# LAYERCANVAS: INFOGRAPHIC GENERATION IN AI LEARNING ASSISTANTS VIA HYBRID LAYER-SET DIFFUSION

**Anonymous authors**
Paper under double-blind review

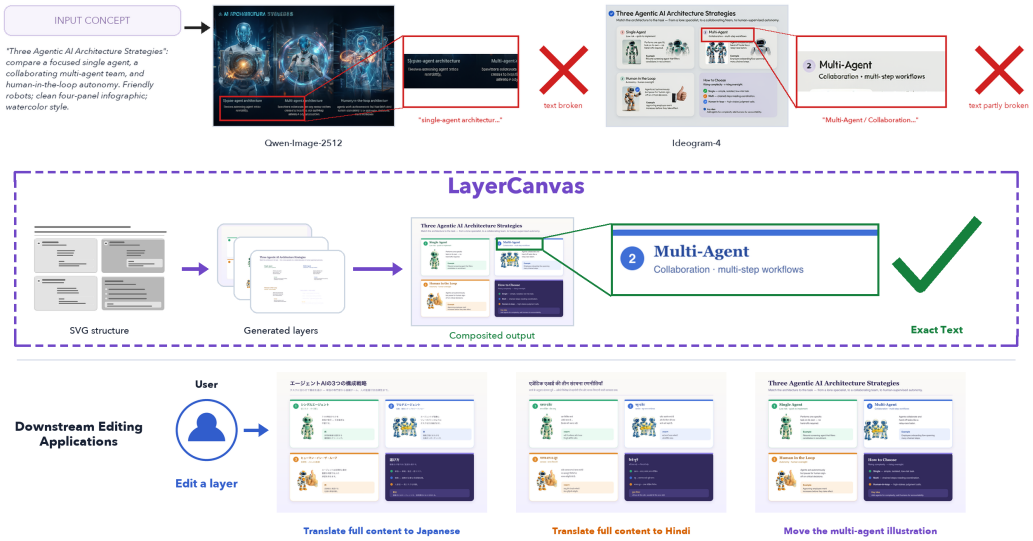


Figure 1: **LayerCanvas generates editable, text-faithful infographics.** Current SOTA text-to-image models such as Qwen-Image-2512 (Qwen Team, 2025a) and Ideogram-4 (Ideogram AI, 2026) flatten the graphic and often garble text (top). Given a vector layout, LayerCanvas generates ordered RGB layers and composes them with exact vector text (middle), enabling localization and illustration edits without changing unrelated content (bottom).

## ABSTRACT

Online learning platforms are increasingly deploying AI assistants to provide personalized, on-demand tutoring. These can leverage infographics to communicate complex information and improve the learning experience. However, automating effective infographic generation remains a challenging task, requiring a combination of visual richness with precise typography, structured layouts, and continued editability. Existing representation methods satisfy these goals only in part: Scalable Vector Graphics (SVG) representations preserve geometry, hierarchy, and text but provide limited aesthetic flexibility, while raster diffusion models produce richer appearances by flattening content and layout into a single image. Conditioning a diffusion model on a rasterized SVG does not resolve this conflict because the raster preserves approximate placement but discards element ownership, occlusion relationships, and the structure needed to modify and recompose the result. We introduce *LayerCanvas*, a hybrid approach that retains SVG as the semantic, structural, and typographic program while using diffusion to aestheticize non-text content. Rather than generating one flattened raster, LayerCanvas partitions SVG elements into z-ordered sets of mutually non-overlapping elements and processes them through communicating parallel RGB generation streams. The resulting



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

layers are independently editable and deterministically recomposed, while text is rendered from preserved vector metadata. Our approach combines rich visual synthesis with accurate typography and localized editing. LayerCanvas substantially improves text fidelity over state-of-the-art raster image generators while achieving the highest overall aesthetic quality. We further demonstrate non-destructive content editing, localization, asset reuse, and animation.

## 1 INTRODUCTION

Visual explanations are a fundamental component of modern learning platforms, where instructional content is increasingly presented through a combination of text, diagrams, illustrations, and infographics. Pairing words with pictures improves comprehension and retention relative to text alone (Mayer, 2009; Clark & Paivio, 1991), while infographics can make dense information more accessible and memorable (Bateman et al., 2010; Bhat & Alyahya, 2024). As learning platforms increasingly deliver and adapt instructional materials digitally, there is a growing need to automatically generate visual explanations that are both pedagogically effective and easy to revise as content evolves. Such explanations must communicate through precise labels and clear visual hierarchy while also providing the visual richness expected from modern image generators (Rombach et al., 2022; Podell et al., 2024; Chen et al., 2025). Meeting these requirements is challenging: precision, structure, and editability are naturally supported by representations such as Scalable Vector Graphics (SVG) (W3C SVG Working Group, 2011), whereas visual richness is most readily produced by raster generative models.

This representational contrast has a fundamental tradeoff. Vector graphics make geometry, grouping, text, and rendering order explicit, which makes direct manipulation possible (W3C SVG Working Group, 2011), but automatically generated SVGs (Rodriguez et al., 2025; Wu et al., 2023; Xing et al., 2025; Wu et al., 2025) often appear visually rigid, and stylizing them by hand is costly and difficult to scale. On the other hand, diffusion and flow models provide richer aesthetic priors (Rombach et al., 2022; Saharia et al., 2022; Esser et al., 2024), but synthesizing the artifact as a single raster makes exact typography and local intervention difficult; even state-of-the-art image models render dense, legible text unreliably, motivating specialized text-rendering methods (Liu et al., 2023b; Chen et al., 2023; Tuo et al., 2024). These limitations are especially consequential for learning platforms, which must serve learners worldwide by localizing explanations into different languages and adapting layouts across display formats, from desktop interfaces to mobile screens. Supporting these uses requires text to remain disentangled from imagery and layout components to remain independently adjustable, properties that are lost when the infographic is flattened into a raster. A natural way then is to rasterize the SVG layout and use it as a conditioning signal for a diffusion model, letting the vector structure guide where content goes while diffusion supplies appearance.

Simply rasterizing an SVG for conditioning does not resolve the mismatch: it preserves coarse spatial placement but discards element ownership, occlusion relationships, and the structure needed to recompose the graphic after an edit. This loss of structure cannot be addressed simply by strengthening raster conditioning. A system designed for editable generation should preserve these compositional units throughout synthesis instead of attempting to recover their ownership from pixels afterward.

Motivated by this principle, LayerCanvas combines the structure of SVGs with the visual capabilities of diffusion models. The SVG defines the semantic layout, hierarchy, paint order, and typography. Its non-text elements are partitioned into ordered, non-overlapping layers and generated through communicating parallel RGB streams, which are then alpha-composited according to the SVG z-order. Text nodes are kept outside the diffusion process entirely and rendered directly from their preserved vector metadata (Figure 1). Outputs remain editable, so users can replace text, move or hide groups, change the z-order, reuse illustrations, or specify animation trajectories. This makes the representation well suited to visual authoring workflows where generated content needs to remain editable and reusable, including educational content creation.

Training such a model requires examples that pair SVG with the corresponding rendered layers and final image. To create this paired supervision, we design a custom multi-agent framework, following recent work on agentic pipelines (Jain et al., 2026; Huang et al., 2026). The agents create concepts and SVG layouts, add illustrations, produce text-free targets and masks, and check that each example renders and reconstructs correctly.

Our contributions are:

- A hybrid formulation that separates deterministic SVG structure and typography from diffusion-based aestheticization;
- A multi-agent pipeline that produces and validates paired SVGs, text-free targets, layers, masks, and provenance metadata;
- A rendering-aligned layer representation that supports parallel generation, explicit occlusion modeling, and deterministic z-order composition, yielding over $25\%$ higher placement-exact OCR character accuracy and approximately $27\%$ higher aggregate aesthetic quality than the strongest raster baseline
- Practical workflows for non-destructive editing, localization, asset reuse, and animated visual explanations.

## 2 Related Work

**Visual documents and typography.** Automatic graphic design combines templates, constraints, retrieval, and learned layouts; neural methods generate sets or sequences of labeled regions, including alignment and overlap structure (Li et al., 2019; Gupta et al., 2021; Inoue et al., 2023). Such representations organize posters, slides, and documents, but do not by themselves resolve illustration appearance, occlusion, or editable object ownership. Programmatic graphics preserve structure, yet hand-authored styling and rigid templates scale poorly to open-ended concepts. Image generators provide richer aesthetics, and text-aware methods introduce character encoders, rendered-glyph conditions, OCR objectives, or localized controls (Chen et al., 2023; Yang et al., 2023; Tuo et al., 2024). Instead, we retain SVG as the source of truth for geometry and typography, remove text from diffusion targets, and render editable vector text after aestheticization.

**Controllable, object-centric, and layered generation.** ControlNet and T2I-Adapter condition pretrained diffusion models on spatial signals (Zhang et al., 2023; Mou et al., 2024), while grounded, reference-conditioned, and regional methods add box, entity, exemplar, or prompt-region controls (Li et al., 2023; Ye et al., 2023; Bar-Tal et al., 2023). These approaches improve spatial control but typically produce a single entangled raster or latent field, limiting direct editing after sampling. Object-centric, compositional, and layer-aware models separate entities, concepts, transparency, or foreground/background content (Burgess et al., 2019; Locatello et al., 2020; Avrahami et al., 2023; Zhang & Agrawala, 2024), with recent methods explicitly generating or recovering editable image layers (Yang et al., 2025; Yin et al., 2025; Wang et al., 2026). In contrast, LayerCanvas uses SVG layer identity as explicit conditioning, packing mutually non-overlapping elements into ordered RGB layer sets. This known layer structure makes output decomposition and local intervention intrinsic to the representation, rather than requiring layers to be recovered post hoc.

**Agentic multimodal curation and differentiation.** Foundation models have increasingly been used to generate instructions, propose examples, and curate synthetic multimodal supervision (Wang et al., 2023; Liu et al., 2023a; Gadre et al., 2023). More recent agentic pipelines extend this paradigm by coordinating specialized agents for scientific tasks, including translating NeRF papers into executable code and automating the generation of editable scientific figures (Jain et al., 2026; Huang et al., 2026). These systems demonstrate the potential of agentic workflows for generating and iteratively refining structured scientific artifacts. Our pipeline builds on this agentic paradigm by combining specialized generative and validation agents with deterministic SVG parsing, rendering, OCR, overlap analysis, and alpha-composition checks, producing mechanically validated training records.

## 3 Problem Setup and Method

### 3.1 Overview

Infographics combine two capabilities that are usually separated. SVGs provide exact layout and typography, whereas diffusion models provide rich raster appearance but collapse the scene into one pixel grid (Podell et al., 2024; Rombach et al., 2022). Conditioning diffusion on a rasterized SVG

preserves coarse geometry but loses element boundaries and information about which elements cover others. LayerCanvas instead keeps the SVG as the main scene description and generates its non-text content as an ordered set of editable raster layers.

Figure 2 summarizes the approach. A renderer creates training layers from the SVG, elements that do not overlap are packed into ordered groups, and all groups are denoised in parallel by a frozen image backbone. A trainable adapter identifies each layer and exchanges messages based on SVG overlap and parent-child links. The generated layers are then combined in SVG paint order and exact vector text is added back.

### 3.2 Problem setup and hybrid output

We represent an infographic as an SVG scene

$$S = (\mathcal{E}, \mathcal{T}, \mathcal{H}, \mathbf{z}, \mathcal{M}), \tag{1}$$

where $\mathcal{E} = \{e_i\}_{i=1}^{N}$ contains non-text elements, $\mathcal{T}$ stores text and typography, $\mathcal{H}$ stores parent-child links, $z_i$ is the paint index, and $\mathcal{M}$ contains element type $\tau_i$, normalized bounding box $b_i$, and renderer-derived masks. In particular, $A_i$ is the full (amodal) mask of element $e_i$, while $V_i$ is the part that remains visible after later elements cover it. Keeping both masks provides training targets for content that becomes visible after deletion, translation, or z-order edits.

The output is a hybrid artifact,

$$\mathcal{A} = \Big(S, \{\widehat{L}_g\}_{g=1}^{G}, R\Big), \qquad \widehat{I} = \mathrm{Over}\Big(R_{\mathrm{text}}(\mathcal{T}), \mathrm{Comp}_z(\widehat{L}_1, \ldots, \widehat{L}_G)\Big), \tag{2}$$

where $\widehat{L}_g = (\widehat{C}_g, \widehat{\alpha}_g)$ is an editable RGB layer and $\mathrm{Comp}_z$ composes layers in paint order. Diffusion therefore generates the non-text appearance, while the SVG continues to define geometry, ownership, which elements cover others, and exact text.

### 3.3 Ordered non-overlapping layer sets

Assigning one diffusion path to every SVG element preserves ownership but is unnecessarily expensive: many elements never overlap and can share a path without competing for pixels. Unlike pixel-derived layer decomposition (Zhang & Agrawala, 2024), we use the SVG's explicit element ownership to form ordered groups $\mathcal{G} = (G_1, \ldots, G_G)$ of $\mathcal{E}$. Elements within a group have non-overlapping full masks,

$$A_i \odot A_j = 0, \qquad i \neq j, \quad e_i, e_j \in G_g, \tag{3}$$

while every overlap respects paint order: if $A_i \odot A_j \neq 0$ and $z_i < z_j$, then $g(i) < g(j)$. Consequently, non-overlapping elements may share one generated layer, but an element in front and the element behind it remain separately editable.

We create the groups by processing elements in ascending $(z_i, i)$ order:

$$g(i) = \begin{cases} 1, & \mathcal{P}_i = \varnothing, \\ 1 + \max_{j \in \mathcal{P}_i} g(j), & \text{otherwise}, \end{cases} \qquad \mathcal{P}_i = \{j : z_j \leq z_i,\ A_j \odot A_i \neq 0,\ j \text{ processed}\}. \tag{4}$$

This rule cannot place a front element behind an element that it covers. Group masks are unions, $A_g = \max_{i \in G_g} A_i$ and $V_g = \max_{i \in G_g} V_i$, and each layer path receives its grouped control raster, type, box, paint order, and owned illustration prompt. The number of paths therefore depends on how many layers overlap, rather than the number of SVG elements.

For premultiplied color $P_g = \alpha_g C_g$, the final raster is obtained by back-to-front alpha composition,

$$(P^{(g)}, \alpha^{(g)}) = \big(P_g + (1 - \alpha_g) P^{(g-1)}, \alpha_g + (1 - \alpha_g)\alpha^{(g-1)}\big), \tag{5}$$

starting from a transparent canvas. Paint order controls composition, not generation order: all valid groups are denoised in parallel, and individual layers can later be translated, removed, or reordered without regenerating unrelated groups.

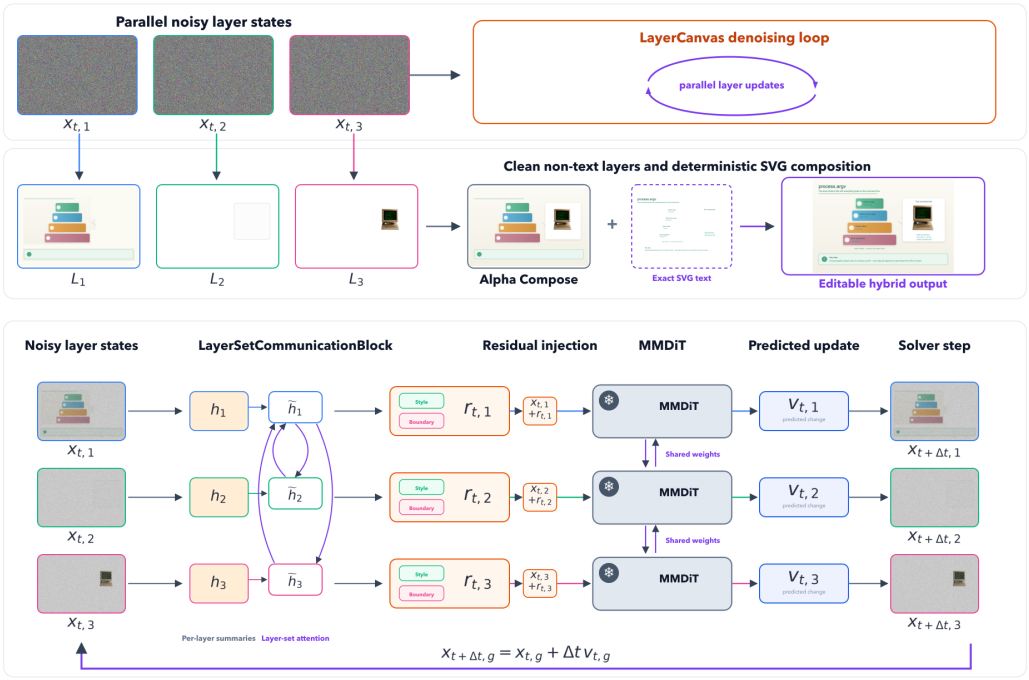


Figure 2: **LayerCanvas layer generation. Top:** Parallel denoising produces aligned non-text layers, which are composed in paint order before exact SVG text is restored. **Bottom:** Layer-set attention produces residuals for shared frozen MMDiT updates, and the solver advances every layer.

### 3.4 Generating coordinated layers

LayerCanvas generates all layers together, but separate layers can easily drift toward different colors, lighting, or object identities. Following the continuous interpolation used in flow matching (Lipman et al., 2023), we give them part of the same starting noise and use the same denoising timestep. This encourages a common look while preserving private noise for layer-specific detail. Shared noise cannot make one layer respond to another, however, so the frozen image backbone needs a small trainable adapter. The adapter follows the efficient frozen-backbone conditioning strategy of ControlNet and T2I-Adapter (Zhang et al., 2023; Mou et al., 2024), but operates across an ordered layer set so that each layer is influenced by the others. It answers three questions: which layer is being generated, which other layers matter to it, and whether their effect should be global or limited to a shared boundary.

**Identifying each layer.** The frozen backbone normally generates one image and has no reason to know whether a batch item represents a background, a foreground illustration, or a later paint-order layer. For group $g$, the model's variational autoencoder (VAE) (Kingma & Welling, 2014) converts the grouped control raster into internal features $s_g$. The trainable grouped layer-control adapter then adds learned identifiers for paint order and the main element type:

$$q_g = \mathrm{LN}(s_g) + E_z[g] + E_\tau[\tau_g], \qquad r_g^{(0)} = W_0 q_g, \quad r_g^{(d)} = W_d \phi(W_s q_g). \tag{6}$$

Here, $\mathrm{LN}(s_g)$ describes the layer's visual control, $E_z[g]$ tells the model where it appears in paint order, and $E_\tau[\tau_g]$ tells it the main element type. The projections $W_0$ and $W_d$ turn this combined description into small adjustments at the transformer input and selected depths. These outputs start at zero, so the adapter initially leaves the pretrained backbone unchanged. The adjustments are added at a small set of backbone depths.

**Sharing information between layers.** Layer identifiers solve ownership, but the backbone still processes every layer independently. Dense communication between every image location would

Table 1: **Primary comparison** on test set. Char and Exact measure placement-exact OCR, while Recall measures text recovered anywhere in the image. Appeal, Typog., and Cov. are VLM scores from 0-5, with Avg denoting their mean. Best synthesis result in each column is **bold**.

| | Text fidelity | | | VLM judge (0-5) | | | |
|---|---|---|---|---|---|---|---|
| Method | Char | Exact | Recall | Appeal | Typog. | Cov. | Avg |
| SVG (LLM-authored) | 0.856 | 0.676 | 0.964 | 2.37 | 4.02 | 4.90 | 3.76 |
| Qwen-Image-2512 (Qwen Team, 2025a) | 0.080 | 0.002 | 0.149 | 2.69 | 1.00 | 1.45 | 1.71 |
| Ideogram-4 (Ideogram AI, 2026) | 0.657 | 0.360 | 0.744 | 2.92 | 2.29 | 3.89 | 3.03 |
| LayerCanvas (Ours) | **0.822** | **0.623** | **0.942** | **2.99** | **3.81** | **4.75** | **3.85** |

be costly and would ignore the overlap and parent-child links already available in the SVG. Instead, each *LayerSetCommunicationBlock* builds one compact summary $h_g$ per layer. For a receiving layer $g$, a sending layer $h$, and attention head $a$, the block first computes a pairwise score $\ell^a_{gh}$, converts the scores into weights $\pi^a_{gh}$, and uses those weights to combine the sending-layer summaries:

$$\begin{aligned} \ell^a_{gh} &= \frac{\langle W^a_Q h_g, W^a_K h_h \rangle}{\sqrt{d_a}} + (w^a_E)^\top E_{gh}, \\ \pi^a_{gh} &= \mathrm{softmax}_h(\ell^a_{gh}), \\ \widetilde{h}_g &= W_O \,\mathrm{Concat}_a \left( \sum_h \pi^a_{gh} W^a_V h_h \right). \end{aligned} \tag{7}$$

The first part of $\ell^a_{gh}$ measures similarity between the two layers; the second adds their SVG relationship from $E_{gh}$. $\pi^a_{gh}$ says how much layer $g$ should listen to layer $h$, and $\widetilde{h}_g$ is the resulting information received by layer $g$ after all attention heads are combined. Because the SVG links have directions, a covered background can receive a different message from the element in front of it.

**Applying shared information.** Not every cross-layer change should affect the same area. A shared palette or lighting choice should act smoothly across a layer, while an overlap correction should stay near the touching edges. We therefore split the received summary $\widetilde{h}_g$ into a style path and a boundary path. Let $x_g$ denote the current spatial features of layer $g$:

$$\begin{aligned} r^{\mathrm{sty}}_g &= F_{\mathrm{sty}}(\widetilde{h}_g), \\ r^{\mathrm{bnd}}_g &= F_{\mathrm{bnd}}(x_g, \widetilde{h}_g), \\ r_g &= \sigma(\gamma^{\mathrm{sty}}_g)\, r^{\mathrm{sty}}_g + \sigma(\gamma^{\mathrm{bnd}}_g)\, B_g \odot r^{\mathrm{bnd}}_g. \end{aligned} \tag{8}$$

The style function turns $\widetilde{h}_g$ into a smooth layer-wide change. The boundary function combines $\widetilde{h}_g$ with the current features $x_g$, while $B_g$ keeps its change near layer edges. The learned gates $\gamma^{\mathrm{sty}}_g$ and $\gamma^{\mathrm{bnd}}_g$ decide how strongly each path should act. Their weighted sum is the final residual $r_g$, which is added back to layer $g$ at that depth of the frozen MMDiT image backbone. As above, the added outputs start at zero, so communication is introduced gradually during training.

**Training and exact text.** Training checks each layer, the final combined image, hidden areas and boundaries. These losses check both individual ownership and the assembled image. Typography is excluded from diffusion targets and added back from the retained SVG strings. This guarantees exact editable text rather than asking diffusion to paint glyphs, as in text-aware raster generators (Chen et al., 2023; Tuo et al., 2024). Appendix B.1 gives the full training path, loss and text-composition procedure.

# 4 Experiments

## 4.1 Setup

**Dataset -** We develop InfoLayers, a corpus of 23,960 paired infographic records on a $768 \times 512$ canvas with a multi-agent pipeline. Each sample pairs a structural SVG with a text-free and a

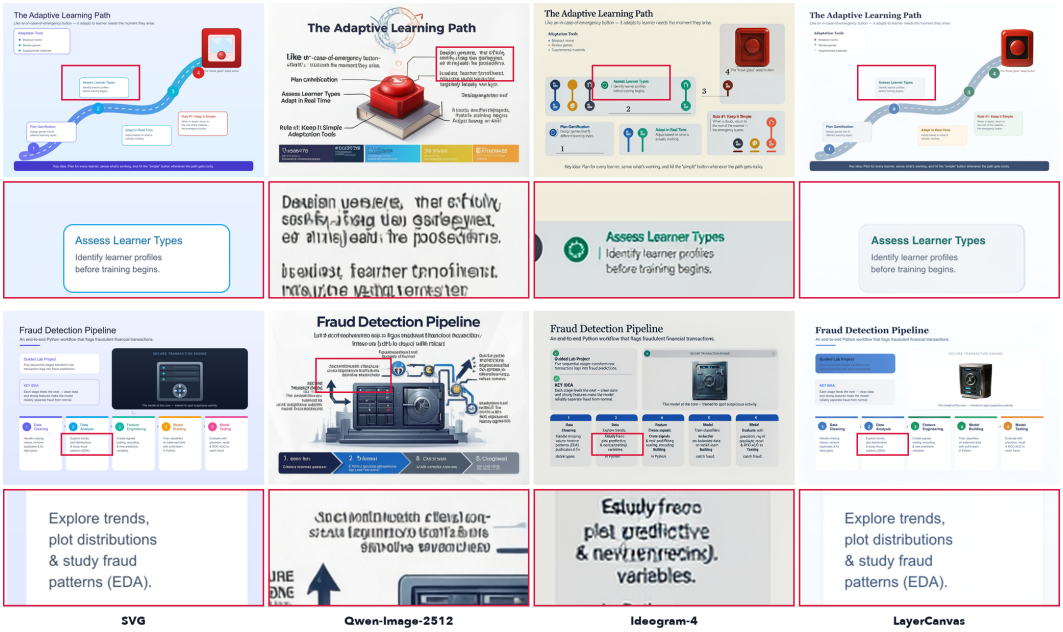


Figure 3: **Qualitative four-way comparison.** Columns show, left to right, the SVG control, two prompt-only SOTA diffusion baselines (Qwen-Image-2512 (Qwen Team, 2025a) and Ideogram-4 (Ideogram AI, 2026)), and LayerCanvas. The row below each example shows zoomed-in crops of the highlighted text regions. Prompt-only models produce appealing raster illustrations but garble or invent text and drift from the prescribed layout whereas LayerCanvas keeps the layer layout and composites exact, legible vector text.

composited target, together with the per-layer supervision our model consumes: amodal and visible masks, paint order, and a hierarchy/overlap graph. Construction and layout statistics are given in Appendix A.

**Baselines -** We compare four methods on shared held-out layouts. The *LLM-authored SVG* reference recolors and completes the structural SVG and renders it, giving exact vector text by construction but no learned aestheticization. The two prompt-only open-weight SOTA diffusion baselines, *Qwen-Image-2512* (Qwen Team, 2025a) and *Ideogram-4* (Ideogram AI, 2026), are each conditioned on the design-concept prompt derived from the input SVG layout; for Ideogram-4 this prompt is constructed from the same SVG layout, with its structural elements serialized into the structured prompt format the model expects. LayerCanvas is our grouped layer-set diffusion model with deterministic exact-SVG text compositing built on the same MMDiT backbone as Ideogram-4 (Ideogram AI, 2026). For evaluation, all methods are placed on a $768\times512$ canvas.

**Metrics -** Text fidelity is scored two ways. *Placement-exact* fidelity uses ground-truth-region OCR (Tesseract OCR contributors, 2026) at $768\times512$, and reports character accuracy and exact-match rate. *Placement-agnostic* fidelity runs full-image OCR at the same resolution and measures word recall, how much of the ground-truth text appears anywhere in the image, so a baseline that renders readable text at the wrong position still gets credit. The two exact-text methods place text from the SVG by construction, so their remaining gap from $1.0$ is OCR error rather than real text mistakes. For visual quality we use a *VLM as a judge*: Claude Opus 4.8 (Anthropic, 2026) scores each design from $0$ to $5$ on illustration appeal, typography, and semantic coverage, and we report *Avg*, the mean of the three. The full evaluation protocol is given in Appendix B.2.

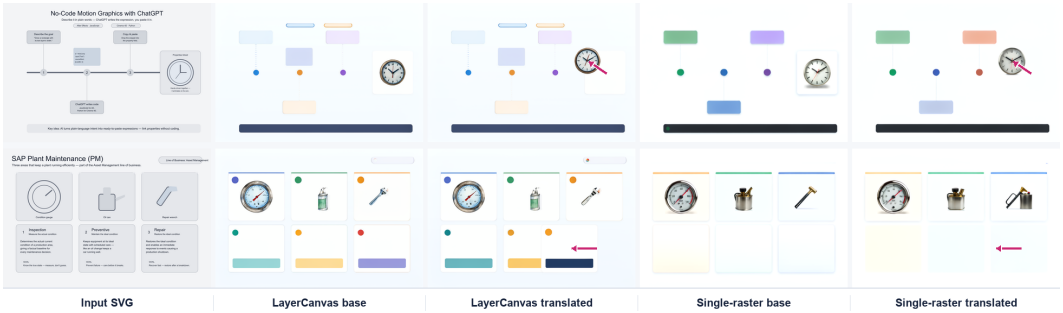


Figure 4: **Representation ablation:** Each column shows the source SVG, fixed-noise base generation, and the translated version by perturbing the base SVG for LayerCanvas and the single-raster as control. Red arrows show the shift from the base position in the translated cases. LayerCanvas better preserves object identity and unrelated layout content.

Table 2: **Comparison of Different Conditioning Representations on validation set**

| Method | Char | Exact | Recall | Appeal | Typog. | Cov. | Avg |
|---|---|---|---|---|---|---|---|
| Single-raster Conditioning | **0.851** | **0.568** | 0.783 | 2.78 | 3.56 | 4.32 | 3.55 |
| LayerCanvas | 0.823 | 0.530 | **0.795** | **2.84** | **3.76** | **4.72** | **3.77** |

### 4.2 Main comparison

Table 1 reports the comparison on the test set (construction in Appendix B). LayerCanvas combines strong visual appeal with text fidelity, outperforming both raster baselines across all text metrics while approaching the exact-text SVG reference. Ideogram-4 (Ideogram AI, 2026) generates aesthetic looking samples, but it still trails LayerCanvas on text fidelity; Qwen-Image-2512 (Qwen Team, 2025a) performs poorly on both text placement and recovery. Under the primary Claude judge, LayerCanvas is the strongest learned method across illustration appeal, typography, semantic coverage, and aggregate quality, and slightly exceeds the SVG reference overall. The same learned-method ranking holds under independent Gemini 3.7 Flash (Google DeepMind, 2026) and GPT-5.6 Sol (OpenAI, 2026) judges (Appendix Table 5). Figure 3 shows the same appeal-text trade-off qualitatively. Additional results are in Appendix B.3. Inference compute details for LayerCanvas are in Appendix B.4.2.

### 4.3 Representation ablation

We compare grouped LayerCanvas with a single-raster conditioning baseline to test whether explicit layer separation improves editability. The baseline conditions the same MMDiT model (Ideogram AI, 2026) directly on the rasterized SVG using T2I-Adapter (Mou et al., 2024). A single stream must learn to render the whole graphic at once, which becomes harder as the layout grows denser. In contrast, each LayerCanvas layer contains non-overlapping elements, making it simpler to learn before the layers are composed. Under fixed noise, LayerCanvas better preserves the moved element and surrounding layout, while the control redraws unrelated content (Figure 4). It also improves text recall and all VLM scores, raising average quality from 3.55 to 3.77 (Table 2).

## 5 Downstream applications

Because LayerCanvas preserves SVG structure alongside independently addressable raster layers, a generated design can be adapted without regenerating the whole image. Vector text supports changes to wording, typography, and language, while separate visual layers enable animation, asset reuse, and local element edits such as removing a card or swapping an icon while preserving unaffected content. Retained grouping and paint order further support layout retargeting, re-theming, and SVG-level corrections while limiting changes to the intended components. Together, these capabilities turn

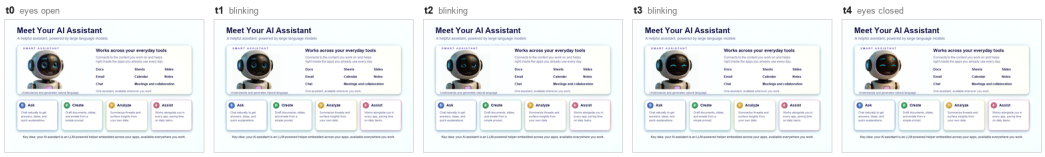


Figure 5: **Independent animation of one LayerCanvas region.** A Text2Video-Zero-inspired image-diffusion trajectory produces an open-to-closed blink. Only the selected eye region changes across the five frames; the surrounding illustration, layout, and vector typography remain fixed.

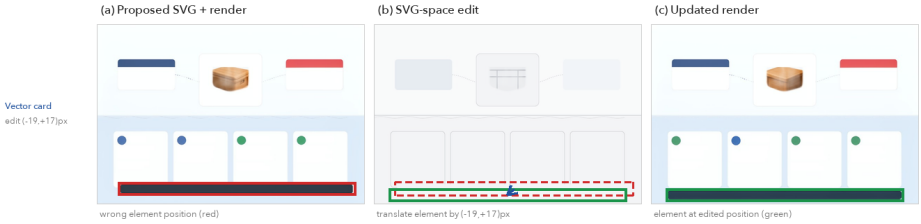


Figure 6: **Closed-loop SVG correction.** Each row shows an incorrect SVG and render, the direct SVG edit, and the corrected render. The edited element moves to the requested location while unrelated content remains stable.

a generated infographic into a reusable authoring artifact. We highlight three below; the rest are detailed in Appendix C.

### 5.1 Text editing and localization

End-to-end raster generators must repaint text whenever wording, typography, or language changes. LayerCanvas instead generates only the visual substrate and preserves typography as styled SVG nodes, so copy, style, and locale changes require no diffusion pass and cannot alter the generated artwork (Figure 13).

### 5.2 Independent layer animation

We adapt the zero-shot animation principle of Text2Video-Zero (Khachatryan et al., 2023), in which the motion and appearance priors of a pretrained text-to-image diffusion model produce a temporally linked sequence without video-specific training. Whereas a flattened document would expose the complete page to this update, LayerCanvas lets the temporal operation target one named image layer while all other layers remain fixed: every unselected image layer and the full-resolution SVG typography are copied identically into every frame.

### 5.3 SVG-space refinement and faithful diffusion propagation

An upstream layout proposal may contain a misplaced, mislabeled, or incorrectly styled element. In LayerCanvas, the SVG is a persistent control surface: the element can be corrected directly, and the model propagates that correction to the raster without retraining or redesigning the rest of the document.

## 6 Conclusion

We formulate infographic generation as structured aestheticization: a deterministic SVG supplies geometry, hierarchy, typography, while parallel layer-set generation supplies visual appearance. Aligning this representation with the renderer supports many downstream applications like editing and artifact reuse. Enforcing generalization to more layouts and improving transformations under changed occlusion remain important research directions.

## ETHICS STATEMENT

This work does not involve human subjects, and no personal or user data was collected. The InfoLayers corpus is not scraped: concepts are sourced from an educational course catalog and used only as textual seeds, and the visual content of every target is generated by our own multi-agent pipeline (Appendix A). Every record retains its course identifier, layout archetype, edit prompts, diffusion style and seed, and compositing parameters, so any sample can be traced back to its concept and regenerated (§A.4).

As discussed in Appendix D, LayerCanvas requires a valid source SVG and inherits errors and biases in its content, layout, and training data. Deterministic typography preserves supplied text but does not ensure factuality, accessibility, or culturally appropriate localization, and edits that change occlusion or z-order may affect unintended regions. Because the intended setting is an AI learning assistant, these failure modes reach learners, so responsible deployment requires documented data provenance, license and bias checks, generated-content disclosure, and human review.

## REPRODUCIBILITY STATEMENT

The method is specified in Section 3, with the adapter, layer-identity conditioning, and cross-layer communication given in full in Appendix B.1. The construction of InfoLayers is documented end to end in Appendix A: concept sourcing (§A.1), the four-stage generation pipeline (§A.2), layer decomposition, augmentation, and the validation checks applied to every record (§A.3), the resulting scale, leakage-free course-disjoint splits, and provenance (§A.4), and the models, versions, and compute used at each stage (§A.6). The evaluation protocol, including the compared methods, both text-fidelity metrics, and the judge rubric and judge models, is given in §B.2, and judge outputs are disk-cached so that reported scores are stable across reruns. Runtime and memory measurements are reported in §B.4. The editing comparisons in Section 5 hold sampling noise fixed between the original and edited conditions so that differences are attributable to the structural edit.

## AI USE STATEMENT

In this work, we used generative AI tools to *generate a synthetic data set* and to *clean and reformat that data set*: the InfoLayers corpus is produced by a multi-agent pipeline in which an LLM design agent (Claude Opus 4.8) writes, validates, and repairs the structural SVG programs that seed each record, as described in §A.2 and §A.6. We also used generative AI tools to *support qualitative analysis*: aesthetic and typographic quality is scored by VLM judges under the rubric and protocol given in §B.2, and an LLM-generated SVG serves as one of the compared baselines. These uses are components of the research itself and are documented in the appendix rather than being incidental.

We have not used generative AI tools to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, to implement methods, to assist with translation, or to interpret results. The remaining tasks with required disclosure, namely developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs, are not applicable to this work, which contains no theoretical claims or proofs.

Additionally, among the tasks with recommended disclosure, we used generative AI tools to *edit this paper to improve readability*, applied to prose first written by the authors. We have reviewed all AI-assisted work: every generated data record passes the automated structural and rendering validation checks described in §A.3 before entering the corpus, judge outputs are cached and were cross-checked by repeating the identical rubric with two further judge models (§B.2), and all edited prose was read and approved by the authors against the original text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Anthropic. Introducing Claude Opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, May 2026. Product announcement, May 28, 2026.

Omri Avrahami, Kfir Aberman, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. Break-A-Scene: Extracting multiple concepts from a single image. In *SIGGRAPH Asia 2023 Conference Papers*, 2023. doi: 10.1145/3610548.3618154.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *International Conference on Machine Learning*, 2023.

Scott Bateman, Regan L. Mandryk, Carl Gutwin, Aaron Genest, David McDine, and Christopher Brooks. Useful junk? the effects of visual embellishment on comprehension and memorability of charts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 2573–2582, 2010.

S. A. Bhat and S. Alyahya. Infographics in educational settings: A literature review. *IEEE Access*, 12:1633–1649, 2024. doi: 10.1109/ACCESS.2023.3348083.

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Hang Chen, Qian Xiang, Jiaxin Hu, Meilin Ye, Chao Yu, Hao Cheng, and Lei Zhang. Comprehensive exploration of diffusion models in image generation: A survey. *Artificial Intelligence Review*, 58, 2025. doi: 10.1007/s10462-025-11110-3.

Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. TextDiffuser: Diffusion models as text painters. In *Advances in Neural Information Processing Systems*, 2023.

James M. Clark and Allan Paivio. Dual coding theory and education. *Educational Psychology Review*, 3(3):149–210, 1991.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Robert Marten, Mitchell Wortsman, et al. DataComp: In search of the next generation of multimodal datasets. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, 2023.

Daniel Gatis. rembg: A tool to remove image backgrounds. `https://github.com/danielgatis/rembg`, 2025. MIT-licensed software.

Google DeepMind. Gemini 3.7 Flash model card. `https://deepmind.google/models/model-cards/gemini-3-7-flash/`, August 2026. Published August 13, 2026.

Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S. Davis, Vijay Mahadevan, and Abhinav Shrivastava. LayoutTransformer: Layout generation and completion with self-attention. In *IEEE/CVF International Conference on Computer Vision*, pp. 1004–1014, 2021.

Siyuan Huang, Yifan Zhou, Yutong Gao, Zi Yin, Juyang Bai, Xinxin Liu, Rama Chellappa, Chun Pong Lau, Cheng Peng, Sayan Nag, and Shraman Pramanick. SciFig: Towards automating editable figure generation for scientific papers. arXiv preprint arXiv:2601.04390, 2026.

Ideogram AI. Ideogram 4.0 technical details: Open model at the forefront of design. `https://ideogram.ai/blog/ideogram-4.0/`, June 2026. Technical blog.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10167–10176, 2023.

Seemandhar Jain, Keshav Gupta, Kunal Gupta, and Manmohan Chandraker. NERFIFY: A multi-agent framework for turning NeRF papers into code. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24384–24394, 2026.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2Video-Zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15954–15964, 2023. doi: 10.1109/ICCV51070.2023.01462.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Dom Lachowicz, Caleb Moore, et al. *rsvg-convert: Turn SVG Files into Raster Images*. librsvg, 2014. URL https://manpages.ubuntu.com/manpages/trusty/man1/rsvg-convert.1.html.

Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. LayoutGAN: Generating graphic layouts with wireframe discriminators. In *International Conference on Learning Representations*, 2019.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023a.

Rosanne Liu, Dan Garrette, Chitwan Saharia, William Chan, Adam Roberts, Sharan Narang, Irina Blok, RJ Mical, Mohammad Norouzi, and Noah Constant. Character-aware models improve visual text rendering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.

Richard E. Mayer. *Multimedia Learning*. Cambridge University Press, 2nd edition, 2009.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 4296–4304, 2024.

OpenAI. GPT-5.6 Sol model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026. Model documentation.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2024.

Qwen Team. Qwen-Image-2512: Finer details, greater realism. https://qwen.ai/blog?id=qwen-image-2512, 2025a. Technical blog.

Qwen Team. Qwen-Image-Edit: Image editing with higher quality and efficiency. https://qwen.ai/blog?id=qwen-image-edit, 2025b. Technical blog.

Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. StarVector: Generating scalable vector graphics code from images and text. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tesseract OCR contributors. Tesseract: Open source OCR engine. `https://github.com/tesseract-ocr/tesseract`, 2026. Software repository; accessed September 2, 2026.

Yuxiang Tuo, Wangmeng Xiang, Jun-Yan He, Yifeng Geng, and Xuansong Xie. AnyText: Multilingual visual text generation and editing. In *International Conference on Learning Representations*, 2024.

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, June 2014. ISSN 2167-8359. doi: 10.7717/peerj.453. URL `http://dx.doi.org/10.7717/peerj.453`.

W3C SVG Working Group. Scalable Vector Graphics (SVG) 1.1 (Second Edition). W3C Recommendation, 2011. URL `https://www.w3.org/TR/SVG11/`.

Binhao Wang, Shihao Zhao, Bo Cheng, Qiuyu Ji, Yuhang Ma, Liebucha Wu, Shanyuan Liu, Dawei Leng, and Yuhui Yin. RevealLayer: Disentangling hidden and visible layers via occlusion-aware image decomposition. *arXiv preprint arXiv:2605.11818*, 2026. URL `https://arxiv.org/abs/2605.11818`.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. IconShop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics*, 42(6), 2023. doi: 10.1145/3618364.

Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2SVG: Vector graphics generation with large language models and image diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23690–23700, 2025.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering LLMs to understand and generate complex vector graphics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jinrui Yang, Qing Liu, Yijun Li, Soo Ye Kim, Daniil Pakhomov, Mengwei Ren, Jianming Zhang, Zhe Lin, Cihang Xie, and Yuyin Zhou. Generative image layer decomposition with visual effects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Yukang Yang, Dongnan Gui, Yuhui Yuan, Weicong Liang, Haisong Ding, Han Hu, and Kai Chen. GlyphControl: Glyph conditional control for visual text generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44050–44066, 2023.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. IP-Adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, Lionel M. Ni, Jingren Zhou, Junyang Lin, and Chenfei Wu. Qwen-Image-Layered: Towards inherent editability via layer decomposition. *arXiv preprint arXiv:2512.15603*, 2025. URL `https://arxiv.org/abs/2512.15603`.

Lvmin Zhang and Maneesh Agrawala. Transparent image layer diffusion using latent transparency. *ACM Transactions on Graphics*, 43(4), 2024. doi: 10.1145/3658150.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE/CVF International Conference on Computer Vision*, 2023.

# Appendix

CONTENTS


- Appendix A: Data
  - A.1 Concept sourcing and topical scope
  - A.2 Multi-agent generation pipeline
  - A.3 Layer decomposition, augmentation, and validation
  - A.4 Dataset composition, splits, and provenance
  - A.5 Dataset analysis: color composition is coupled and learnable
  - A.6 Implementation details
- Appendix B: Additional Details and Experiments
  - B.1 Additional method details
  - B.2 Evaluation details
  - B.3 Additional comparisons
  - B.4 Systems profile
- Appendix C: Downstream Applications
  - C.1 Text editing and localization
  - C.2 Animation and progressive reveal
  - C.3 Asset reuse and cross-document recomposition
  - C.4 Layout retargeting and responsive resize
  - C.5 Command-driven interactive editing
  - C.6 SVG-space refinement and faithful diffusion propagation
- Appendix D: Limitations, ethics, and broader impact


## A DATA

Training LayerCanvas requires a form of supervision that, to our knowledge, no existing corpus provides: infographics paired at the level of individual, renderer-aligned layers. A usable training record must expose, for a single graphic, (i) a structural program that fixes geometry, hierarchy, paint order, and typography; (ii) an aestheticized, text-free rendering of that program; and (iii) a decomposition of that rendering into ordered, mutually non-overlapping RGBA layers with amodal and visible masks. Public SVG datasets supply programs but no rich appearance, while raster infographic collections supply appearance but discard element ownership, occlusion, and text metadata. We therefore *synthesize* the corpus with a multi-agent generation pipeline that couples a large language model, which designs and validates editable SVG programs, with a diffusion image editor, which aestheticizes only the non-text visual content. The pipeline is designed so that every artifact LayerCanvas consumes at training time, namely the colorless structural input, the text-free target, the per-layer amodal/visible assets, and the graph of overlap and hierarchy relations, is produced deterministically from an SVG the agent has already checked, rather than recovered from pixels after the fact.

This section describes how concepts are sourced from an educational course catalog (§A.1), how the multi-agent pipeline turns each concept into a validated paired record (§A.2), how those records are decomposed into layers, augmented, and validated (§A.3), the resulting scale, splits, and provenance (§A.4), an analysis of the appearance statistics the pipeline induces (§A.5), and the implementation and compute details needed to reproduce it (§A.6).

### A.1 CONCEPT SOURCING AND TOPICAL SCOPE

We seed generation with concepts drawn from popular online educational courses. Each concept is a short, self-contained instructional idea (for example, “handling missing values in Python,” “the tip-of-the-iceberg model of specialist agents,” or “the Power BI $\leftrightarrow$ Excel round-trip”), and it carries the identifier and title of the course it was extracted from. Anchoring the corpus in real course material keeps the distribution of topics, terminology, and visual conventions close to genuine educational

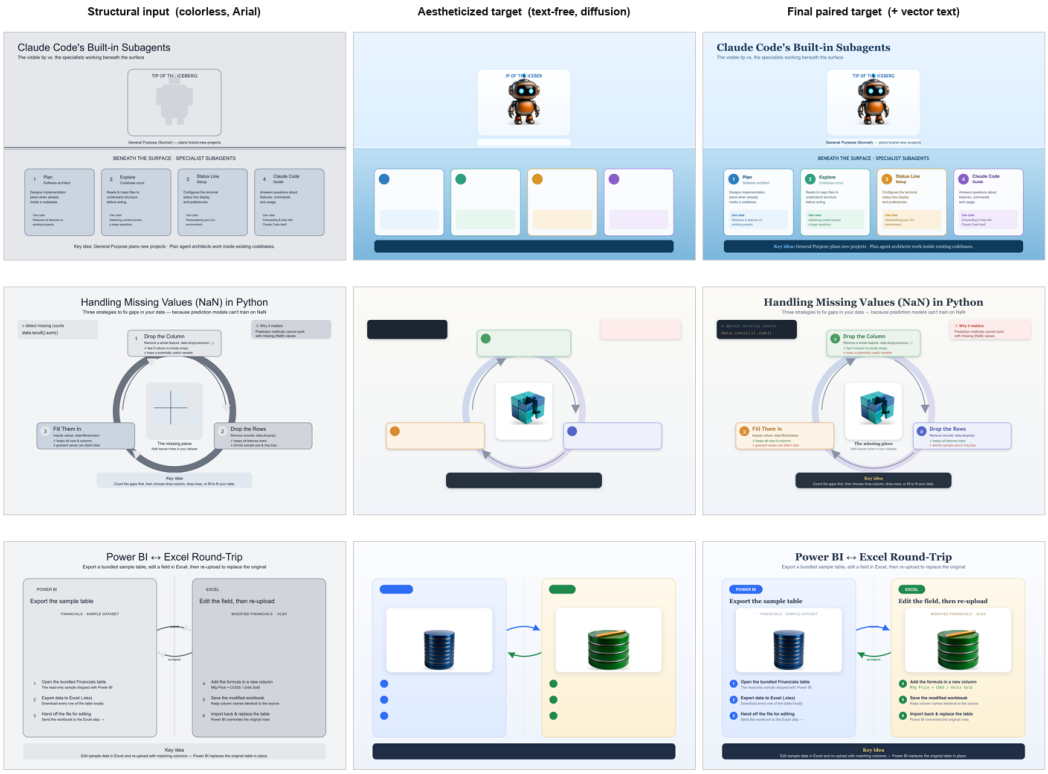


Figure 7: **The paired training record.** Each column is one artifact the pipeline emits for the same sample. *Left:* the structural input, an SVG with color, fonts, and textures stripped and text set in a neutral face, which supplies geometry, hierarchy, and paint order without leaking target appearance. *Middle:* the aestheticized, *text-free* target that the diffusion streams are trained to produce; the editable-visual regions have been restyled into rich illustrations while every text region is left blank. *Right:* the final paired target after the preserved vector text layer is composited on top. Diffusion is never asked to render glyphs.

infographics (dense labels, step-wise procedures, comparisons, cycles, and metaphorical scenes) rather than to generic stock imagery. The course identifier is retained on every derived record and is used both for provenance and, critically, for constructing leakage-free splits (§A.4).

### A.2 Multi-agent generation pipeline

The pipeline proceeds in four stages. A design stage authors a structured, editable SVG program from a concept; an extraction stage renders the structural input and the text-free layout and serializes structured conditioning; an aestheticization stage restyles each editable region with a diffusion image editor; and a composition stage mattes and reinserts the styled illustrations in paint order. All appearance in the final target originates in the SVG program or in the diffusion edits of its tagged regions, never in a free-form raster of the whole scene.

**Stage A: structured SVG design (LLM).** A design agent, built on a large language model, turns a concept into a complete, standalone SVG on a fixed $1536 \times 1024$ canvas. Beyond producing an aesthetic educational layout, the agent must mark at least one *editable-visual* group: a coarse vector placeholder standing in for a tangible, photographable subject (a robot, a database cylinder, a brass scale, a stack of coins). Each such group is emitted with a strict, machine-parseable signature,

```
<g id="editable-visual-k" data-role="editable-visual"
  data-visual-desc="..." data-edit-prompt="...">...</g>,
```

where `data-edit-prompt` is the exact, self-contained instruction later handed to the diffusion editor. To prevent the corpus from collapsing onto a single visual template, the agent is assigned one of fourteen layout archetypes per concept (hero-left, hub-and-spoke, comparison-split, circular-cycle, above-below, quadrant-grid, timeline, and others), rotated deterministically across a batch and treated as an adaptable starting structure rather than a rigid template. The prompt enforces explicit layout discipline (shared alignment gridlines, equal gaps for repeated elements, symmetry matched to the number of editable visuals) and an aesthetic direction tuned for legibility (light airy backgrounds, per-section color coding, a strong title/subtitle/takeaway hierarchy). A guard explicitly forbids screens, dashboards, spreadsheets, and other flat, text-bearing objects from being marked editable, because such subjects photorealize poorly and reintroduce garbled text; those elements remain crisp vector content. The agent operates in a render-and-inspect loop: it rasterizes its SVG, examines the image, and performs at least one revision pass to correct misalignment, uneven spacing, dead space, or low-contrast text before emitting a manifest of its editable groups.

**Stage B: structural extraction and conditioning.** From the validated SVG we deterministically derive the model's inputs. The *structural input* (Fig. 7, left) is produced by "rawifying" the SVG, that is, stripping color, fonts, gradients, and textures, setting text in a neutral face, and rendering to a flat raster, so that it encodes placement, grouping, and paint order without revealing target appearance. We also render a *text-free layout* in which all recursive text nodes are removed. The record captures the scene as typed metadata: every text run with its content, semantic role (title/heading/label), normalized bounding box, font family, size, weight, and fill; every illustration with its identifier, description, the `data-edit-prompt`, its bounding box, and its z-index; and a global style block listing the design palette (hex swatches), font families, and mean corner radius. This record is the interface between deterministic structure and diffusion aestheticization used throughout the method.

**Stage C: illustration aestheticization (diffusion editor).** Each editable-visual crop is rendered in isolation and passed to a diffusion image editor (Qwen-Image-Edit (Qwen Team, 2025b)) that repaints *only* that region into a rich illustration on a plain white studio background. A style controller makes the requested look dominant: it strips residual realism adjectives from the agent's subject description, prepends a leading style directive, and swaps a matching negative prompt, so a single subject can be realized in any of six styles (*photoreal*, *flat2d*, *soft3d*, *cartoon*, *watercolor*, *isometric*) while its identity and composition are preserved. The middle column of Fig. 7 shows the resulting text-free targets, in which the coarse placeholders have become detailed objects but every label region remains blank.

**Stage D: matting and z-ordered composition.** The styled illustration is cut from its white background with an alpha matte (rembg/u2net (Gatis, 2025)), cleaned to its largest connected component and feathered, fitted to the placeholder's bounding box, and composited back into the scene at exactly the illustration's original z-index. Because the surrounding layout is rebuilt from the SVG with the editable groups removed, the realistic render occupies the placeholder footprint without disturbing any text or vector element, and soft contact shadows are added programmatically and kept off text and table regions. The output is the final paired target (Fig. 7, right): the aestheticized, text-free composite with the preserved vector text layer rendered crisply on top.

### A.3 Layer decomposition, augmentation, and validation

**Ordered non-overlap layer decomposition.** For every sample we additionally emit the amodal layer decomposition. Each top-level layer is rendered in isolation to recover its *amodal* silhouette (its complete support as if nothing occluded it) and its *visible* mask (the pixels surviving occlusion by all higher-z layers); their difference is exactly the occluded region. Editable layers take their amodal pixels from the matted realistic render, reconstructed with the same parameters used in composition, so that they agree with the final target. We also record, for every pair of layers whose amodal silhouettes intersect, an overlap edge annotated with the back/front assignment and the covered fraction. These masks and edges are what make the ordered non-overlap layer sets and the graph-biased communication of the method well defined; they are computed deterministically from the renderer, not discovered from pixels.

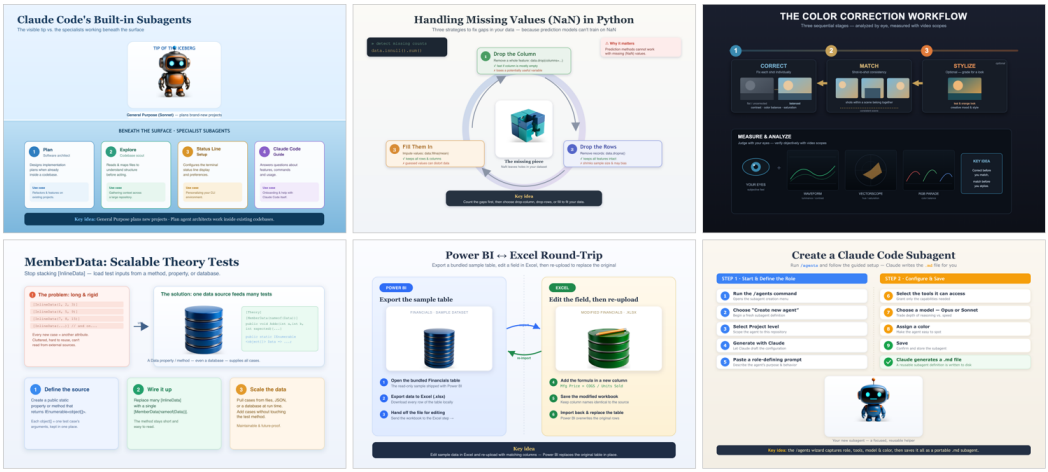


Figure 8: **Representative dataset infographics.** The corpus spans diverse layout archetypes (above-below iceberg, circular cycle, hero-plus-steps, comparison split, guided procedure, staged timeline), both light and dark themes, and a range of illustration styles, while preserving crisp typography and clean editable-visual footprints in every sample. Text is always rendered from vector metadata, never by the diffusion editor.

**Style augmentation via illustration reuse.** Because Stage A (LLM design) and Stage B (structural extraction) are the expensive, language-model-dependent parts of the pipeline, we amortize them: given an already-built sample with its per-layer amodal RGBA and its tagged editable-visual illustration, we generate additional (input $\rightarrow$ target) pairs whose only difference is the visual *style* of the illustration. The existing target layers and structured conditioning are reused verbatim; only Stage C (re-style the illustration) and Stage D (re-matte, re-compose) are rerun on-GPU. This yields multiple stylistic realizations of the same structural layout at a fraction of the cost, and a denser-composition variant reuses previously generated illustrations inside new, higher-overlap layouts to increase occlusion depth.

**Validation and quality control.** Records are filtered at several points. The design agent's own inspect-and-revise loop rejects layouts with misalignment, empty regions, or unreadable text before a sample is emitted. Structural extraction validates that text nodes are fully removed from visual targets and that every illustration crop is isolated with clear margins. Composition validates alpha coverage of the matte and geometric fit within the target footprint, and layer reconstruction is checked against the composite so that within-group elements do not overwrite one another and every cross-group overlap follows SVG paint order. Samples failing these checks are regenerated or dropped.

### A.4 Dataset composition, splits, and provenance

The merged corpus contains 23,960 paired records, combining an initial generation pass (roughly 8.4K pairs) with the style-augmentation and dense-reuse passes (roughly 15.5K pairs); the passes are merged so that no structural layout appears across the boundary in conflicting roles. Every record carries the full training tuple: the structural input, the text-free and final targets, the per-layer amodal/visible masks, the overlap and hierarchy graph, and the structured conditioning metadata. Illustration styles are distributed roughly evenly across the six style families, and the samples span the fourteen layout archetypes. Figure 8 shows representative final targets.

**Leakage-free, course-disjoint splits.** Splits are constructed at the granularity of the *source course*, not the individual sample, so that no concept from a course used for evaluation can also appear in training. Courses are ordered by a fixed hash of their identifier, and a subset-sum procedure selects a set of whole courses summing to 200 samples for the test split and, from the remainder, another 200 for validation; all remaining courses form the training split. Because a course contributes all of its

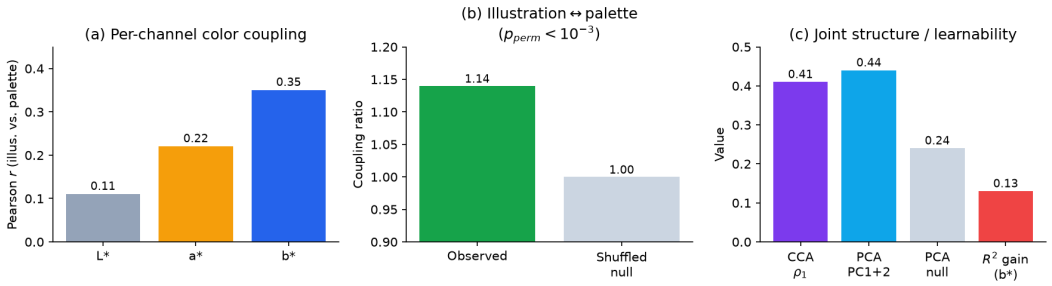


Figure 9: **Cross-element color structure** ($n = 1{,}000$)**.** (a) Per-channel Pearson correlation in CIELAB between mean illustration color and mean palette color; coupling is strongest on the $b^*$ (blue-yellow) axis. (b) Observed illustration↔palette coupling exceeds a shuffled null ($p_{\text{perm}} < 10^{-3}$). (c) Joint-structure and learnability summaries: canonical correlation $\rho_1$, variance explained by the first two joint principal components versus a null, and the $R^2$ gain on $b^*$ from conditioning on the palette. Together these show the dataset induces coordinated, non-random color composition across layers.

concepts to exactly one split, the evaluation sets measure generalization to unseen courses and topics rather than memorization of a familiar course's visual style.

**Provenance.** Each record retains its course identifier and title, the assigned layout archetype, the editable-visual descriptions and edit prompts, the diffusion style and seed, and the compositing parameters, so that any sample can be traced back to its concept and regenerated. Concepts are used as textual seeds for synthesis; the visual content of every target is generated by our own pipeline.

### A.5 Dataset analysis: color composition is coupled and learnable

A natural question for a synthesized corpus is whether it contains *learnable structure* beyond each element in isolation, in particular, whether the colors of different elements within a graphic are related, as they are in human-designed infographics where palette choices are made jointly. We quantify this on a $1{,}000$-sample subset by comparing, within each graphic, the colors of the vector design and the colors of the diffusion-generated illustrations, all in perceptual CIELAB space.

Two findings emerge (Fig. 9). First, the vector side is palette-coherent by construction: text fills are drawn *verbatim* from the design palette (median $\Delta E = 0$ to the nearest palette swatch), confirming that the LLM designs a single coherent color scheme per graphic. Second, and less trivially, the *diffusion-generated illustrations are measurably coupled to that vector palette* even though the editor sees only a per-illustration subject prompt. Aggregate illustration color and palette color are positively correlated per channel, most strongly on the blue-yellow ($b^*$, $r = 0.35$) and red-green ($a^*$, $r = 0.22$) axes and weakest on lightness ($L^*$, $r = 0.11$); a permutation test rejects independence ($p_{\text{perm}} < 10^{-3}$, observed-to-null coupling ratio $1.14\times$); and a canonical correlation analysis between illustration and palette color reaches a first correlation of $\rho_1 = 0.41$. The coupling is also *learnable*: knowing the palette improves prediction of an illustration's $b^*$ by $\Delta R^2 = +0.13$ beyond the style label alone, and the leading two principal components of the joint color representation explain $0.44$ of the variance versus $0.24$ under a shuffled null.

Mechanistically, this coupling is *authorial* rather than injected: the same language model writes both the design palette and each illustration's subject prompt, so color intent leaks through word choice, and it survives the editor's independent re-styling. This is a desirable property for our setting: it means the corpus rewards a model that coordinates appearance *across* layers rather than styling each layer in isolation, which is precisely the behavior the correlated, communicating streams of LayerCanvas are designed to capture.

### A.6 IMPLEMENTATION DETAILS

The design agent is driven by a large language model (Claude Opus 4.8 (Anthropic, 2026)), accessed through a standard chat-completion API, and invoked with a render-and-inspect tool loop that supplies the rasterized SVG back to the model for revision. SVGs are rendered with `rsvg-convert` (Lachowicz et al., 2014) at the fixed $1536 \times 1024$ canvas. Illustration aestheticization uses the Qwen-Image-Edit (Qwen Team, 2025b) diffusion image editor in `bfloat16`, with classifier-free guidance $5.0$, $40$ denoising steps, and a fixed seed for reproducibility; crop sizes are snapped to multiples of $32$ within $[256, 1024]$. Standalone asset generation, where needed, uses the Qwen-Image (Qwen Team, 2025a) text-to-image model. Background removal uses rembg (Gatis, 2025) with the u2net matting model; a border-connected white-key matte is available as a fast alternative for saturated subjects. Color analysis is performed in CIELAB (via `scikit-image` (van der Walt et al., 2014)) with $k$-means dominant-color extraction and permutation, canonical-correlation, and PCA baselines.

The stages have different resource needs, so we separate them. The language-model stage requires only API access and runs on a CPU host, while the GPU stages (aestheticization, matting, composition, and layer rendering) run on NVIDIA H100 and H200 GPUs. Large batches are executed on $8\times$H100 and $8\times$H200 nodes. The style-augmentation and dense-reuse passes require no language-model access, since they reuse the previously validated structural layers and only rerun the GPU restyle-and-compose steps.

# B Additional Details and Experiments

This section provides the complete method specification followed by supplementary experimental details, comparisons, and analyses.

## B.1 Additional method details

**Renderer-derived masks and hidden areas.** The offline renderer removes all SVG text nodes and records a text-free target, per-element RGBA assets, element types, normalized boxes, parent-child links, paint order, and masks. For element $e_i$, the full (amodal) mask $A_i$ covers its entire area before later elements are painted, while $V_i$ retains only the pixels visible in the final rendering. Their difference $H_i = (A_i - V_i)_+$ identifies content hidden by an element in front. Because per-element assets are rendered before composition, $H_i$ contains the element's true appearance rather than a blank region. This provides a direct training target for content revealed by deletion, translation, or z-order edits. Illustration prompts and text regions are routed to their owning visual elements, although text remains absent from image targets.

**Properties of ordered grouping.** The rule in Eq. (4) never places an element in the same or an earlier group than an already processed element that it overlaps. Thus every overlap keeps SVG paint order, while Eq. (3) guarantees that elements sharing a generated layer cannot overwrite one another. The groups are not based on meaning or element type: different element types may share a generated layer whenever their full masks do not overlap. The layers are also not generated one at a time; the group index determines composition order but all layers are sampled simultaneously. After a geometric edit, grouping and graph information are recomputed without resampling unaffected layers.

**Correlated layer-wise flow matching.** Let $x_{1,g} \in \mathbb{R}^{K \times D}$ be the packed internal VAE features of the clean target for group $g$. We combine shared and private Gaussian noise,

$$\epsilon_g = \sqrt{\rho}\,\epsilon_{\mathrm{sh}} + \sqrt{1-\rho}\,\epsilon_g^{\mathrm{pr}}, \quad \epsilon_{\mathrm{sh}}, \epsilon_g^{\mathrm{pr}} \sim \mathcal{N}(0, I), \tag{9}$$

where the shared component encourages a consistent scene-level look and the private component preserves layer-specific variation. The current schedule samples $\rho \sim \mathcal{U}(0.5, 1)$. Each example uses one common timestep $t$, so communication always relates layers at the same noise level. The implemented linear path is

$$x_{t,g} = (1-t)\epsilon_g + t x_{1,g}, \qquad u_g = x_{1,g} - \epsilon_g, \qquad t \sim \mathcal{U}(0,1), \tag{10}$$

with primary objective

$$\mathcal{L}_{\mathrm{layer}} = \frac{\sum_g m_g \|v_{\theta,g} - u_g\|_2^2}{\sum_g m_g}, \qquad \widehat{x}_{1,g} = x_{t,g} + (1-t)v_{\theta,g}, \tag{11}$$

where $m_g$ ignores empty layer slots added for batching. During generation, these partly shared starting states are updated in parallel. Shared noise only helps align the overall look: without the communication block, the frozen backbone still processes each group independently and cannot react to another layer's evolving content or boundary. Figure 2 shows the full noise-to-layer path and expands one repeated denoising step.

**Communication summaries and graph bias.** At a selected backbone depth, each layer is summarized using nine compact feature groups: summary statistics from the full and hidden areas, averages from visible areas and boundary bands, $2 \times 2$ pooled spatial context, type, box, paint order, timestep, local text context, and a global feature average. These values are combined into the group summary $h_g$. The graph tensor $E_{gh} \in \mathbb{R}^4$ stores directed back-to-front overlap, front-to-back overlap, parent-to-child links, and child-to-parent links. A learned linear map produces one added bias per attention head, as formalized in Eq. (7); invalid padded groups are masked from both keys and outputs.

**Style and boundary paths.** The communicated summary $\widetilde{h}_g$ is returned to the image features through the two paths in Eq. (8). The style path predicts a coarse $2 \times 2$ cell grid and enlarges it to provide smooth appearance corrections. The boundary path combines local hidden states with $\widetilde{h}_g$ and is multiplied by a narrow band $B_g$, limiting its effect to silhouettes and cross-layer contacts. Their learned gate biases are initialized to $[-2.0, -2.5]$, giving initial gate strengths of approximately $0.12$ and $0.076$. These gates start small but not at zero. The block makes no change at the start of training because the style and boundary output layers start at zero.

**Composition and structure losses.** The loss for each layer does not by itself check the result after the layers are combined. The composition loss sorts groups by paint order and compares their masked reconstruction with the clean composite. $\mathcal{L}_{\text{hid}}$ supervises the hidden area $H_g = (A_g - V_g)_+$, and $\mathcal{L}_{\text{bnd}}$ matches values and spatial gradients near ancestor boundaries. For every auxiliary term, we first recover the one-step clean estimate $\widehat{x}_{1,g}$ for each group from its noisy state and predicted velocity using Eq. (11). This estimate remains in the packed model representation; training does not decode a full RGB image through the VAE for these losses. Only the composition loss combines the estimated groups: it applies their amodal masks, sorts them by SVG paint order, and compares the result with the corresponding composition of the clean target groups. The remaining auxiliary losses operate directly on individual clean group estimates.

The full objective is

$$\mathcal{L} = \lambda_{\text{layer}}\mathcal{L}_{\text{layer}} + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}} + \lambda_{\text{hid}}\mathcal{L}_{\text{hid}} + \lambda_{\text{bnd}}\mathcal{L}_{\text{bnd}} \tag{12}$$

**Exact typography.** All SVG text nodes are removed from diffusion targets, and prompts request blank title, label, and caption regions. We retain the Unicode string, role, bounding box, font family and style, alignment, transforms, hierarchy, and owning visual element. During generation, the text-free composite is placed beneath a transparent vector-rendered text layer $R_{\text{text}}(\mathcal{T})$, as specified in Eq. (2). Spelling, line breaks, alignment, and font edits therefore require only vector re-rendering, not another diffusion pass.

### B.2 Evaluation details

**Compared methods.** The primary comparison (Table 1) evaluates four methods: (i) **SVG (LLM-generated)**, a reference in which an LLM (Claude Opus 4.8 (Anthropic, 2026)) recolors and completes the structural SVG, which is then rasterized with `rsvg-convert` (Lachowicz et al., 2014); this yields exact vector text by construction but is not a photorealistic synthesizer; (ii) **Qwen-Image-2512** (Qwen Team, 2025a), a prompt-only diffusion model conditioned on the design-concept prompt which is itself derived from the same SVG layout (for fairness in evaluation); (iii) **Ideogram-4** (Ideogram AI, 2026), a prompt-only open-weights system whose prompt is constructed from the same SVG layout, with its structural elements serialized into the structured prompt format the model expects; and (iv) **LayerCanvas**, grouped layer-set diffusion followed by exact SVG text compositing. Prompt-only baselines receive the concept prompt reconstructed from each record's structured conditioning; LayerCanvas additionally consumes the structural layer set.

**Generation configuration.** Diffusion baselines and LayerCanvas are run at matched prompts and seeds; the editor backbone runs in bf16 with classifier-free guidance 5.0 and 40 denoising steps. LayerCanvas adds text exactly from the source SVG after raster synthesis.

**Text-fidelity protocol.** Text fidelity is measured two ways. *Placement-exact* fidelity uses ground-truth-region OCR: for each ground-truth text box (taken from the stored structured conditioning) we crop the corresponding region and OCR it with Tesseract (Tesseract OCR contributors, 2026). The primary results use a strict $768\times512$ evaluation canvas. Both exact-text methods are rendered with `rsvg-convert` (Lachowicz et al., 2014) on the same host with Georgia, Helvetica, and Arial available through fontconfig. We report character accuracy ($1-$ normalized edit distance, averaged over regions) and exact-match rate. *Placement-agnostic* fidelity instead runs location-independent full-image OCR and reports word recall, the fraction of ground-truth content tokens (length $\geq 3$, NFKC-normalized) recovered anywhere in the image; this credits a baseline that renders legible text at the wrong position. For the two exact-text methods the remaining gap from $1.0$ reflects OCR error rather than genuine text mistakes, since the text is placed by construction. The two prompt-only

Table 3: Full evaluation, including the *craft* judge criterion omitted from Table 1. Text fidelity is measured at $768\times512$ and includes placement-exact ground-truth-region OCR (char / exact) and placement-agnostic full-image word recall (*Recall*); judge criteria are 0-5, higher better; *Avg* is the mean of appeal, typography, and coverage. †LLM-authored SVG recolor reference.

| | Text fidelity | | | VLM judge (0-5) | | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Char | Exact | Recall | Appeal | Typog. | Cov. | Craft | Avg |
| SVG (LLM-authored)† | 0.856 | 0.676 | 0.964 | 2.37 | 4.02 | 4.90 | 3.91 | 3.76 |
| Qwen-Image-2512 | 0.080 | 0.002 | 0.149 | 2.69 | 1.00 | 1.45 | 1.51 | 1.71 |
| Ideogram-4 | 0.657 | 0.360 | 0.744 | 2.92 | 2.29 | 3.89 | 2.49 | 3.03 |
| LayerCanvas (Ours) | 0.822 | 0.623 | 0.942 | 2.99 | 3.81 | 4.75 | 3.59 | 3.85 |

Table 4: **OCR resolution sensitivity.** Placement-exact character accuracy (*Char*) and exact-match rate (*Exact*), together with placement-agnostic word recall (*Recall*).

| | $768\times512$ OCR | | | 4096-pixel OCR | | |
|---|---|---|---|---|---|---|
| Method | Char | Exact | Recall | Char | Exact | Recall |
| SVG (LLM-authored)† | 0.856 | 0.676 | 0.964 | 0.883 | 0.595 | 0.814 |
| Qwen-Image-2512 | 0.080 | 0.002 | 0.149 | 0.077 | 0.002 | 0.150 |
| Ideogram-4 | 0.657 | 0.360 | 0.744 | 0.643 | 0.268 | 0.727 |
| LayerCanvas (Ours) | 0.822 | 0.623 | 0.942 | 0.853 | 0.543 | 0.806 |

baselines behave differently under the two scores: Qwen-Image-2512 is near zero on both placement-exact OCR and word recall because its text is genuinely garbled and rarely lands in the ground-truth boxes, whereas Ideogram-4 recovers most text on word recall but trails on placement-exact OCR, since its glyphs are partially readable yet often mislocated.

**Judge protocol.** Aesthetic and typographic quality is scored by a VLM judge using the human-preference variant of our static-quality rubric. The primary judge (Claude Opus 4.8) scores each image on 0-5 across three criteria-*illustration appeal*, *typography quality*, *semantic coverage* plus a holistic overall score that we omit in the reported tables in favor of *Avg*, the transparent mean of illustration appeal, typography, and semantic coverage. The appeal criterion uses strict, anchored bands so that coarse, flat, or generic illustrations stay in the low range regardless of how clean the surrounding layout is, and richly rendered, detailed artwork is required for the top bands; this keeps the score aligned with what a human would actually prefer without the rubric explicitly naming photorealism. The design concept passed to the judge is reconstructed from the stored structured conditioning, and judgments are disk-cached for reproducibility. We additionally repeat the identical image-rubric judgments with Gemini 3.7 Flash (Google DeepMind, 2026) and GPT-5.6 Sol (OpenAI, 2026). All three judges receive the same $768\times512$ resolution.

**Full results.** Table 3 reports the judge criteria alongside both text-fidelity metrics.

Across judges, LayerCanvas's Avg is $3.88 \pm 0.08$ (population standard deviation), compared with $1.90 \pm 0.13$ for Qwen and $3.30 \pm 0.21$ for Ideogram. This slightly exceeds the SVG reference's $3.83 \pm 0.07$ cross-judge mean. Judges differ in absolute calibration but preserve the learned-method ranking.

### B.3 Additional comparisons

Figures 10 and 11 provide additional qualitative comparisons, showing that LayerCanvas preserves the source layout and legible text while enriching the illustrations and avoiding the text failures of raster baselines.

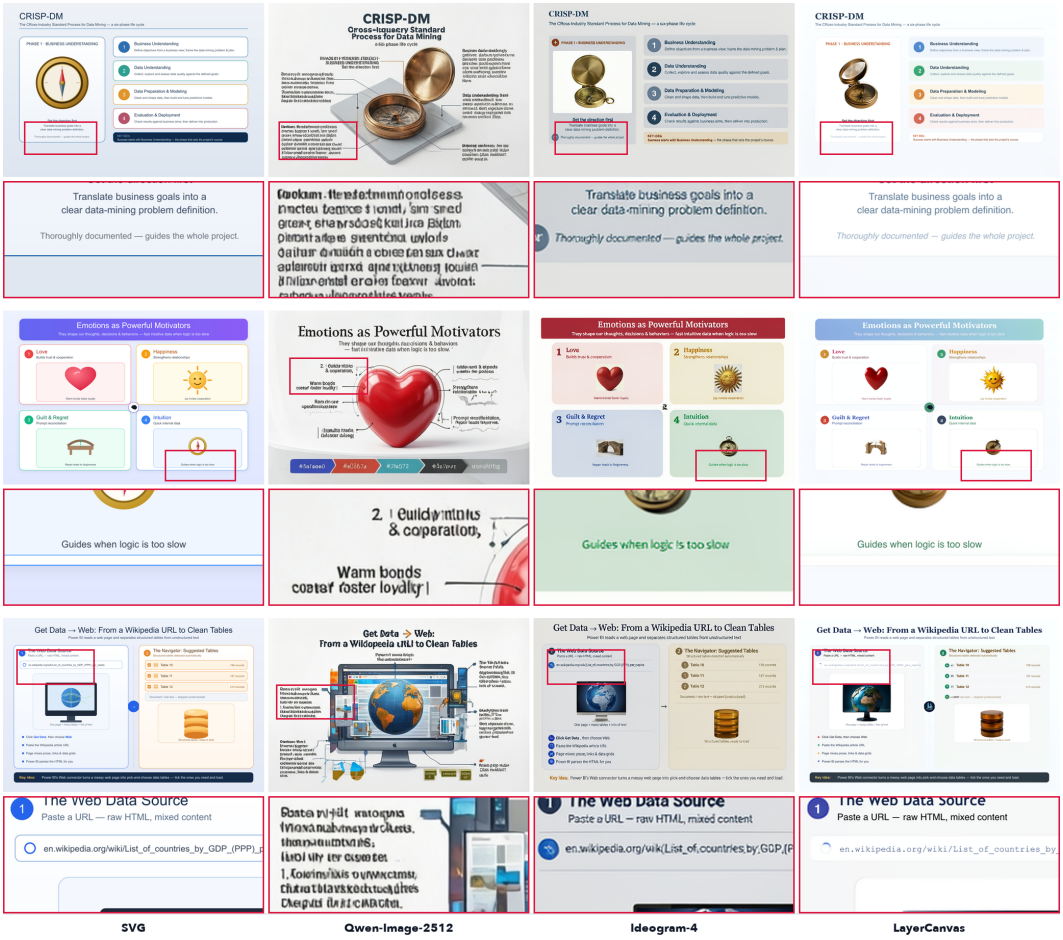


Figure 10: **Additional qualitative comparisons.** Columns show the LLM-authored SVG reference, Qwen-Image-2512, Ideogram-4, and LayerCanvas, respectively.

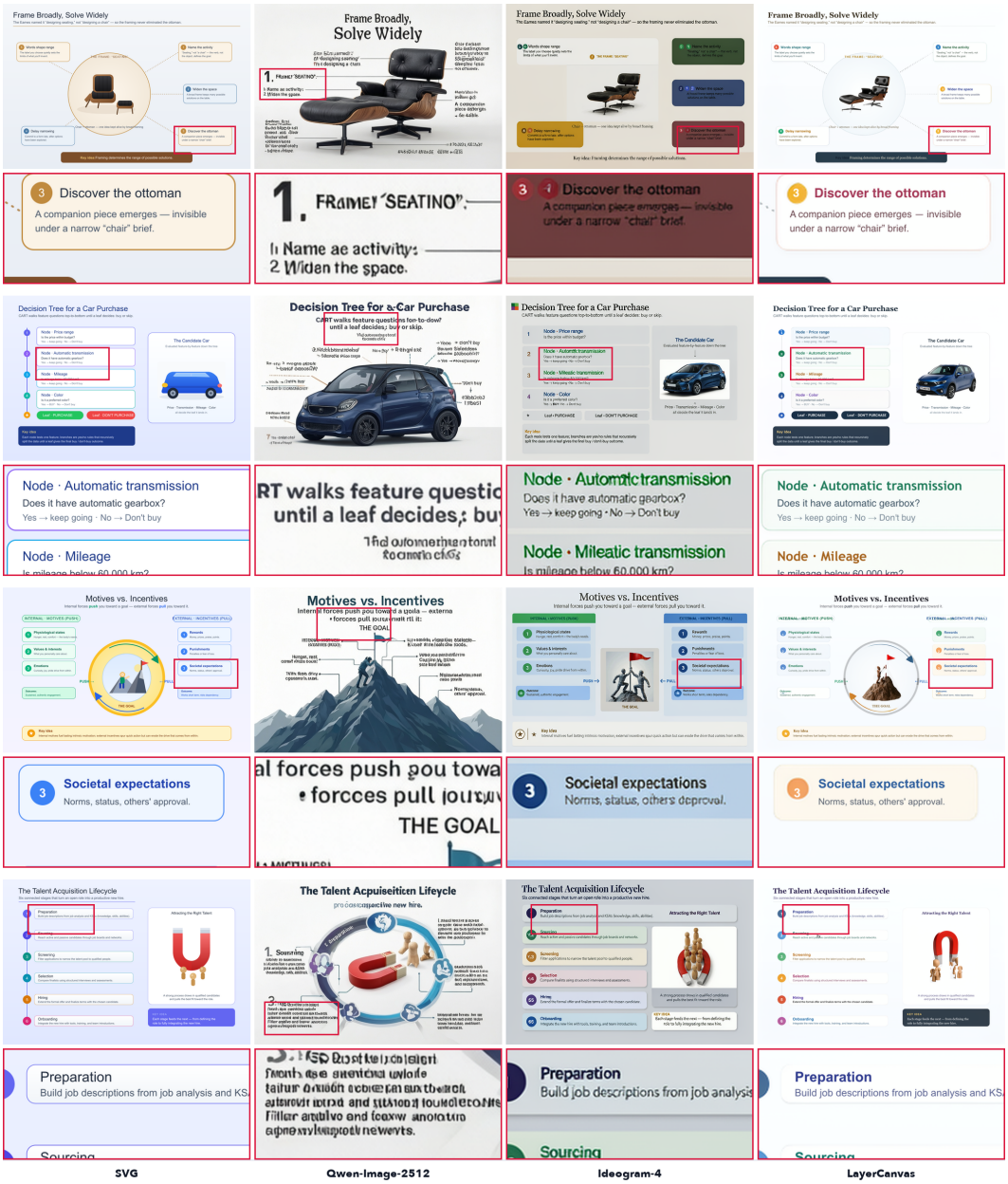


Figure 11: **Additional qualitative comparisons.** Columns show the LLM-authored SVG reference, Qwen-Image-2512, Ideogram-4, and LayerCanvas, respectively.

Table 5: **Multi-judge robustness at** $768\times512$**.** Each entry is the mean of illustration appeal, typography quality, and semantic coverage. LayerCanvas is the highest-scoring learned synthesis method under every judge.

| Judge | SVG | Qwen | Ideogram | LayerCanvas |
|---|---|---|---|---|
| Claude Opus 4.8 (Anthropic, 2026) | 3.76 | 1.71 | 3.03 | **3.85** |
| Gemini 3.7 Flash (Google DeepMind, 2026) | 3.93 | 1.99 | 3.54 | **3.99** |
| GPT-5.6 Sol (OpenAI, 2026) | 3.81 | 1.99 | 3.32 | **3.81** |

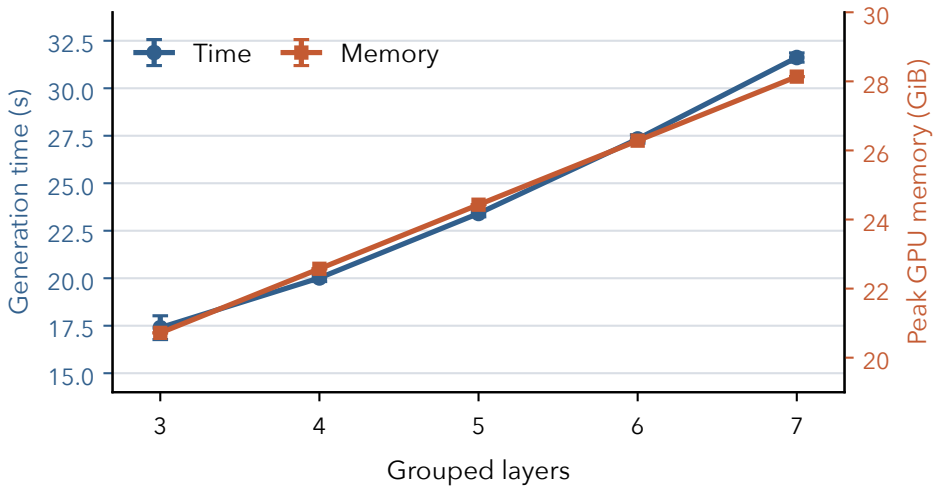


Figure 12: **Diffusion inference compute.** Generation time and peak allocated GPU memory on one H200 increase with the number of grouped layers. Each point is the mean of two 20-step generations; error bars show the measured minimum and maximum.

### B.4 Systems profile

#### B.4.1 Training compute

With full gradient checkpointing, a single-H200 training step takes 10.33 s and uses 30.8 GiB of peak GPU memory. Backward recomputation takes 7.18 s, forward execution 2.29 s, and input preparation 0.83 s. Checkpointing half of the transformer blocks reduces the step time to approximately 9.2 s but raises peak GPU memory to approximately 108 GiB. Quarter and no-checkpointing profiles exceed 133 GiB and run out of memory on a 141 GB H200.

#### B.4.2 Inference compute

**Layer generation.** A matched single-H200 profile uses two validation examples at each grouped layer count from three through seven. End-to-end generation takes 24.0 s per sample on average and 23.4 s at the median, with a measured range of 16.8-31.9 s over 10 generations. Peak allocated GPU memory rises from 20.7 GiB at three grouped layers to 28.1 GiB at seven. The trend is shown in Figure 12.

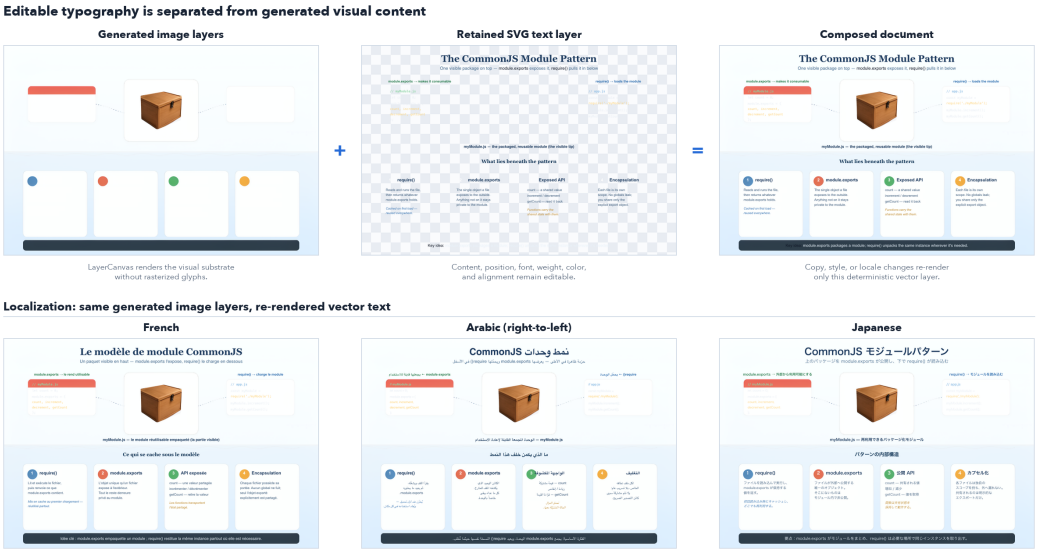


Figure 13: **Editable text representation and localization.** A held-out LayerCanvas result is generated without glyphs, while the source SVG text nodes remain a separate editable vector layer. Their composition produces the final document. The lower row localizes the same unchanged generated image layers into French, Arabic, and Japanese by re-rendering only the text layer.

# C Downstream Applications

This section demonstrates how the hybrid raster-SVG layer-set representation supports deterministic editing and reuse after generation. Each application uses the trained grouped layer-set model or a previously generated output from that model; operations performed only on the persistent structured representation are identified explicitly.

## C.1 Text editing and localization

End-to-end raster generators must repaint text whenever wording, typography, or language changes. Because their glyph generation is data-driven, they may also favor the visual conventions of a dominant training language rather than faithfully rendering a requested, less represented script. LayerCanvas instead generates only the visual substrate and preserves typography as styled SVG nodes. Copy, style, and locale changes therefore require no diffusion pass and cannot alter the generated artwork.

**Implementation.** Each text node retains its Unicode content, position, font family, size, weight, style, anchor, fill color, transforms, and parent ownership. Editing replaces only the requested attributes before the text-only SVG is rasterized and alpha-composited over the unchanged model output. Localization uses a script-appropriate local font and shaping direction; nonlinguistic values such as numbers and code may remain unchanged. Figure 13 shows the same generated layers with text rendered in four languages.

**Deterministic behavior.** This operation is a property of the representation rather than a learned prediction: the requested node attributes are written directly into SVG and rendered by the vector-text runtime. Thus the rendered language follows the provided Unicode content and selected font rather than a language distribution learned by the image model. The only external requirements are an available font with the requested glyphs and enough space in the retained text box for the new copy. If a translation changes line length substantially, standard SVG layout operations can resize or reflow the text box without regenerating any image layer.

### C.2 INDEPENDENT LAYER ANIMATION

We adapt the zero-shot animation principle of Text2Video-Zero (Khachatryan et al., 2023), which shows that the motion and appearance priors of a pretrained text-to-image diffusion model can produce a temporally linked sequence without video-specific training. A flattened document would still expose the complete page to this generative update, allowing unrelated illustrations, layout, and typography to drift. LayerCanvas instead lets the temporal operation target one named image layer while all other layers remain fixed.

**Implementation.** Following Text2Video-Zero, we derive a short motion trajectory from an image diffusion prior rather than training a video model. The user selects an image layer, optionally together with a bounded subregion, and specifies endpoint states or a motion transform. Intermediate states are produced through latent interpolation or motion-guided warping, decoded only for the selected content, and alpha-composited over the unchanged layer set at each timestep. This final step is specific to our representation: every unselected image layer and the full-resolution SVG typography are copied identically into every frame. In Figure 5, the selected subregion is an eye band covering $6.6\%$ of the canvas.

**Discussion.** The generative component supplies motion only; spatial isolation comes from the LayerCanvas representation. Pixels outside the selected region and the complete vector-text layer are not predicted at subsequent timesteps, so they remain bit-identical by construction. This converts a global zero-shot image animation method into a local document operation and avoids spending temporal model capacity on content that should not move.

### C.3 ASSET REUSE AND RECOMPOSITION

A flattened image binds every illustration, diagram, and decorative element to its original page. Reusing one of them requires segmentation followed by inpainting or regeneration of the surrounding content. LayerCanvas instead keeps generated visual groups as addressable image layers with their own alpha and paint order, so an element can become a reusable asset after generation.

**Implementation.** A selected layer is copied with its image content and alpha mask. Standard layer operations then translate or scale it, adjust its appearance within the mask, and assign a new position in the target paint order. Multiple selected layers may be placed onto a new background and combined with newly authored SVG text. These operations act on the saved representation and do not require segmentation, inpainting, or another diffusion pass. Figure 14 shows three layers reused in a new composition.

**Discussion.** Reuse preserves the generated appearance of an asset while decoupling it from the document in which it first appeared. A collection of generated layers can therefore serve as a project-specific asset library: recurring illustrations or diagram components may be placed across related documents while layout and text remain independently editable. The resulting composition is still structured, so the reused layer can be moved, hidden, or replaced again later.

### C.4 LAYOUT RETARGETING AND RESPONSIVE RESIZE

A design may need to appear as a presentation slide, poster, and mobile graphic. Resizing a flattened page either distorts its elements or removes content through cropping. LayerCanvas retains the document as addressable groups, allowing the same generated content to be arranged for a different canvas.

**Implementation.** Each illustration, diagram, card, and text block carries a bounding box and group assignment. A simple reflow solver packs these groups in reading order, using more columns on wide canvases and a vertical stack on narrow ones. Generated image groups are scaled uniformly, while SVG text can be re-rendered for the new box. The operation changes only layout metadata and recomposition; the image layers are not generated again. Figure 15 shows the same content adapted to slide, poster, and mobile layouts.

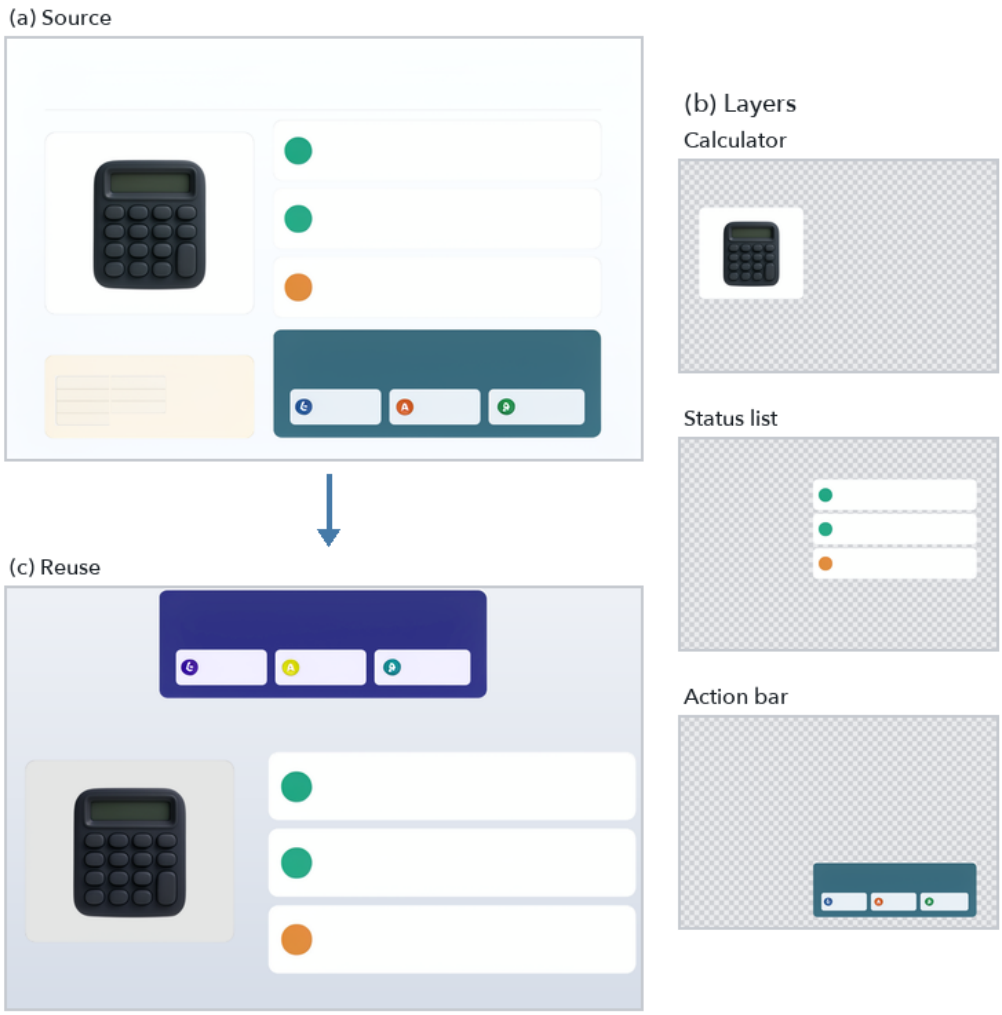


Figure 14: **Asset reuse.** Three generated layers are taken from one document and arranged into a new composition. Each layer carries its own alpha, allowing independent repositioning, scaling, restyling, and reordering.

**Discussion.** Retargeting becomes a document-layout problem rather than an image-generation problem. Groups may be reprioritized, resized, or moved between rows without changing their generated appearance, and the result remains editable after recomposition. More advanced layout objectives or design templates can replace the simple solver without changing the underlying representation.

### C.5 Command-driven interactive editing

Interactive authoring is naturally expressed through commands such as "remove this card," "emphasize this panel," or "swap this icon." Prompt-based regeneration may redraw the entire canvas. Named layer groups instead allow the command to be grounded to one element and executed as a local, reversible edit.

**Implementation.** A lightweight interpreter maps a command to $\{\text{ACTION}, \text{TARGET}\}$ using action phrases and aliases for the document's semantic groups. We support removing, emphasizing, swapping, and adding elements. The executor applies the operation to the targeted SVG group or




Figure 15: **Layout retargeting.** The same generated module illustration, code panels, and explanatory cards are rearranged for slide, poster, and mobile canvases. Image groups preserve their aspect ratios, while text remains separately renderable for each target layout.

model-rendered layer. Full-resolution alpha masks define the affected region, allowing the result to be recomposited without resampling unrelated content. Figure 16 shows remove, swap, and add commands applied to selected regions.

**Discussion.** The language interface does not edit pixels directly. It selects an operation and resolves a semantic target, after which the persistent layer representation defines the exact region and compositing behavior. Commands can therefore be previewed, reversed, or chained without resampling the full document. A learned language parser could replace the lightweight interpreter without changing the underlying editing operations or their locality.

### C.6 SVG-SPACE REFINEMENT AND FAITHFUL DIFFUSION PROPAGATION

An upstream layout proposal may contain a misplaced, mislabeled, or incorrectly styled element. In a raster-only pipeline this often requires a new prompt or manual pixel editing. In LayerCanvas, the SVG is a persistent control surface: the element can be corrected directly, and the model propagates that correction to the raster without retraining or redesigning the rest of the document.

**Implementation.** We deliberately perturb one SVG element by translating, resizing, recoloring, relabeling, or replacing it. The incorrect and corrected conditions use the same sampling noise so that differences are attributable to the structural edit. Composition uses each group's full-resolution vector alpha mask rather than an upsampled latent-grid mask. Figure 6 presents the incorrect condition, the SVG edit, and the corrected render as a matched triptych.

**Discussion.** The structured intermediate remains editable after an initial generation rather than becoming a disposable prompt. A user or upstream layout system can inspect the result, correct the responsible SVG group, and regenerate the affected visual content under the revised condition. Fixed noise helps preserve the remaining appearance, while recomputing masks and overlap relations allows changes in position, scale, or occlusion to propagate consistently.

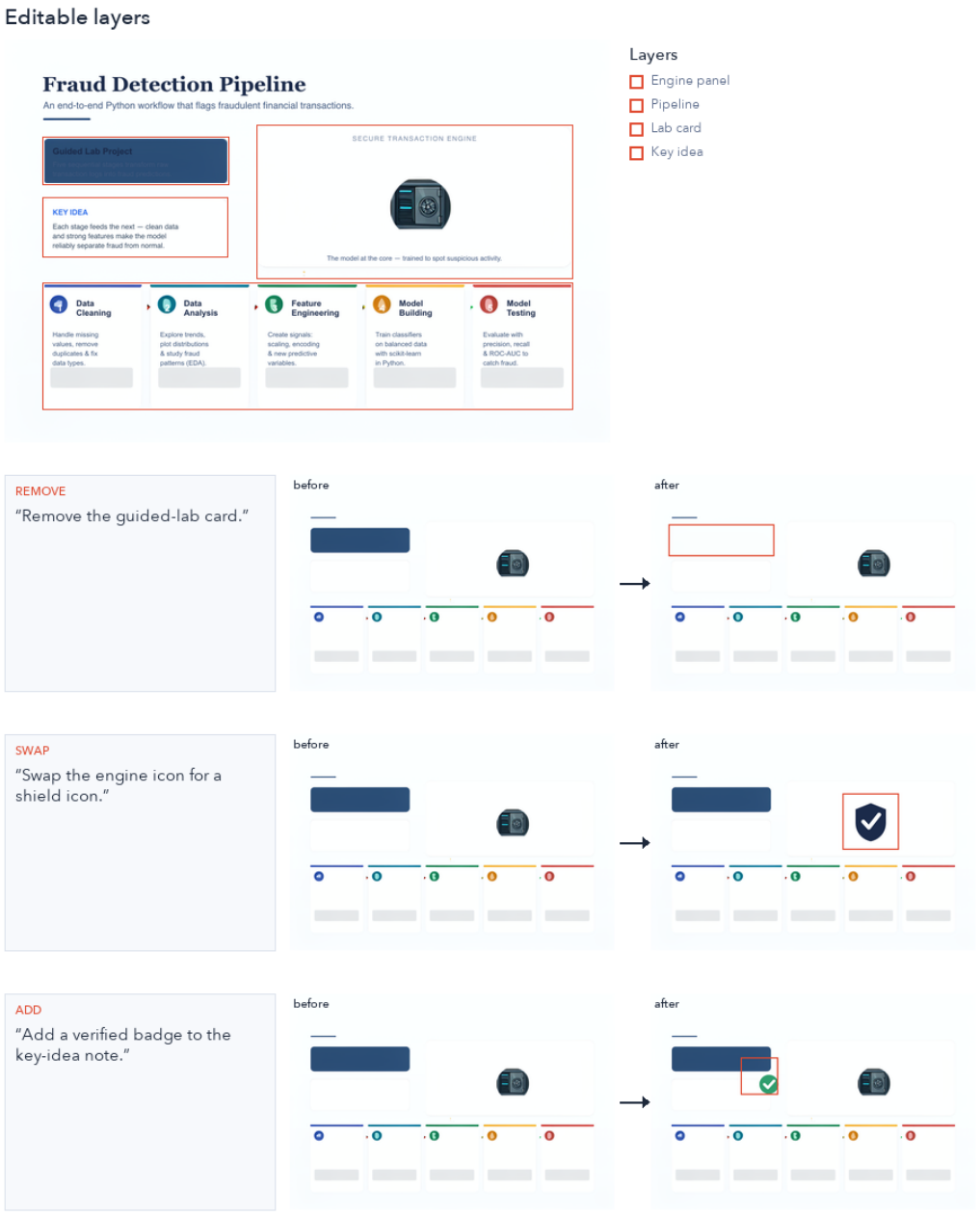


Figure 16: **Command-driven editing.** Three natural-language commands are converted to targeted remove, swap, and add operations. Red boxes indicate the edited regions; unselected layers are reused unchanged.

## D LIMITATIONS, ETHICS, AND BROADER IMPACT

LayerCanvas requires a valid source SVG and inherits errors and biases in its content, layout, and training data. Deterministic typography preserves supplied text but does not ensure factuality, accessibility, or culturally appropriate localization, while edits that change occlusion or z-order may affect unintended regions. Responsible deployment therefore requires documented data provenance, license and bias checks, generated-content disclosure, and human review.