

## A Supplementary Material

This supplementary material provides further details to the main paper. It covers the following aspects:

- **Section A.1: Experiment Details for Real-world Image Super-Resolution:** Including dataset descriptions, backbone model configurations, an in-depth discussion of the patch-wise KDS mechanism with ablation studies, interaction of KDS with various diffusion samplers (DDIM, DPM-Solver++), and additional comparative results on severe degradation datasets.
- **Section A.2: Experiment Details for Image Inpainting:** Covering the datasets used and the hyperparameter settings for the inpainting task.
- **Section A.3: Additional Visual Results:** Presenting more qualitative examples for both super-resolution and image inpainting tasks to further demonstrate the efficacy of KDS.

### A.1 Experiment details: Real-world Image Super Resolution

#### A.1.1 Datasets

Our real-world image super-resolution experiments utilized a synthesized dataset derived from DIV2K [55], alongside two real-world datasets: RealSR [56] and DRealSR [57]. For the DIV2K-based synthesized data, we used the same dataset provided by StableSR [11], which consists of 3,000 randomly cropped patches (resolution:  $512 \times 512$  each) from the DIV2K validation set [55]. Subsequently, we generated corresponding low-resolution (LR) images (resolution:  $128 \times 128$ ) using the degradation model adopted in Real-ESRGAN [65]. For the RealSR [56] and DRealSR [57] datasets, we adhered to the protocol from [11] to center-crop the provided LR images to  $128 \times 128$ .

#### A.1.2 Backbone Model Configurations

To assess the effectiveness of our proposed KDS method, we applied it to three established backbone models: LDM-SR [24], DiffBIR [4], and SeeSR [33]. For LDM-SR, the LR image was directly employed as a conditional input, concatenated with the LDM’s primary input. For DiffBIR [4] and SeeSR [33], we adopted the hyperparameter settings recommended on their official GitHub pages. This included their specified parameters for text prompting, classifier-free guidance, and the use of their official pre-trained model checkpoints.

#### A.1.3 Patch-wise KDS Mechanism: Configuration and Discussion

As introduced in Section 3.1, we employ a patch-wise mechanism to facilitate mode-seeking within the high-dimensional latent space. The motivation is that the accuracy of the mode-seeking process is intrinsically linked to both the number of samples available for kernel density estimation (KDE) and the dimensionality of these samples. In our case, direct mode-seeking on the entire  $64 \times 64 \times 8$  latent vector  $z$  (which encodes features for the  $512 \times 512 \times 3$  image space) is impractical with a practical number of particles (e.g., 10-20).

**Patch Size Configuration:** To effectively implement our patch-wise strategy, the patch size is chosen to balance accurate mode-seeking with computational efficiency. Accurate mode-seeking across the entire latent space would necessitate thousands of samples, which is impractical for real-world applications. Therefore, to achieve robust estimation with minimal samples, we set the patch size to  $1 \times 1$ . This processes each spatial location  $(h, w)$  in the latent map as an individual 8-dimensional vector (i.e., batches of  $1 \times 1 \times 8$  vectors). This strategic choice significantly reduces the dimensionality for each kernel density estimation (KDE), enabling accurate estimation with limited samples while leveraging the rich, learned features of the 8 channels. Consequently, this facilitates more effective mode-seeking with a limited particle count. Ablation studies (Table 5) demonstrate that with a restricted particle count ( $N = 10$ ), larger patch sizes lead to inaccurate mode-seeking and sub-optimal performance.

**Steering Strength  $\delta_t$  Configuration:** Another key hyperparameter in KDS is the steering strength, denoted as  $\delta_t$ . During the diffusion process, the sampling procedure exhibits varying sensitivity to guidance. Specifically, at later stages of the sampling process (i.e., smaller  $t$  values, approaching

Table 5: Ablation Study on Patch-size ( $N = 10$ )

	PSNR	SSIM	LPIPS ↓	FID
DDIM	22.05	0.531	0.307	20.89
+ KDS (patch-size=1)	22.52	0.555	0.289	20.33
+ KDS (patch-size=4)	22.35	0.547	0.296	20.77
+ KDS (patch-size=16)	22.17	0.538	0.304	20.86

the data), the model can be more sensitive. To ensure stability and effective guidance, we define  $\delta_t$  conditionally based on the timestep  $t$ . Assuming  $T$  is the total number of diffusion steps, we set:

$$\delta_t = \begin{cases} 0 & \text{if } t/T < 0.3 \\ 0.3 & \text{if } t/T \geq 0.3 \end{cases} \quad (10)$$

Note that, this hyperparameter setting is fixed for all the experiments across different applications and backbones.

#### 496 A.1.4 Integration with Standard Diffusion Samplers

497 As discussed in the main paper, KDS is a flexible, plug and play approach that can be applied to  
 498 all existing samplers. In this section, we will detailed introduce how to apply KDS to DDIM, and  
 499 DPM-solver and how to interatve with Classifier-Free Guidance.

500 **Interaction with DDIM:** To better illustrate the plug-and-play nature of KDS, we provide pseu-  
 501 docode for three scenarios:

- 502 1. **Algorithm 1:** Standard DDIM sampling.
- 503 2. **Algorithm 2:** DDIM integrated with KDS.
- 504 3. **Algorithm 3:** DDIM combined with Classifier-Free Guidance (CFG) and KDS.

505 As demonstrated in the pseudocode, KDS functions as a straightforward plug-in module (Line 5 - 15  
 506 in Algorithm 2). It enhances the predicted ensemble  $\hat{\mathbf{z}}_{0|t}^{\text{pred}}$  and maintain the rest sampling design of  
 507 the base sampler.

508 **Interaction with Classifier-Free Guidance (CFG):** CFG is frequently used in conditional diffusion  
 509 models for SR (like DiffBIR [4], SeeSR [33]) to further improve perceptual quality. CFG is applied  
 510 by adjusting the noise prediction, commonly via the extrapolation formula

$$\tilde{\epsilon}(\mathbf{z}_t, t, \mathbf{c}) = \epsilon(\mathbf{z}_t, t, \emptyset) + w(\epsilon(\mathbf{z}_t, t, \mathbf{c}) - \epsilon(\mathbf{z}_t, t, \emptyset)), \quad (11)$$

511 where  $w$  is the guidance scale. While higher  $w$  can enhance perception, it sometimes introduces  
 512 artifacts. We investigated how KDS interacts with CFG by varying  $w$  in the DiffBIR model using  
 513 DDIM sampling on the DrealSR dataset. As shown in Table 6, KDS consistently boosts performance  
 514 across different CFG strengths ( $w = 1, 2, 4$ ). For each value of  $w$ , adding KDS leads to substantial  
 515 improvements in PSNR and SSIM, along with generally better perceptual metrics (LPIPS, NIMA,  
 516 CLIPQA). This suggests that KDS provides benefits complementary to CFG, enhancing fidelity  
 517 without hindering the perceptual adjustments offered by CFG.

Table 6: Performance of DDIM with Varying CFG Weights  $w$  on DrealSR Dataset.

Method	PSNR	SSIM	LPIPS (↓)	DISTS(↓)
DDIM ( $w = 1$ )	25.11	0.576	0.492	0.298
+ KDS ( $N = 10$ )	<b>26.94</b>	<b>0.677</b>	<b>0.427</b>	<b>0.283</b>
DDIM ( $w = 2$ )	24.75	0.569	0.486	0.293
+ KDS ( $N = 10$ )	<b>26.60</b>	<b>0.667</b>	<b>0.428</b>	<b>0.278</b>
DDIM ( $w = 4$ )	23.99	0.551	0.491	0.293
+ KDS ( $N = 10$ )	<b>25.63</b>	<b>0.645</b>	<b>0.422</b>	<b>0.276</b>
DDIM ( $w = 6$ )	23.27	0.534	0.497	0.296
+ KDS ( $N = 10$ )	<b>25.04</b>	<b>0.629</b>	<b>0.440</b>	<b>0.282</b>

---

#### Algorithm 1 Standard DDIM Sampling

---

**Require:** Model  $\epsilon_\theta$ , condition:  $\mathbf{c}$ , Schedule  $\bar{\alpha}_t$

- 1:  $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$  ▷ Init noise
  - 2: **for**  $t = T, \dots, 1$  **do**
  - 3:    $\epsilon_{\theta,t} \leftarrow \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c})$  ▷ Predict noise
  - 4:    $\hat{\mathbf{z}}_{0|t} \leftarrow (\mathbf{z}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_{\theta,t}) / \sqrt{\bar{\alpha}_t}$  ▷ Predict  $\mathbf{z}_0$
  - 5:    $\epsilon'_t \leftarrow (\mathbf{z}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{z}}_{0|t}) / \sqrt{1 - \bar{\alpha}_t}$  ▷ Update direction based on  $\hat{\mathbf{z}}_{0|t}$
  - 6:    $\mathbf{z}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \hat{\mathbf{z}}_{0|t} + \sqrt{1 - \bar{\alpha}_{t-1}} \epsilon'_t$  ▷ DDIM step
  - 7: **end for**
  - 8: **return**  $\mathbf{z}_0$
-

---

**Algorithm 2** DDIM + KDS

---

**Require:** Model  $\epsilon_\theta$ , Schedule  $\bar{\alpha}_t$ , condition:  $\mathbf{c}$ , Number of particles:  $N$ , bandwidth:  $h$ , steering strength:  $\delta_t$ , PatchSize

- 1:  $\mathbf{Z}_T \sim \mathcal{N}(0, \mathbf{I})$  (ensemble of  $N$  samples) ▷ Init noise ensemble
- 2: **for**  $t = T, \dots, 1$  **do**
- 3:    $\mathbf{E}_{\theta,t} \leftarrow \epsilon_\theta(\mathbf{Z}_t, t, \mathbf{c})$  ▷ Predict ensemble noise
- 4:    $\hat{\mathbf{Z}}_{0|t}^{\text{pred}} \leftarrow (\mathbf{Z}_t - \sqrt{1 - \bar{\alpha}_t} \mathbf{E}_{\theta,t}) / \sqrt{\bar{\alpha}_t}$  ▷ Predict  $\mathbf{x}_0$  ensemble
- 5:    $\mathbf{Patches} \leftarrow \text{Patchify}(\hat{\mathbf{Z}}_{0|t}^{\text{pred}})$  ▷ Extract all non-overlapped patches:  $\mathbf{Patches}[k, loc]$ .
- 6:   **for** each patch location  $j$  **do** ▷ This loop over patch locations, can be executed **in parallel**.
- 7:      $\mathbf{P}_j \leftarrow \mathbf{Patches}[:, j]$  ▷ Ensemble of  $N$  original patches at location  $j$ .
- 8:     **for**  $i = 1, \dots, N$  **do** ▷ For particle  $i$ 's patch at location  $j$ , can be computed **in parallel**.
- 9:        $\mathbf{p}_j^{(i)} \leftarrow \mathbf{P}_j[i]$  ▷ Patch from particle  $i$  at location  $j$ .
- 10:        $\mathbf{m}(\mathbf{p}_j^{(i)}) \leftarrow \frac{\sum_{k=1}^N G\left(\frac{\|\mathbf{p}_j^{(i)} - \mathbf{p}_j^{(k)}\|^2}{h^2}\right) \mathbf{p}_j^{(k)}}{\sum_{k=1}^N G\left(\frac{\|\mathbf{p}_j^{(i)} - \mathbf{p}_j^{(k)}\|^2}{h^2}\right)} - \mathbf{p}_j^{(i)}$  ▷ Mean shift vector (Eq. 7).
- 11:        $\hat{\mathbf{p}}_j^{(i), \text{KDS}} \leftarrow \mathbf{p}_j^{(i)} + \delta_t \mathbf{m}(\mathbf{p}_j^{(i)})$  ▷ Apply steering (Eq. 8)
- 12:        $\mathbf{Patches}[i, j] \leftarrow \hat{\mathbf{p}}_j^{(i), \text{KDS}}$  ▷ Update the patch set with the guided patch.
- 13:   **end for**
- 14: **end for**
- 15:    $\hat{\mathbf{Z}}_{0|t}^{\text{KDS}} \leftarrow \text{Unpatchify}(\mathbf{Patches})$  ▷ Reconstruct guided latent prediction.
- 16:    $\mathbf{E}'_t \leftarrow (\mathbf{Z}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{Z}}_{0|t}^{\text{KDS}}) / \sqrt{1 - \bar{\alpha}_t}$  ▷ Update direction
- 17:    $\mathbf{Z}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \hat{\mathbf{Z}}_{0|t}^{\text{KDS}} + \sqrt{1 - \bar{\alpha}_{t-1}} \mathbf{E}'_t$  ▷ DDIM step
- 18: **end for**
- 19: **return**  $\hat{\mathbf{Z}}_0^{\text{KDS}}$  ▷ Return KDS-guided result

---

---

**Algorithm 3** DDIM + CFG + KDS

---

**Require:** Model  $\epsilon_\theta$ , Schedule  $\bar{\alpha}_t$ , condition:  $\mathbf{c}$ , Number of particles:  $N$ , bandwidth:  $h$ , steering strength:  $\delta_t$ , CFG strength:  $w$ , PatchSize

- 1:  $\mathbf{Z}_T \sim \mathcal{N}(0, \mathbf{I})$  (ensemble of  $N$  samples) ▷ Init noise ensemble
- 2: **for**  $t = T, \dots, 1$  **do**
- 3:    $\mathbf{E}_{\theta,t} \leftarrow \epsilon_\theta(\mathbf{Z}_t, t, \emptyset) + w(\epsilon_\theta(\mathbf{Z}_t, t, \mathbf{c}) - \epsilon_\theta(\mathbf{Z}_t, t, \emptyset))$  ▷ Predict ensemble noise with CFG
- 4:    $\hat{\mathbf{Z}}_{0|t}^{\text{pred}} \leftarrow (\mathbf{Z}_t - \sqrt{1 - \bar{\alpha}_t} \mathbf{E}_{\theta,t}) / \sqrt{\bar{\alpha}_t}$  ▷ Predict  $\mathbf{x}_0$  ensemble
- 5:    $\hat{\mathbf{Z}}_{0|t}^{\text{KDS}} \leftarrow \text{Patch-wise KDS}(\hat{\mathbf{Z}}_{0|t}^{\text{pred}})$  ▷ Same as Step 5-15 in Algorithm 2
- 6:    $\mathbf{E}'_t \leftarrow (\mathbf{Z}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{Z}}_{0|t}^{\text{KDS}}) / \sqrt{1 - \bar{\alpha}_t}$  ▷ Update direction based on KDS-guided  $\mathbf{x}_0$
- 7:    $\mathbf{Z}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \hat{\mathbf{Z}}_{0|t}^{\text{KDS}} + \sqrt{1 - \bar{\alpha}_{t-1}} \mathbf{E}'_t$  ▷ DDIM step with KDS-guided  $\mathbf{x}_0$
- 8: **end for**
- 9: **return**  $\hat{\mathbf{Z}}_0^{\text{KDS}}$  ▷ Return KDS-guided result

---



---

**Algorithm 4** DPM-Solver++.

**Require:** initial value  $Z_T$ , time steps  $\{t_i\}_{i=0}^M$  and  $\{s_i\}_{i=1}^M$ , data prediction model  $Z_\theta$ .  
1:  $Z_T \sim \mathcal{N}(0, \mathbf{I})$  (ensemble of  $N$  samples)  $\triangleright$  Init noise ensemble  
2:  $\tilde{Z}_{t_0} \leftarrow Z_T$ .  
3: **for**  $i \leftarrow 1$  to  $M$  **do**  
4:  $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$   
5:  $r_i \leftarrow \frac{\lambda_{s_i} - \lambda_{t_{i-1}}}{h_i}$   
6:  $\hat{Z}_{0|t}^{\text{pred}} \leftarrow Z_\theta(\tilde{Z}_{t_{i-1}}, t_{i-1})$   
7:  $U_i \leftarrow \frac{\sigma_{s_i}}{\sigma_{t_{i-1}}} \tilde{Z}_{t_{i-1}} - \alpha_{s_i} (e^{-r_i h_i} - 1) \hat{Z}_{0|t}^{\text{pred}}$   
8:  $\hat{U}_{0|s}^{\text{pred}} \leftarrow Z_\theta(U_i, s_i)$   
9:  $D_i \leftarrow (1 - \frac{1}{2r_i}) \hat{Z}_{0|t}^{\text{pred}} + \frac{1}{2r_i} \hat{U}_{0|s}^{\text{pred}}$   
10:  $\tilde{Z}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{Z}_{t_{i-1}} - \alpha_{t_i} (e^{-h_i} - 1) D_i$   
11: **end for**  
12: **return**  $\tilde{Z}_{t_M}$

---

---

**Algorithm 5** DPM-Solver++ with KDS.

**Require:** initial value  $Z_T$ , time steps  $\{t_i\}_{i=0}^M$  and  $\{s_i\}_{i=1}^M$ , data prediction model  $Z_\theta$ .  
1:  $Z_T \sim \mathcal{N}(0, \mathbf{I})$  (ensemble of  $N$  samples)  $\triangleright$  Init noise ensemble  
2:  $\tilde{Z}_{t_0} \leftarrow Z_T$ .  
3: **for**  $i \leftarrow 1$  to  $M$  **do**  
4:  $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$   
5:  $r_i \leftarrow \frac{\lambda_{s_i} - \lambda_{t_{i-1}}}{h_i}$   
6:  $\hat{Z}_{0|t}^{\text{pred}} \leftarrow Z_\theta(\tilde{Z}_{t_{i-1}}, t_{i-1})$   
7:  $\hat{Z}_{0|t}^{\text{KDS}} \leftarrow \text{Patch-wise KDS}(\hat{Z}_{0|t}^{\text{pred}})$   $\triangleright$  Same as Step 5-15 in Algorithm 2  
8:  $U_i \leftarrow \frac{\sigma_{s_i}}{\sigma_{t_{i-1}}} \tilde{Z}_{t_{i-1}} - \alpha_{s_i} (e^{-r_i h_i} - 1) \hat{Z}_{0|t}^{\text{KDS}}$   
9:  $\hat{U}_{0|s}^{\text{pred}} \leftarrow Z_\theta(U_i, s_i)$   
10:  $\hat{U}_{0|s}^{\text{KDS}} \leftarrow \text{Patch-wise KDS}(\hat{U}_{0|s}^{\text{pred}})$   $\triangleright$  Same as Step 5-15 in Algorithm 2  
11:  $D_i \leftarrow (1 - \frac{1}{2r_i}) \hat{Z}_{0|t}^{\text{KDS}} + \frac{1}{2r_i} \hat{U}_{0|s}^{\text{KDS}}$   
12:  $\tilde{Z}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}} \tilde{Z}_{t_{i-1}} - \alpha_{t_i} (e^{-h_i} - 1) D_i$   
13: **end for**  
14: **return**  $\tilde{Z}_{t_M}$

---

### A.1.5 Additional Comparisons on Various Degradations and Samplers

In this subsection, we further evaluate KDS’s performance on several additional real-world SR degradations and its effectiveness as a plug-in module for DPM-Solver++ [66]. We introduce a novel dataset, DF2K, generated by synthesizing 3,000 randomly degraded image pairs from the original DF2K dataset. While adopting the Real-ESRGAN pipeline, we employed hyperparameters that introduce more significant blur, noise, and JPEG artifacts, making it a more challenging benchmark compared to standard degradation levels, such as those in DIV2K. As shown in Table 7, KDS consistently improves performance on both the challenging DF2K dataset and the standard DIV2K degradation dataset.

Table 7: Performance with LDM-SR backbone on different Real-world SR degradation levels.

Datasets	DF2k					DIV2k				
	PSNR	SSIM	LPIPS ( $\downarrow$ )	NIMA	FID ( $\downarrow$ )	PSNR	SSIM	LPIPS ( $\downarrow$ )	NIMA	FID ( $\downarrow$ )
DPM-Solver++	23.11	0.579	0.276	4.968	18.76	22.06	0.532	0.306	4.922	20.88
+ KDS	<b>23.70</b>	<b>0.594</b>	<b>0.265</b>	<b>4.972</b>	<b>18.38</b>	<b>22.29</b>	<b>0.542</b>	<b>0.290</b>	<b>4.947</b>	<b>20.65</b>
DDIM	22.88	0.542	0.276	4.930	18.59	22.05	0.531	0.307	4.922	20.89
+ KDS	<b>23.71</b>	<b>0.597</b>	<b>0.261</b>	<b>4.943</b>	<b>18.11</b>	<b>22.37</b>	<b>0.549</b>	<b>0.292</b>	<b>4.949</b>	<b>20.78</b>

### A.1.6 Additional comparison with Best-of-N approaches:

While inference-time scaling allows for generating multiple candidate solutions, particularly useful for low-quality inputs, selecting the optimal one from an N-particle ensemble remains a challenge. In our main paper, we didn’t cover the full scope of this experiment. Here, we expand on that by including more metrics. As Table 8 now demonstrates, using non-reference metrics like LIQE [64] or ClipiQA to pick the best particle from an N-selection (i.e., ”best LIQE best of N” or ”best CLIPQA best of N”) results in significantly lower performance compared to our Kernel Density Steering (KDS) method. This shows that, despite comparable computational costs, traditional post-sampling selection methods don’t achieve the same performance level as KDS. As shown in Figure 8, KDS method achieve the most stable performance compared with both BoN baselines, which suffers from the artifacts which confused the non-reference metrics.

Table 8: Comparison of BoN Selection Methods

Method	PSNR	SSIM	LPIPS ( $\downarrow$ )	DISTS ( $\downarrow$ )	LIQE	CLIPQA
DDIM	23.21	0.610	0.370	0.250	4.046	0.689
+ BoN (LIQE)	23.72	0.622	0.361	0.246	<b>4.351</b>	0.741
+ BoN (CLIPQA)	23.01	0.592	0.382	0.247	4.187	<b>0.774</b>
+ KDS	<b>24.39</b>	<b>0.669</b>	<b>0.339</b>	<b>0.245</b>	3.819	0.692

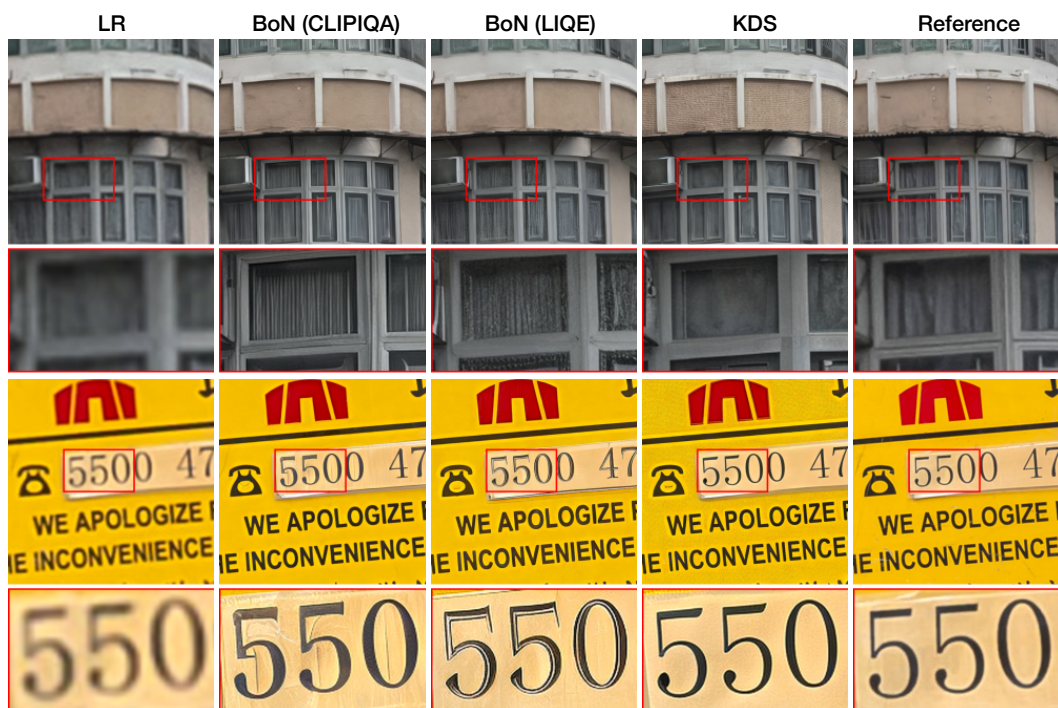


Figure 8: Visual comparison between KDS and Best-of-N (BoN) selection. BoN (CLIPQA) and BoN (LIQE) means the best particle in terms of these two metrics correspondingly.

## 538 A.2 Experiment details: Image Inpainting

539 **Datasets** We generated our inpainting test set by center corpped 30% square region of each image.  
540 We generate used first 1,000 images from ImageNet testset.

541 **Hyperparameters:** Similar to real-world SR settings, we fixed the patch size to 1, bandwidth  $h$  to  
542 0.3, steering strength  $\delta_t$  same as introduced in (Eq: 10).

## 543 A.3 Additional Visual Results

544 This section provides additional qualitative results to visually demonstrate the effectiveness of Kernel  
545 Density Steering (KDS). The figures included are:

- 546 • **Figure 9, Figure 10 and Figure 11:** These figures showcase super-resolution performance  
547 on the DIV2K dataset using LDM-SR, DiffBIR and SeeSR backbones, respectively. They  
548 illustrate improvements in sharpness and detail recovery achieved with KDS.
- 549 • **Figure 12:** This figure highlights KDS’s robustness, demonstrating its performance on the  
550 more challenging DF2K real-world SR dataset with the LDM-SR backbone.
- 551 • **Figures 13, 14, and 15:** These figures display image inpainting results on the ImageNet  
552 dataset using LDM-inpainting. Each figure presents all 10 particles sampled with standard  
553 DDIM versus DDIM enhanced with KDS. They visually confirm KDS’s ability to improve  
554 fidelity and reduce artifacts across the ensemble for the inpainted regions.



Figure 9: Real-world image super-resolution performance with LDM-SR on DIV2K dataset.



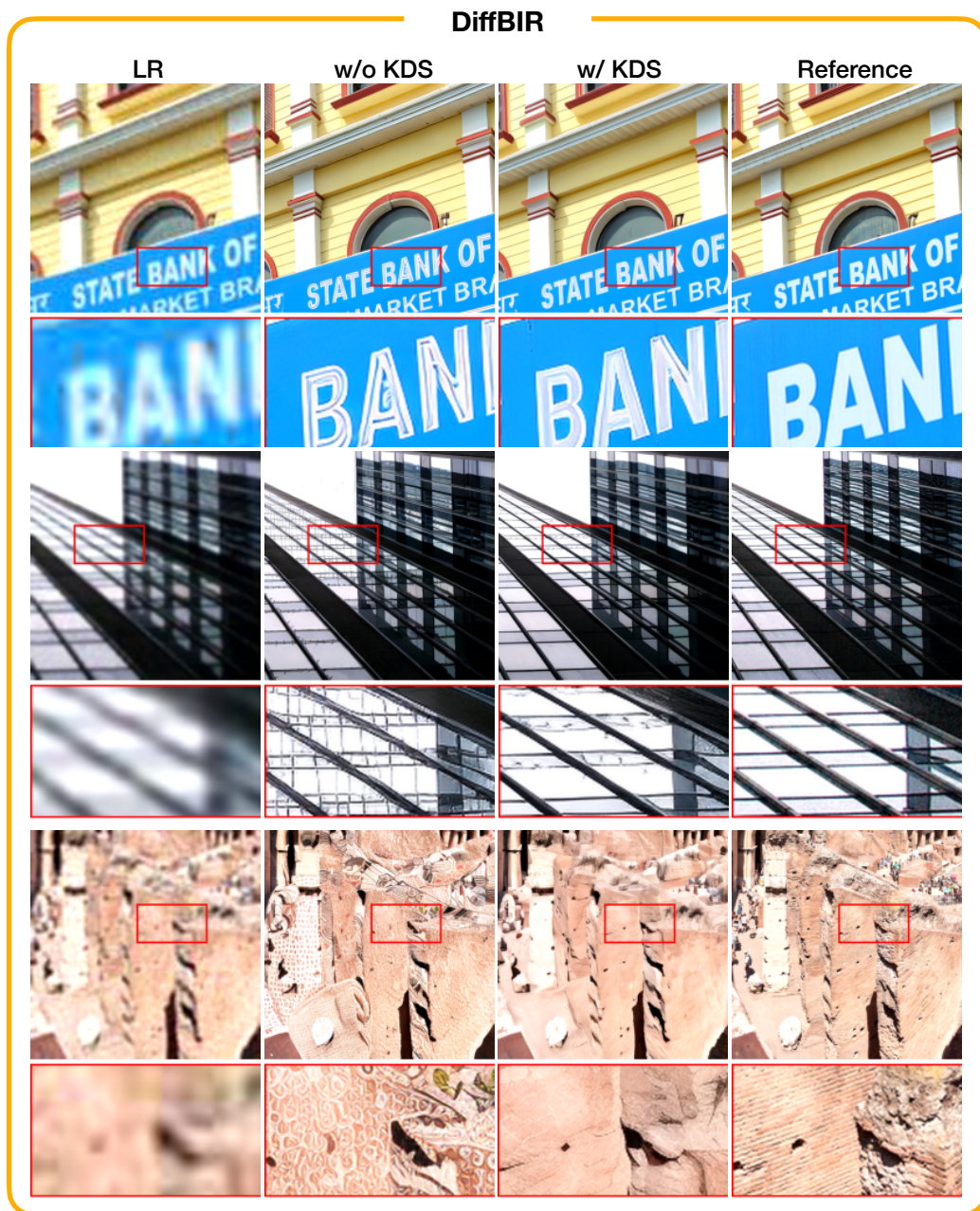


Figure 10: Real-world image super-resolution performance with DiffBIR on DIV2K dataset.

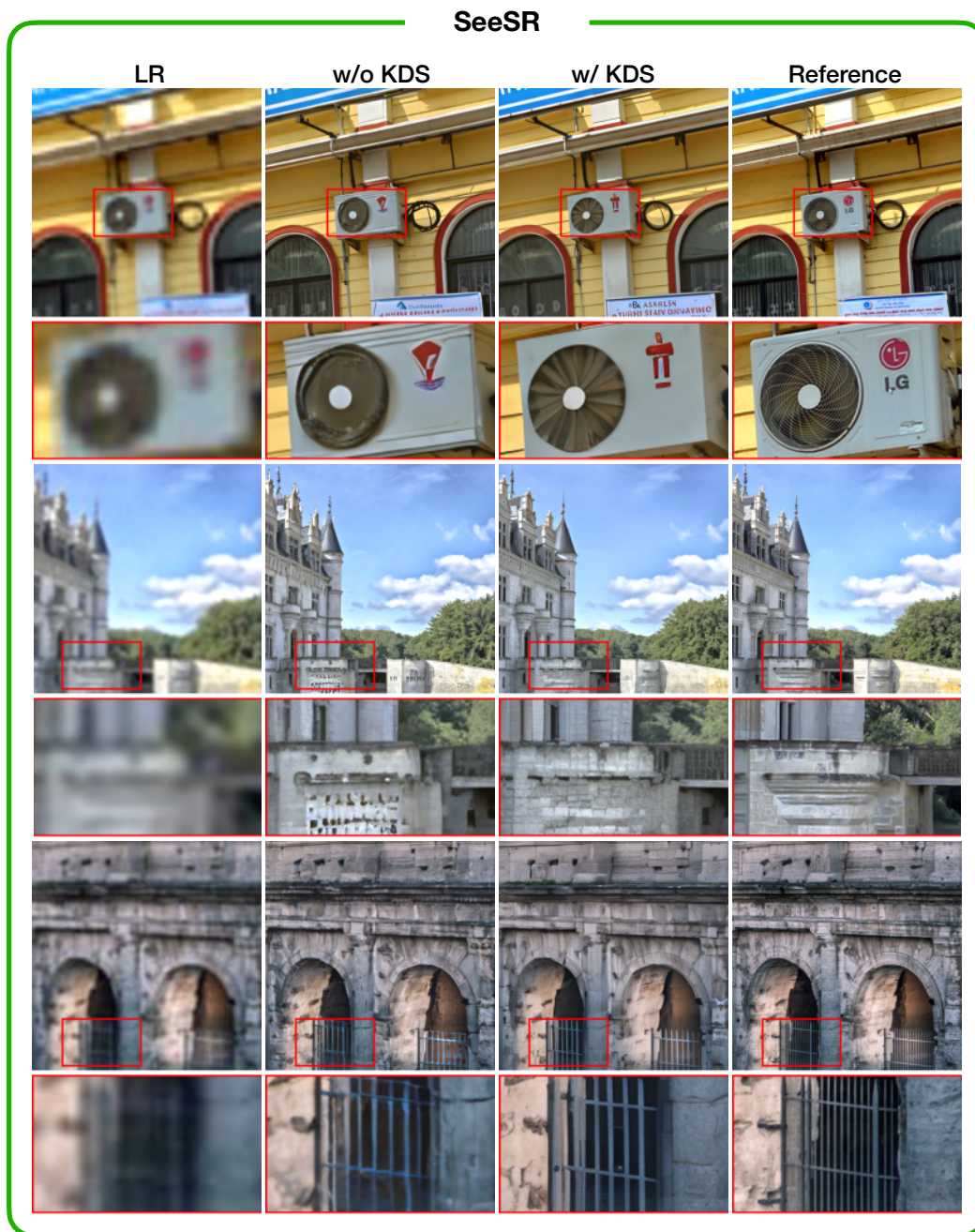


Figure 11: Real-world image super-resolution performance with SeeSR on DIV2K dataset.



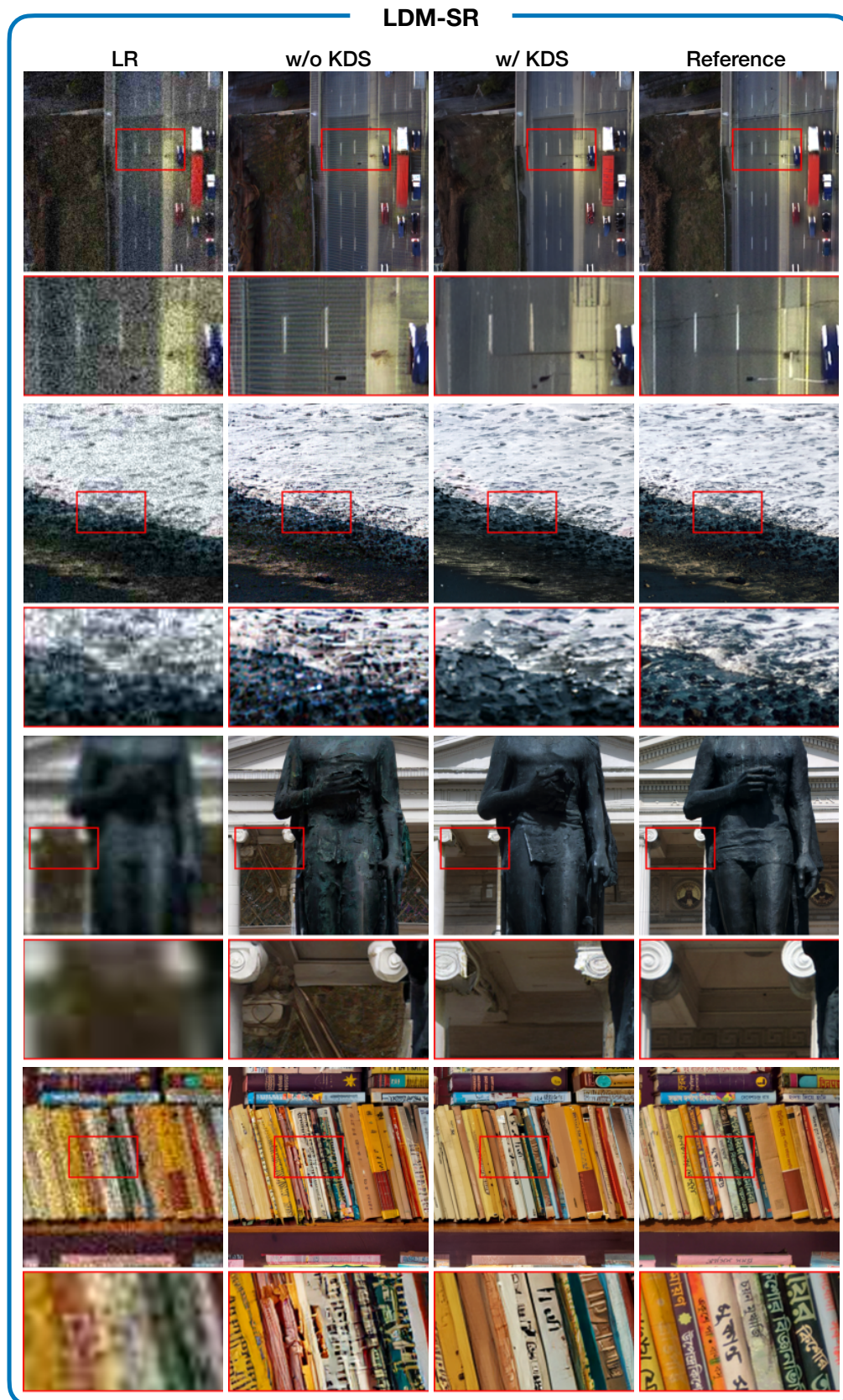


Figure 12: Real-world image super-resolution performance with LDM-SR on DF2K dataset.



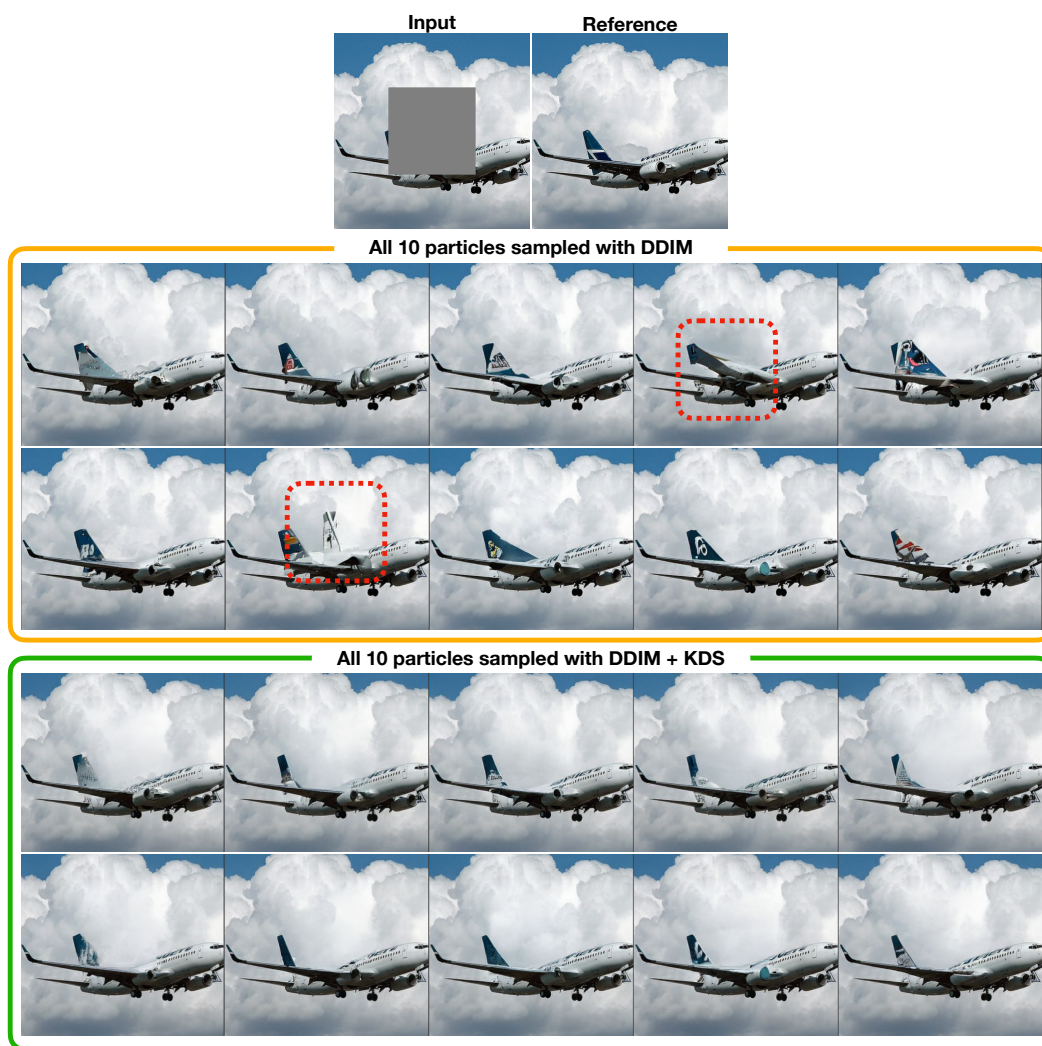


Figure 13: Image Inpainting performance with LDM-inpainting on ImageNet dataset. Visualizes all 10 particles for DDIM vs. DDIM + KDS. Regions with artifacts were highlighted with red box.

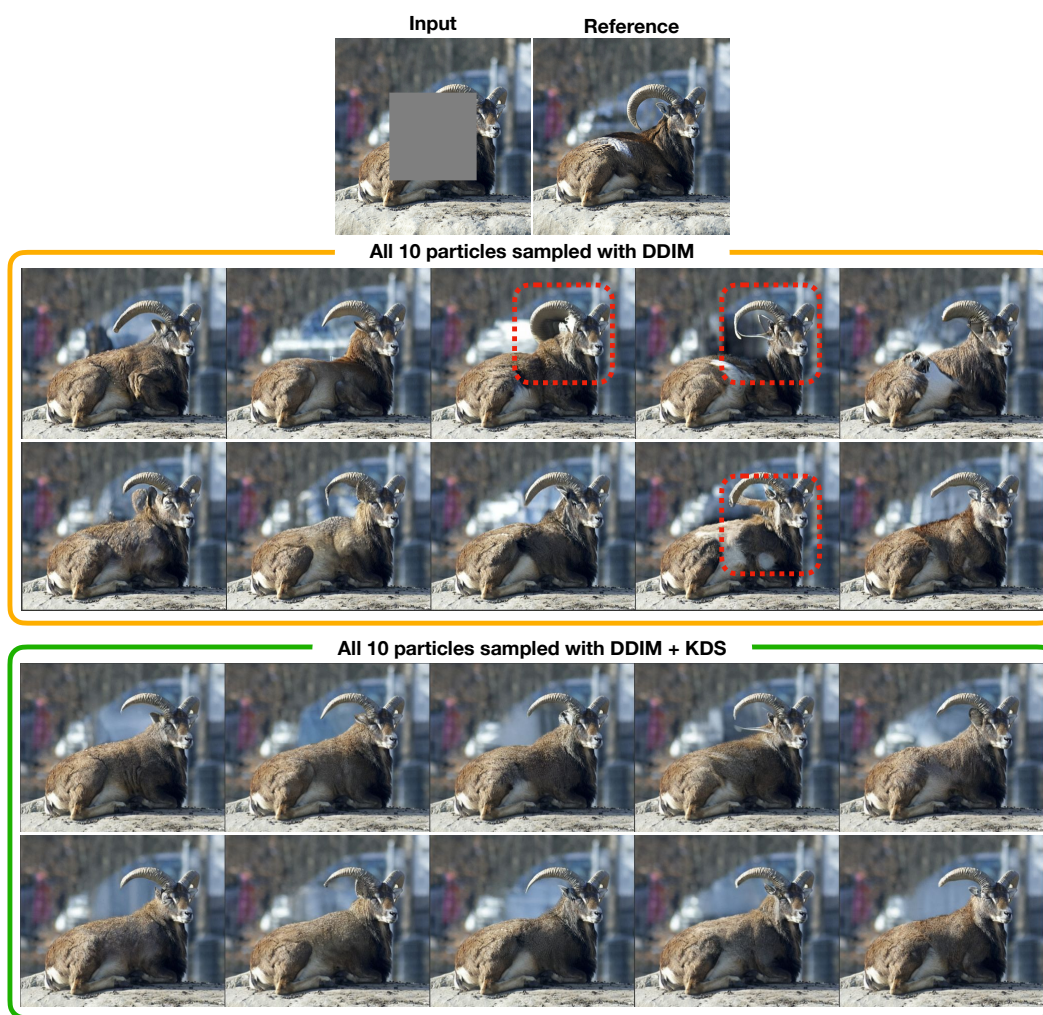


Figure 14: Image Inpainting performance with LDM-inpainting on ImageNet dataset. Visualizes all 10 particles for DDIM vs. DDIM + KDS. Regions with artifacts were highlighted with red box.

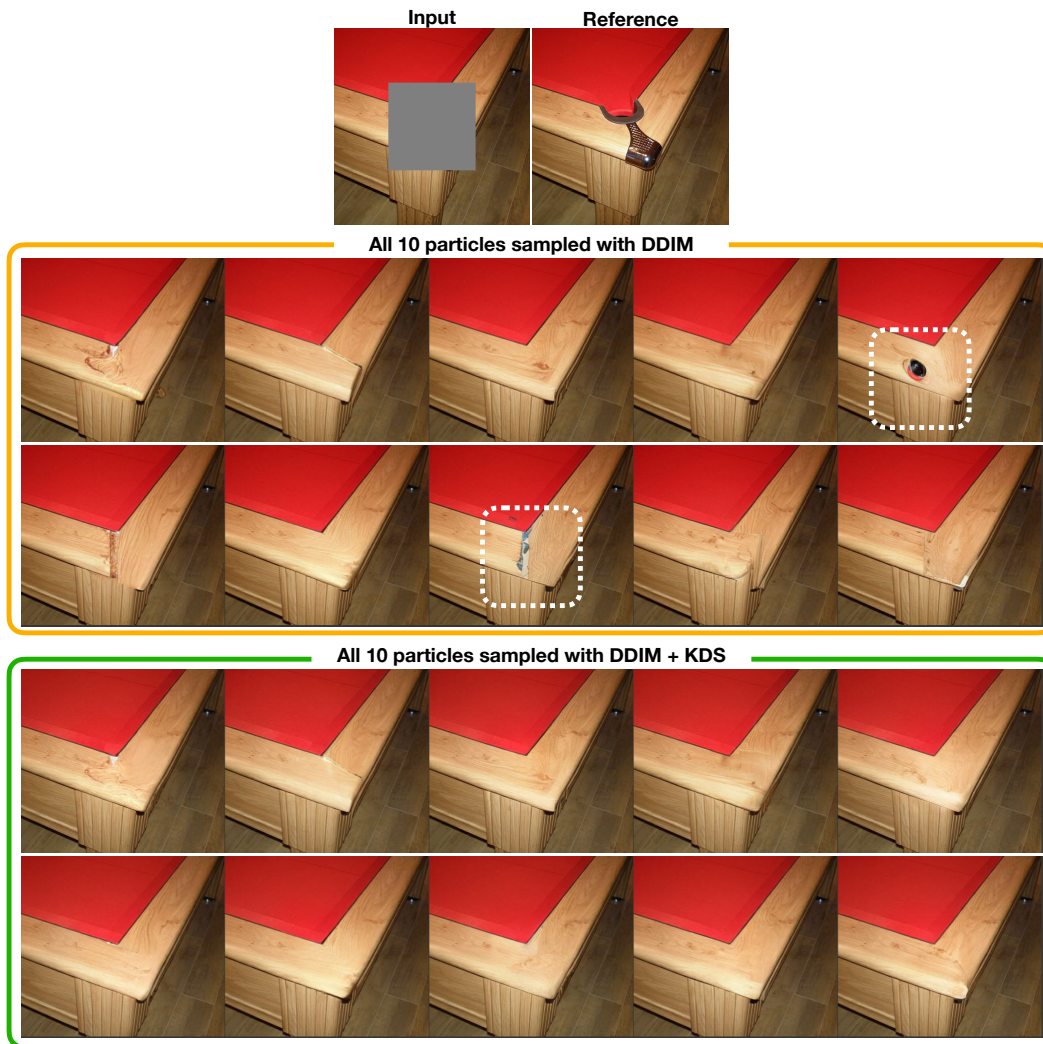


Figure 15: Image Inpainting performance with LDM-inpainting on ImageNet dataset. Visualizes all 10 particles for DDIM vs. DDIM + KDS. Regions with artifacts were highlighted with white box.