PixelART: Image-to-Layer Decomposition without Latents or Text-to-Image Pretraining

Zelin Jia[1] **Zhao Zhang**[2] **Zhicong Tang**[2] **Yuhui Yuan**[2*] **Shixia Liu**[1*]
[1]School of Software, BNRist, Tsinghua University [2]Canva Research
ryanyuan@canva.com, shixia@tsinghua.edu.cn

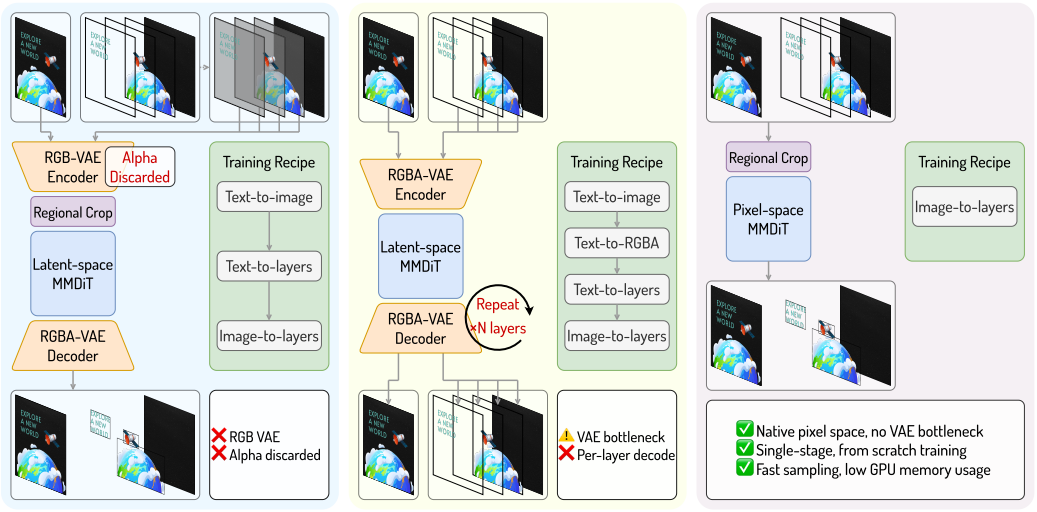


Figure 1: Comparison with previous Image-to-Layer decomposition systems. Left: ART [1] and its variant; Middle: Qwen-Image-Layered [2]; Right: Ours. Our PixelART is much simpler and trained from scratch.

Abstract

Image-to-layer (I2L) decomposition converts a flattened image into editable RGBA layers, enabling element-level editing in design workflows. Existing diffusion-based systems typically adapt large pretrained text-to-image (T2I) models and introduce RGBA autoencoders or variable-layer architectural modules. We revisit this design choice and ask whether I2L decomposition truly requires these heavyweight components. In light of this, we formulate I2L decomposition as a *pixel-native assignment-and-completion problem*: assigning visible pixels to editable layers and completing occluded content behind them. Based on this formulation, we introduce PixelART, a pixel-space rectified-flow Transformer trained from scratch for I2L decomposition. We further observe that high-noise timesteps determine layer assignment and coarse layer organization, while low-noise timesteps mainly refine color, alpha, texture, and boundaries. This motivates a *terminal-boosted timestep sampling* strategy that increases training coverage in the high-noise assignment regime. Trained on approximately 4M multi-layer design templates, PixelART achieves state-of-the-art layer decomposition and composite reconstruction on DESIGN-MULTI-LAYER-BENCH and LICA with over 80% fewer parameters, 98% lower latency, and 85% lower memory than the recent Qwen-Image-Layered model. Ablation experiments show that pixel-space $\mathbf{x}$-prediction, terminal-boosted timestep sampling, and data/model scaling are critical.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

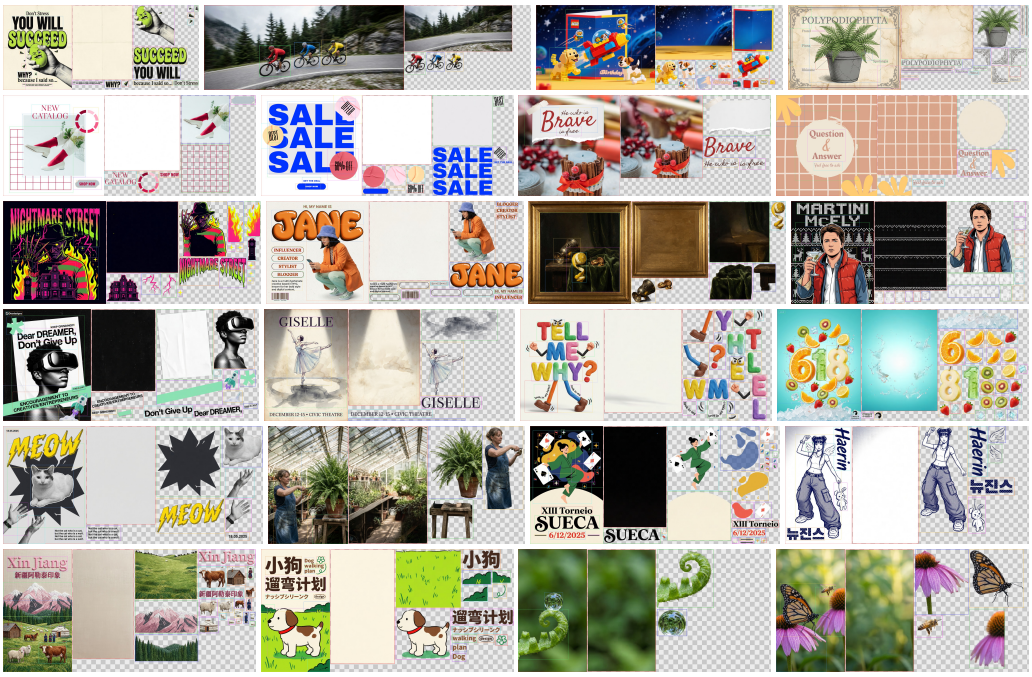


Figure 2: Image-to-layer decomposition results with our PixelART-H/32.

1 Introduction

Flat image generators, such as Nano-Banana [3, 4, 5] and GPT-Image [6, 7, 8], can synthesize visually rich raster images, but their outputs remain difficult to edit. In contrast, professional design tools rely on layered representations, where text, graphics, backgrounds, shadows, and decorative elements can be edited independently [9]. Image-to-layer (I2L) decomposition aims to recover this editable structure by converting a flattened RGB image into an ordered stack of RGBA layers, enabling element-level editing, restyling, and recomposition. Recent diffusion-based systems, including ART [1], Qwen-Image-Layered (QIL) [2], OmniPSD [10], and OmniAlpha [11], have made rapid progress on layered generation and decomposition. Despite their differences, they largely follow a common recipe: adapt a large pretrained text-to-image (T2I) backbone, extend an RGB autoencoder into an RGBA autoencoder, and add task-specific mechanisms for transparent and variable-layer outputs. While natural, this design is complex, and the latent autoencoder can become a bottleneck when modeling design layers with transparency, small text, sparse strokes, flat colors, and sharp alpha boundaries.

We revisit this recipe and ask *whether* strong I2L performance can be obtained without latent autoencoding or T2I *pretraining*. To this end, we propose PixelART, a pixel-space rectified-flow Transformer trained from scratch. PixelART directly denoises regional RGBA pixel patches using a single-stream Transformer, avoiding RGBA-VAEs, pretrained T2I backbones, layer-specific decoders, and variable-layer modules (Figure 1). We formulate the I2L task as a pixel-native assignment-and-completion problem, in which the model must assign visible pixels to layers, complete occluded content, and refine RGBA values. We observe that layer assignment and detail refinement occur at different stages of the diffusion trajectory. High-noise timesteps determine layer assignment and structure; low-noise timesteps refine appearance like color, alpha, texture, and boundaries. We propose a novel *terminal-boosted timestep sampling* strategy to increase training coverage in the high-noise assignment regime.

PixelART achieves state-of-the-art layer decomposition and composite reconstruction quality on DESIGN-MULTI-LAYER-BENCH and LICA while using substantially fewer parameters, lower latency, and memory than VAE-based diffusion baselines. Ablations show that pixel-space $\mathbf{x}$-prediction, high-noise timestep coverage, and data/model scaling are all critical, whereas T2I initialization reduces early loss but yields no consistent final gains in our graphic-design setting. Figure 2 presents additional I2L decomposition visual results. The main contributions of our work are:

- We show that I2L decomposition of graphic designs can be effectively solved by a pixel-space diffusion Transformer trained from scratch, without latent autoencoding or T2I pretraining.
- We introduce PixelART, a single-stream regional Transformer that directly predicts editable RGBA pixel patches without layer-specific decoders or variable-layer modules.
- We identify the importance of high-noise timesteps for layer assignment and propose terminal-boosted timestep sampling to improve decomposition reliability. System-level comparisons and ablations demonstrate that PixelART matches or outperforms larger T2I-initialized pipelines while maintaining a simpler and more efficient design.

2 Related Work

Pixel-space Diffusion Models. Diffusion-based image generation has evolved through three waves. The *first wave* operates directly in RGB pixel space with U-Net backbones, including DDPM [12], ADM [13], GLIDE [14], Imagen [15], DALL-E 2 [16], and Cascaded Diffusion [17]. These models rely on high-capacity convolutional architectures and dense multi-scale processing to handle the high dimensionality of pixel-space generation. The *second wave* shifts denoising into compressed VAE latents, including LDM [18], SDXL [19], eDiff-I [20], DALL-E 3 [21], FLUX [22], SD3 [23], and Qwen-Image [24]. This dramatically reduces per-step computation but introduces a dependence on the VAE representation, which is less natural for tasks such as I2L decomposition whose inputs and outputs are inherently pixel-native. The *third wave* revisits pixel space with Transformer backbones. SiD2 [25], PixelFlow [26], and PixNerd [27] show that ViT-based pixel diffusion can scale competitively to high resolutions when combined with techniques such as loss weighting, cascaded multi-resolution training, and auxiliary patch decoding. JiT [28], pMF [29], and PixelGen [30] go further, showing that plain ViTs with $\mathbf{x}$-prediction over large pixel patches can approach latent diffusion performance without a learned image tokenizer. PixelART builds directly on this third wave. Specifically, we adopt JiT's $\mathbf{x}$-prediction ViT formulation and extend it from T2I to I2L decomposition, for which its pixel-native input–output formulation provides a natural fit.

Multi-Layer Generation and Decomposition. Layered RGBA generation has mainly followed two paradigms. *Sequential* methods instead generate or extract layers one at a time, including LayerDiffuse [31], COLE [32], OpenCOLE [33], LayerD [34], and OmniPSD [10]. *Simultaneous* methods generate all layers in a single forward pass, including Text2Layer [35], LayerDiff [36], ART [1], PrismLayer [37], and DreamLayer [38]. Recent simultaneous I2L systems, notably QIL [2] and CLD [39], adapt pretrained T2I backbones with substantial task-specific engineering, such as RGBA VAEs, variable-layer tokenization, region-conditioned Transformers, and multi-stage training. Commercial systems such as Magic Layers [40], Lovart [41], and Ideogram [42] appear to adopt similar strategies at product scale. Beyond direct multi-layer generation and decomposition, prior studies have also explored matting, segmentation, and alpha-aware representation learning. Matting- and segmentation-based pipelines, including LayerD and trimap-free methods built on SAM [43], SAM2 [44], HQ-SAM [45], and ZIM [46], decompose images through cascaded detection, segmentation, matting, and inpainting. Such pipelines are not optimized end-to-end and can struggle with complex boundaries and semi-transparent content. In parallel, transparency-aware VAE methods, including LayerDiffuse [31], AlphaVAE [47], Wan-Alpha [48], and OmniAlpha [11], extend RGB autoencoders to RGBA latent spaces for downstream generation. Overall, prior methods rely on pretrained T2I model, RGBA-specific tokenizers, or cascaded expert modules. PixelART instead uses a single pixel-space diffusion Transformer trained from scratch, without a VAE or pretrained backbone, achieving competitive performance.

3 Method

3.1 Image-to-Layer as Pixel-Space Inverse Generation

To avoid reliance on pretrained T2I backbones and RGBA-specific components, we formulate I2L decomposition as a *pixel-native inverse problem*: assigning visible pixels to editable layers and completing occluded content behind them. Both subproblems are naturally defined in RGB/RGBA pixel space, motivating direct learning from layered supervision.

Formulation. Given a flattened RGB image $\mathbf{I} \in [0,1]^{H\times W\times 3}$,[2] I2L decomposition aims to recover an ordered stack of editable RGBA layers $\mathcal{L} = \{\mathbf{L}_0, \mathbf{L}_1, \ldots, \mathbf{L}_K\}$, where $\mathbf{L}_i \in [0,1]^{H\times W\times 4}$. Here $\mathbf{L}_0$ denotes the background layer, and $\{\mathbf{L}_i\}_{i=1}^{K}$ are the foreground layers in design z-order. The predicted layers satisfy the following reconstruction constraint under alpha compositing:

$$\Pi_{\mathrm{RGB}}\left(\mathrm{Over}(\mathbf{L}_0, \mathbf{L}_1, \ldots, \mathbf{L}_K)\right) \approx \mathbf{I}, \tag{1}$$

where $\mathrm{Over}(\cdot)$ denotes source-over compositing of the layers in z-order, and Π_{RGB} extracts the RGB channels.

Regional layer parameterization. Following the anonymous regional formulation of ART [1], we assign no semantic labels to layer slots. Instead, each layer is specified only by its spatial region and z-order. Let $\mathcal{R} = \{r_0, r_1, \ldots, r_K\}$ denote the region set, where r_0 is the full canvas and $r_i = (x_i, y_i, h_i, w_i)$ for $i > 0$ is the padded bounding box of the non-transparent support of layer i. During training, $\mathcal{R}$ is derived from the ground-truth design file. At inference, $\mathcal{R}$ can be provided by a user or obtained from an automatic layout detector. PixelART is agnostic to how these regions are obtained.

For each RGBA crop $\mathbf{X}_i = \mathrm{crop}(\mathbf{L}_i, r_i) \in [0,1]^{h_i\times w_i\times 4}$, we pad its spatial dimensions to multiples of the patch size P and partition it into non-overlapping pixel patches:

$$\phi_P(\mathbf{X}_i) \in \mathbb{R}^{N_i\times 4P^2}, \qquad N_i = \lceil h_i/P \rceil \, \lceil w_i/P \rceil \, . \tag{2}$$

The clean target sequence is then formed directly in RGBA pixel-patch space:

$$\mathbf{Y} = [\phi_P(\mathbf{X}_0); \phi_P(\mathbf{X}_1); \ldots; \phi_P(\mathbf{X}_K)]. \tag{3}$$

The flattened input image is converted to normalized RGBA by appending an opaque alpha channel, $\widetilde{\mathbf{I}} = [\mathbf{I}; \mathbf{1}]$. Variable-length regional sequences are padded to a common length, with padding tokens masked out in both attention and loss computation.

3.2 PixelART Architecture

Pixel patch embedding. To process raw RGBA patches with a Transformer of hidden width d, PixelART maps each $4P^2$-dimensional pixel patch to a d-dimensional token, while retaining the denoising target in exact pixel space. Let $E(\cdot)$ denote this patch embedding. We consider two choices. By default, we use a shared linear projection from $4P^2$ to d. Alternatively, following the spirit of JiT [28], we use a lightweight convolutional bottleneck consisting of a $P \times P$, stride-P convolution from 4 channels to d_b, followed by a 1×1 convolution from d_b to d. The embedding is used only for Transformer computation and does not introduce a VAE or latent decoder. The output head still predicts RGBA pixel patches directly in $\mathbb{R}^{N_i\times 4P^2}$.

Anonymous regional tokens with 3D RoPE. PixelART treats all visual regions as anonymous RGBA patch sequences. We do not provide regional prompts or use learned role embeddings to distinguish the input image, the background layer, or foreground layers, nor do we use layer-specific decoders. For patch j from region r_i, the initial Transformer token is $\mathbf{h}_{i,j} = E(\mathbf{X}_i)_j$. Spatial and layer-slot information is injected only through 3D RoPE inside the attention layers. Each visual token is assigned a 3D coordinate that specifies the x, y, and z positions, where x and y correspond to the spatial position within the image, and z refers to layer index. For clean conditioning-image tokens, we use the full-canvas spatial coordinates and a single fixed layer slot reserved for the input image; for target layer tokens, z_i follows the anonymous layer order.

Single-stream pixel diffusion Transformer. Inspired by single-stream diffusion Transformers [49], PixelART jointly models the conditioning signals and target layers by concatenating optional text tokens, clean conditioning-image tokens, and noisy target-layer tokens into a single sequence:

$$\mathbf{S}_t = [\mathbf{T};\, \mathbf{G};\, E(\mathbf{Y}_t^{(0)});\, E(\mathbf{Y}_t^{(1)});\, \ldots;\, E(\mathbf{Y}_t^{(K)})], \tag{4}$$

where $\mathbf{T}$ denotes the text tokens, $\mathbf{G} = E(\widetilde{\mathbf{I}})$ denotes the clean conditioning-image tokens, and $\mathbf{Y}_t^{(i)}$ denotes the noised pixel-patch sequence of target region i. After lightweight modality-specific encoding, all tokens are processed together by shared self-attention layers, allowing target-layer tokens to attend to the input image, text, and all other layers. The output head produces predictions only at target-layer positions:

$$\widehat{\mathbf{Y}}_\theta = f_\theta(\mathbf{S}_t, t, \mathcal{R})_{\mathrm{target}}. \tag{5}$$

[2] All RGB and RGBA tensors are normalized to $[0, 1]$ unless otherwise specified.



Figure 3: **High-noise timesteps are critical for layer assignment.** We compare layer states along the sampling trajectory for checkpoints with and without terminal boosting. The boosted model (1st row) assigns regions correctly at the earliest high-noise stage, whereas the unboosted model (2nd row) makes a persistent layer-assignment error. Later steps mainly refine RGBA details. Green/red boxes mark correct/incorrect assignments.

A diffusion mask separates conditioning from target tokens: conditioning tokens remain clean, while target-layer tokens are noised and supervised.

At inference, we feed the conditioning-image and text tokens, initialize the target pixel-patch sequence $\mathbf{Y}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and integrate the rectified-flow ODE from $t = 0$ to $t = 1$. At each step $t < 1$, the model predicts clean patches $\widehat{\mathbf{Y}}_\theta$ and estimates the velocity following Equation (7). We adopt classifier-free guidance (CFG) [50] in velocity space. After sampling, each predicted region sequence $\widehat{\mathbf{Y}}_i$ is unpatchified, cropped to remove padding, and pasted back to the canvas:

$$\widehat{\mathbf{L}}_i(\mathbf{p}) = \begin{cases} \mathrm{clip}_{[0,1]}\left(\widehat{\mathbf{X}}_i(\mathbf{p} - \mathbf{o}_i)\right), & \mathbf{p} \in r_i, \\ (0,0,0,0), & \text{otherwise}, \end{cases} \tag{6}$$

where $\widehat{\mathbf{X}}_i = \mathrm{crop}_{h_i,w_i}(\phi_P^{-1}(\widehat{\mathbf{Y}}_i))$ and $\mathbf{o}_i = (x_i, y_i)$ is the origin of region r_i. The final output is the ordered editable RGBA layers $\{\widehat{\mathbf{L}}_0, \widehat{\mathbf{L}}_1, \ldots, \widehat{\mathbf{L}}_K\}$.

3.3 Learning Objective

PixelART follows rectified-flow training, where $t = 0$ denotes pure noise and $t = 1$ denotes clean data. Given clean target pixel patches $\mathbf{Y}$, Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a timestep $t \sim q(t)$, we define $\mathbf{Y}_t = t\mathbf{Y} + (1-t)\boldsymbol{\epsilon}$. The model directly predicts clean pixel patches $\widehat{\mathbf{Y}}_\theta$. Following the $\mathbf{x}$-prediction with $\mathbf{v}$-loss formulation, we convert this prediction to a velocity estimate:

$$\widehat{\mathbf{v}}_\theta = \frac{\widehat{\mathbf{Y}}_\theta - \mathbf{Y}_t}{1-t}, \qquad \mathbf{v} = \mathbf{Y} - \boldsymbol{\epsilon}. \tag{7}$$

We then optimize the velocity-space flow-matching objective:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\mathbf{Y},\boldsymbol{\epsilon},t}\left[\frac{1}{|\Omega|}\sum_{j\in\Omega}\|\widehat{\mathbf{v}}_{\theta,j} - \mathbf{v}_j\|_2^2\right], \tag{8}$$

where Ω denotes the set of valid, non-padding target tokens. Equivalently, this objective performs clean-pixel prediction with a timestep-dependent weight of $1/(1-t)^2$. Following [29], we further incorporate an LPIPS perceptual loss based on VGG and ConvNeXt-V2.

3.4 Timestep Sampling Strategy

The flow-matching objective above depends on the timestep distribution $q(t)$. Since layer assignment is largely determined at high-noise timesteps, we further introduce terminal-boosted timestep sampling to emphasize this stage during training. Figure 3 visualizes several examples illustrating failures caused by incorrect layer assignments at high-noise timesteps.

Logit-normal base schedule. Modern rectified-flow Transformers such as SD3, FLUX, and JiT [28] commonly sample timesteps from a logit-normal distribution [23, 22, 24]. A latent variable $u \sim \mathcal{N}(\mu, \sigma^2)$ is mapped to $t \in (0,1)$ by $t = \mathrm{sigmoid}(u)$, yielding density

$$q_{\text{base}}(t) = \frac{1}{\sigma\sqrt{2\pi}}\frac{1}{t(1-t)}\exp\left(-\frac{(\mathrm{logit}(t)-\mu)^2}{2\sigma^2}\right). \tag{9}$$

With $t = 0$ as pure noise and $t = 1$ as clean data, μ controls which endpoint receives more mass; the standard logit-normal schedule concentrates mass at intermediate timesteps, with little near $t = 0$.

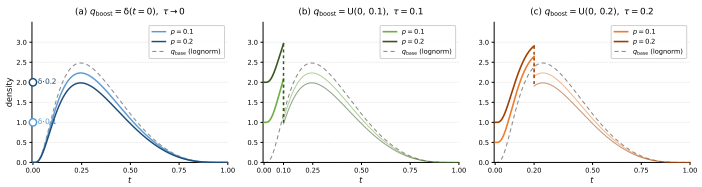


Figure 4: **Terminal-boosted timestep distributions.** We mix the base schedule with either a point mass at $t = 0$ or a uniform distribution $\mathcal{U}(0, \tau)$ to increase the mass in the high-noise region.

High-noise layer assignment. I2L decomposition differs from standard image generation because it must first solve a discrete assignment problem: which visible pixels belong to which output layer. We interpret the output space as a disjoint union of manifolds, one for each valid layer assignment. The model must choose the correct branch at the beginning of the sampling trajectory, when $t \approx 0$. Once denoising has progressed, later steps mainly refine continuous RGBA values and often cannot switch to a different layer-assignment branch. This motivates increasing the training probability of the high-noise region near $t = 0$.

Terminal-boost formulation. We augment base schedule with an auxiliary high-noise component:

$$q(t) = p \cdot q_{\text{boost}}(t; \tau) + (1 - p) \cdot q_{\text{base}}(t), \tag{10}$$

where p is the boost probability and q_{boost} is supported on $[0, \tau]$. We consider two simple choices:

$$q_{\text{boost}}(t; \tau) = \begin{cases} \delta(t = 0), & \tau \to 0, \\ \mathcal{U}(0, \tau), & \tau > 0. \end{cases} \tag{11}$$

The point-mass boost emphasizes the pure-noise endpoint, while the uniform boost spreads probability over an early high-noise interval. Figure 4 visualizes the resulting mixture density for $p \in \{0.1, 0.2\}$ and $\tau \in \{0, 0.1, 0.2\}$. All these variants allocate more probability mass to high-noise regions. Terminal boosting improves assignment under the layer definition specified by the layout and annotations, rather than resolving semantic ambiguity in layer granularity. The compatibility of the terminal-boosted schedule with $\mathbf{x}$-prediction $\mathbf{v}$-loss is discussed in Supplementary G.

4 Experiments

Model configuration. We instantiate three model variants, PixelART-B/L/H, following the ViT [51] naming convention. They use $12/24/32$ transformer blocks, $768/1024/1280$ hidden dimensions, $12/16/16$ attention heads, $128/128/256$ bottleneck dimensions, and $168\text{M}/523\text{M}/1.05\text{B}$ parameters, respectively.

Training setup. Unless otherwise specified, ablation experiments use PixelART-B/16 at 256×256 resolution, with a global batch size of 1024 for 100K optimizer updates. A default PixelART-B run takes approximately one day on $8\times$ H200 GPUs. For system-level experiments, we train PixelART-H/32 on $32\times$ H200 GPUs, first at 256×256 resolution with patch size 32 for 80 epochs, then progressively increase the resolution to 512×512 and 1024×1024 for additional 60 and 10 epochs, respectively. The full internal multi-layer design dataset contains approximately 4M high-quality samples, with subsets used for data-scale ablations.

Evaluation setup. We mainly evaluate on DESIGN-MULTI-LAYER-BENCH [1][3] and LICA [53][4]. We report *Layer* L1/PSNR/SSIM on non-transparent pixels using the ground-truth alpha mask, and *Composite* L1/PSNR/SSIM between the composited prediction and the input image. We also report VLM-based ratings of *Alpha* (matte and transparency quality) and *Visual* (layer integrity and separation), evaluated by GPT-5.5 [54] and Gemini-3.1 Pro [55]. The evaluation prompt is provided in Supplementary D.4. For the user study, we collect two-dimensional pairwise preferences from 10 annotators with diverse backgrounds.

Layout setting. To extract the regional tokens from flat design pixels, we adopt a graphic design layout detector DAD [56] based on Qwen3-VL-2B [57]. The additional latency introduced by the layout detector is only 1 second when predicting around 20 bounding boxes in z-order.

[3] A graphic-design benchmark curated from VistaCreate [52]

[4] PixelART evaluation code and LICA results: `https://github.com/hmk114/pixelart-eval`.

Method	T2I Pre.	VAE	#Params (Layout)	DESIGN-MULTI-LAYER-BENCH Pixel PSNR↑ Comp.	DESIGN-MULTI-LAYER-BENCH VLM (GPT/Gem.)↑ Alpha	DESIGN-MULTI-LAYER-BENCH VLM (GPT/Gem.)↑ Visual	LICA Pixel PSNR↑ Comp.	LICA VLM (GPT/Gem.)↑ Alpha	LICA VLM (GPT/Gem.)↑ Visual	Inference Efficiency Time (s)↓	Inference Efficiency Mem (GB)↓
CLD [39]	✓	✓	14B (12+2)	29.79	4.24/4.83	4.50/4.68	28.84	4.21/4.78	4.38/4.62	58.7 (+0.7)	55.6 (+4.2)
QIL [2]	✓	✓	20B	32.46	4.30/4.73	4.14/4.37	31.99	4.22/4.65	3.93/4.20	213.4	72.4
PixelART	✗	✗	3.05B (1.05+2)	**36.92**	**4.32/4.91**	**4.57/4.81**	**36.39**	**4.26/4.87**	**4.44/4.75**	**3.1** (+0.7)	**6.6** (+4.2)
LayerD† [34]	✓	✓	0.27B	36.07	4.12/3.94	3.49/3.13	37.22	4.06/4.01	3.64/3.37	15.0	3.4

Table 1: **System-level comparison**. *T2I Pre.*: initialization from a pretrained T2I model. *Pixel PSNR*: PSNR between the alpha-composited reconstruction and the input image. *VLM*: alpha and visual quality scores (1–5) from GPT-5.5 / Gemini-3.1 Pro, separated by '/'. *Efficiency*: per-image inference latency and peak GPU memory on a single H200. Parenthesized values for PixelART and CLD indicate the additional layout detector overhead (#parameters, latency, GPU memory). LayerD† uses segmentation and inpainting rather than diffusion, achieving high Composite PSNR but the lowest VLM Visual score. Bold is the best result among diffusion-based methods.

	DESIGN-MULTI-LAYER-BENCH									LICA								
	Composite			Layer			Assignment			Composite			Layer			Assignment		
Method	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	α-IoU↑	Owner↑	Leak.↓	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	α-IoU↑	Owner↑	Leak.↓
GT-box conditioning																		
CLD	0.0205	29.84	0.9225	0.0850	22.74	0.7574	0.9047	0.9071	0.1033	0.0238	28.39	0.9088	0.1240	20.21	0.6233	0.8706	0.7983	0.2108
QIL (LC)	0.0308	28.17	0.9129	0.2165	20.09	0.5955	0.8161	0.6103	0.3939	0.0632	25.24	0.8711	0.2570	18.71	0.4459	0.7094	0.4263	0.5734
MRT	0.0150	33.16	0.9511	0.0523	29.25	0.8938	0.9433	**0.9590**	0.0597	0.0149	33.22	0.9605	0.0735	28.47	0.8010	0.8944	0.8311	0.1774
PixelART	**0.0069**	**36.85**	**0.9678**	**0.0437**	**30.70**	**0.8978**	**0.9632**	0.9589	**0.0573**	**0.0068**	**36.37**	**0.9686**	**0.0563**	**31.90**	**0.8670**	**0.9154**	**0.8544**	**0.1571**
GT Layer-Merge evaluation																		
QIL	0.0149	32.46	0.9404	0.0654	26.89	0.8707	0.8420	0.8654	0.1455	0.0153	31.99	0.9443	0.0955	25.57	0.8209	0.8176	0.7516	0.2582
LayerD	0.0030	36.07	0.9867	0.1115	25.21	0.7895	0.8663	0.6671	0.3368	0.0027	37.22	0.9881	0.1379	27.19	0.7366	0.8193	0.5579	0.4462

Table 2: **Decomposition quality.** LC denotes Layer-Crop, Owner denotes ownership accuracy, GT Layer-Merge evaluates bounding-box-free predictions against merged GT layers, and Bold indicates the best GT-box-conditioned scores.

4.1 Main Results

Comparison with State-of-the-Art Methods. We compare PixelART-H with state-of-the-art I2L decomposition methods on DESIGN-MULTI-LAYER-BENCH [1] and LICA [53]. The baselines include QIL [2], which adapts Qwen-Image-20B [24] using a VLD-MMDiT architecture, and CLD [39], an ART-style [1] variant built on FLUX.1[dev]-12B [22]. Both methods rely on pretrained T2I backbones and RGBA latent autoencoders. We also include LayerD [34], a non-diffusion pipeline combining matting with LaMa [58] inpainting. For the system-level comparison, we use each baseline's official checkpoint and codebase with its default inference configuration. PixelART and CLD use the same layout detector, whereas QIL and LayerD are evaluated in their native bounding-box-free setting.

Table 1 shows that PixelART achieves the highest Composite PSNR and VLM rating while using $6\times$ fewer parameters than QIL and $4\times$ fewer than CLD. PixelART-H is also substantially more efficient. Evaluated on 100 randomly sampled 1024×1024 inputs from DESIGN-MULTI-LAYER-BENCH [1] using a single H200 GPU, without additional acceleration techniques such as quantization, distillation, or step reduction, it runs $19\times$ faster than CLD and $69\times$ faster than QIL while requiring 8–$11\times$ less peak GPU memory. These gains come from its smaller diffusion transformer backbone and the elimination of VAE decoding during sampling.[5]

We include MRT [59] as an additional baseline for evaluating decomposition quality and examine whether PixelART's gains are attributable solely to layout conditioning. Unless otherwise noted, layout-conditioned methods receive ground-truth (GT) boxes. For methods that do not take bounding boxes as input, we use two protocols: Layer-Crop (LC) crops QIL's visual tokens to regions defined by GT boxes, and GT Layer-Merge, which merges GT layers to match predicted partitions (Supplementary E). As shown in Table 2, PixelART achieves the best overall performance on both benchmarks. In particular, compared with QIL (LC) using the same DAD-predicted layouts, PixelART retains its advantage under this setting and exhibits limited sensitivity to detector box quality (Supplementary C).

[5]Despite its small parameter count, LayerD [34] incurs non-negligible latency due to costly CPU-side segmentation, matting, and inpainting operations in its official implementation.

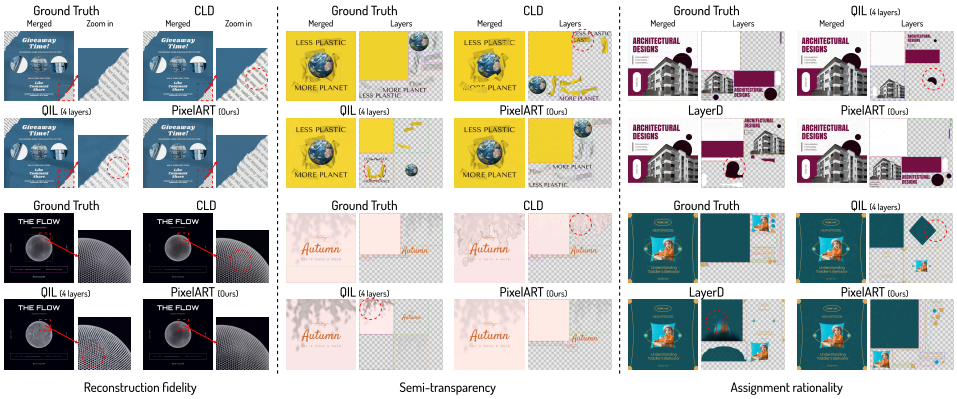


Figure 5: Qualitative comparison with state-of-the-art methods. QIL: Qwen-Image-Layered.

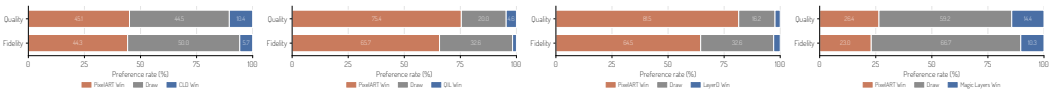


Figure 6: **User study.** Comparison with state-of-the-art methods and the commercial system Magic Layers [40].

Qualitative Comparison. Figure 5 compares PixelART with prior methods along three complementary qualitative dimensions. (1) *Reconstruction fidelity*: CLD and QIL often introduce artifacts in fine textures, small text, or fuse adjacent characters, which is consistent with the information loss of latent autoencoding. PixelART directly denoises RGBA pixels and better preserves high-frequency details. (2) *Semi-transparency*: for soft shadows, feathered boundaries, and gradient overlays, CLD tends to produce nearly binary alpha masks, failing to capture partial transparency, while QIL bakes background colors into foreground layers and can introduce color shifts. In contrast, PixelART better preserves RGB consistency and smooth alpha transitions. (3) *Assignment rationality*: bounding-box-free baselines, such as QIL and LayerD, do not require layout input, but may mix unrelated elements, split coherent ones across layers, or exhibit segmentation leakage and incomplete extraction. PixelART conditions on bounding boxes, enabling explicit control over layer assignment and producing more coherent, editable layers. Fine-tuning on approximately 2K synthetic layered images also improves natural-image decomposition (Supplementary J).

User Study. Figure 6 summarizes the user study results. Our method is consistently preferred over prior state-of-the-art methods and achieves comparable or superior performance to the latest commercial system, Canva Magic Layers [40].

4.2 Ablation Study

Pixel-space outperforms latent-space diffusion for I2L. Table 3 compares pixel-space and latent-space diffusion with matched diffusion-Transformer capacity and training budgets. For the latent baseline, we pair the frozen RGBA-VAE encoder from QIL [2] with a separately trained ViT-based regional RGBA decoder. Its setup and reconstruction quality are detailed in Supplementary H. Pixel diffusion outperforms the latent baseline on every metric, including a +5.57 dB gain on Composite PSNR. These results support pixel-space modeling for I2L under the evaluated settings, and we therefore adopt pixel-space diffusion by default unless otherwise specified. Smaller pixel patches improve PSNR and SSIM but increase latency (Supplementary B).

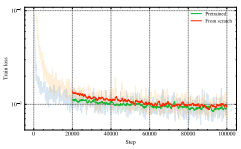
	Layer metrics			Composite metrics		
Method	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
Latent VAE	0.0749	26.37	0.8172	0.0202	28.79	0.9155
Pixel	**0.0626**	**28.66**	**0.8609**	**0.0132**	**34.36**	**0.9539**

Table 3: Pixel-space vs. latent-space diffusion

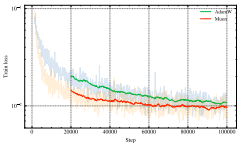
	Layer metrics			Composite metrics		
Initialization	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
From scratch	**0.0626**	**28.66**	0.8609	**0.0132**	**34.36**	**0.9539**
From T2I pretrained	0.0639	28.27	**0.8614**	0.0132	33.97	0.9504

Table 4: Effect of T2I pretraining.

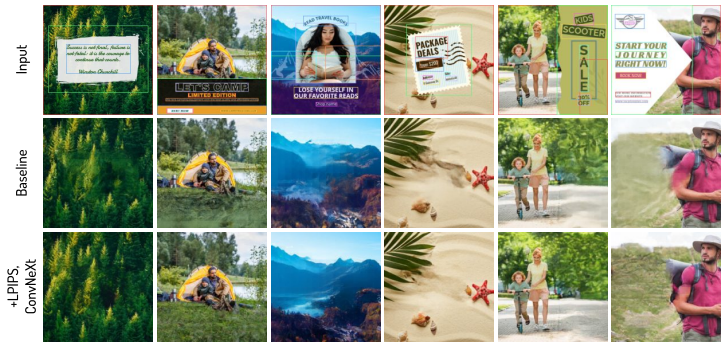
T2I initialization yields no consistent final gains. We evaluate whether T2I pretraining improves I2L decomposition in our graphic-design setting. Following PixelGen [30], we pretrain PixelART-B



(a) T2I initialization.



(b) Optimizer.



(c) Perceptual loss.

Figure 7: **Ablation visualizations. Left top:** I2L fine-tuning loss curves for T2I-pretrained vs. from-scratch initializations on PixelART-B. **Left bottom:** training loss curves of Muon vs. AdamW under the same recipe. **Right:** qualitative effect of perceptual losses on background reconstruction.

on the public BLIP-3o corpus [60] (∼36 million text–image pairs) for 250K steps with standard flow matching, then fine-tune on I2L using the same training budget as the from-scratch baseline. As shown in Table 4, T2I pretraining brings a marginal improvement in the Layer SSIM metric and a slight regression on Composite PSNR (-0.39 dB). Figure 7(a) shows that the T2I-initialized model starts with a substantially lower loss, but the from-scratch run closes the gap within roughly 60K training steps. Reducing the I2L corpus to 1M samples also yields no consistent final gains from T2I initialization (Supplementary I). These findings are specific to the evaluated graphic-design data, PixelART-B model, and training budget. Training costs are reported in Supplementary I.

Prediction	Layer metrics			Composite metrics		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
$\mathbf{x}$-pred	**0.0626**	**28.66**	**0.8609**	**0.0132**	**34.36**	**0.9539**
$\mathbf{v}$-pred	0.2504	9.92	0.1718	0.2703	9.05	0.0383

Table 5: Effect of prediction space.

Optimizer	Layer metrics			Composite metrics		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
AdamW [61]	0.0682	27.75	0.8475	**0.0129**	33.76	0.9501
Muon [62]	**0.0626**	**28.66**	**0.8609**	0.0132	**34.36**	**0.9539**

Table 6: Effect of optimizer.

$\mathbf{x}$-prediction is critical for I2L. Table 5 compares $\mathbf{x}$- and $\mathbf{v}$-prediction. $\mathbf{v}$-prediction fails to converge in our setting. A possible explanation is a representational bottleneck: each $16\times16\times4$ RGBA patch has 1024 dimensions, exceeding the model's hidden width of 768, which may hinder the modeling of full-dimensional velocity targets. In contrast, $\mathbf{x}$-prediction targets clean images, which lie on a lower-dimensional manifold and thus admit a more compact representation in the model's hidden space. This is consistent with the manifold argument of JiT [28]. Even at PixelART-H scale (d=1280), where the hidden width exceeds the patch dimensionality, $\mathbf{v}$-prediction still underperforms $\mathbf{x}$-prediction (Supplementary G, Table 15).

Muon outperforms AdamW. Table 6 compares Muon [62] with AdamW [61] using the same learning rate, warmup, batch size, and training budget. Muon improves Layer PSNR by $+0.91$ dB and Composite PSNR by $+0.60$ dB, with consistent SSIM gains across both. Figure 7(b) further shows that Muon achieves a lower training loss under the same training recipe.

Training data	Layer metrics			Composite metrics		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
40K	0.0971	24.02	0.7671	0.0160	31.23	0.9292
400K	0.0698	27.67	0.8424	0.0144	33.71	0.9479
4M	**0.0626**	**28.66**	**0.8609**	**0.0132**	**34.36**	**0.9539**

Table 7: Effect of training data scale.

Model variant	Layer metrics			Composite metrics		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
PixelART-B/16	0.0626	28.66	0.8609	0.0132	34.36	0.9539
PixelART-L/16	0.0475	31.78	0.8968	**0.0094**	36.79	0.9677
PixelART-H/16	**0.0451**	**32.71**	**0.9011**	0.0096	**37.96**	**0.9696**

Table 8: Effect of model size scaling.

I2L benefits substantially from data scale. Table 7 compares training on 40K, 400K, and 4M real multi-layer designs with all other settings fixed. All metrics improve with more data. From 40K to 4M, Layer and Composite PSNR increase by 4.64 dB and 3.13 dB, respectively. Although the marginal gain from 400K to 4M ($+0.99$ dB Layer PSNR) is smaller than from 40K to 400K ($+3.65$ dB), the curve has not yet plateaued, suggesting that further real-data scaling remains a promising direction. We use 4M designs by default.

PixelART improves with model size. Table 8 compares three model variants, PixelART-B/16 (168M), PixelART-L/16 (523M), and PixelART-H/16 (1.05B), trained with the same data and training budget. Nearly all metrics improve monotonically with model size. As the model grows from 168M to 1.05B parameters, Layer and Composite PSNR increase by 4.05 dB and 3.60 dB, respectively. Together with the data-scaling results in Table 7, these findings show that PixelART benefits from both more training data and larger models within the evaluated ranges.

	Background metrics		Layer metrics			Composite metrics		
Loss	FID↓	LPIPS↓	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
MSE	45.623	0.2910	**0.0603**	**27.72**	**0.8431**	**0.0139**	**33.03**	**0.9410**
+ LPIPS (VGG)	41.478	0.2739	0.0656	26.86	0.8324	0.0155	31.94	0.9314
+ LPIPS (VGG, ConvNeXt)	**35.846**	**0.2577**	0.0631	27.28	0.8383	0.0145	32.46	0.9360

Table 9: Effect of perceptual losses evaluated on the non-solid-background subset.

Perceptual loss improves background reconstruction. Table 9 evaluates LPIPS-based perceptual supervision for the background reconstruction, where large occluded regions make artifacts particularly noticeable. To avoid solid-color backgrounds dominating the evaluation, we report all metrics on the $1{,}474$-sample with non-solid-backgrounds. A VGG-based loss [63] reduces background FID from 45.6 to 41.5 and LPIPS from 0.291 to 0.274. Adding a ConvNeXt-based loss [64] further reduces these values to 35.8 and 0.258, respectively, while Layer and Composite metrics remain comparable. Figure 7(c) confirms the trend qualitatively: perceptual supervision yields sharper and more coherent background completions. We therefore use both perceptual losses in the final system.

		Layer metrics			Composite metrics			Assignment metrics		
q_{boost}	p	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	α-IoU↑	Owner Acc.↑	Leak.↓
–	–	0.0626	28.66	0.8609	0.0132	34.36	0.9539	0.8983	0.9040	0.1245
$\delta(t{=}0)$	0.1	0.0588	27.89	0.8563	0.0137	33.06	0.9449	0.9471	0.9292	0.0987
	0.2	0.0609	27.01	0.8444	0.0157	31.90	0.9327	0.9468	0.9347	0.0943
$\mathcal{U}(0, 0.1)$	0.1	0.0535	28.89	0.8778	0.0125	34.08	0.9545	0.9476	0.9341	0.0951
	0.2	0.0523	28.87	0.8763	0.0125	33.82	0.9509	**0.9501**	**0.9395**	**0.0909**
$\mathcal{U}(0, 0.2)$	0.1	**0.0516**	**29.51**	**0.8802**	**0.0119**	**34.53**	**0.9568**	0.9466	0.9358	0.0944
	0.2	0.0534	28.86	0.8765	0.0125	34.01	0.9530	0.9476	0.9377	0.0944

Table 10: Effect of the boost component q_{boost} and mixing weight p in terminal-boosted timestep sampling.

Terminal-boosted sampling improves layer assignment. Table 10 compares the boost component q_{boost} and mixing probability p. All point-mass boosts at $t{=}0$ and truncated-uniform boosts $\mathcal{U}(0, \tau)$ improve each assignment metric over the no-boost baseline. These gains confirm that additional supervision in the high-noise regime improves layer assignment. Truncated-uniform boosts yield better reconstruction than the point-mass variants, but varying p and τ does not consistently improve assignment metrics. We use $\mathcal{U}(0, 0.2)$ with $p{=}0.1$ by default, as it achieves the best reconstruction metrics while maintaining strong assignment performance. This configuration also outperforms shifted logit-normal baselines on all nine metrics (Supplementary G.3).

5 Conclusion

We introduce PixelART, a pixel-space rectified-flow Transformer trained from scratch for I2L decomposition. PixelART models decomposition as a pixel-space assignment-and-completion problem: high-noise timesteps determine layer assignment, while low-noise timesteps refine RGBA details. It directly denoises regional RGBA pixel patches, removing the RGBA-VAE bottleneck without requiring pretrained T2I backbones or layer-specific decoders. On DESIGN-MULTI-LAYER-BENCH, PixelART achieves state-of-the-art performance in both I2L decomposition and composite reconstruction, with substantially lower latency and memory than VAE-based diffusion baselines. Ablations show the benefits of pixel-space $\mathbf{x}$-prediction, high-noise timestep coverage, perceptual supervision, and data/model scaling. T2I initialization, however, provides no consistent final gains in our graphic-design setting. These findings support from-scratch pixel-space diffusion as a simple, scalable method for graphic-design I2L decomposition.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under grant 62636008. The authors thank Canva for providing computational resources. The authors have no additional competing interests to declare.

References

[1] Y. Pu, Y. Zhao, Z. Tang, R. Yin, H. Ye, Y. Yuan *et al.*, "Art: Anonymous region transformer for variable multi-layer transparent image generation," in *CVPR*, 2025, pp. 7952–7962.

[2] S. Yin, Z. Zhang, Z. Tang, K. Gao, X. Xu, K. Yan, J. Li, Y. Chen, Y. Chen, H.-Y. Shum, L. M. Ni, J. Zhou, J. Lin, and C. Wu, "Qwen-image-layered: Towards inherent editability via layer decomposition," *arXiv preprint arXiv:2512.15603*, 2025.

[3] Google, "Introducing Gemini 2.5 Flash Image, our state-of-the-art image model," https://developers.googleblog.com/introducing-gemini-2-5-flash-image/, 2025, also known as Nano Banana. Accessed: 2025-08-26.

[4] Google DeepMind, "Nano Banana 2: Google's latest AI image generation model," https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/, 2026, accessed: 2026-02-26.

[5] ——, "Nano Banana Pro: Gemini 3 Pro Image model from Google DeepMind," https://blog.google/innovation-and-ai/products/nano-banana-pro/, 2025, accessed: 2025-11-20.

[6] OpenAI, "GPT Image 1 Model," https://developers.openai.com/api/docs/models/gpt-image-1, 2025, accessed: 2025-03-25.

[7] ——, "GPT Image 1.5 Model," https://developers.openai.com/api/docs/models/gpt-image-1.5, 2025, accessed: 2025-12-16.

[8] ——, "GPT Image 2 Model," https://developers.openai.com/api/docs/models/gpt-image-2, 2026, accessed: 2026-04-21.

[9] Z. Li, D. Li, Y. Guo, X. Guo, B. Li, L. Xiao, S. Qiao, J. Chen, Z. Wu, H. Zhang, and S. Liu, "Chartgalaxy: A dataset for infographic chart understanding and generation," in *ICLR*, 2026, pp. 52 484–52 546.

[10] C. Liu, Y. Song, H. Wang, and M. Z. Shou, "Omnipsd: Layered psd generation with diffusion transformer," *arXiv preprint arXiv:2512.09247*, 2025.

[11] H. Yu, J. Wang, J. Zhan, R. Chen, Z. Wang, H. Zhang *et al.*, "OmniAlpha: Aligning transparency-aware generation via multi-task unified reinforcement learning," *arXiv preprint arXiv:2511.20211*, 2025.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.

[13] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *NeurIPS*, 2021.

[14] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," in *ICML*, 2022.

[15] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," in *NeurIPS*, 2022.

[16] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," *arXiv preprint arXiv:2204.06125*, 2022.

[17] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *JMLR*, vol. 23, no. 47, pp. 1–33, 2022.

[18] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022, pp. 10 684–10 695.

[19] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," in *ICLR*, 2024.

[20] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, Q. Zhang, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, T. Karras, and M.-Y. Liu, "eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers," *arXiv preprint arXiv:2211.01324*, 2022.

[21] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, W. Manassra, P. Dhariwal, C. Chu, Y. Jiao, and A. Ramesh, "Improving image generation with better captions," *OpenAI Technical Report*, 2023, https://cdn.openai.com/papers/dall-e-3.pdf.

[22] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[23] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *ICML*, 2024.

[24] C. Wu, J. Li, J. Zhou, J. Lin, K. Gao, K. Yan *et al.*, "Qwen-image technical report," 2025. [Online]. Available: https://arxiv.org/abs/2508.02324

[25] E. Hoogeboom, T. Mensink, J. Heek, K. Lamerigts, R. Gao, and T. Salimans, "Simpler diffusion (SiD2): 1.5 FID on ImageNet512 with pixel-space diffusion," *arXiv preprint arXiv:2410.19324*, 2024.

[26] S. Chen, C. Ge, S. Zhang, P. Sun, and P. Luo, "PixelFlow: Pixel-space generative models with flow," *arXiv preprint arXiv:2504.07963*, 2025.

[27] S. Wang, Z. Gao, C. Zhu, W. Huang, and L. Wang, "PixNerd: Pixel neural field diffusion," *arXiv preprint arXiv:2507.23268*, 2025.

[28] T. Li and K. He, "Back to basics: Let denoising generative models denoise," *arXiv preprint arXiv:2511.13720*, 2025.

[29] Y. Lu, S. Lu, Q. Sun, H. Zhao, Z. Jiang, X. Wang, T. Li, Z. Geng, and K. He, "One-step latent-free image generation with pixel mean flows," *arXiv preprint arXiv:2601.22158*, 2026.

[30] Z. Ma, R. Xu, and S. Zhang, "PixelGen: Pixel diffusion beats latent diffusion with perceptual loss," *arXiv preprint arXiv:2602.02493*, 2026.

[31] L. Zhang and M. Agrawala, "Transparent image layer diffusion using latent transparency," *ACM Transactions on Graphics*, vol. 43, no. 4, 2024.

[32] P. Jia, C. Li, Y. Yuan, Z. Liu, Y. Shen, B. Chen *et al.*, "Cole: A hierarchical generation framework for multi-layered and editable graphic design," *arXiv preprint arXiv:2311.16974*, 2023.

[33] N. Inoue, K. Masui, W. Shimoda, and K. Yamaguchi, "Opencole: Towards reproducible automatic graphic design generation," in *CVPR Workshops*, 2024.

[34] T. Suzuki, K.-J. Liu, N. Inoue, and K. Yamaguchi, "Layerd: Decomposing raster graphic designs into layers," in *ICCV*, 2025.

[35] X. Zhang, W. Zhao, X. Lu, and J. Chien, "Text2layer: Layered image generation using latent diffusion model," *arXiv preprint arXiv:2307.09781*, 2023.

[36] R. Huang, K. Cai, J. Han, X. Liang, R. Pei, G. Lu, S. Xu, W. Zhang, and H. Xu, "Layerdiff: Exploring text-guided multi-layered composable image synthesis via layer-collaborative diffusion model," in *ECCV*, 2024.

[37] J. Chen, H. Jiang, Y. Wang, K. Wu, J. Li, C. Zhang, K. Yanai, D. Chen, and Y. Yuan, "Prismlayers: Open data for high-quality multi-layer transparent image generative models," *arXiv preprint arXiv:2505.22523*, 2025.

[38] J. Huang, P. Yan, J. Cai, J. Liu, Z. Wang, Y. Wang, X. Wu, and G. Li, "Dreamlayer: Simultaneous multi-layer generation via diffusion model," in *ICCV*, 2025.

[39] Z. Liu, Z. Xu, S. Shu, J. Zhou, R. Zhang, Z. Tang, and X. Li, "Controllable layer decomposition for reversible multi-layer image generation," *arXiv preprint arXiv:2511.16249*, 2025.

[40] Canva, "Using magic layers to turn images into editable layers," https://www.canva.com/help/editable-magic-layers/, 2026, accessed 2026-05-02.

[41] Lovart, "Lovart: Ai design agent," https://www.lovart.ai/, 2026, accessed 2026-05-02.

[42] Ideogram, "Ideogram canvas, magic fill, and extend," https://ideogram.ai/features/canvas, 2026, accessed 2026-05-02.

[43] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," in *ICCV*, 2023.

[44] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[45] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, "Segment anything in high quality," in *NeurIPS*, 2023.

[46] B. Kim, C. Shin, J. Jeong, H. Jung, S.-Y. Lee, S. Chun *et al.*, "Zim: Zero-shot image matting for anything," in *ICCV*, 2025.

[47] Z. Wang, H. Yu, J. Zhan, and C. Yuan, "Alphavae: Unified end-to-end rgba image reconstruction and generation with alpha-aware representation learning," *arXiv preprint arXiv:2507.09308*, 2025.

[48] H. Dong, W. Wang, C. Li, J. Lyu, and D. Lin, "Video generation with stable transparency via shiftable rgb-a distribution learner," *arXiv preprint arXiv:2509.24979*, 2025.

[49] Z-Image Team, H. Cai, S. Cao, R. Du, P. Gao, S. Hoi, Z. Hou *et al.*, "Z-Image: An efficient image generation foundation model with single-stream diffusion transformer," *arXiv preprint arXiv:2511.22699*, 2025.

[50] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS Workshops*, 2021.

[51] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[52] VistaCreate Team, "Vistacreate (formerly crello) graphic design platform," https://create.vista.com/, 2025, accessed: 2025-11-09.

[53] E. Hirsch, S. Yadav, M. Garg, and P. Mehta, "Lica: Layered image composition annotations for graphic design research," *arXiv preprint arXiv:2603.16098*, 2026.

[54] OpenAI, "GPT-5.5 Model," https://developers.openai.com/api/docs/models/gpt-5.5, 2026, accessed: 2026-05-07.

[55] Google, "Gemini 3.1 Pro Preview Model," https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview, 2026, model ID: `gemini-3.1-pro-preview`. Accessed: 2026-05-07.

[56] J. Zhu, B. Li, S. Qiao, Y. Gu, Z. Zhang, Y. Yuan, and S. Liu, "Detect anything in graphic design: Element-level rewards for autoregressive detection," *arXiv preprint arXiv:2609.07072*, 2026.

[57] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng *et al.*, "Qwen3-VL technical report," *arXiv preprint arXiv:2511.21631*, 2025.

[58] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov *et al.*, "Resolution-robust large mask inpainting with fourier convolutions," in *WACV*, 2022, pp. 2149–2159.

[59] Z. Tang, J. Chen, Z. Zhang, M. Zhou, Y. Liu, Y. Pu, Y. Bai, E. Smith, and Y. Yuan, "Masked region transformer for layered image generation and editing at scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2026, pp. 40 622–40 632. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2026/html/Tang_Masked_Region_Transformer_for_Layered_Image_Generation_and_Editing_at_CVPR_2026_paper.html

[60] J. Chen, Z. Xu, X. Pan, Y. Hu, C. Qin, T. Goldstein *et al.*, "BLIP3-o: A family of fully open unified multimodal models—architecture, training and dataset," *arXiv preprint arXiv:2505.09568*, 2025.

[61] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[62] K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein, "Muon: An optimizer for hidden layers in neural networks," https://kellerjordan.github.io/posts/muon/, 2024.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[64] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "ConvNeXt V2: Co-designing and scaling convnets with masked autoencoders," in *CVPR*, 2023, pp. 16 133–16 142.

[65] D. Ghosh, H. Hajishirzi, and L. Schmidt, "GenEval: An object-focused framework for evaluating text-to-image alignment," in *NeurIPS*, 2023.

A Data Source

Training Data. Our multi-layer training corpus contains approximately 4M internal design templates that are explicitly licensed for model training. Each template is stored as an editable design file with vector and raster layers in z-order, and is rasterized at multiple resolutions (256, 512, 1024) into the `{flat image, ordered RGBA layer stack, layer bounding boxes, layer order}` tuples that PixelART consumes.

Evaluation Data. We use DESIGN-MULTI-LAYER-BENCH [1], which is curated from VistaCreate [52], and LICA [53] as evaluation datasets; we do not include any of their test samples in training. No personally identifying information, copyrighted third-party images, or human-subject content were used, and no human annotators were employed for the data collection.

B Detailed Model and Training Configuration

Table 11 summarizes the architecture, training, and sampling hyperparameters for all PixelART configurations used in the main paper. PixelART-B/L/H are the three variants used in the model-size ablation (PixelART-B is also the default for all other ablations); PixelART (main model) is a separately trained system-level model used for the comparison in Table 1, sharing the -H scale but with a coarser patch size and a multi-resolution training curriculum. Hyperparameters are inherited from JiT [28] and SD3 [23] defaults if not listed below.

Patch size. Table 12 compares patch sizes 8, 16, and 32 at the PixelART-B scale. Smaller patches improve PSNR and SSIM but increase latency and peak memory. We use $P = 16$ for B-scale ablations as a compromise between reconstruction quality and efficiency. Larger patches also produce less accurate boundaries and more blur and artifacts within patches. The system-level model uses H/32 with the separate multi-resolution curriculum in Table 11, keeping token sequences shorter at high resolution.

C Layout Detector

PixelART is a region-conditioned I2L model: given a z-ordered set of regional bounding boxes $\mathcal{R}$, it predicts an editable RGBA crop for each region. During training, $\mathcal{R}$ is obtained from the ground-truth design file. At inference time, $\mathcal{R}$ can either be specified by users to control the desired layer granularity, or be predicted by a layout detector from the flattened RGB image to enable fully automatic I2L decomposition.

For system-level comparisons, we use DAD [56], an internal graphic-design layout detector based on Qwen3-VL-2B [57]. The detector predicts foreground bounding boxes in back-to-front z-order. PixelART only uses the predicted box coordinates and their order; no masks, alpha mattes, RGB content, semantic labels, or layer pixels are passed from the detector. Thus, the layout detector serves as an automatic interface for providing the structural input required by PixelART.

The detector is trained on internal design data disjoint from the DESIGN-MULTI-LAYER-BENCH and LICA test splits. On the 1024×1024 DESIGN-MULTI-LAYER-BENCH inputs reported in Table 1, it adds 0.7 s latency and 4.2 GB peak GPU memory per image, reported separately in the parenthesized terms of Table 1.

Comparison under the same predicted layout. To test whether layout access alone explains the gap in Table 1, we condition QIL on the same DAD-predicted layouts as PixelART through Layer-Crop. This protocol applies regional cropping to QIL's full-resolution visual tokens while retaining its diffusion process. Predicted regions need not match GT layers, so we report composite metrics and GPT ratings instead of layer-wise scores. PixelART performs better on both benchmarks (Table 13), indicating that layout access alone does not explain the gap.

Table 11: **Configurations of PixelART variants.** PixelART-B is the default for all ablations and is also the small model in the model-size ablation, alongside PixelART-L and PixelART-H. PixelART (main model) is the separately trained system-level model reported in Table 1; it shares the H scale but uses a coarser patch size and a three-stage resolution curriculum. Settings that are common across multiple columns are merged. The pixel-space versus latent-space comparison (Table 3) uses the same PixelART-B architecture, but the latter operates on the RGBA-VAE latents of QIL with a patch size of 2, with the parameter count differing by $\sim$1M due to the smaller patch size.

	PixelART-B	PixelART-L	PixelART-H	PixelART (main)
Architecture				
# Transformer blocks	12	24	32	32
hidden dim d	768	1024	1280	1280
# attention heads	12	16	16	16
patch size P	16			32
input noise scale	1			2
patch-embed bottleneck dim d_b	128	128	256	256
# parameters	168M	523M	1052M	1057M
MLP expansion ratio	4			
positional encoding	3D RoPE over (x, y, z)			
Training				
prediction target	$\mathbf{x}$-prediction with $\mathbf{v}$-loss			
base timestep schedule	$\mathrm{logit}(t) \sim \mathcal{N}(\mu{=}{-}0.8, \sigma{=}0.8)$			
optimizer	Muon [62]			
learning rate	$1\mathrm{e}{-4}$ (constant)			
warmup steps	1000			
weight decay	0			
EMA decay	0.9996			
text-condition drop (CFG)	0.1			
boost component q_{boost}	not used			$\mathcal{U}(0, 0.2)$ with $p{=}0.1$
perceptual loss (at $t > 0.3$)	not used			LPIPS (VGG, ConvNeXt-V2)
hardware	$8\times$ H200			$32\times$ H200
training resolution	256			$256 \to 512 \to 1024$
global batch size (per resolution)	1024			2048/2048/1024
training duration (per resolution)	100K optimizer updates			80/60/10 epochs
Sampling				
inference time shift	not used			3 (at 1024 px only)
ODE solver	Euler			
# sampling steps	25			
timestep grid	linear in $[0, 1]$			
CFG scale	2.2			

	Composite			Layer			Mean inference cost	
P	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	Time (s)↓	Peak mem. (GiB)↓
8	**0.0111**	**38.04**	**0.9627**	0.0630	**31.69**	**0.8660**	2.699	5.13
16	0.0132	34.36	0.9539	**0.0626**	28.66	0.8609	0.824	4.83
32	0.0261	26.87	0.8544	0.1323	21.72	0.6650	**0.629**	**4.76**

Table 12: **Patch-size ablation on PixelART-B.** The default patch size is $P = 16$.

	Composite metrics			GPT ratings	
Method	L1↓	PSNR↑	SSIM↑	Alpha↑	Visual↑
DESIGN-MULTI-LAYER-BENCH					
QIL (LC)	0.0361	27.39	0.9048	3.97	3.74
PixelART	**0.0066**	**36.92**	**0.9704**	**4.32**	**4.57**
LICA					
QIL (LC)	0.0909	24.11	0.8367	3.85	3.53
PixelART	**0.0066**	**36.39**	**0.9695**	**4.26**	**4.44**

Table 13: **Comparison using the same DAD-predicted layouts.** QIL uses the Layer-Crop variant rather than the bounding-box-free setting in Table 1.

Sensitivity to detector training. We hold PixelART fixed and vary the DAD checkpoint, using detectors trained for 10K, 100K and 400K iterations. On DESIGN-MULTI-LAYER-BENCH, Composite PSNR increases with detector training and remains within 0.46 dB of the GT-box result (Table 14). These runs use a different sample set from the comparisons in Tables 1 and 13.

Layout source	Composite PSNR↑
GT boxes	36.85
DAD (10K iterations)	36.39
DAD (100K iterations)	36.58
DAD (400K iterations)	36.72

Table 14: **Layout sensitivity on DESIGN-MULTI-LAYER-BENCH.** The PixelART checkpoint is fixed across all rows.

D Evaluation Metric Definitions

We evaluate three groups of pixel-level metrics: *Layer* metrics for decomposition fidelity, *Composite* metrics for visible reconstruction fidelity, and *Assignment* metrics for layer-assignment correctness. Throughout, let $\hat{\mathbf{L}}_i \in [0,1]^{H\times W\times 4}$ and $\mathbf{L}_i \in [0,1]^{H\times W\times 4}$ denote the predicted and ground-truth RGBA layer i, respectively, with $\mathbf{L}_i = (\mathbf{L}_i^{\mathrm{rgb}}, \alpha_i)$. We use $\mathbf{I}, \hat{\mathbf{I}} \in [0,1]^{H\times W\times 3}$ for the input flat image and the predicted alpha-composite, and $K{+}1$ for the total number of layers (background plus K foreground).

D.1 Layer Metrics

Layer metrics measure per-layer reconstruction fidelity. To jointly capture errors in both the predicted alpha and the predicted RGB channels, we compare alpha-premultiplied color, $\hat{\alpha}_i(p)\,\hat{\mathbf{L}}_i^{\mathrm{rgb}}(p)$ versus $\alpha_i(p)\,\mathbf{L}_i^{\mathrm{rgb}}(p)$, with the per-pixel error further weighted by the ground-truth alpha. A pixel for which the predicted alpha is wrong then contributes a non-zero error even if its RGB happens to match. Concretely, for each layer i and pixel p, the layer L1 is

$$\mathrm{L1}^{(i)}_{\mathrm{layer}} = \frac{\sum_p \alpha_i(p)\,\big\|\hat{\alpha}_i(p)\,\hat{\mathbf{L}}_i^{\mathrm{rgb}}(p) \;-\; \alpha_i(p)\,\mathbf{L}_i^{\mathrm{rgb}}(p)\big\|_1}{3\sum_p \alpha_i(p) + \varepsilon}. \tag{12}$$

PSNR and SSIM are computed analogously on the same alpha-premultiplied RGB pair, weighted by α_i. We then average over all layers in the test sample, and finally over the test set.

D.2 Composite Metrics

Composite metrics measure visible reconstruction fidelity. We first composite the predicted layers in z-order:

$$\hat{\mathbf{I}} = \Pi_{\mathrm{RGB}}\left(\mathrm{Over}(\hat{\mathbf{L}}_0, \hat{\mathbf{L}}_1, \ldots, \hat{\mathbf{L}}_K)\right), \tag{13}$$

and report L1, PSNR, and SSIM between $\hat{\mathbf{I}}$ and the input flat image $\mathbf{I}$ over all pixels.

D.3 Layer Assignment Metrics

Pixel reconstruction metrics alone do not penalize a model for confusing two visually similar layers, so we additionally report three layer-assignment metrics. Let $\hat{\alpha}_i, \alpha_i \in [0,1]^{H\times W}$ denote the predicted and ground-truth alpha of layer i, with index $i = 0$ for the background and $i \geq 1$ for foreground layers in increasing z-order.

Alpha-IoU (soft IoU on alpha masks). We measure how well the predicted alpha masks overlap with the ground-truth ones via a continuous (soft) intersection-over-union, computed independently per layer and averaged across all layers including the background:

$$\alpha\text{-IoU}_i = \frac{\sum_p \min\big(\hat{\alpha}_i(p),\, \alpha_i(p)\big)}{\sum_p \max\big(\hat{\alpha}_i(p),\, \alpha_i(p)\big) + \varepsilon}, \qquad \alpha\text{-IoU} = \frac{1}{K+1}\sum_{i=0}^{K} \alpha\text{-IoU}_i. \tag{14}$$

The min/max formulation reduces to the standard IoU when $\hat{\alpha}_i, \alpha_i \in \{0,1\}$ and remains well-defined for soft alpha. Higher is better.

Visible-pixel ownership (Owner Acc.). Owner accuracy answers that for each pixel the user actually sees, whether the model assigns it to the right layer. For a layer stack with index i increasing toward the front, the visible contribution of layer i at pixel p after z-ordered source-over compositing is

$$w_i(p) \;=\; \alpha_i(p)\prod_{k>i}\big(1-\alpha_k(p)\big), \tag{15}$$

and the per-pixel owner is the layer with the largest visible contribution:

$$o^{\mathrm{gt}}(p) = \arg\max_i w_i^{\mathrm{gt}}(p), \qquad o^{\mathrm{pred}}(p) = \arg\max_i w_i^{\mathrm{pred}}(p). \tag{16}$$

We restrict the metric to foreground pixels with a clearly dominant ground-truth owner: $\Omega_{\mathrm{fg}} = \{p : o^{\mathrm{gt}}(p) > 0,\ \max_i w_i^{\mathrm{gt}}(p) > \tau_{\mathrm{vis}},\ w_{(1)}^{\mathrm{gt}}(p) - w_{(2)}^{\mathrm{gt}}(p) > \delta\}$, where $w_{(1)}, w_{(2)}$ are the top-1 and top-2 visible weights and we use $\tau_{\mathrm{vis}} = \delta = 0.05$. The background layer $i = 0$ is excluded from Ω_{fg} to prevent it from dominating the metric. Owner accuracy is the fraction of Ω_{fg} for which the predicted and ground-truth owners agree:

$$\text{Owner-Acc.} = \frac{1}{|\Omega_{\mathrm{fg}}|}\sum_{p\in\Omega_{\mathrm{fg}}} \mathbf{1}\big[o^{\mathrm{pred}}(p) = o^{\mathrm{gt}}(p)\big]. \tag{17}$$

Wrong-layer leakage (Leak.). Wrong-layer leakage complements Owner accuracy by measuring *how much* predicted mass is misplaced rather than only whether the top-1 owner is correct. On each ground-truth owner mask $M_i^{\mathrm{gt}} = \{p : o^{\mathrm{gt}}(p) = i\}$, it reports the fraction of predicted visible contribution that lands on layers other than i. We use the visible-weight form w_j^{pred} rather than raw alpha so that the metric remains stable in regions with semi-transparent overlays such as drop shadows, glows, and gradient fills:

$$\text{Leak.} = \frac{\sum_i \sum_{p\in M_i^{\mathrm{gt}}} \sum_{j\neq i} w_j^{\mathrm{pred}}(p)}{\sum_i \sum_{p\in M_i^{\mathrm{gt}}} \sum_j w_j^{\mathrm{pred}}(p) + \varepsilon}. \tag{18}$$

Lower is better. The denominator normalizes by the total predicted visible mass on the same set of pixels, so the metric is a unitless leakage ratio independent of overall alpha calibration.

D.4 VLM Quality Metrics

For each test sample, we render a single PNG with the original input on the left and a packed grid of the predicted RGBA layers on the right (each cell labeled `L1`, `L2`, …, in z-order), and ask a frontier VLM to score every layer along two dimensions on a 1–5 scale: *Alpha* (matte and edge quality, opacity correctness) and *Visual* (content integrity, object completeness, clean foreground/background separation). The exact prompt is shown in the listing below; it is identical across all methods and datasets compared in Table 1, with only the candidate decomposition image differing. The VLM returns a strict JSON record with one Alpha and one Visual score per layer. We parse this record, average per sample, and then average across the test set. We report results from GPT-5.5 [54] (with *high* image detail and default reasoning effort) and Gemini-3.1 Pro [55] (with *high* media resolution and default thinking level) separately.

VLM judge prompt for layer-decomposition quality.

You are an expert judge for layer decomposition quality. You will evaluate **all layers shown in a single concatenated image**.

Image Layout

The input image has this structure:

- **Left side**: The original composite image (reference).
- **Right side**: A grid of layer cutouts (collage layout). Each cell is labeled `LN` (e.g., `L1`, `L2`, …) at the bottom. Larger elements occupy bigger cells.

Your Task

Score **ALL** the layers shown in the grid, using **two dimensions** (1–5, higher is better).

Score Scale:

- **5** = Excellent, no visible issues.
- **4** = Good with minor issues.
- **3** = Acceptable with noticeable problems.
- **2** = Poor, significant problems.
- **1** = Unusable, severe failures.

Dimension 1: Alpha Quality (Transparency & Edge Integrity)

Evaluate the **matte quality**, edge precision, and transparency correctness. This dimension focuses on the alpha channel and edge processing.

What to check:

- Edge artifacts: Haloing (light/dark outline), fringing (color bleeding at edges), jagged/aliased edges.
- Opacity correctness: Is transparency logical? (glass = semi-transparent, solid objects = opaque).
- Processing artifacts: Grid patterns in semi-transparent areas, alpha channel noise.

Score Scale:

- **5**: Excellent matte. Crisp edges, smooth transitions, no halos or artifacts.
- **4**: Good. Minor edge softness or faint halo, clean overall.
- **3**: Acceptable. Visible halo, aliasing, or minor grid artifacts.
- **2**: Poor. Strong jagged edges, heavy halos, distinct grid patterns.
- **1**: Unusable. Massive halos, wrong transparency, severe alpha failure.

Dimension 2: Visual Quality (Content Integrity & Segmentation)

Evaluate the pixel integrity, semantic correctness, **completeness**, and **clean separation** of the visible content.

CRITICAL — Object Completeness Check: For EACH layer, carefully compare it with the original image and ask:

1. **Is this a complete semantic object?** Look at the original — if you can see a whole surfboard, person, logo, etc., the layer should contain the ENTIRE object, not just a piece of it.
2. **Is the layer cleanly separated?** Background layers should NOT contain fragments of foreground objects.

What to check:

- Visual artifacts: Strange gray values, noise, compression artifacts, color bleeding.
- Structural issues: Distorted anatomy (extra fingers, melted faces), garbled text, twisted shapes.
- Content fidelity: Does it match the corresponding element in the original?

Segmentation Failures (penalize these severely):

- **Incomplete object**: The layer contains only PART of an object that is COMPLETE in the original. Example: Original shows a full surfboard → Layer shows only half of it. This is WRONG — the entire surfboard should be in one layer.
- **Foreground residue**: Background layers contain leftover pieces of foreground objects. Example: A background layer has someone's arm or part of an object floating in it.
- **Missing inpainting**: An object is partially hidden behind another object in the original, but the layer doesn't complete the hidden portion.

Note on image boundary: If an object is cut off at the IMAGE BOUNDARY in the original (e.g., half a person at the edge of the photo), the layer cannot show the missing part because the output layer has the same dimensions as the original — there's simply no canvas space beyond the boundary. This is NOT a failure. However, if an object is fully WITHIN the original image, it must be fully present in the layer.

Score Scale:

- **5**: Excellent. All objects are complete, clean separation, no artifacts.
- **4**: Good. Minor artifacts. All objects complete and cleanly separated.
- **3**: Acceptable. Some artifacts. Minor incompleteness or small residue.
- **2**: Poor. **Object is incomplete** (cut off mid-object when it is complete in the original), OR noticeable foreground residue on the background layer.
- **1**: Unusable. Severe incompleteness, major residue, or unrecognizable content.

Output Requirements

A) Return **strict JSON only** (no markdown fences, no extra text).
B) Output scores for **ALL** layers visible in the grid (`L1`, `L2`, `L3`, ...).
C) Use concise **failure_tags**:

- *Alpha*: `["halo", "jagged", "fringing", "grid_artifacts", "wrong_opacity", "edge_noise"]`.

- *Visual*: `["distorted_anatomy", "bad_text", "noise", "color_bleed", "blur", "pixel_corruption", "incomplete_object", "foreground_residue", "missing_inpainting", "abnormal_gray"]`.
- `incomplete_object` = **IMPORTANT**: object is complete in the original but layer only shows a fragment.
- `foreground_residue` = the background layer has leftover pieces of foreground objects (not cleanly removed).
- `missing_inpainting` = hidden/occluded parts are not filled in.

D) Provide a short, specific **reason** for each score.

Output JSON Schema

```
{
  "layer_count": <number of layers evaluated>,
  "layers": [
    {
      "layer_id": "L1",
      "alpha_quality": {
        "score": 1-5,
        "failure_tags": [],
        "reason": "string"
      },
      "visual_quality": {
        "score": 1-5,
        "failure_tags": [],
        "reason": "string"
      }
    }
  ],
  "overall_summary": "string"
}
```

Output strict JSON only.

We invoke the VLM in a single round per sample. Responses are validated against the JSON schema above; samples whose response does not parse are re-queried up to three times before being discarded.

E Decomposition Quality Comparison

For layer-level evaluation, CLD, MRT and PixelART take GT boxes as input, while QIL (LC) applies them through regional cropping of full-resolution visual tokens without changing its diffusion process. For bounding-box-free QIL and LayerD, GT Layer-Merge merges GT layers into the predicted partition using layer-wise similarity. This changes the evaluation references rather than model inputs, so these results are reported in a separate group in Table 2.

F User Study Protocol

The user study referenced in the main text complements the VLM-based metrics with pairwise human preferences. We describe recruitment and informed consent, the evaluation interface and protocol, and data-handling procedures.

Recruitment and informed consent. We recruit 10 annotators with diverse backgrounds. Before participation, each annotator receives a written informed-consent document covering: (i) the purpose of the study (comparing layer-decomposition methods on layer quality and composite fidelity); (ii) the procedure and expected duration (approximately 20–40 minutes); (iii) the data we collect (a self-chosen username, per-question choices, optional failure-mode tags, and optional free-text notes; no audio, video, or sensitive personal information); (iv) voluntary participation and the right to withdraw at any time, including post-hoc deletion of submitted data on request; and (v) the intended use of the results (anonymized, aggregated reporting). Annotators sign the document before beginning the task.

Evaluation interface and protocol. Annotators complete the study through a web-based questionnaire. Each question presents the original input image alongside two anonymized layer-decomposition outputs labelled A and B, produced by two different methods. For each pair, the annotator independently selects one of three options—"A is better", "B is better", or "tie"—on each of two dimensions:

- **Layer quality**: how suitable the predicted layers are for downstream editing, including segmentation accuracy, alpha and edge quality, missing or mixed objects, residual foreground content in the background layer, and plausibility of inpainted regions.
- **Composite consistency**: how faithfully the alpha-composited reconstruction matches the input, including text, small icons, edges, colour and texture, blur, and localized artefacts.

The annotator may optionally add failure-mode tags and free-text notes; these fields are not required. Method identities (the A/B assignment) are randomized per question and not disclosed during the study. Figure 8 shows the annotation interface.

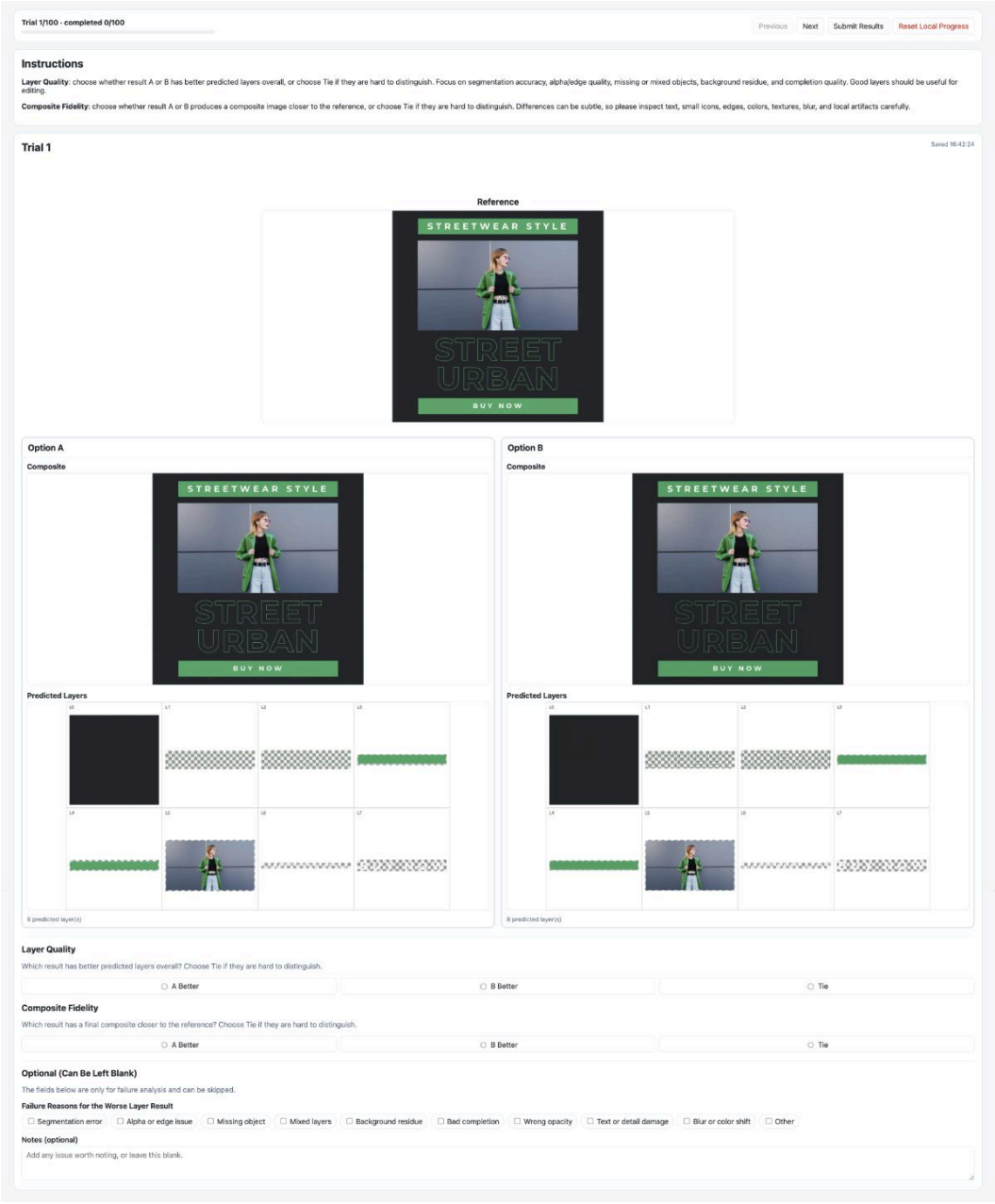


Figure 8: **Web interface used in the user study.** Each question shows the original input image, two anonymized layer-decomposition outputs (A/B), the two-dimension ternary choice, and optional fields for failure tags and free-text notes.

Data handling and privacy. Submitted responses are stored on internal infrastructure accessible only to the research team. Beyond the self-chosen username we do not collect age, contact information, or any identifying data, and annotators are instructed not to enter sensitive personal information in free-text notes. We report only aggregated, anonymized statistics; raw responses are retained internally and are deleted on participant request.

G Prediction Space and Timestep Sampling

This section discusses the prediction space and timestep sampling of PixelART. We first explain why our $\mathbf{x}$-prediction $\mathbf{v}$-loss formulation is compatible with terminal-boosted timestep sampling (Sec. 3.4), and then complement the main-text ablation (Table 5) with $\mathbf{x}$- vs. $\mathbf{v}$-prediction at the PixelART-H scale.

G.1 Compatibility with $\mathbf{x}$-prediction $\mathbf{v}$-loss

The boost distribution must avoid the clean-data endpoint. Because our $\mathbf{x}$-prediction $\mathbf{v}$-loss has an equivalent clean-prediction weight $1/(1-t)^2$, any boost density with mass arbitrarily close to $t = 1$ would make the expected loss unstable. Our candidates, $\delta(t = 0)$ and $\mathcal{U}(0, \tau)$ with $\tau < 1$, are bounded away from $t = 1$, so

$$\mathbb{E}_{t\sim q_{\text{boost}}}\left[\frac{1}{(1-t)^2}\left\|\widehat{\mathbf{Y}}_\theta - \mathbf{Y}\right\|_2^2\right] < \infty. \tag{19}$$

For $\mathcal{U}(0, \tau)$, the weight is upper-bounded by $1/(1-\tau)^2$.

G.2 PixelART-H $\mathbf{x}$- vs. $\mathbf{v}$-Prediction

The main-text ablation (Table 5) shows that $\mathbf{v}$-prediction fails to converge at the PixelART-B scale, where each $16\times16\times4$ RGBA patch has dimensionality 1024 exceeding the model's hidden width of 768. To test whether the failure is purely a capacity issue, we repeat the comparison at the PixelART-H scale (d=1280, exceeding the patch dimensionality) with the convolutional bottleneck removed, so the per-token dimensionality entering the Transformer is 1024 rather than the bottleneck width. The training recipe otherwise follows the PixelART-H ablation default in Table 11.

Table 15: **$\mathbf{x}$- vs. $\mathbf{v}$-prediction at PixelART-H scale** (d=1280, no bottleneck). Both runs use identical recipes apart from the prediction target. At this width, $\mathbf{v}$-prediction *does* converge but still trails $\mathbf{x}$-prediction, indicating that the manifold mismatch is not solely a capacity issue.

	Layer metrics			Composite metrics		
Prediction	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
$\mathbf{x}$-pred	**0.0453**	**32.88**	**0.8995**	**0.0086**	**38.55**	**0.9727**
$\mathbf{v}$-pred	0.0526	30.17	0.8691	0.0114	36.08	0.9441

At PixelART-H, $\mathbf{v}$-prediction trails $\mathbf{x}$-prediction by -2.71 dB Layer PSNR and -2.47 dB Composite PSNR. This residual gap is consistent with the manifold argument of JiT [28]: design layers occupy a much lower-dimensional manifold than the velocity target, and predicting clean pixels remains a better-conditioned regression problem than predicting velocities, even at high width.

G.3 Shifted Logit-Normal Baseline

As a simpler alternative to terminal boosting, we shift the logit-normal mean from $\mu = -0.8$ to -1.0, -1.25, and -1.5 to emphasize high-noise timesteps. All three shifts improve assignment metrics over the base schedule but reduce both Layer and Composite PSNR (Table 16). Our default terminal boost achieves the best score on all nine metrics.

H RGBA-VAE Decoder for the Latent-Space Baseline

The pixel-space versus latent-space comparison (Table 3) trains a PixelART-B variant on latents produced by the frozen RGBA-VAE encoder from QIL [2], using a latent patch size of 2 to account

Schedule	Composite			Layer			Assignment		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	α-IoU↑	Owner↑	Leak.↓
Base ($\mu = -0.8$)	0.0132	34.36	0.9539	0.0626	28.66	0.8609	0.8983	0.9040	0.1245
Shifted ($\mu = -1.0$)	0.0133	33.90	0.9511	0.0606	28.52	0.8634	0.9012	0.9248	0.1058
Shifted ($\mu = -1.25$)	0.0166	31.65	0.9287	0.0660	26.83	0.8476	0.9104	0.9224	0.1084
Shifted ($\mu = -1.5$)	0.0195	30.06	0.9045	0.0731	25.36	0.8232	0.9158	0.9312	0.1016
Terminal boost	**0.0119**	**34.53**	**0.9568**	**0.0516**	**29.51**	**0.8802**	**0.9466**	**0.9358**	**0.0944**

Table 16: **Shifted logit-normal versus terminal boosting on PixelART-B.** The logit-normal standard deviation is $\sigma = 0.8$. Terminal boost uses $\mathcal{U}(0, 0.2)$ with $p = 0.1$.

for spatial down-sampling. The diffusion training settings otherwise follow the PixelART-B ablation default in Table 11. PixelART operates on bounding-box-conditioned regional latents, but the official QIL decoder reconstructs only full-canvas latents and cannot be applied directly. We therefore reuse a separately trained ViT-based regional RGBA decoder (300M parameters) that maps cropped regional latents to cropped RGBA layers, which are then composited into the canvas. Both the QIL encoder and this decoder remain frozen during latent PixelART training. The matched parameter budget in Table 3 covers only the diffusion Transformer and excludes the latent baseline's encoder and decoder.

Table 17: **RGBA-VAE decoder reconstruction quality** on DESIGN-MULTI-LAYER-BENCH at 256×256. Each decoder takes the official QIL encoder's latent of a ground-truth RGBA layer and produces a reconstruction, which is compared against the original. Both decoders share the same encoder. Layer L1/PSNR are alpha-pre-multiplied (Appendix D); layer SSIM is alpha-weighted.

Decoder	Composite metrics			Layer metrics		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑
QIL (official)	0.0132	32.52	0.9568	0.0363	31.66	0.9289
Ours (300M ViT)	**0.0083**	**33.69**	**0.9686**	**0.0254**	**37.49**	**0.9569**

Table 17 reports reconstruction metrics for our trained decoder and the official QIL decoder under the same encoder. Our decoder is competitive with the official one on every metric, providing a comparable upper bound for any latent diffusion model that uses this encoder.

I T2I Pretraining

For the T2I-pretraining ablation in Table 4, we pretrain PixelART-B (168M parameters) on the public BLIP-3o corpus [60] ($\sim$36M text–image pairs) for 250K steps with standard rectified-flow training, then fine-tune on our I2L corpus with the same recipe as the from-scratch baseline. Unless stated otherwise, all training hyperparameters follow the PixelART-B ablation default in Table 11. To verify that the pretrained checkpoint produces coherent T2I outputs, we evaluate it on GenEval [65] *before* I2L fine-tuning. The full breakdown is:

Table 18: **GenEval scores of the BLIP-3o-pretrained PixelART-B** (168M params, batch 1024, 250K iters), evaluated before I2L fine-tuning. Overall is the per-task average. The model does not adopt any specialized optimization recipes common in dedicated T2I models, yet still produces non-degenerate outputs across all six task categories at a level consistent with this parameter scale and training budget. This checkpoint serves as the T2I-initialized starting point for the I2L ablation in Table 4.

	single obj.	two obj.	counting	colors	color attr.	position	**overall**
PixelART-B (BLIP-3o, T2I)	74.38	24.24	18.12	61.70	12.75	7.50	**33.12**

For completeness, the prompt-level statistics are: total images 2,212, total prompts 553, correct images 31.87%, correct prompts 53.16%.

Effect of I2L data scale. We repeat the PixelART-B initialization comparison on 1M layered design samples with the same 100K-step I2L budget. Table 19 shows mixed effects on final quality,

consistent with the 4M results in Table 4. These experiments do not establish how stronger generative or inpainting pretraining would affect natural images with complex occlusions.

Initialization	Composite			Layer			Assignment		
	L1↓	PSNR↑	SSIM↑	L1↓	PSNR↑	SSIM↑	α-IoU↑	Owner↑	Leak.↓
From scratch	**0.01308**	**34.24**	**0.9537**	0.06416	**28.55**	0.8569	0.8936	**0.9214**	**0.1078**
T2I pretrained	0.01392	33.92	0.9495	**0.06407**	28.29	**0.8607**	**0.8999**	0.9032	0.1257

Table 19: **Effect of T2I initialization with 1M I2L training samples.** Both PixelART-B runs use 100K I2L updates.

Training cost at PixelART-L scale. To quantify the additional pretraining cost, we estimate 1.8×10^{20} denoiser training FLOPs and measure 1280 H200 GPU-hours for T2I pretraining (250K steps), compared with 1.3×10^{20} FLOPs and 376 GPU-hours for I2L training (100K steps), using PixelART-L for both stages. Although runtimes depend on implementation choices such as caching text embeddings for I2L, this comparison highlights the computational benefit of omitting T2I pretraining while keeping the I2L budget unchanged.

J Generalization and Adaptation to Natural Images

The base PixelART model is trained exclusively on multi-layer design templates. Figure 9 shows representative successful zero-shot decompositions on natural images outside this training distribution.

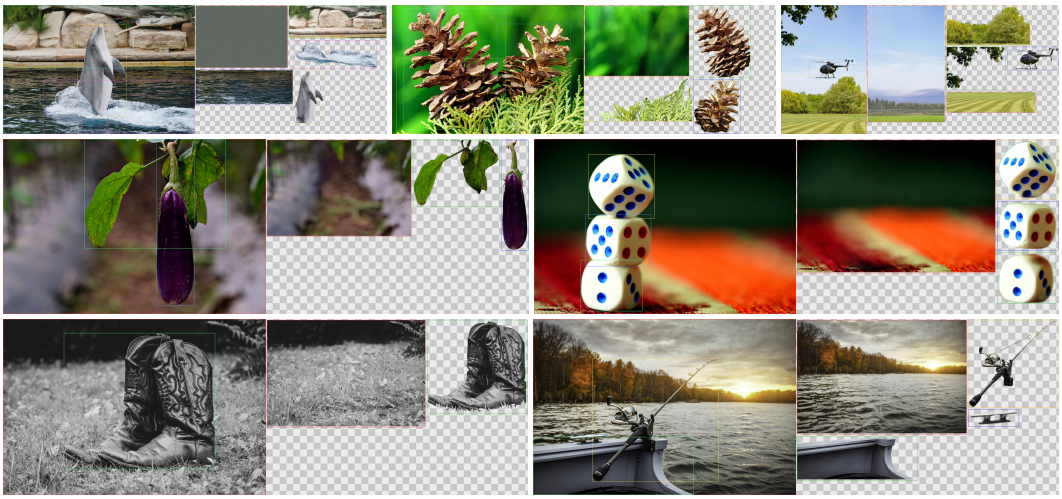


Figure 9: **Zero-shot decomposition on natural images.** Although trained only on multi-layer design templates, PixelART produces coherent foreground/background separation and plausible occluded-region completion on selected natural-image inputs.

We fine-tune PixelART for 3K steps on approximately 2K layered natural-like images from an internal 3D synthetic dataset. On nearly 100 natural images, fine-tuning improves all five metrics and yields better composite reconstruction and visual-quality ratings than QIL (Table 20).

Model	Composite metrics			GPT ratings	
	L1↓	PSNR↑	SSIM↑	Alpha↑	Visual↑
QIL	0.02120	30.89	0.8932	3.702	2.854
PixelART	0.01487	33.21	0.9227	3.604	3.044
PixelART + fine-tuning	**0.01409**	**33.58**	**0.9266**	**3.707**	**3.208**

Table 20: **Natural-image adaptation.** Evaluation on nearly 100 natural images.

K Limitations

We summarize the most prominent limitations of PixelART in this section.

- **Complex occlusion completion.** Filling in regions hidden behind large foregrounds is the model's clearest weakness, particularly on natural photographs. We suspect that limited coverage of dense photographic occlusions in the training data contributes to these failures.
- **Reduced reliability on natural images.** As shown in Appendix J, the base model can produce coherent decompositions on natural photographs, but the success rate is noticeably lower and less stable than on design content. Beyond the completion artefacts above, failures on natural inputs frequently take the form of *layer assignment errors*: a near-empty foreground layer, residual foreground content in the background, multiple elements merged into one layer, or a single object split across layers. Our fine-tuning experiment shows that additional layered supervision improves natural-image decomposition.
- **Unusually large K.** PixelART supports variable K through anonymous regional tokens, but only up to a maximum of $K{=}32$ layers seen during training. For inputs that exceed or approach this limit, reconstruction precision drops noticeably and assignment errors become more frequent.
- **Requires external layout.** PixelART consumes a bounding-box layout, supplied either by the user or by an automatic detector (Appendix C); it does not operate end-to-end on the input image alone, unlike bounding-box-free baselines such as QIL.
- **Limited reproducibility from public assets.** Because the 4M training corpus and the internal layout detector are not publicly released, the full system cannot be exactly reproduced from public assets alone. To support independent verification, we provide PixelART predictions, references, layouts, per-sample metrics and evaluation scripts for the GT-box and DAD-box settings on LICA, plus frozen VLM scores and judge grids for the DAD setting at `https://github.com/hmk114/pixelart-eval`. The full architecture and training recipe are specified in Appendix B, enabling reimplementation on alternative layered datasets.

L Broader Impacts

Positive impacts. PixelART targets design and editing workflows. By recovering editable RGBA layer stacks from rasterized images, it reduces the manual cost of repurposing, retargeting, or updating visual content, and lowers the barrier to image editing for users without professional design experience.

Potential misuse. As with other generative editing systems, PixelART can be misused. The most direct risks are: (i) image manipulation with deceptive intent, such as removing or altering disclaimers, context-providing details, or other visible information; (ii) decomposition of copyrighted design assets to facilitate unauthorized reuse; and (iii) removal of watermarks, signatures, or other provenance markers. These risks are bounded relative to full T2I generation, since PixelART operates on an existing input rather than fabricating new scenes from scratch, but they are direct enough to warrant explicit acknowledgement.

Mitigations. For any production deployment of this technology, we recommend a small set of pre-input safeguards: (i) automatic detection of copyrighted or otherwise protected material in the input image, with processing refused on positive matches; and (ii) automatic rejection of requests whose bounding-box layout targets watermarks, signatures, or other provenance markers. We also encourage the broader community to develop forensic methods that can detect layer-level re-editing, complementing existing image-tampering detection work.

M External Assets and Licenses

We list the external assets used in this work, citing the original creators and the licenses under which the assets are released. All assets are used in accordance with their respective licenses.

Models.

- **Qwen3-VL-2B-Instruct** [57], used as the base model for the internal layout detector. Apache License 2.0.

- **QIL** [2], used as a system-level baseline (Table 1) and as the source of the RGBA-VAE encoder for the latent-space ablation (App. H). Apache License 2.0.
- **CLD** [39], used as a system-level baseline (Table 1). License inherited from FLUX.1-dev: FLUX.1 [dev] Non-Commercial License.
- **LayerD** [34], used as a non-diffusion paradigm reference (Table 1). Apache License 2.0.

Datasets and benchmarks.

- **LICA** [53], used as an evaluation benchmark in Table 1. CC BY 4.0.
- **BLIP-3o pretraining corpus** [60], used for the T2I-pretraining ablation (App. I). Apache License 2.0.
- **GenEval** [65], used to evaluate the T2I-pretrained checkpoint (App. I). MIT License.

Closed external APIs.

- **GPT-5.5** (OpenAI), used as a VLM judge for layer quality scoring (App. D.4). Use is subject to OpenAI's API usage policies.
- **Gemini-3.1 Pro** (Google), used as a second VLM judge. Use is subject to Google's Generative AI API terms of service.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Main claims in the abstract and introduction are backed by the method description in Section 3, and the experiments in Section 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations as future work in Sec. K.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The algorithmic details—architecture, training recipe, sampling procedure, and evaluation protocol—are fully specified in Section 3 and Appendix B, D, F, C, and H, enabling reimplementation on alternative layered datasets. Exact reproduction of the main model is limited because the 4M training corpus and the internal layout detector are not publicly released; we therefore provide a verification path on the public LICA benchmark by releasing PixelART predictions, references, layouts, per-sample metrics and evaluation scripts for the GT-box and DAD-box settings, plus frozen VLM scores and judge grids for the DAD setting (Appendix K).

Guidelines:

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

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: The 4M training corpus and internal layout detector cannot be publicly released due to policy constraints. To support independent verification despite this, we provide PixelART predictions, references, layouts, per-sample metrics and evaluation scripts for the GT-box and DAD-box settings on LICA, plus frozen VLM scores and judge grids for the DAD setting. The full architecture, training recipe, and evaluation protocol are specified in Appendix B, D, and F so that the system can be reimplemented on alternative layered datasets. See Appendix K for further discussion of reproducibility scope.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have provided details for the experimental setup in Section 4 and Table 11.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We did not report the variance due to the resource limitation.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the compute resources to conduct our experiments in Table 11.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and made sure that the paper conforms to it.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

Justification: Appendix L discusses both positive impacts and potential negative impacts, along with possible pre-input mitigations.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not foresee any high risk for misuse of this work.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: External models, datasets, and APIs used in this work are listed in App. M with citations to the original creators and the licenses under which each asset is released. The internal training corpus and layout detector are described separately in Apps. A and C.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The PixelART evaluation release includes documentation of its LICA coverage, data formats, verification commands and known limitations.

    Guidelines:

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

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

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

    Answer: [Yes]

    Justification: We provide the screenshot of the web-based user study page and compensation details in Appendix F.

    Guidelines:

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

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

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

    Answer: [Yes]

Justification: The paper describes the human evaluation protocol in Appendix F. The study has obtained IRB approval.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: We describe the usage of LLMs in our evaluation process in D.4.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.