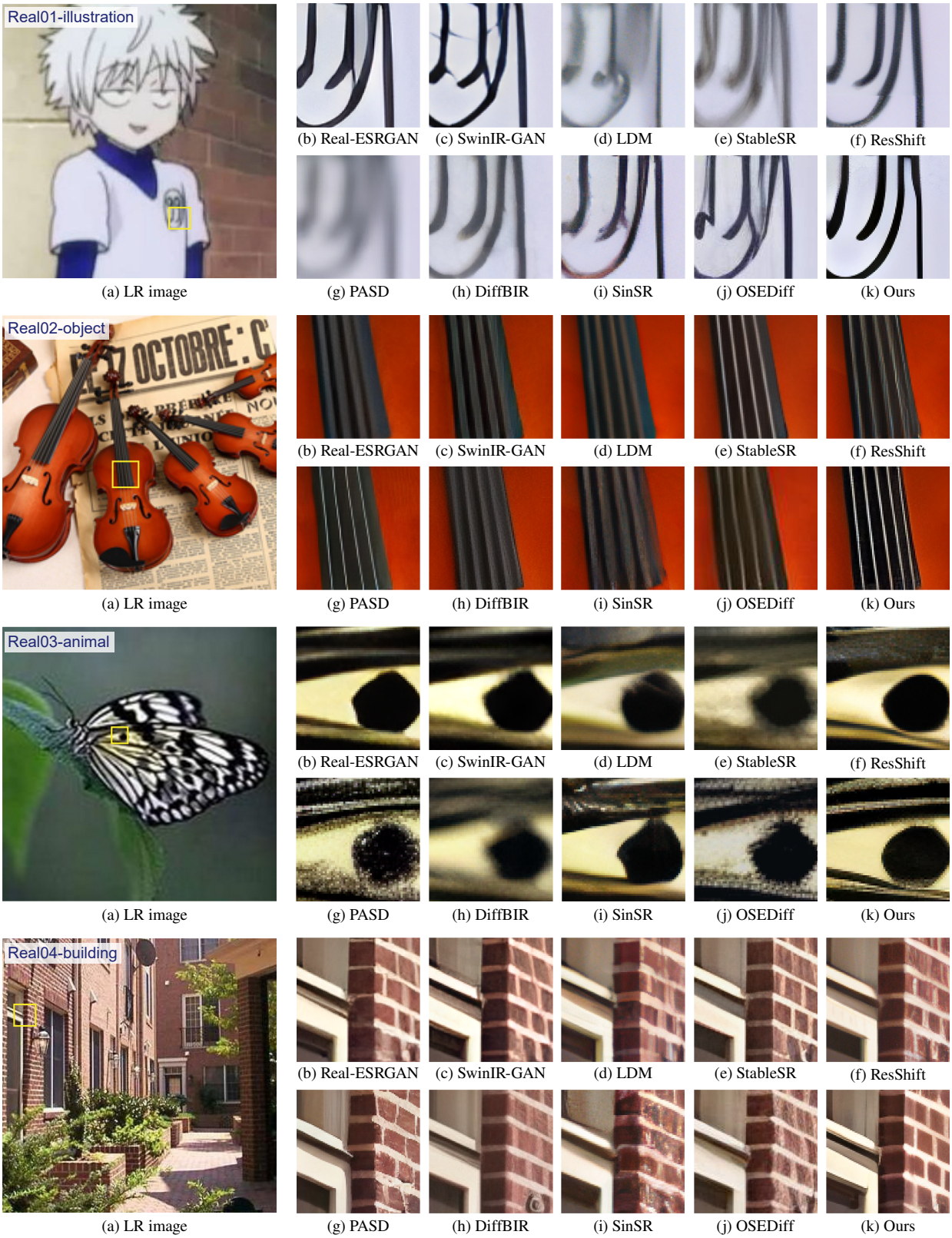
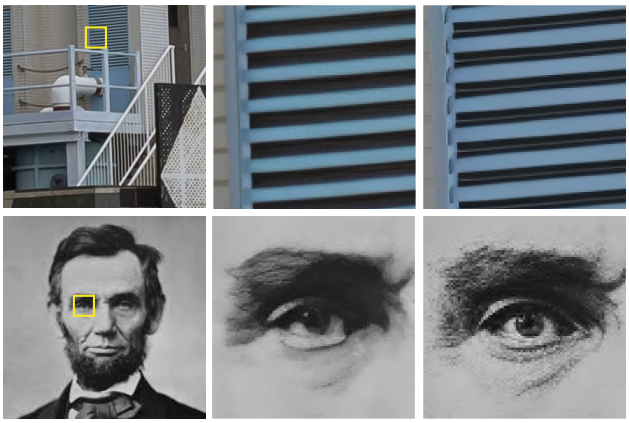


Figure 13. More visual comparison between PatchScaler and state-of-the-art SR methods on real-world low-resolution images.

Methods	RealSR			RealSet110			Runtime(s) (#Params)
	ManIQA	CLIPIQA	MUSIQ	ManIQA	CLIPIQA	MUSIQ	
S3Diff-s1*	0.4593	0.5835	67.82	0.4802	0.8115	70.95	15.92 (1327M)
AddSR-s1	0.5105	0.5479	**71.34**	0.5310	0.8096	**71.19**	37.65 (2511M)
TAD-SR-s1*	0.4589	0.5828	65.83	0.4466	0.6786	57.24	OOM (119M)
AdcSR-s1*	0.5175	0.5628	69.87	0.4742	0.7925	70.28	3.80 (456M)
InvSR-s1	0.4573	0.5670	68.55	0.4589	0.7931	70.03	4.96 (1290M)
Ours	**0.5225**	**0.5839**	69.50	**0.5442**	**0.8213**	70.97	**3.41** (875M)

Table 10. More comparisons with latest distillation-based (*) and one-step (s1) SR models. Runtime is measured for upscaling images to a resolution of 2048×2048. Existing one-step methods suffer from limited flexibility and inefficiency on high-resolution inputs, while our method achieves significant efficiency gains with competitive reconstruction quality.



LR image StableSR StableSR+PGS

Figure 14. Comparison between StableSR and StableSR+PGS.

Methods	StableSR [48]	OSEDiff [53]	Ours
GFLOPs	6412	2161	5190
#Param (M)	1409	1776	875

Table 11. Analysis of FLOPs and parameters with state-of-the-art SR methods.

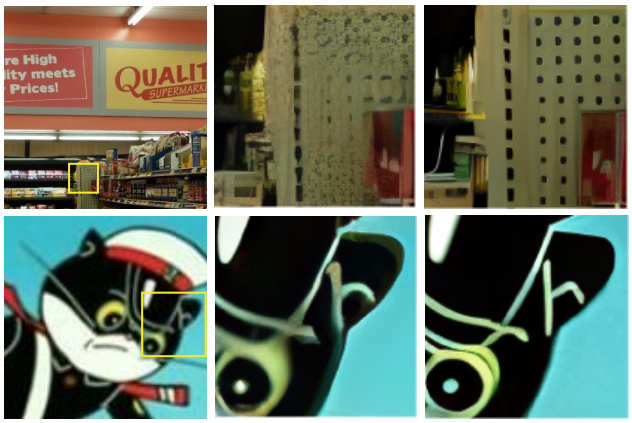
C.4. FLOPs and Parameters

To assess the efficiency of PatchScaler, we compare its runtime against other methods in Tab. 2 of the main paper. As shown in Tab. 11, we further report the FLOPs and parameters with two typical diffusion-based methods, as a reference. It can be seen that our method has significantly fewer parameters than the other two methods, and its FLOPs is second only to OSEDiff [53].

D. More Results

D.1. More Visual Comparisons with SOTA Methods

As shown in Fig. 13, we present more visual evaluation results of the proposed PatchScaler with state-of-the-art SR methods on real-world LR images. These results consistently demonstrate the superiority of PatchScaler in both artifact removal and texture detail enhancement.



LR image w/ text prompt w/ texture prompt

Figure 15. Visual comparisons of different prompts.

D.2. Comparison with More One-Step SR Methods

We compare PatchScaler with more one-step SR models (including distillation-based methods) in Tab. 10, demonstrating its efficiency and quality benefits from adaptive sampling and texture prompt.

D.3. Extension of PGS

To verify the extension of the proposed Patch-adaptive Group Sampling (PGS), we applied the proposed PGS and GRM to StableSR [48] in a trainable-free manner. We present more visual comparisons in Fig. 14. It can be seen that our method can be seamlessly applied to other baselines, yielding substantial improvements in reconstruction quality.

D.4. Text Prompt vs. Texture Prompt

As shown in Fig. 15, we provide more visual comparisons of different prompt. It can be seen that the model with texture prompt can generate clearer detailed information. This is due to the challenging misalignment [7] between the text content and image content in the SISR task, which often leads to degraded performance. On the other hand, the proposed texture prompt effectively assists the reconstruction process by providing high-quality texture references that are similar to the target patch.