GRAPE (Gaussian Rendering for Accelerated Pixel Enhancement) Brings Fast and Lightweight Arbitrary Super-Resolution

Supplementary Materials

1. Perceptual Evaluation

We compare our proposed GRAPE method with bicubic interpolation, LIIF [3], GaussianSR [4], and GSASR [2]. EDSR [6] is adopted as the encoder backbone. The evaluation is conducted on widely used benchmark datasets: Set5 [1], Set14 [11], Urban100 [5], BSD100 [7], and DIV2K [9]. Since PSNR and FPS are already reported in the main paper, we additionally present Structural Similarity Index Measure (SSIM) [10] and Learned Perceptual Image Patch Similarity (LPIPS) [12] in the supplementary material. Both PSNR and SSIM are computed on the Y channel of the YCbCr color space. For all comparisons, we used the official models provided by the respective repositories. This ensures a fair evaluation setting and allows us to directly compare against both traditional interpolation methods and state-of-the-art learning-based approaches. Table 1 reports results on Urban100, Table 2 presents results on Set5 and Set14, and Table 3 shows results on BSD100 and DIV2K. Overall, although GRAPE is not always the best, it is competitive across datasets.

Table 1. Quantitative comparison on Urban100 [5] (average resolution 985×798). SSIM [10] ↑ and LPIPS [12] ↓ are reported for scales $\times 4 \sim \times 30$.

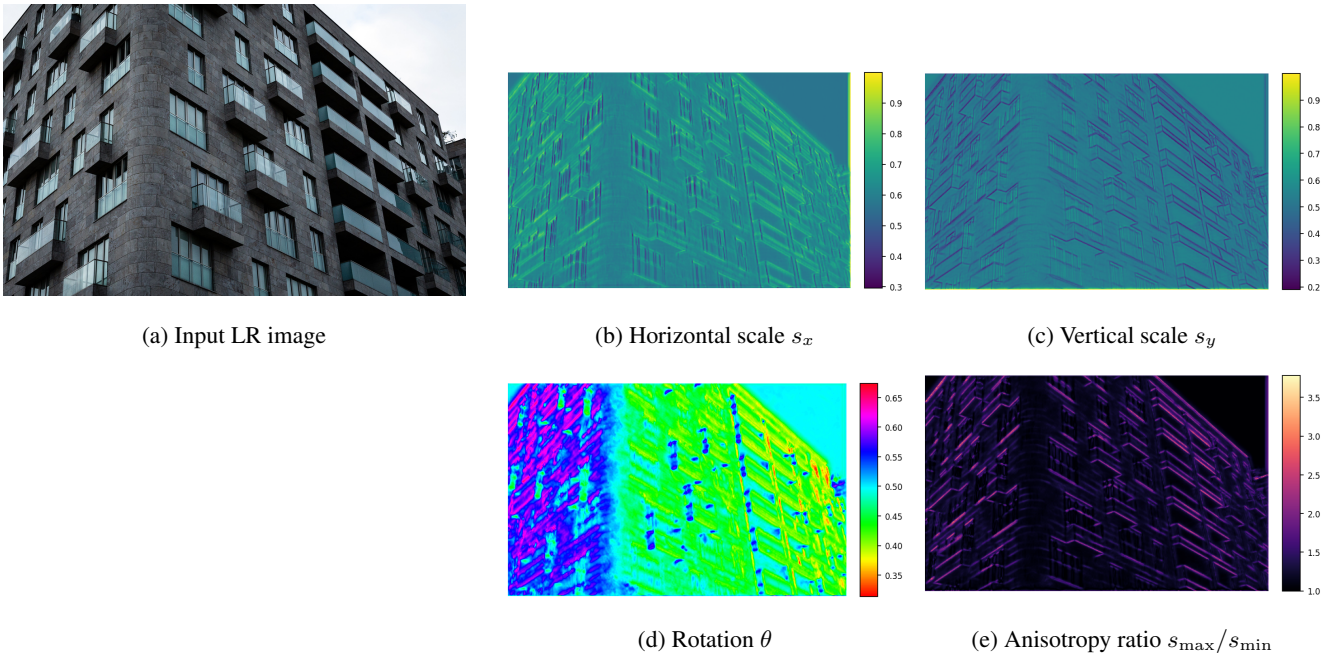
Method	SSIM ↑						LPIPS ↓					
	×4	×8	×12	×16	×24	×30	×4	×8	×12	×16	×24	×30
Bicubic	0.6412	0.3839	0.3311	0.3115	0.2934	0.2887	0.3816	0.6412	0.6906	0.7367	0.7748	0.7763
LIIF [3]	0.7707	0.5926	0.5084	0.4646	0.4215	0.4063	0.2895	0.4529	0.5410	0.5972	0.6600	0.6866
GaussianSR [4]	0.7716	0.5810	0.4904	0.4469	0.4090	0.3973	0.2881	0.4639	0.5686	0.6327	0.6966	0.7183
GSASR [2]	0.8142	0.6480	0.4994	0.5158	0.4686	0.4509	0.1987	0.2515	0.3135	0.5608	0.6312	0.6643
Ours	0.7625	0.5706	0.4880	0.4468	0.4103	0.3979	0.2614	0.4307	0.5147	0.5603	0.6082	0.6315

Table 2. Quantitative comparison on Set5 [1] and Set14 [11]. SSIM [10] ↑ and LPIPS [12] ↓ at various scales using a common EDSR [6] encoder. The average image size for each dataset is noted in parentheses.

Method	Set5 [1] (313×336)						Set14 [11] (491×446)					
	SSIM ↑			LPIPS ↓			SSIM ↑			LPIPS ↓		
	×4	×6.4	×12.8	×4	×6.4	×12.8	×4	×6.4	×12.8	×4	×6.4	×12.8
Bicubic	0.7836	0.6669	0.5310	0.2820	0.3992	0.5722	0.6766	0.5700	0.4688	0.3656	0.4765	0.6272
LIIF [3]	0.8688	0.7803	0.6252	0.2170	0.3032	0.4700	0.7525	0.6473	0.5234	0.3067	0.4098	0.5502
GaussianSR [4]	0.8690	0.7741	0.5985	0.2177	0.3042	0.4958	0.7531	0.6433	0.5088	0.3068	0.4108	0.5689
GSASR [2]	0.8731	0.8218	0.6908	0.2095	0.2909	0.4472	0.7528	0.6893	0.5712	0.2932	0.3928	0.5316
Ours	0.8657	0.7620	0.6023	0.1863	0.3017	0.4678	0.7498	0.6334	0.5092	0.2790	0.3777	0.5285

Table 3. Comparison on BSD100 [7] and DIV2K [9]. SSIM [10] ↑ and LPIPS [12] ↓ at various scales using a shared EDSR [6] encoder. The average image size for each dataset is noted in parentheses.

Method	BSD100 [7] (444×358)						DIV2K [9] (1979×1451)					
	SSIM ↑			LPIPS ↓			SSIM ↑			LPIPS ↓		
	×4	×6.4	×12.8	×4	×6.4	×12.8	×4	×6.4	×12.8	×4	×6.4	×12.8
Bicubic	0.6558	0.5598	0.4780	0.3896	0.5026	0.6566	0.7580	0.6693	0.5851	0.3606	0.4630	0.5998
LIIF [3]	0.7220	0.6174	0.5136	0.3455	0.4506	0.5909	0.8221	0.7328	0.6294	0.3060	0.4023	0.5267
GaussianSR [4]	0.7223	0.6142	0.5050	0.3464	0.4510	0.6049	OOM	OOM	OOM	OOM	OOM	OOM
GSASR [2]	0.7220	0.6454	0.5415	0.3294	0.4348	0.5751	0.8486	0.6454	0.6695	0.2518	0.4348	0.5072
Ours	0.7201	0.6070	0.5032	0.3113	0.4141	0.5585	0.8199	0.7157	0.6121	0.2694	0.3704	0.5039



(a) Input LR image

(b) Horizontal scale s_x

(c) Vertical scale s_y

(d) Rotation θ

(e) Anisotropy ratio $s_{\max}/s_{\min}$

Figure 1. **Learned Gaussian field of GRAPE.** The left panel shows the LR input image. The right panels visualize Gaussian parameters predicted by the model: horizontal scale s_x, vertical scale s_y, rotation θ. The anisotropy ratio $s_{\max}/s_{\min}$ is obtained and visualized from the predicted scales.

2. Gaussian Parameter Visualization

To better understand the behavior of GRAPE, we visualize the Gaussian parameters predicted by the model for a representative example from Urban100 [5]. Unlike implicit decoders that produce RGB values directly, GRAPE predicts a structured field of Gaussian primitives whose scales, rotations, and associated anisotropy jointly determine how each high-resolution pixel is reconstructed. Figure 1 illustrates these parameter maps. The leftmost panel (a) shows the input LR image, and the right four panels depict the Gaussian parameters associated with each primitive:

(b) Horizontal scale s_x. This heatmap reflects the horizontal extent of each Gaussian kernel. Larger values tend to appear along horizontally extended structures such as balcony railings and window boundaries, indicating that GRAPE adaptively broadens its kernel along stable horizontal edges. Texture-rich regions exhibit more rapid variations, suggesting that the model contracts the receptive field to preserve high-frequency details.

(c) Vertical scale s_y. The vertical scale follows a similar trend, highlighting vertically oriented edges such as building corners and window columns. The coherent patterns across s_x and s_y show that the model learns direction-aware receptive fields that align with the underlying geometric layout rather than relying on fixed, isotropic filters.

(d) Rotation θ. This map reveals the dominant in-plane rotation of each Gaussian. Strong edge structures yield consistent rotations that flow smoothly along object boundaries, while textureless regions—where structural cues are weak—exhibit noisier responses. The model thereby learns to rotate Gaussians only when directional information is meaningful, which helps avoid ringing artifacts and oversharpening.

(e) Anisotropy ratio $s_{\max}/s_{\min}$. This ratio quantifies how extended each Gaussian is. High anisotropy occurs along straight building façades, narrow beams, or repetitive grid patterns, where the model stretches kernels along the dominant direction while shrinking them in the orthogonal direction. Conversely, smooth areas remain near-isotropic. This behavior confirms that GRAPE selectively applies directional smoothing only where it benefits reconstruction fidelity.

Discussion. We visually examined the effectiveness of the anisotropic formulation and confirmed that it better aligns with dominant structures in the image. In our implementation, we constrain the ellipse parameters using a sigmoid function to keep them within a stable 0–1 range, preventing overly elongated or degenerate kernels while still allowing meaningful anisotropic variation. However, replacing this with a softplus parameterization caused the values to grow uncontrollably and led to training instability. Developing a more robust parameterization that avoids such numerical issues remains an important direction for future work.

3. Analysis of the Quality–Efficiency Trade-Off via Ensemble Size K

To investigate the quality–efficiency behavior of GRAPE, we examine the effect of varying the ensemble size $K \in \{1, 2, 4, 8, 16\}$. Since the ensemble serves as a primary capacity-control mechanism, this experiment illustrates how the number of Gaussian primitives influences both reconstruction performance and runtime. The results are visualized in Fig. 2.

Increasing K generally improves reconstruction quality: PSNR increases from 25.81 dB at $K = 1$ to 25.87 dB at $K = 4$, while inference speed decreases as more primitives must be evaluated. This reflects the expected trade-off between model capacity and computational cost.

Quality improvements saturate beyond $K = 4$. Larger ensembles such as $K = 8$ or $K = 16$ do not yield consistent gains and may introduce minor redundancy, with the slight fluctuation observed at $K = 16$ remaining within normal variance. Thus, $K = 4$ provides a balanced operating point, whereas smaller values favor efficiency and larger values incur additional overhead with limited benefit.

This analysis shows that adjusting K enables GRAPE to flexibly traverse different quality–efficiency settings, allowing the model to accommodate varying accuracy and runtime requirements.

4. Comparison with ContinuousSR

ContinuousSR [8] is one of the closest prior works to GRAPE in that both approaches utilize 2D Gaussian splatting for arbitrary-scale super-resolution (ASSR). While they share this overall direction, their internal formulations and computational choices differ in meaningful ways. Below we provide a concise comparison that highlights these distinctions.

Gaussian parameterization. ContinuousSR constructs Gaussians using a learned dictionary of covariance atoms, together with covariance weighting and adaptive position drifting. According to [8], directly estimating all Gaussian parameters end-to-end presents considerable optimization difficulty, often leading to suboptimal convergence. To better understand this behavior, the authors analyze Gaussian parameters obtained by converting a large collection of high-resolution images into Gaussian space, revealing consistent and bounded covariance distributions referred to as the Deep Gaussian Prior (DGP). These findings motivate their dictionary-assisted formulation, which uses predefined covariance atoms and learned weights to stabilize parameter construction. GRAPE takes a more direct approach. We regress rotation, scales, offsets, and colors through a single 1×1 convolutional head with a lightweight K-ensemble, relying on a rotation–scale factorization to ensure valid covariance matrices. This compact formulation

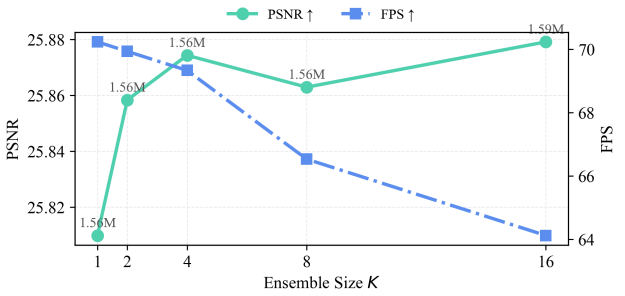


Figure 2. Effect of ensemble size K on inference speed, model size, and PSNR. The quality gains saturate beyond $K = 4$, while computational cost continues to increase.

Table 4. Comparison between ContinuousSR [8] and GRAPE.

	ContinuousSR	**GRAPE (ours)**
Parametrization	Dictionary-based	Direct regression
Params (M)	5.57	1.56
Training Prior	Uses DGP prior	Not required
FPS (Urban100 $\times4$)	12.31	69.33

enables end-to-end learning without external priors or pre-analyzed Gaussian statistics, and we find it provides stable optimization in practice. As a result, GRAPE avoids the additional structural components required in the dictionary-based pipeline.

Architectural simplicity and efficiency. These design choices lead to clear architectural differences. ContinuousSR includes dictionary lookup and covariance assembly modules, resulting in a model containing 5.57M parameters and additional inference overhead. GRAPE, by predicting complete Gaussian parameters directly, eliminates these intermediate steps and operates with 1.56M parameters. This design yields higher runtime efficiency. On Urban100 $\times4$, ContinuousSR reaches 12.31 FPS while GRAPE achieves 69.33 FPS. The efficiency gap is consistent with the reduced indirection in GRAPE's parameterization. These differences are summarized in Table 4.

Summary. Overall, while both approaches build on Gaussian-based rendering for ASSR, they differ substantially in their underlying methodology. ContinuousSR leverages a dictionary-assisted pipeline informed by statistical analysis, whereas GRAPE uses a compact, fully end-to-end parameterization that learns all Gaussian attributes directly from features. These differing design priorities lead to a simpler model and notably higher computational efficiency in GRAPE.

References

[1] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. pages 135.1–135.10, 2012. 1

[2] D. Chen et al. Generalized and efficient 2d gaussian splatting for arbitrary-scale super-resolution. https://arxiv.org/abs/2501.06838, 2025. 1

[3] Y. Chen, S. Liu, and X. Wang. Learning continuous image representation with local implicit image function. pages 8628–8638, 2021. 1

[4] J. Hu, B. Xia, B. Chen, W. Yang, and L. Zhang. Gaussiansr: High-fidelity 2d gaussian splatting for arbitrary-scale image super-resolution. pages 3554–3562, 2025. 1

[5] J. B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. pages 5197–5206, 2015. 1, 2

[6] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. pages 136–144, 2017. 1

[7] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. pages 416–423, 2001. 1

[8] L. Peng et al. Pixel to gaussian: Ultra-fast continuous super-resolution with 2d gaussian modeling. https://arxiv.org/abs/2503.06617, 2025. 3

[9] R. Timofte, E. Agustsson, L. Van Gool, M. H. Yang, and L. Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. pages 1110–1121, 2017. 1

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. 13(4):600–612, 2004. 1

[11] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse representations. In *Int. Conf. Curves and Surfaces*, pages 711–730, 2010. 1

[12] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. https://arxiv.org/abs/1801.03924, 2018. 1