Bézier Splatting for Fast and Differentiable Vector Graphics Rendering

Xi Liu[1*], **Chaoyi Zhou**[1*], **Nanxuan Zhao**[2], **Siyu Huang**[1]
[1]Clemson University, [2]Adobe Research

Figure 1: This work proposes *Bézier Splatting*, a new differentiable vector graphics (VGs) renderer that achieves an order-of-magnitude computational speedup in comparison with the state-of-the-art method DiffVG [15] (tested on a NVIDIA RTX 4090 GPU).

Abstract

Differentiable vector graphics (VGs) are widely used in image vectorization and vector synthesis, while existing representations are costly to optimize and struggle to achieve high-quality rendering results for high-resolution images. This work introduces a new differentiable VG representation, dubbed Bézier Splatting, that enables fast yet high-fidelity VG rasterization. Bézier Splatting samples 2D Gaussians along Bézier curves, which naturally provide positional gradients at object boundaries. Thanks to the efficient splatting-based differentiable rasterizer, Bézier Splatting achieves 30× and 150× faster per forward and backward rasterization step for open curves compared to DiffVG. Additionally, we introduce an adaptive pruning and densification strategy that dynamically adjusts the spatial distribution of curves to escape local minima, further improving VG quality. Furthermore, our new VG representation supports conversion to standard XML-based SVG format, enhancing interoperability with existing VG tools and pipelines. Experimental results show that Bézier Splatting significantly outperforms existing methods with better visual fidelity and significant optimization speedup. The project page is xiliu8006.github.io/Bezier_splatting_project.

1 Introduction

Vector graphics (VGs) represent images through parametric primitives such as points, curves, and shapes. Unlike raster images, they enable structured representations, lossless resizing, compact storage, and precise content editing, making them crucial for various applications such as user interfaces and animation.

Recently, differentiable VG rasterization gains significant attention, as it allows raster-based algorithms to edit or synthesize VGs through gradient-based optimization. DiffVG [15] is the first differentiable VG framework that leverages the anti-aliasing algorithm to differentiate the vector curves that are inherently discontinuous in pixel space. However, it suffers from slow training and low-fidelity rendering, particularly for high-resolution images. LIVE [18] further introduces a

Corresponding author: Siyu Huang

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

layer-wise coarse-to-fine strategy to improve quality and topology. However, it is highly computationally expensive, requiring 5 hours to vectorize a 2K-resolution image. Additionally, learning-based methods such as Im2Vec [21] train neural networks to map pixels to VGs, but they are restricted to simple graphics and struggle with out-of-domain generalization. Towards scalable applications of VGs, these challenges highlight the need for a differentiable VG method with better efficiency, fidelity, and generalization capability.

This work presents *Bézier Splatting*, a new differentiable VG representation that optimizes Bézier curves through Gaussian splatting-based rasterization. We sample 2D Gaussian points along Bézier curves and their interior regions, then leverage the Gaussian Splatting framework [12] for efficient curve rasterization. Unlike DiffVG [15] which requires computationally intensive boundary sampling and gradient computation, 2D Gaussians inherently provide direct positional gradients at object boundaries through its differentiable Gaussian formulation, enabling over *150*$\times$ faster backward computation over DiffVG for open curves. We further introduce an adaptive pruning and densification strategy that adaptively removes redundant curves while adding new ones to necessary regions during optimization. It helps the optimization process escape the local minima of current spatial distributions of curves, formulating a "global receptive field" for further improving the VG optimization process. As shown in Fig. 1, Bézier Splatting outperforms the state-of-the-art differentiable VG rasterizer DiffVG [15] for both open and closed curves in terms of efficiency and rendering quality. We summarize the contribution of this work as follows.

- We propose a novel differentiable vector graphic representation, Bézier Splatting, which achieves an order-of-magnitude computational speedup while producing high-quality rendering results.
- We present an adaptive pruning and densification strategy to improve the optimization process of Bézier curves by escaping the local minima of the spatial distributions of curves.
- Extensive experiments demonstrate that Bézier Splatting outperforms existing differentiable VG rendering methods in efficiency and visual quality.

2 Related works

2.1 Vectorization and Rasterization

Image vectorization and vector graphic (VG) rasterization are important research topics in computer graphics and computer vision. DiffVG [15] introduces the first differentiable VG rasterization framework, laying a foundation for optimizing and generating vectorized representations through gradient-based methods. It expands the applicability of VGs to a broader range of tasks including image vectorization, text-to-VG generation, and painterly rendering. Based on DiffVG [15], LIVE [18] and O&R [7] further proposes a layer-wise path initialization strategy, vectorizing raster images into compact and semantically consistent VG representations while preserving image topology. Du *et al.* [4] use linear gradients to fill the regional colors, and Chen *et al.* [3] propose a specific implicit neural representation to model regional color distributions, enhancing the color representation within closed Bézier curves. However, these DiffVG-based VG rasterization approaches [15, 18, 4, 3] suffer from slow optimization, often requiring several hours to process a 2K-resolution image with 1,024 curves.

With the advancement of deep learning, another line of approaches directly learns deep neural networks for vector synthesis. Lopes *et al.* [16] combine image-based encoder and VG decoder to generate fonts. Img2Vec [21] integrates the encoder with recurrent neural networks (RNNs) by leveraging sequential modeling for structured vector synthesis. SVGFormer [2] further adopts a Transformer-based architecture [25] to improve the capacity for representing complex geometric structures. More recently, diffusion models [8] have been applied to text-to-VG synthesis [33, 11, 30, 29]. However, existing learning-based VG synthesis approaches are limited to generating simple graphics, and struggle with out-of-domain data. In this work, we propose a novel VG representation dubbed Bézier Splatting that achieves high-fidelity VG rendering within minutes of optimization for high-resolution images.

2.2 Gaussian Splatting

3D Gaussian Splatting [12, 38] emerges as a promising approach for novel view synthesis (NVS) and attracts significant attention from the community. Its explicit 3D Gaussian-based volumetric

representation enables high-fidelity 3D reconstruction, while the differentiable tile-based rasterization pipeline ensures real-time and high-quality rendering. It has been applied to various domains and tasks, such as 4D modeling [17, 28] and 3D scene generation [31, 23]. Several works further improve the Gaussian primitives for better representation quality. SuGaR [5] proposes to approximate 3D Gaussians with 2D Gaussians for enhanced surface reconstruction. 2D Gaussian Splatting [9] directly adopts 2D Gaussians for 3D reconstruction for simplified optimization and improved geometric fidelity. TetSphere splatting[6] further employs tetrahedral meshes as the geometric primitives to achieve high-quality geometry. Particularly for 2D image representation, GaussianImage [35] adopts 2D Gaussians [9] for efficient image representation, achieving a compact and expressive alternative to rasters or implicit representations [22, 19]. Image-GS [36] further enhances it through a content-adaptive compression approach. This work proposes to integrate Gaussian splatting with vector representations for differentiable VG rasterization. By sampling 2D Gaussians on Bézier curves and rasterizing through the efficient Gaussian splatting pipeline, the proposed Bézier Splatting representation enables fast yet high-quality VG rendering, even for high-resolution images with complex structures.

3 Method

3.1 Overall

Given a raster image, our goal is to efficiently vectorize it into a VG representation that closely resembles the input while preserving the details. Existing methods, including DiffVG [15] and its following work [18, 3], incur substantial computational costs due to the pixel color accumulation computation. Specifically, DiffVG first constructs a bounding volume hierarchy (BVH) tree to determine the curves that intersect with each individual pixel, then solves equations to precisely determine whether a pixel lies within a region and to compute the inward or outward gradients at boundary points.

To overcome this inefficiency, this work proposes a novel VG representation, Bézier Splatting, which is inspired by the high computational efficiency and expressive fitting capacity of Gaussian splatting [35] for rasterization. Bézier Splatting *samples 2D Gaussian points along Bézier curves and their interior regions*, then *leverages the Gaussian splatting method for efficient rasterization*, enjoying the following advantages:

- This design significantly accelerates the forward and backward pass of Bézier curve rasterization, achieving an order-of-magnitude speedup without specialized optimization techniques;
- The 2D Gaussian representation inherently provides direct position gradients for object boundaries, eliminating the need for additional computations such as boundary sampling and gradient derivation via the Reynolds transport theorem in DiffVG [15];
- It supports richer texture representation by allowing properties such as spatially varying opacity and width, facilitating complex effects such as linear-gradient color transitions within a single curve.

To further improve the fidelity and expressiveness of Bézier Splatting, we introduce a pruning and densification approach (Sec. 3.5), which dynamically removes redundant Bézier curves while adaptively adding necessary curves in regions that have high reconstruction error. Fig. 2 illustrates the algorithm flow of Bézier Splatting. More details are discussed in the following sections.

3.2 Primitives of Bézier Splatting

Bézier curves. We adopt Bézier curves as the parametric primitives of VGs. The representation includes N Bézier curves with a degree of M, as:

$$\mathcal{B}_i(t) = \sum_{j=0}^{M} B_j^M(t) P_j^{(i)}, \; t \in [0,1], \; i \in \{1, \ldots, N\}, \tag{1}$$

where t is a normalized position on the curve, $P_j^{(i)}$ represents the j-th control point of the i-th Bézier curve, and $B_j^M(t)$ is the Bernstein polynomial of degree M, given by:

$$B_j^M(t) = \binom{M}{j} (1-t)^{M-j} t^j. \tag{2}$$

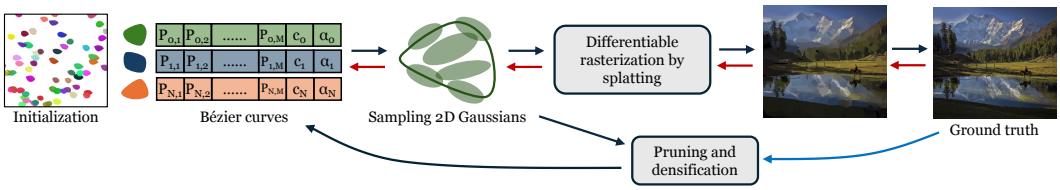

Figure 2: An illustration of the algorithm flow of Bézier Splatting. It begins by randomly initializing Bézier curves and uniformly sampling Gaussians points along them. These Gaussians are then rasterized into the image, enabling gradient-based computation to optimize parameters of both Bézier curves and Gaussians. Curves with negligible opacity or extremely small shapes are removed, while new curves are adaptively added into areas with high reconstruction error, ensuring curves are placed in areas requiring finer details. $\rightarrow$ forward, $\leftarrow$ backpropagation, $\leftarrow$ error map.

Each Bézier curve $\mathcal{B}_i(t)$ is associated with an RGB color parameter $c_i \in \mathbb{R}^3$, and an opacity parameter $o_i \in [0, 1]$ that defines the transparency of the curve.

To formulate an open curve, we follow DiffVG [15] to adopt three sequentially connected Bézier curves with two control points on each. An open curve requires an additional width parameter to define the stroke thickness. For a closed curve, we adopt two connected Bézier curves, enabling a more efficient sampling in enclosed regions. For closed curves, color filling is applied. The connected Bézier curves in either an open or closed curve share the color parameters, but they have separate opacity parameters to better model the opacity changes along the curves or within closed areas for enriching the texture representation capacity.

2D Gaussians on curves. Our Bézier Splatting novelly associates 2D Gaussians with each Bézier curve. The standard formulation of 2DGS [9] parameterizes each 2D Gaussian by position, color, rotation, scale, opacity, and depth. However, to ensure a compact and differentiable representation, these parameters of Gaussians in our Bézier Splatting are inherited from the corresponding control points. We discuss more details of the sampling of Gaussians, rasterization process, and backaward computation in the following sections.

3.3 Sampling Gaussians on Bézier Curves

This work introduces a fast differentiable VG rasterizer based on Gaussian splatting, allowing gradients from raster images to be backpropagated to the 2D Gaussians, then further backpropagated to the Bézier curves through a differentiable sampling strategy, resulting in a highly efficient optimization of the Bézier curves.

Specifically, for each Bézier curve $\mathcal{B}_i(t)$, we uniformly sample K points along it based on Eq. 1. The sampled point set $\mathbf{b}_i$ is:

$$\mathbf{b}_i = \left[\mathcal{B}_i(t_0), \mathcal{B}_i(t_1), \ldots, \mathcal{B}_i(t_{K-1})\right], \tag{3}$$

where t_k is uniformly sampled from $[0, 1]$.

Sampling 2D Gaussians on open curves. We represent an open curve by using 3 sequential Bézier curves $\mathcal{B}_i(t)$ with a degree of 4 to form a single continuous stroke, by following the same setting as DiffVG [15]. The stroke consists of 10 control points, as the end point of each Bézier curve serves as the start point of the next. To ensure that the final rendering result generates a continuous stroke with consistent width and color, we calculate the x-direction scale of a Gaussian point by the distance between neighboring points by following Eq. 7, while the y-direction scale is a learnable parameter that represents the stroke width.

For both closed and open curves, the depth d of a Gaussian is assigned based on the area of curves, ensuring smaller curves are not occluded by larger ones. It prevents them from being ignored during optimization, as the gradient would not count Gaussians when the accumulated opacity exceeds 1.

Sampling 2D Gaussians on closed curves. Achieving accurate color filling is non-trivial for closed curves. A straightforward approach involves uniformly sampling a large number of points, identifying intersecting curves, solving equations to determine which curves cover them, and then computing scaling factors based on the sampled points. However, this process is computationally expensive, making it inefficient for curve rasterization and optimization.

To address this limitation, we propose a new structure named *paired Bézier curve structure*, which supports two groups of Bézier curves with equal cardinality and arbitrary degrees, forming a closed region. This structure enables efficient and flexible sampling within the enclosed region between the two curve groups. Specifically, the two Bézier curves $\mathcal{B}_1(t)$ and $\mathcal{B}_{R+1}(t)$ share the same start and end points, forming the inside curves and boundaries of a closed region. A total of R intermediate Bézier curves are generated by linearly interpolating the corresponding control points $P_j^{(0)}$ and $P_j^{(R+1)}$ as:

$$P_j^{(k)} = (1 - t_k)P_j^{(0)} + t_k P_j^{(R+1)}, \quad k = 1, \ldots, R, \tag{4}$$

where $t_k \in [0, 1]$ are sampled from a normalized cumulative distribution function (CDF). The interpolated control points $\{P_j^{(k)}\}$ define intermediate curves that form a dense strip between the two boundaries. Note that this paired structure can naturally extend to cases where each curve is composed of multiple connected Bézier segments, provided that all segments have the same number of control points, similar to the representation used in DiffVG [15].

This non-uniform sampling ensures that points near the boundary curves have small scales, mitigating the influence of interior Gaussians to the exterior of closed area. The interpolated Bézier curves are:

$$\mathcal{B}_k^{\text{interp}}(t) = \sum_{j=0}^{M} B_j^M(t) P_j^{(k)}, \quad k = 1, \ldots, R. \tag{5}$$

Then, 2D Gaussians are sampled on these interpolated curves, by following the same procedure as open curves (Eq. 3), resulting in a structured and efficient point sampling within the enclosed region. Furthermore, to mitigate artifacts caused by non-convex curve shapes during interpolation, we incorporate the Xing loss from LIVE [18], which enforces convexity constraints on curve shapes, to improve the stability of the interpolation process. The full set of sampled points on the i-th Bézier curve is:

$$\mathbf{X} = \left[\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{R+1}\right] \in \mathbb{R}^{(R+2)\times K\times 2}. \tag{6}$$

Let $\mathbf{X}_{r,k}$ denote the 2D position of the k-th sampled point on the r-th interpolated curve. To construct anisotropic 2D Gaussian primitives aligned with the local geometry, we define the spatial scales as follows. The *x-direction* of each 2D Gaussian is defined to align with the local tangent direction of the curve (i.e., along the curve), while the *y-direction* is perpendicular to it (i.e., across adjacent curves).

The scale $\sigma_x(r, k)$ is computed from the Euclidean distance between consecutive points along the curve, and $\sigma_y(r, k)$ is computed from the distance between corresponding points on adjacent curves:

$$\begin{aligned} \sigma_x(r, k) &= |\mathbf{X}_{r,k+1} - \mathbf{X}_{r,k}|_2/\rho, \\ \sigma_y(r, k) &= |\mathbf{X}_{r+1,k} - \mathbf{X}_{r,k}|_2/\rho. \end{aligned} \tag{7}$$

Here, ρ is a global constant that controls the overall density and overlap of the Gaussians. The rotation $\theta_{r,k}$ of each Gaussian is defined by the angle of the local tangent vector, estimated from neighboring points as

$$\theta_{r,k} = \operatorname{atan2}\left(y_{r,k+1} - y_{r,k-1}, x_{r,k+1} - x_{r,k-1}\right). \tag{8}$$

For boundary points, the rotation is set to align with the nearest available neighbor.

3.4 Splatting-based Differentiable Rasterization

Once all Gaussians on Bézier curves are sampled, the rasterization process follows the Gaussian splatting pipeline [12], which is very fast in both forward and backward computation. Different from GaussianImage [35], we use α-blending for pixel rendering instead of the accumulation-based blending. Accumulation-based blending calculates the pixel value based on all overlapping Gaussians, such that it conflicts with the rendering principles of VGs, where occlusion plays a crucial role in defining vector structures. α-blending ensures proper occlusion handling, as it allows foreground elements to contribute more to the final pixel value. The rendering of a pixel is:

$$C_n = \sum_{i \in M} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{9}$$

C_n is the n-th pixel color, $\mathbf{c}_i$ is the color of corresponding Gaussians, α_i is computed by the projected 2D covariance Σ_i:

$$\alpha_i = o_i \exp^{-\sigma_i}, \quad \sigma_i = \frac{1}{2}\boldsymbol{d}_n^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{d}_n. \tag{10}$$

where d_n is the distance between the pixel and Gaussian center, and Σ_i can be modeled by θ_i, σ_x^i and σ_y^i as:

$$\boldsymbol{\Sigma}_i = (\boldsymbol{R}_i \boldsymbol{S}_i)(\boldsymbol{R}_i \boldsymbol{S}_i)^T. \tag{11}$$

$$\boldsymbol{R}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}, \quad \boldsymbol{S}_i = \begin{bmatrix} \sigma_x^i & 0 \\ 0 & \sigma_y^i \end{bmatrix}. \tag{12}$$

Discussion on method efficiency. The rasterization process of Bézier Splatting is very efficient, since we directly sample 2D Gaussians from Bézier curves in a differentiable manner, then splat them to the 2D plane. The rasterization pipeline remains end-to-end differentiable while being highly optimized for parallel computation and large-scale matrix operations. As a result, Bézier Splatting is a highly efficient and fully differentiable VG representation, which ensures both fast rendering and gradient-based optimization. It not only preserves the flexibility of VGs but also allows seamless integration into deep learning frameworks, making it well-suited for tasks requiring high-quality and editable vector representations.

3.5 Optimization

Training objective. Given a raster image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, the goal of image vectorization is to vectorize the image into Bézier curves while ensuring a high-fidelity reconstruction. We first randomly generate a set of Bézier curves to lay on the canvas, then employ the differentiable rasterizer discussed in Sec. 3.4 to render a raster image $\hat{\mathcal{I}} \in \mathbb{R}^{H \times W \times 3}$. The Bézier curves can then be optimized through any gradient-based loss functions. In this work, we formulate the optimization objective as minimizing the loss between $\mathcal{I}$ and $\hat{\mathcal{I}}$ while enforcing the curves to be convex. Therefore, we only adopt an L_2 loss and a Xing loss L_{Xing} [18], as:

$$L = \lambda_1 \|\hat{\mathcal{I}} - \mathcal{I}\|_2^2 + \lambda_2 L_{\text{Xing}} \tag{13}$$

where λ_1 and λ_2 are hyperparameters that trade off the two loss functions.

Adaptive curve pruning and densification. The gradients of 2D Gaussians are influenced by local pixels only. Therefore, it is hard for 2D Gaussians to dynamically reallocate to regions requiring finer-grained details. The Gaussians would be trapped in local minima, leading to redundant Gaussians that are optimized as either low opacity or excessively large size, resulting in artifacts in rendering results.

In standard 3D Gaussian Splatting pipeline [12], Gaussians with low opacity or excessive size are pruned, whereas those with high gradient responses are split into two. However, this strategy is not directly applicable to Bézier curves. Note that in volumetric representations, large Gaussians are usually unnecessary for modeling any particular structure. In contrast, VG representations often encompass large uniform regions such as backgrounds or areas with homogeneous colors (*e.g.,* walls). Consequently, the size-based pruning strategy would remove critical structures in VG representations. Similarly, splitting a Bézier curve into two can introduce significant randomness, as high-gradient regions do not always indicate poor reconstruction in VGs. Since VGs assume a uniform color within each enclosed region, complex textures naturally produce high gradients. This does not imply the region should be split, as it may disrupt the semantic consistency of the VG representation.

To address this issue, this work introduces a new pruning and densification strategy to dynamically adjust the density of Bézier curves throughout the optimization process. For pruning redundant Bézier curves, we apply three criteria to ensure a stable and precise optimization process. First, we remove curves with opacity below a dynamic threshold that gradually decreases as optimization progresses, ensuring that weakly contributing curves are eliminated while preserving essential structures. Second, we remove curves with an area below a predefined threshold, as they contribute minimally to the final representation. To remove visually insignificant or noisy Bézier curves, we apply an opacity-based filtering strategy: for per-segment opacities, we discard curves whose middle segment is significantly fainter than both ends—often indicating that a single curve improperly spans a region better represented by multiple curves; otherwise, we discard curves with overall opacity below a threshold (e.g., 0.2) to eliminate globally low-visibility curves. Third, we remove curves that exhibit high color similarity with surrounding curves and have significant overlap, as they provide little

Table 1: Computational speed for rendering a 2,040×1,344 image with 2,048 curves.

	Open curve			Closed curve		
	DiffVG [15]	Bézier Splatting	Speedup	DiffVG [15]	Bézier Splatting	Speedup
Forward pass	141.3ms	**4.5ms**	*31.4*×	85.2ms	**14.1ms**	*6.0*×
Backward pass	701.3ms	**4.7ms**	*149.2*×	448.3ms	**24.58ms**	*18.2*×

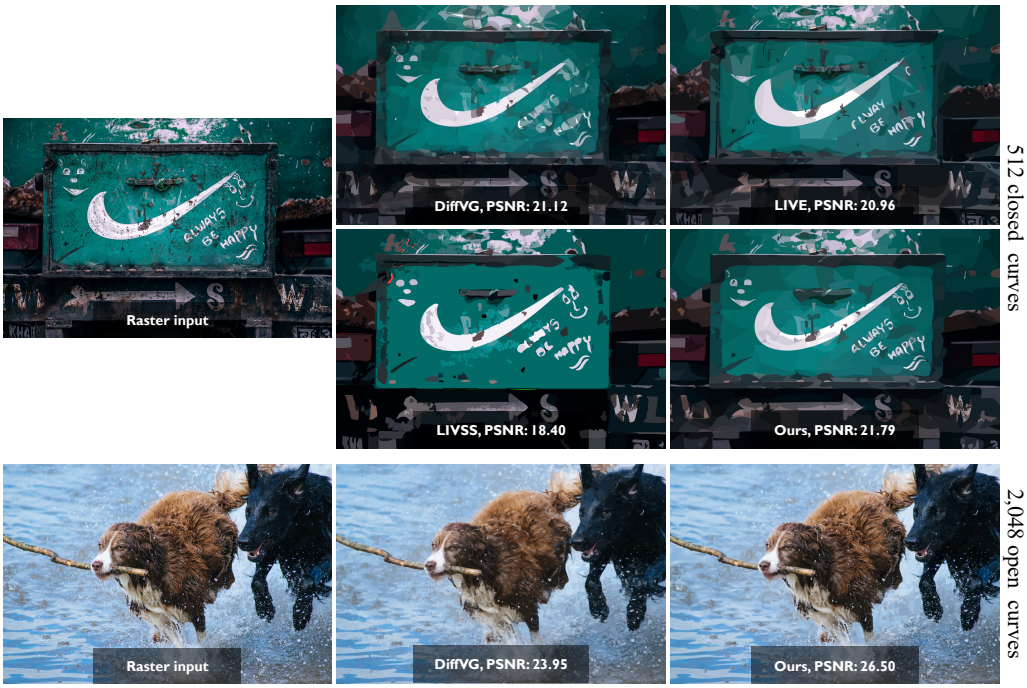

Figure 3: A qualitative comparison of our method and the state-of-the-art differentiable VG rasterization methods, including DiffVG [15], LIVE [18], and LIVSS [27].

additional information and can be pruned without affecting the final representation. By removing such curves, we allow the optimization to introduce more appropriate replacements, leading to a more accurate and visually coherent vectorized representation. For curve densification, we adopt an error-driven curve allocation strategy inspired by LIVE [18]. Specifically, we compute connected error regions, rank them by area, and add new Bézier curves into the highest-error regions. This adaptive redistribution mechanism ensures that curves are allocated to where they would contribute the most to the reconstruction fidelity. The pruning and densification strategy enables a "global receptive field" for redistributing curve density, effectively preserving visual details for high-fidelity rendering while maintaining a compact vector representation.

4 Experiments

4.1 Experimental Setups

Implementation details. We implement Bézier Splatting in PyTorch [20] and optimize it by using the Adam optimizer [13] with a StepLR learning rate scheduler. The learning rate is initialized at 0.01 for color, 2e-4 for Bézier curve control points, and 0.1 for opacity. For the pruning and densification strategy, the opacity threshold is set to 0.02, and the overlap threshold based on Axis-Aligned Bounding Boxes (AABB) is set to 0.9. Following the approach in LIVE [18], new curves are initialized in a circular pattern, and the number of added curves matches the number of removed ones to maintain a constant total curve count. Open curves are optimized for 15,000 iterations, and closed curves are optimized for 10,000 iterations. Pruning and densification are applied every 400 steps until the last but 1,000 steps, after which they are halted to stabilize the representation.

Datasets. We comprehensively evaluate our method across different image domains. We use the publicly available DIV2K [24] dataset for evaluating natural images. Due to the high computational cost of baseline method LIVE [18], we uniformly subsample the DIV2K dataset by selecting one out of every four images, resulting in a final evaluation set of 200 images from the original 800-image

Table 2: Quantitative evaluation and optimization efficiency of differentiable VG methods on the DIV2K dataset [24].

	Method	256 curves				512 curves				1024 curves			
		$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Opt.$^{\downarrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Opt.$^{\downarrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Opt.$^{\downarrow}$
Open	DiffVG [15]	0.552	19.83	0.563	18.9min	0.587	21.47	0.537	22.0min	0.616	22.62	0.517	30.6min
	Ours	**0.600**	**22.17**	**0.540**	**3.4min**	**0.646**	**23.79**	**0.498**	**3.3min**	**0.699**	**25.45**	**0.448**	**3.2min**
Closed	DiffVG [15]	0.578	20.69	0.548	16.4min	0.601	21.82	0.531	18.5min	0.631	22.95	0.509	25.1min
	LIVE [18]	0.576	20.09	0.543	2.6h	0.611	21.70	**0.521**	4.2h	0.648	23.11	0.495	5.1h
	LIVSS [27]	**0.586**	17.71	**0.542**	39.2min	**0.630**	18.71	0.530	54.3min	**0.678**	19.83	0.517	1.4h
	Ours	0.580	**20.74**	0.546	**7.8min**	0.607	**22.11**	0.528	**8.3min**	0.639	**23.45**	**0.507**	**8.6min**

Figure 4: Our Bézier Splatting achieves high-quality image vectorization results for various types of images including artworks, cartoons, and natural images. Curve type and count are indicated at the bottom right of each sample.

dataset. Additionally, we test our method on non-photorealistic images, including the artwork images from Clipart1K dataset [10] and cartoon images from Danbooregions dataset [32], to demonstrate its effectiveness for diverse types of images, as shown in Fig. 4.

4.2 Method Comparison

Table 1 reports the forward and backward runtime for processing 2,048 curves on an image with a resolution of 2,040×1,344. Compared to DiffVG [15], our method significantly accelerates VG rasterization by 31.4× faster per forward step and 149.2× faster per backward step for open curves. For closed curves, our color filling strategy requires sampling 20 additional Bézier curves per closed curve, while our method remains highly efficient, achieving 6× faster forward and 18.2× faster backward computation.

Table 2 quantitatively evaluates the quality of differentiable VG representations by three commonly used metrics, MS-SSIM [26], PSNR, and LPIPS [34]. Our method demonstrates higher optimization efficiency and rendering fidelity for both open and closed curves with different curve numbers.

Fig. 3 visually compares DiffVG [15], LIVE [4], [27], and our Bézier Splatting under 512 closed curves and 2048 open curves. Compared to DiffVG [15], our method captures significantly more fine-grained textures. Compared to LIVE [18], our method effectively enhances rendering quality, resulting in higher-fidelity results. This improvement stems from our method's ability to globally optimize all curves, rather than LIVE's layer-by-layer path-adding strategy. Compared to LIVSS [27], our method demonstrates superior ability in preserving fine structural details, especially in text regions. Due to its heavy reliance on layer-wise semantic simplification, LIVSS struggles to optimize regions where semantic information is ambiguous or difficult to extract. Moreover, by optimizing the entire VG simultaneously, our approach ensures a cohesive and natural appearance, avoiding the accumulation of errors and inconsistencies introduced by layer-wise updates.

As shown in Fig. 4, Bézier Splatting consistently renders high-quality VGs across various image domains from photorealistic natural images, watercolor paintings, to cartoon images. Unlike learning-based image vectorization methods [16, 21, 2], which can not easily generalize to out-of-domain data, our method can flexibly handle different types of images.

4.3 Layer-wise Image Vectorization

Existing works such as LIVE [18], SGLIVE [37] and LIVSS [27] have demonstrated that layer-wise vectorization strategies can significantly improve the topology and compositionality of DiffVG-

Figure 5: As a differentiable VGs renderer, Bézier Splatting can integrate with topology-aware strategies such as layer-wise vectorization [18] to further improve compositionality and detail.

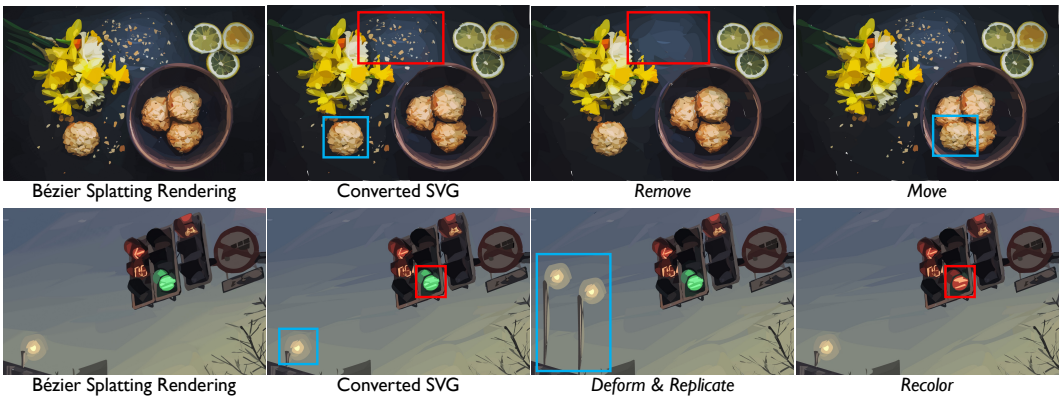

Figure 6: Our Bézier Splatting representation is compatible with the standard SVG XML format and supports flexible vector editing operations.

generated vector graphics [15]. Since the proposed Bézier Splatting is a fast and differentiable VGs renderer alternative to DiffVG, is naturally compatible with these topology-aware strategies and can leverage them to further enhance the structural quality of the optimized vector outputs. As shown in Fig. 5, we incorporate the layer-wise vectorization approach from LIVE [18] into the Bézier Splatting optimization process This integration yields improved compositional integrity and finer details for regions with complex patterns such as the butterfly's wings.

4.4 Editability and Compatibility with SVG XML Format

Our Bézier Splatting representation is fully compatible with the standard SVG XML format, ensuring the same editability as standard SVGs and seamless integration with existing SVG tools. As shown in Fig. 6, a high-fidelity SVG can be obtained via a simple conversion algorithm (see Appendix for details). Once converted to SVG, the vectorized output is fully editable at the primitive level, allowing users to freely deform, remove, move, replicate, and recolor individual elements.

5 Conclusion

This work has presented Bézier Splatting, a novel differentiable vector graphics (VGs) representation that leverages Gaussian splatting for efficient Bézier curve optimization. Our method achieves 30× faster forward computation and 150× faster backward computation in rasterization compared to existing methods, while also delivering high rendering fidelity. Additionally, our adaptive pruning and densification strategy improves optimization by dynamically adjusting curve placement during optimization. Extensive experiments have demonstrated that Bézier Splatting outperforms existing differentiable VG methods in both training efficiency and visual quality, making it a promising solution for scalable applications of VGs.

Limitation and future work. The closed curves in Bézier Splatting are enforced to have convex shapes using Xing loss [18] to prevent false interpolation results, which may slightly reduce the capacity of VG representations. Additionally, the closed curves require sampling more Gaussian points to represent area boundaries precisely, leading to slower computation compared to open curves. An interesting future direction is to leverage the efficiency and differentiability of our approach for high-quality VG synthesis applications such as text-to-VG generation or text-guided VG editing.

6 Acknowledgement

This work was supported in part by the National Science Foundation (NSF) and SC EPSCoR Program under award number #OIA-2242812. This research used in part resources on the Palmetto Cluster at Clemson University under NSF awards MRI 1228312, II NEW 1405767, MRI 1725573, and MRI 2018069. The views expressed in this article do not necessarily represent the views of NSF or the United States government. The authors also thank Dr. Yiqi Zhong and Dr. Luming Liang for beneficial discussions.

References

[1] Adobe Inc. Adobe illustrator. https://www.adobe.com/products/illustrator.html, 2025. 14

[2] Defu Cao, Zhaowen Wang, Jose Echevarria, and Yan Liu. Svgformer: Representation learning for continuous vector graphics using transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10093–10102, 2023. 2, 8

[3] Ye Chen, Bingbing Ni, Jinfan Liu, Xiaoyang Huang, and Xuanhong Chen. Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15877–15886, 2024. 2, 3

[4] Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. Image vectorization and editing via linear gradient layer decomposition. *ACM Transactions on Graphics (TOG)*, 42 (4), 2023. 2, 8

[5] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 3

[6] Minghao Guo, Bohan Wang, Kaiming He, and Wojciech Matusik. Tetsphere splatting: Representing high-quality geometry with lagrangian volumetric meshes. *arXiv preprint arXiv:2405.20283*, 2024. 3

[7] Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: a top-down approach for image vectorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2148–2156, 2024. 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[9] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 3, 4

[10] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5001–5009, 2018. 8

[11] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2023. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 5, 6

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. 7

[14] Kodak. Kodak lossless true color image suite, 1999. Accessed: 2025-03-07. 17, 24

[15] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 1, 2, 3, 4, 5, 7, 8, 9

[16] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939, 2019. 2, 8

[17] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 3

[18] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2022. 1, 2, 3, 5, 6, 7, 8, 9, 13, 17

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15, 2022. 3

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 7

[21] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. 2, 8

[22] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 3

[23] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[24] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 7, 8, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[26] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multi-scale structural similarity for image quality assessment. *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 2:1398–1402, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers ; Conference date: 09-11-2003 Through 12-11-2003. 8

[27] Zhenyu Wang, Jianxi Huang, Zhida Sun, Yuanhao Gong, Daniel Cohen-Or, and Min Lu. Layered image vectorization via semantic simplification. *arXiv preprint arXiv:2406.05404*, 2024. 7, 8

[28] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 3

[29] Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems*, 36:15869–15889, 2023. 2

[30] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4546–4555, 2024. 2

[31] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *CVPR*, 2024. 3

[32] Lvmin Zhang, Yi JI, and Chunping Liu. Danbooregion: An illustration region dataset. In *European Conference on Computer Vision (ECCV)*, 2020. 8, 17, 25

[33] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 2

[34] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8

[35] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, 2024. 3, 5

[36] Yunxiang Zhang, Alexandr Kuznetsov, Akshay Jindal, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. Image-gs: Content-adaptive image representation via 2d gaussians, 2024. 3

[37] Hengyu Zhou, Hui Zhang, and Bin Wang. Segmentation-guided layer-wise image vectorization with gradient fills. In *European Conference on Computer Vision*, pages 165–180. Springer, 2024. 8

[38] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Visualization, 2001. VIS 01. Proceedings*, pages 29–538. IEEE, 2001. 2

A Ablation Study

The effectiveness of adaptive pruning and densification and layer-wise vectorization strategy. We conduct an ablation study to evaluate the effectiveness of our adaptive pruning and densification strategy. Quantitative results are shown in Table 3. To further investigate the flexibility of our framework, we incorporate a layer-wise curve addition strategy inspired by LIVE [18], where curves are progressively added during training. Although this strategy achieves slightly better reconstruction metrics, it doubles the training time compared to our adaptive pruning and densification approach, making it less favorable for time-sensitive applications. Fig. 7 shows our result with layer-wise image vectorization strategy.

Table 3: An ablation study on adaptive pruning and densification and layer-wise training strategy, evaluated on 512 closed curves with DIV2K dataset [24].

Method	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	$Opt^{\downarrow}$
No Strategy	0.590	21.10	0.530	**8.3 min**
– w/ Prune&Densify	0.607	22.11	0.528	8.3 min
– w/ Layer-wise	**0.613**	**22.21**	**0.521**	16.2 min

Figure 7: Our Bézier Splatting is fully compatible with the layer-wise vectorization strategies [18].

Comparing different numbers of curves. We progressively increase the number of curves for vectorizing a watercolor image (Fig. 8) with numerous small spots. As the number of curves grows, our approach first reconstructs the foreground object, the bird, then gradually refines the smaller spots. This demonstrates that our method prioritizes the primary structure before optimizing finer details, effectively distributing curves to balance global structure and local texture representation, thanks to our adaptive pruning and densification strategy.

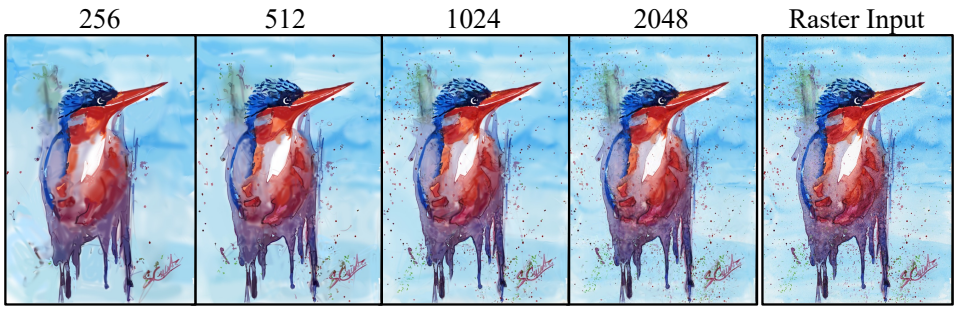

Figure 8: A comparison of different curve numbers by Bézier Splatting.

A systematic evaluation of computation time. To systematically evaluate the computational efficiency of our framework, we report detailed timing statistics across varying configurations of

Table 4: A systematic evaluation of computation time. We test the forward and backward time (ms) under varying numbers of sampled curves within one closed Bézier curve (Inter. #), and the total number of Bézier curves (Curve #).

Inter. #	Curve #	Forward w/ gradient	Sample Gaussians	Splatting	Backward	Render (FPS)
20	2048	10.66	4.12	6.11	21.64	103.45
40	2048	14.79	5.11	9.32	25.06	68.30
80	2048	27.01	8.22	17.74	33.40	37.90
40	256	7.20	3.81	3.09	16.72	150.30
40	512	7.70	3.88	3.67	17.42	144.30
40	1024	10.39	4.06	5.96	19.96	114.30
40	2048	14.79	5.11	9.32	25.06	68.30

interpolation curve numbers and curve resolutions, as summarized in Table 4. The total forward time includes two major parts: the Gaussian sampling and the Gaussian splatting. Increasing the number of interpolated curves within the closed Bézier curves leads to a near-linear growth in both forward and backward runtimes, demonstrating good computational scalability. When fixing the interpolation number (*e.g.*, at 40) and varying the number of Bézier curves, the forward time increases moderately, reflecting the additional cost of handling finer spatial detail. The splatting stage dominates the forward time at higher number of curves due to the increased number of sampled points per region, while the Gaussian sampling time is the primary bottleneck at lower number of curves. This is attributed to overheads from sequential memory allocation and data transfer operations, which are not effectively parallelized on the GPU, suggesting potential for further optimization.

Table 5: A quantitative comparison between Adobe Image Trace [1] and our method across different images.

Image ID	0004		0008		0012		0016		0020	
Method	Adobe [1]	Ours	Adobe [1]	Ours	Adobe [1]	Ours	Adobe [1]	Ours	Adobe [1]	Ours
$SSIM^{\uparrow}$	0.835	0.849	0.645	0.637	0.658	0.658	0.627	0.628	0.814	0.822
$PSNR^{\uparrow}$	26.43	28.62	23.26	24.33	22.29	22.87	23.87	24.54	26.67	28.31
$LPIPS^{\uparrow}$	0.380	0.387	0.489	0.489	0.523	0.518	0.534	0.543	0.505	0.503

Comparison with conventional image vectorization methods. We conduct a comparison against the Image Trace from Adobe Illustrator [1] on five images from the DIV2K dataset. Both methods are evaluated using the same number of parameters (around 20K). As shown in Table 5, our method achieves higher fidelity, with an average PSNR of 25.734 db compared to 24.904 db of Image Trace.

Arbitrary resolution rendering. Our Bézier Splatting representation naturally supports rendering at arbitrary resolutions because all Gaussian parameters are analytically derived from the underlying Bézier curves. When higher-resolution images are required, proportionally increasing the sampling rate is sufficient to maintain reconstruction fidelity. For example, doubling the resolution simply doubles the sampling density. As shown in Table 6, rendering at higher resolutions (4K or 8K) with increased sampling preserves a comparable level of visual quality to the 2K baseline, without introducing noticeable artifacts. We report results on the first four images from the [24] to illustrate this property.

Table 6: Results of Bézier Splatting rendered at different resolutions and sampling settings. Notation: "orig." denotes the baseline sampling rate used during optimization, and "×S" indicates S-times higher sampling density.

Image ID	**2K (orig.)**	**4K**		**8K**	
		orig.	**(×2S)**	**orig.**	**(×4S)**
0004	26.8976	27.1553	27.2761	25.9800	26.6594
0008	21.2063	21.0614	21.4340	19.8318	21.1214
0012	19.8816	19.9079	20.0900	18.9389	19.7760
0016	22.3456	22.4327	22.5408	21.7088	22.3140

Algorithm 1 Adaptive Pruning and Densification for Closed Bézier Curves

Notation:
$\mathbf{opacity}(b)$ — mean opacity of curve b.
$\mathbf{area}(b)$ — area enclosed by curve b.
$\mathbf{colordiff}(b_i, b_j)$ — Euclidean color difference between curves b_i and b_j.
$\mathbf{IoU}(b_i)$ — intersection-over-union between the bounding box of b_i and neighboring curves with $\mathbf{colordiff} < 0.03$.
$\mathbf{ConnectedComponents}(E)$ — connected error regions extracted from the quantized error map.

Input : Closed Bézier curve set $\mathcal{B} = \{b_1, \ldots, b_N\}$; opacity map α; error map E; iteration index t.
Output: Updated curve set $\mathcal{B}'$.
Initialize $\mathcal{B}' \leftarrow \mathcal{B}$.
Pruning phase
for $i \leftarrow 1$ **to** N **do**
  if $\mathbf{opacity}(b_i) < \tau_{\mathbf{opacity}}(t)$ **then**
    Remove b_i from $\mathcal{B}'$. {Low-opacity pruning}
  end if
  if $\mathbf{area}(b_i) < \tau_{\mathbf{area}}$ **then**
    Remove b_i from $\mathcal{B}'$. {Small-area pruning}
  end if
  if $\mathbf{IoU}\big(b_i, \{\, b_j \in \mathcal{B} \mid \mathbf{colordiff}(b_i, b_j) < 0.03 \,\}\big) > 0.9$ **then**
    Remove b_i from $\mathcal{B}'$. {Redundant-overlap pruning}
  end if
end for
Densification phase
$\mathcal{R} \leftarrow \mathbf{ConnectedComponents}(E)$.
$\mathcal{R}_{\mathbf{sorted}} \leftarrow \mathbf{SortByArea}(\mathcal{R})$.
for each region $r_j \in \mathcal{R}_{\mathbf{sorted}}$ **do**
  if curve budget allows **then**
    Insert a new closed Bézier curve into r_j.
  end if
end for
return $\mathcal{B}'$.

B Details of Adaptive Pruning and Densification

To provide a clearer understanding of our optimization behavior, we present here the detailed procedure of the pruning–densification algorithm 1 used in our Bézier-splatting framework.This algorithm is designed to mitigate the local-minima issue observed in 3D Gaussian Splatting (3DGS) and is adapted in our Bézier-splatting framework, where suboptimal primitive initialization leaves certain regions insufficiently covered. Inspired by the pruning–splitting mechanism of 3DGS [1], our method dynamically reallocates Bézier curves according to the reconstruction error map: curves with low opacity, small area, or high overlap with nearby curves of similar color are pruned, while new curves are inserted into regions with high reconstruction error to improve coverage. This reallocation keeps the total curve budget fixed, enhances reconstruction quality, and helps the optimization escape poor local minima. For open curves, we additionally introduce a splitting rule: if the middle opacity of a curve is more than 0.5 lower than that of both endpoints, the curve is split into two segments to better adapt to local structure variations.

C Convert Bézier Splatting to Standard SVG Format

As described in Algorithm 2, we convert our optimized Bézier Splatting representation into a standard SVG file for compatibility with downstream vector editing tools. Each curve is represented by a set of $3k + 1$ control points $\mathcal{C}_i \in \mathbb{R}^{(3k+1)\times 2}$, corresponding to k continuous cubic Bézier segments. These control points are normalized to the range $[-1, 1]$ and are first transformed into pixel coordinates according to the canvas size (W, H).

For each curve i, we initialize the SVG path string d with a `[M` P_i^0`]` command. We then iteratively construct each cubic segment using three internal control points: $(p_1, p_2, p_3) = (P_i^{3j+1}, P_i^{3j+2}, P_i^{3j+3})$, and append the segment as a path command `[C` p_1`,` p_2`,` p_3`]` to d. After all segments are processed, a `[Z]` is appended to close the path.

Each path d is then associated with a fill color (r, g, b), computed via sigmoid activation on the optimized feature vector $\mathcal{F}_i$, and opacity 1.0. The result is stored in a global path set $\mathcal{S}$, and all paths are assembled into the final SVG file S. This output is directly compatible with standard vector graphics tools such as Adobe Illustrator.

Algorithm 2 Convert Bézier Splatting to Standard SVG

Input: Optimized control points $\mathcal{C} \in \mathbb{R}^{N\times(3k+1)\times 2}$, feature colors $\mathcal{F} \in \mathbb{R}^{N\times 3}$, canvas size (W, H)
Output: SVG file S containing cubic Bézier paths
for each curve $i = 1$ to N **do**
  $P_i \leftarrow \frac{\mathcal{C}_i+1}{2} \cdot (W, H)$
  $(r, g, b) \leftarrow \text{sigmoid}(\mathcal{F}_i) \times 255$
  Initialize path $d \leftarrow$ `[M` P_i^0`]`
  for each segment $j = 1$ to k **do**
    $(p_1, p_2, p_3) \leftarrow (P_i^{3j+1}, P_i^{3j+2}, P_i^{3j+3})$
    $d \oplus$ `[C` p_1`,` p_2`,` p_3`]`
  end for
  $d \oplus$ `Z`
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{path}(d, \text{ fill} = (r, g, b), \text{ opacity} = 1.0)\}$
end for
Return: SVG file assembled from all paths

D Bézier Splatting Supports Flexible Curve Attributes

All 2D Gaussians in our method are generated via a differentiable sampling algorithm, which makes it straightforward to incorporate user-defined shape attributes, such as linear-gradient fills in color or opacity. To demonstrate the extensibility of our model, we evaluate the multi-opacity scheme for open curves: each Bézier segment within an open curve is assigned an independent opacity value. This design enables users to either maintain consistent sampling point appearance across segments or apply customized interpolation strategies. This strategy significantly improves vectorization quality for open curves, as shown in Table 7. Since closed curves are the primary representation format in SVG, we adopt a uniform setting for opacity and color in those regions to ensure a full compatibility with existing vector graphic tools. Nonetheless, users can still follow the above multi-opacity scheme to implement linear-gradient fills for both opacity and color, enhancing the flexibility and expressiveness of vector graphics.

Table 7: An ablation study on the number of opacity parameters per curve, evaluated on 512 open curves with DIV2K dataset [24].

Method	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$
3-opacity	**0.65**	**23.79**	**0.50**
1-opacity	0.63	22.97	0.51

E More Comparisons

We present more qualitative comparisons on the DIV2K [24] dataset. As shown in Fig. 9, Fig. 10, Fig. 11, Fig. 12, and Fig. 13, our method shows consistent improvements over the baselines, including better preservation of fine details, higher rendering fidelity, fewer visual artifacts, and more accurate geometric structures. Furthermore, as the number of curves increases, our method demonstrates a significantly improved representational ability, enabling more precise reconstruction of complex shapes, and fine-grained structures.

We further evaluate our method on another natural image dataset, Kodak [14]. Due to the slow vectorization speed of LIVE [18], we compare with DiffVG on open curves only. As shown in Table 8, our method consistently outperforms DiffVG across quantitative metrics including PSNR, SSIM, and LPIPS.

Table 8: A quantitative evaluation on the Kodak dataset [14] with 256 to 1024 curves. We report results on open curves.

	Method	**256**			**512**			**1024**		
		$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$
Open	DiffVG	0.601	23.43	0.535	0.645	24.70	0.495	0.699	26.04	0.439
	Ours	**0.679**	**26.18**	**0.457**	**0.743**	**27.90**	**0.383**	**0.797**	**29.24**	**0.310**
Closed	DiffVG	**0.622**	24.11	**0.513**	**0.666**	25.34	0.475	**0.719**	26.66	**0.420**
	Ours	0.621	**24.19**	0.519	0.664	**25.61**	0.485	0.708	**26.91**	0.448

F More Image Vectorization Results

Fig. 14, Fig. 15, and Fig. 16 show more results on natural images from DIV2K [24] and Kodak [14], as well as animation images from DanbooRegion [32], respectively. The results demonstrate that both the global structure and local texture of the images are well reconstructed, highlighting the effectiveness of our approach in capturing fine details and complex shapes.

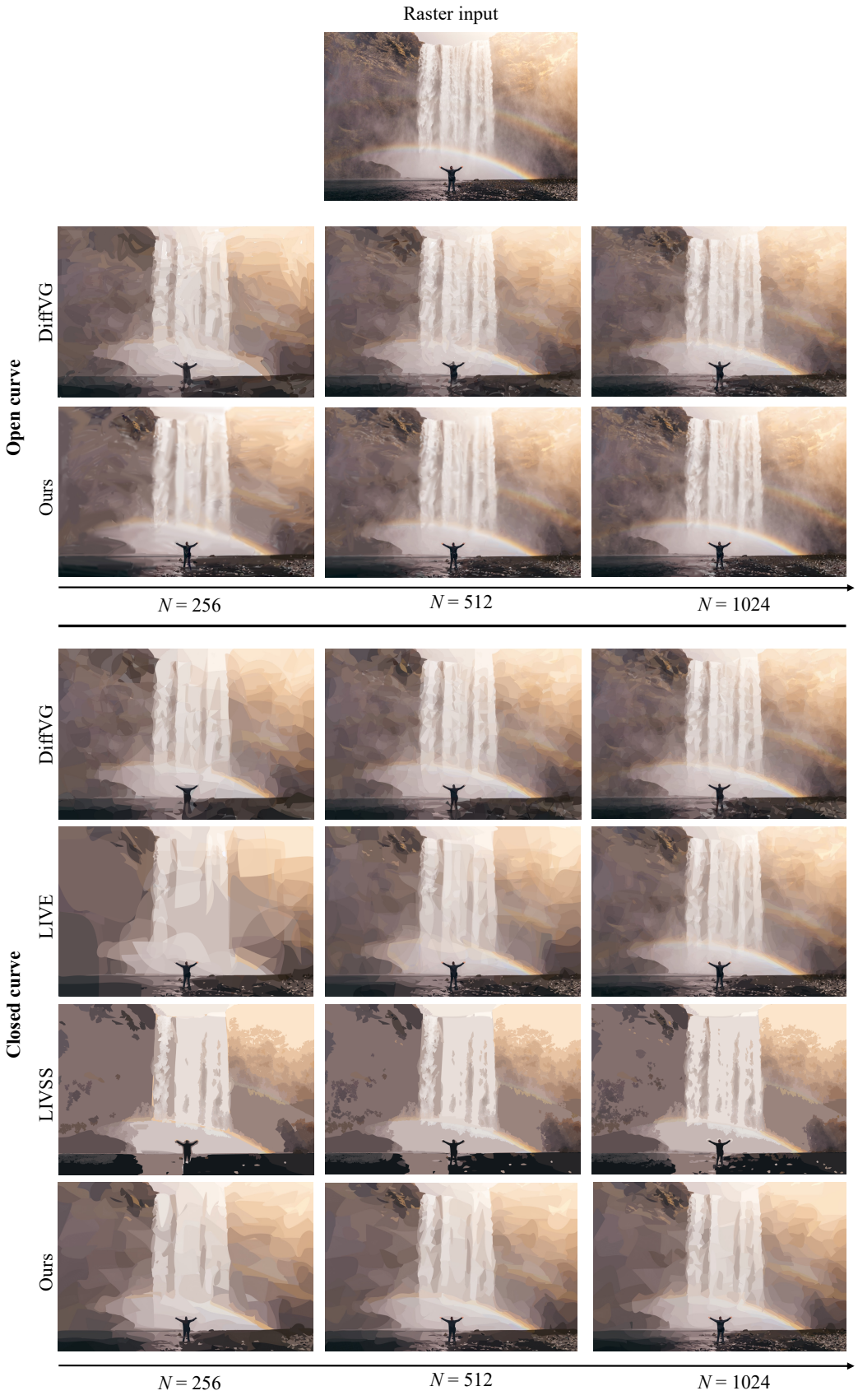

Figure 9: A qualitative comparison of our method and the existing differentiable VG rasterization method on DIV2K dataset [24].

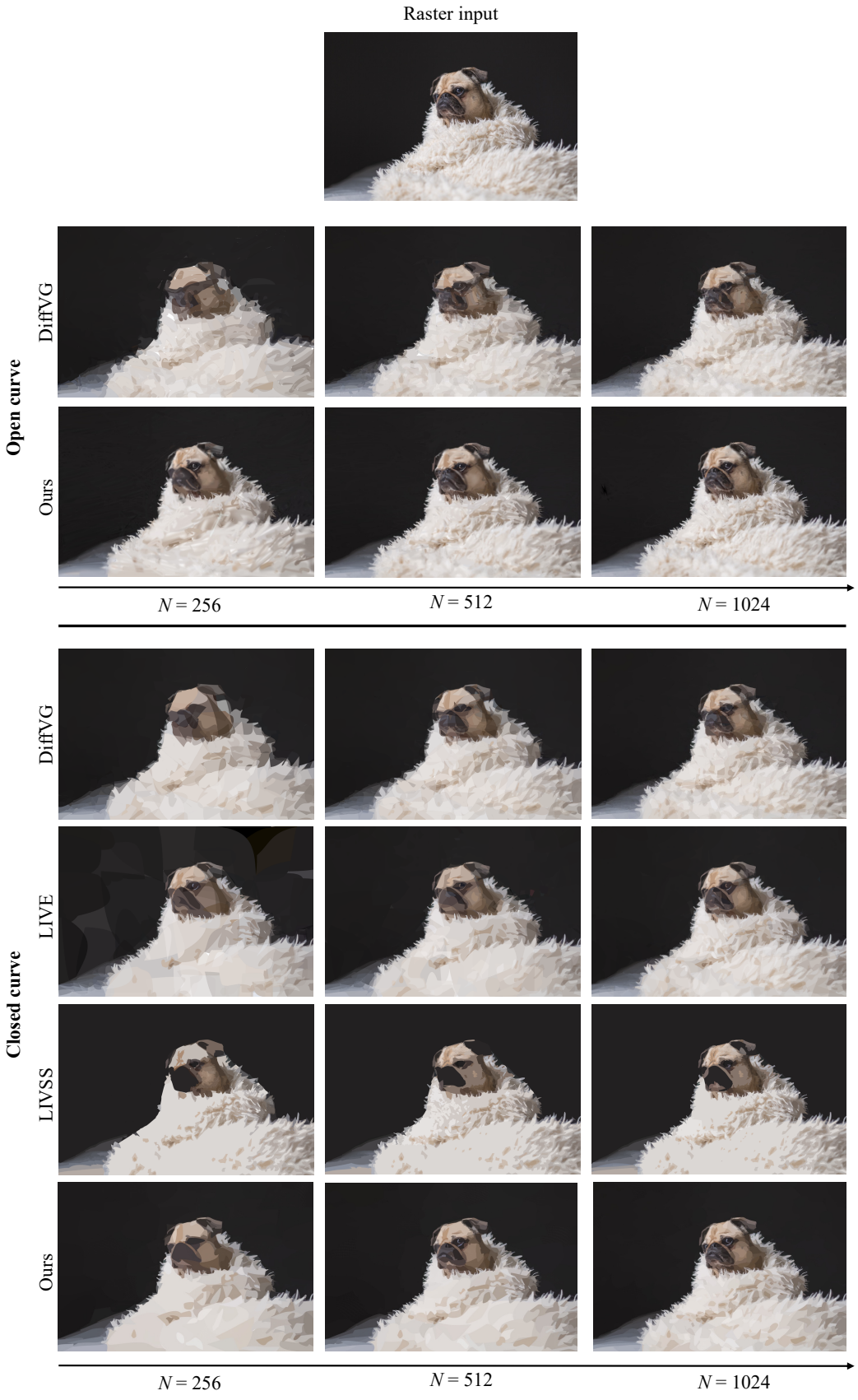

Figure 10: More qualitative comparisons of our method and the existing differentiable VG rasterization method on DIV2K dataset [24].

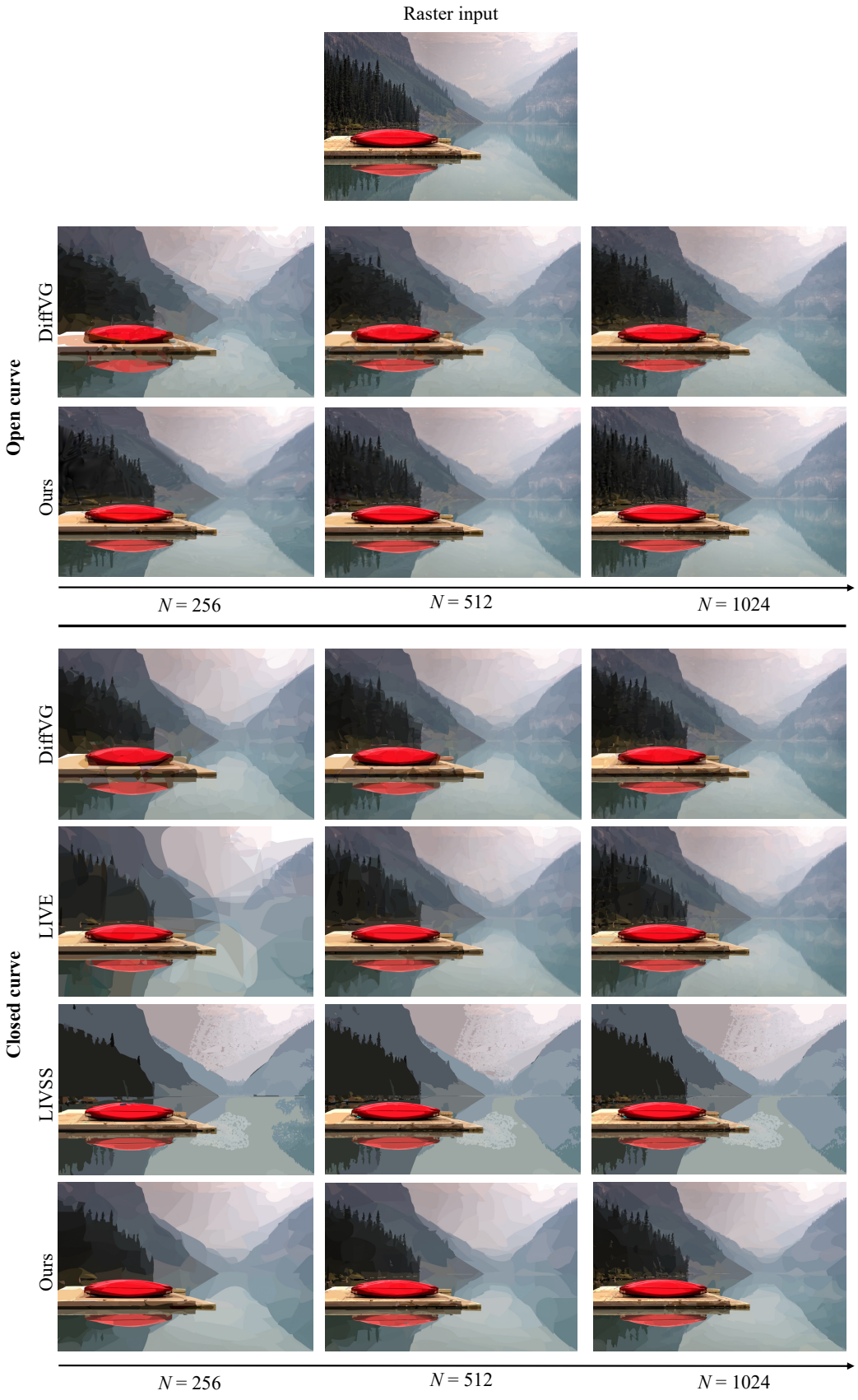

Figure 11: More qualitative comparisons of our method and the existing differentiable VG rasterization method on DIV2K dataset [24].

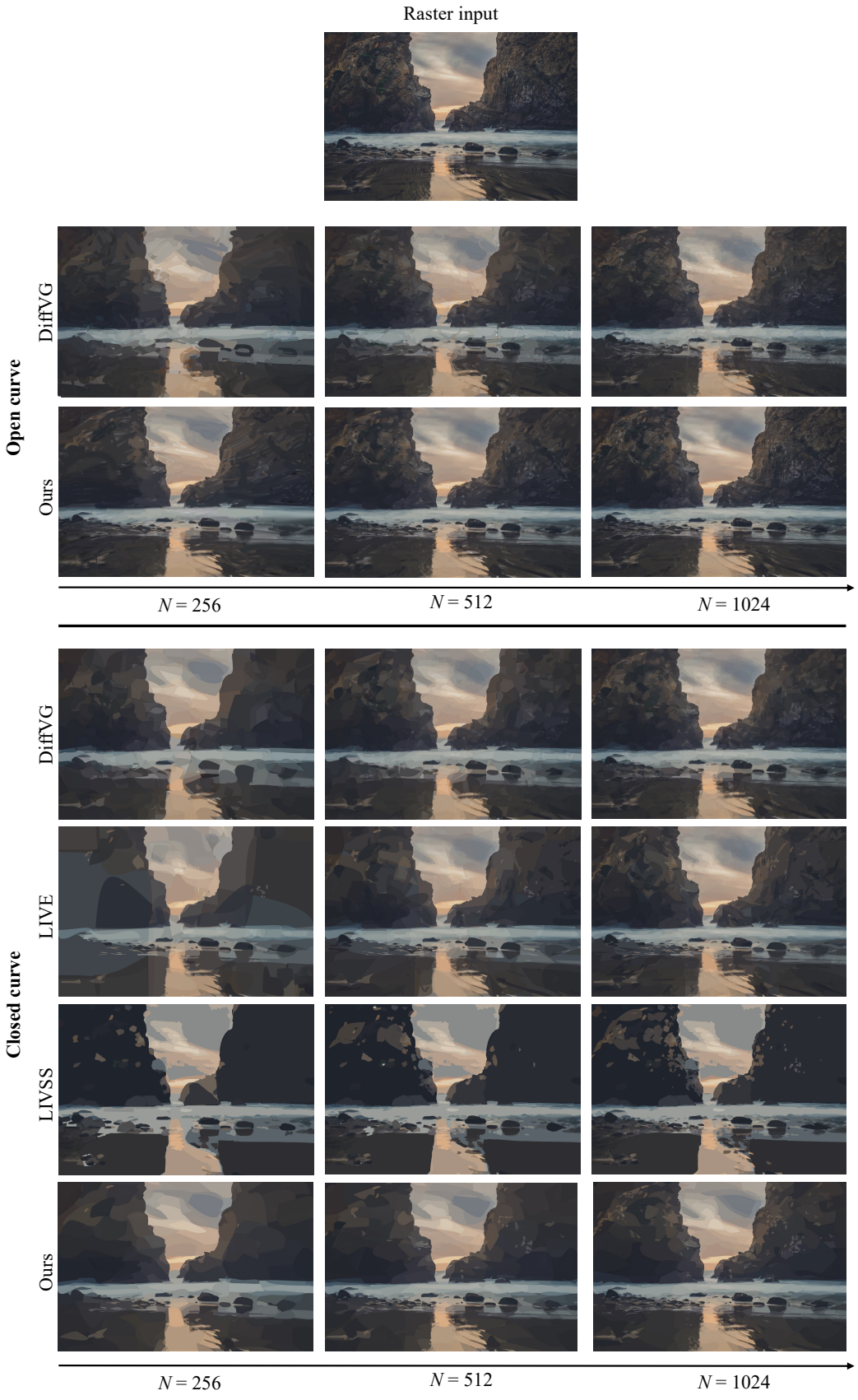

Figure 12: More qualitative comparisons of our method and the existing differentiable VG rasterization method on DIV2K dataset [24].

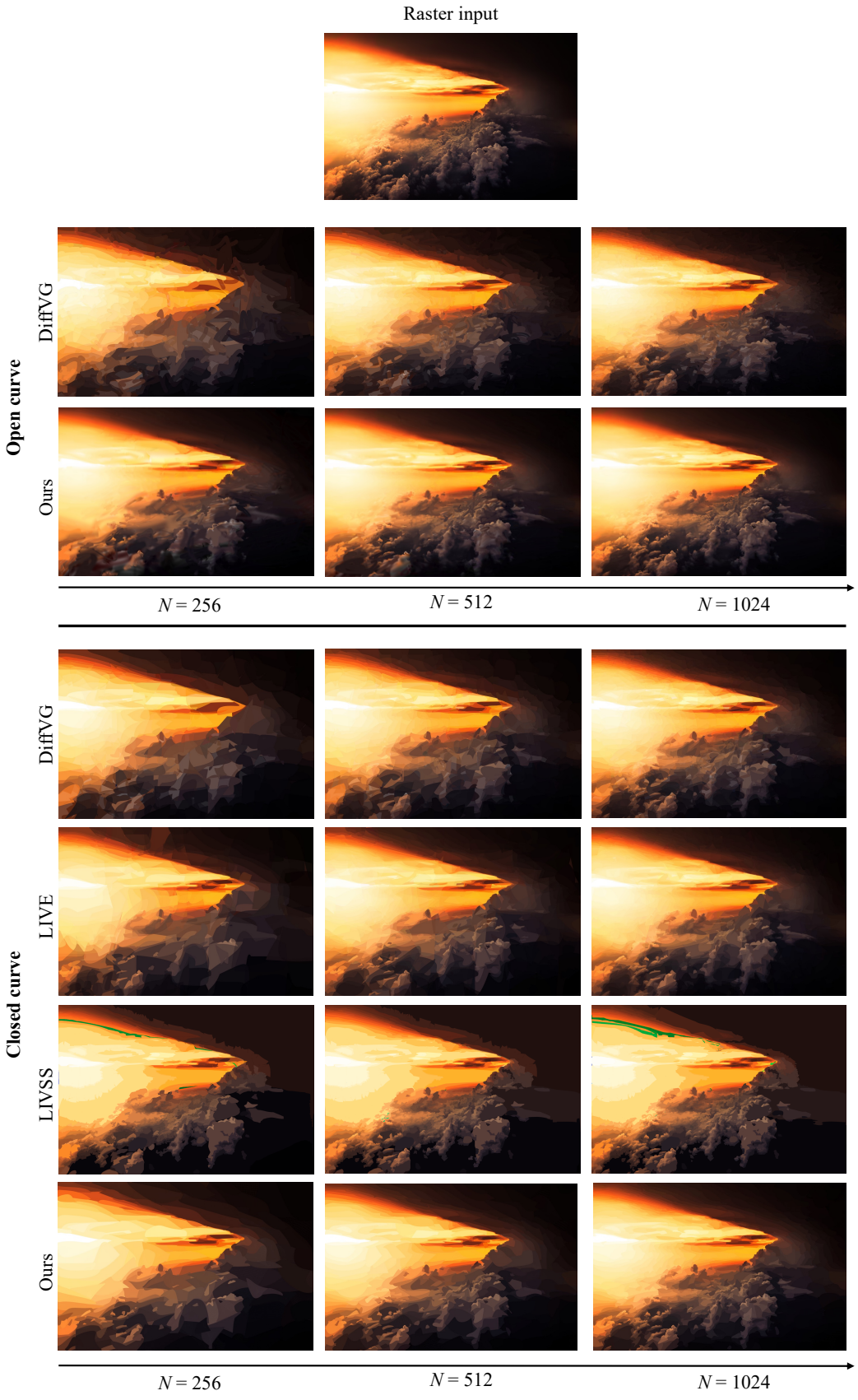

Figure 13: More qualitative comparisons of our method and the existing differentiable VG rasterization method on DIV2K dataset [24].

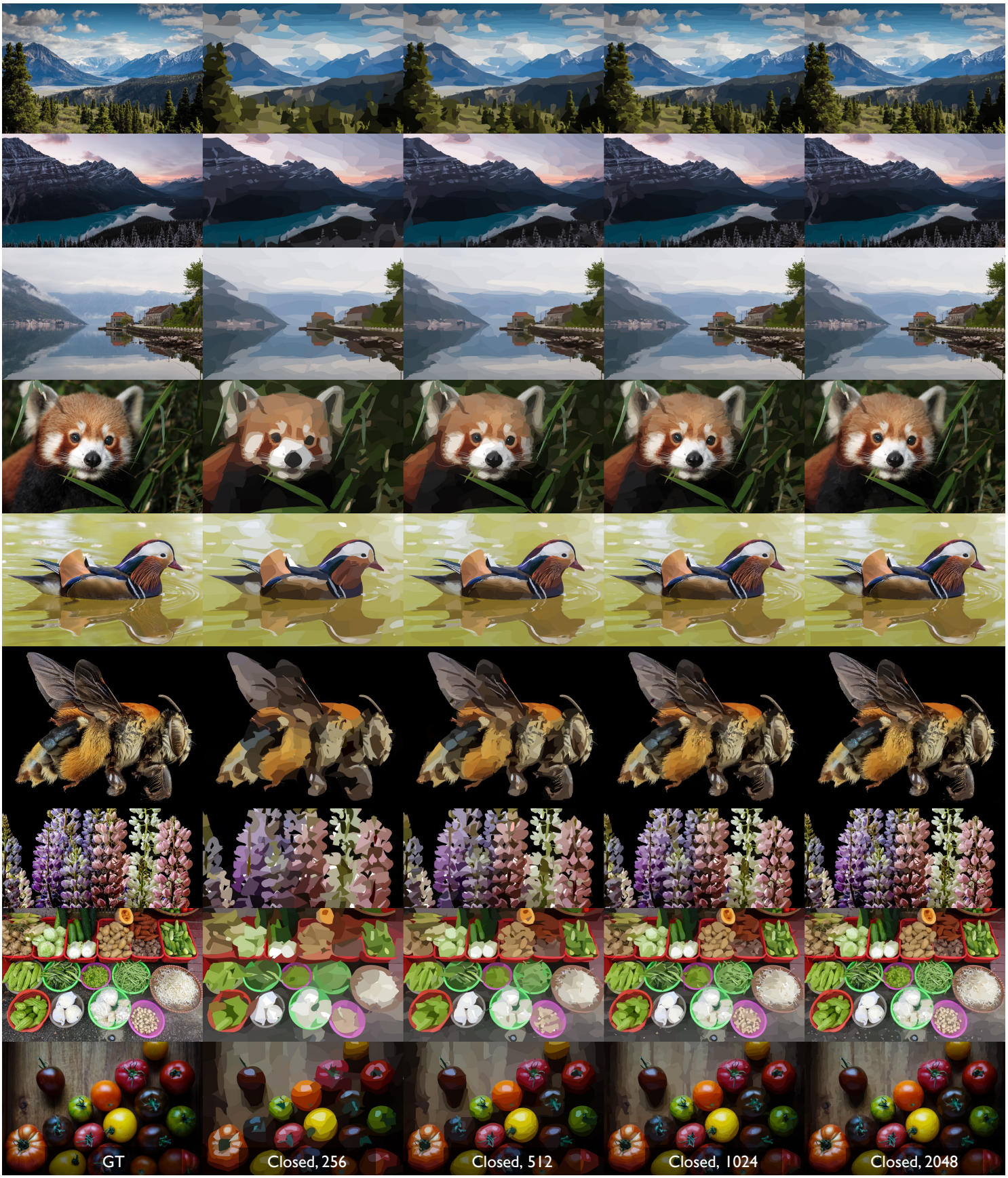

Figure 14: More image vectorization results by Bézier Splatting on DIV2K dataset [24].

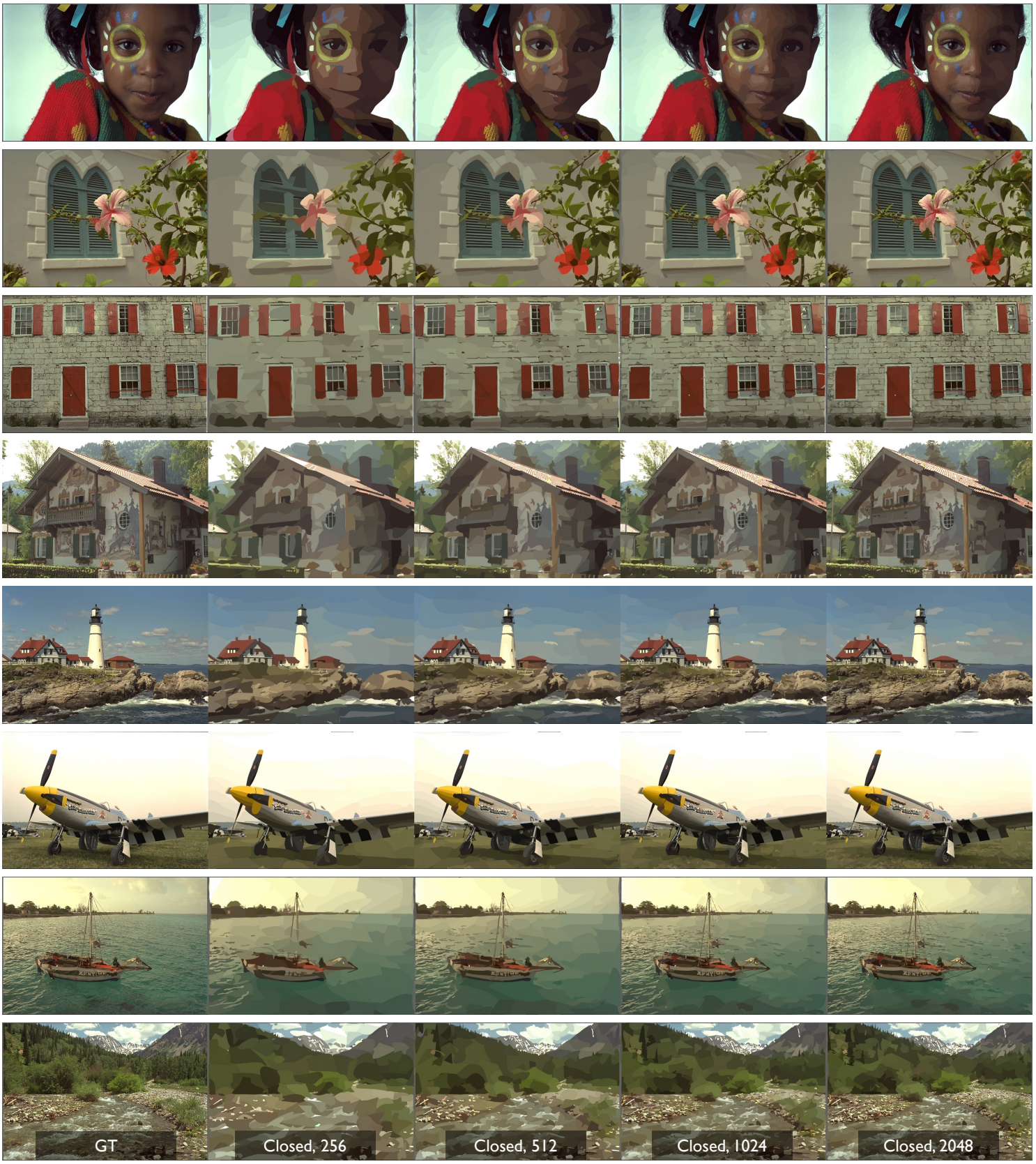

Figure 15: More image vectorization results by Bézier Splatting on Kodak [14] dataset.

Figure 16: More image vectorization results by Bézier Splatting on DanbooRegion dataset [32].

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We ensured that the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussions on limitations of this work can be found in the last section of the main paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have discussed all implementation details of the proposed method in Section 4 and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The project page has been online. All datasets used in this work are publicly available. The code will be publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have discussed all implementation details of the proposed method in Section 4 and Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Existing literature did not provide error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details of compute resources can be found in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have ensured that this research followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the societal impacts in the last section of main paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper does not have such risks for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the original papers that produced the code package and datasets. The licenses for re-packaged datasets will be provided on the dataset website.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.