SemLayer: Semantic-aware Generative Segmentation and Layer Construction for Abstract Icons

Supplementary Material

A. Implementation Details

A.1. Segmentation Post-processing

Raw segmentation results can be influenced by anti-aliasing artifacts already present in the training data, which arise when the original segmentations are rasterized. Because supervision occurs on a pixel grid, boundaries between adjacent regions contain mixed or partially transparent pixels, making the interfaces inherently ambiguous. As a result, our predictions may show thin black gaps between segments, and boundary pixels may carry colors that are not strictly consistent with either side. To obtain clean, fully assigned regions, and to ensure positional consistency between the input mask and our prediction, since slight spatial shifts can occur—we treat all black pixels as unlabeled queries and assign each one the color of its nearest labeled pixel in the predicted map. This nearest-neighbor retrieval produces crisp, piecewise-constant regions with perfectly abutting boundaries, making them more robust to small geometric misalignments.

A.2. Completion Post-processing

Raw completion outputs may contain thin artifacts or small isolated noises near completion boundaries. We address these imperfections with a lightweight post-processing pipeline. First, we identify valid completion regions using a distance-based heuristic. We then retain the largest connected component, remove boundary rings via distance-transform filtering, and enforce topological consistency by preserving only regions connected to the expanded original mask. This yields clean, structurally coherent binary masks suitable for downstream vectorization.

A.3. Curve Reuse

A key advantage of working with parametric Bézier curves (instead of polygonal approximations) is that we can perform exact local edits while preserving the original high-quality geometry elsewhere. We formulate completion as a local curve surgery problem: maximally reuse the original curves and only replace the missing region with a small number of newly constructed bridge segments.

Given an original incomplete contour $\mathcal{C}_A$ and a completed proposal $\mathcal{C}_B$, we first detect their contact regions by adaptive sampling and nearest-neighbor distance tests. These regions indicate where the two contours overlap and where replacement should happen. We then perform parametric curve cutting at the boundaries of each contact region. Unlike polygon-based methods, Bézier curves can be split exactly at arbitrary parameters, allowing us to remove only the local overlapping segments while keeping the rest of the curve bitwise identical to the original.

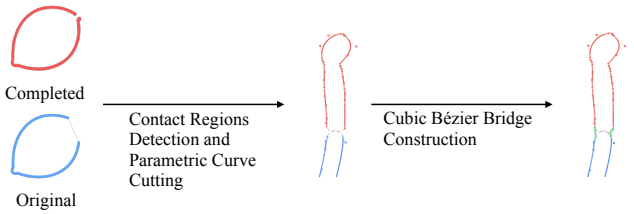

Figure 1. Illustration of our curve reuse and local surgery pipeline. We (1) detect the contact region between the original contour and the completed contour, (2) cut both curves locally, (3) keep the high-quality original segments (blue), (4) reuse the reliable part of the completion (red), and (5) connect them using newly constructed smooth bridge curves (green).

This produces two kept curve chains: (1) the preserved high-quality part of the original contour, and (2) the reliable part of the completion. To reconnect them, we construct cubic Bézier bridge curves that enforce G^1 continuity (matching position and tangent) at both ends. This ensures visually smooth transitions without kinks or cusps. Only these short bridge segments are newly created; all other geometry is reused verbatim.

Finally, we assemble the merged contour by choosing the correct traversal order (forward or reversed) based on endpoint proximity.

B. More Baselines

B.1. Generative vs. Simpler Baselines.

To validate the necessity of a generative approach, we implement classification and regression baselines adapted from the architecture of NIVEL [**?**]. As shown in Tab. 1, these simpler models achieve substantially lower performance than our generative pipeline, confirming that the strong shape priors learned by diffusion models are crucial for both semantic segmentation and amodal completion of abstract icons.

B.2. LLM-based Vector-Space Baselines

A natural question is whether semantic layer construction can be performed directly in the vector domain by leveraging recent large language models (LLMs). To investigate this, we provided raw SVG code as input to Gemini-3 and

Method	$\text{mIoU}_{\text{Seg.}}(\%) \uparrow$	$\text{mIoU}_{\text{Comp.}}(\%) \uparrow$
Simpler Baseline	46.6	63.9
Ours	**84.3**	**85.2**

Table 1. Comparison with simpler classification/regression baselines. Our generative approach significantly outperforms non-generative alternatives.

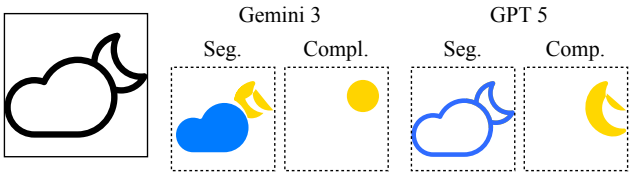

Figure 2. LLM-based vector-space baselines. Gemini-3 and GPT-5 fail to perform valid segmentation or completion when operating directly on SVG code.

GPT-5, prompting them to (1) assign semantic part labels to path groups (segmentation) and (2) complete occluded regions by generating new SVG paths (completion). As shown in Fig. 2, both models fail to produce valid results: segmentation outputs are semantically incoherent, and completion outputs contain structurally broken or nonsensical paths. We attribute this to the fact that flattened SVG code consists of compound, semantic-less paths that lack the visual grounding necessary for spatial reasoning, validating our pixel-based approach.

C. Runtime Analysis

Tab. 2 reports average per-icon runtime on a single NVIDIA A6000 GPU, evaluated over 48 test samples. Our segmentation model runs on par with SAM2 and 3.7× faster than gpt-image-1. For completion, our model is 3× faster than MP3D and over 22× faster than gpt-image-1.

Segmentation		**Completion**	
Method	Sec/icon ↓	Method	Sec/icon ↓
SAM2	14.32±3.72	MP3D	49.27±17.21
gpt-image-1	54.45±6.38	gpt-image-1	379.73±153.96
Ours	**14.56±0.30**	Ours	**16.83±6.32**

Table 2. Runtime comparison. Average seconds per icon on a single NVIDIA A6000 GPU (48 test samples).

D. Failure Cases

We present representative failure cases in Fig. 3. The first example illustrates failure under highly ambiguous and tightly tangled structures, where overlapping strokes make it difficult for the segmentation model to identify distinct semantic parts. The second example shows that our model struggles with pure generation: it fails to recover semantically expected yet entirely unseen structures (*e.g.*, invisible sun rays fully occluded by overlapping elements). These cases highlight the remaining challenges in handling extreme occlusion and structural ambiguity.

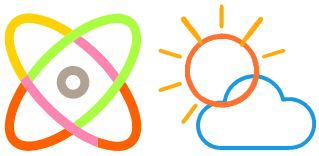

Figure 3. Representative failure cases. *Left:* tightly tangled structures cause segmentation errors. *Right:* the model fails to hallucinate entirely unseen structures (sun rays occluded by cloud).