ALPHAANYTHING: EXTENDING PRETRAINED RGB GENERATORS TO RGBA

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing RGBA generation methods commonly learn a transparency-specific representation through reconstruction, introducing an additional training stage and potentially altering the RGB decoding function inherited from pretrained models. We present ALPHAANYTHING, a lightweight framework that extends pretrained RGB generators to RGBA directly in their original latent space, without RGBA reconstruction pretraining. ALPHAANYTHING inserts LoRA modules into the generative transformer and attaches an alpha branch to the frozen RGB VAE decoder. Unidirectional RGB-to-alpha feature sharing allows opacity supervision to shape the generative latent while guaranteeing that, for any fixed latent, RGB decoding remains identical to that of the pretrained decoder. We further introduce Endpoint-Aware Alpha Loss (EA Loss), which derives opacity supervision from the complementary foreground–background weights used in compositing. It supports exact opacity endpoints and continuous partial transparency while emphasizing localized alpha errors. Experiments on text-to-RGBA and image-to-RGBA tasks across multiple pretrained RGB backbones demonstrate competitive generation quality, and controlled ablations validate the shared-latent training design and EA Loss.

1 INTRODUCTION

Pretrained image generators can synthesize high-quality visual content from text or image conditions (Rombach et al., 2022; Wu et al., 2025; 2026). Many downstream applications, however, require assets that can be directly layered and composited into new scenes, rather than RGB images tied to a fixed background (Fontanella et al., 2024; Yin et al., 2026). RGBA images support this use case by representing appearance and opacity separately: the RGB channels describe foreground color, while the alpha channel controls how the foreground is blended with its background (Xu et al., 2017; Zhang & Agrawala, 2024). A practical RGBA extension should therefore add transparency generation while reusing the conditioning and appearance priors of a pretrained RGB model.

Existing RGBA generation methods commonly introduce a transparency-specific representation through reconstruction training. LayerDiffuse learns a transparency encoder and decoder with reconstruction supervision (Zhang & Agrawala, 2024), while AlphaVAE and Qwen-Image-Layered train dedicated four-channel autoencoders (Wang et al., 2025b; Yin et al., 2026). Although effective, this strategy adds a separate representation-learning stage and couples subsequent generative training to the reconstructed RGBA latent space. Updating an RGB autoencoder for RGBA reconstruction can also change its pretrained RGB decoding function, with potential losses in color or high-frequency detail. Zippo (Xie et al., 2024) shows that a dedicated RGBA VAE is not strictly necessary by encoding replicated alpha with a pretrained RGB VAE, but it still models appearance and opacity with separate latent states. These observations motivate a simpler question: can a pretrained RGB generator learn opacity directly in its existing latent space, using generative supervision alone?

We answer this question with ALPHAANYTHING, a lightweight framework that augments a pretrained RGB generator with a shared-latent alpha pathway. We insert LoRA modules (Hu et al., 2022) into the generative transformer and attach a trainable alpha branch to the frozen RGB VAE decoder. During training, flow-matching supervision updates the transformer LoRA, while opacity supervision passes through the frozen decoder features to shape the same predicted latent. Within the decoder, information flows only from the RGB pathway to the alpha branch; the alpha state is never fed back

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



Figure 1: Representative text- and image-to-RGBA results. Each prompt specifies an asset to generate or extract. For text-to-RGBA, independently generated layers are manually arranged into the displayed compositions.

into the RGB pathway. Consequently, for any fixed latent, the augmented decoder produces exactly the same RGB output as the original pretrained decoder, while predicting alpha from the RGB feature hierarchy. RGB and alpha are thus obtained from one latent and one augmented decoder traversal, without an RGBA autoencoder or a separate alpha latent.

Learning opacity also requires an objective that reflects its role in compositing. Alpha is not merely a bounded image channel: $a = 0$ and $a = 1$ denote exact transparency and opacity, intermediate values encode partial visibility, and $(1 - a, a)$ gives the complementary background–foreground mixing weights. We therefore introduce *Endpoint-Aware Alpha Loss (EA Loss)*. EA Loss represents

this compositing pair on the Bernoulli simplex and uses its Fisher–Rao coordinate (Miyamoto et al., 2024) to parameterize the full opacity range. An endpoint-compatible Fenchel–Young formulation represents exact endpoints with finite scores and retains corrective gradients when a prediction saturates at the wrong endpoint. The same complementary weights further separate background- and foreground-associated errors before root aggregation, increasing the relative contribution of localized defects such as thin structures, small holes, and background leakage. We characterize these properties theoretically and evaluate the complete objective against standard alpha L_1 supervision.

The paper makes three contributions. **(1) Shared-latent RGBA extension.** We show that pretrained RGB generators can acquire text- and image-conditioned RGBA generation without RGBA reconstruction pretraining. A unidirectional RGB-to-alpha decoder branch lets opacity supervision shape the generative latent while preserving the pretrained RGB decoding function for every fixed latent. **(2) Endpoint-aware opacity supervision.** We derive EA Loss from the complementary compositing weights of alpha. The objective accommodates exact endpoints and continuous partial transparency while emphasizing spatially localized alpha errors, and we establish its key optimization and error-control properties. **(3) Evaluation across tasks and backbones.** We instantiate ALPHAANYTHING on multiple pretrained RGB generators and evaluate it in both text-to-RGBA and image-to-RGBA settings, where it achieves competitive generation quality without an RGBA autoencoder or a separate alpha latent.

2 RELATED WORK

RGB image generation. Modern diffusion and flow-based image generators learn powerful text- and image-conditioned visual priors in pretrained RGB latent spaces (Rombach et al., 2022; Peebles & Xie, 2023; Wu et al., 2025; 2026). These models provide strong appearance synthesis and editing capabilities, but their outputs remain conventional three-channel images whose content is tied to an RGB background. Parameter-efficient training methods such as LoRA (Hu et al., 2022) further show that new generative capabilities can be introduced while largely reusing a pretrained backbone. Our work builds on this premise, extending pretrained RGB generators with opacity generation rather than replacing their learned RGB representation.

RGBA generation and representations. Image matting estimates opacity for content already observed in an RGB image, classically with auxiliary spatial guidance such as a trimap or mask (Xu et al., 2017). It therefore differs from RGBA generation: matting operates on visible image evidence rather than semantic generation instructions, does not jointly synthesize appearance and opacity, and typically recovers the visible matte rather than completing object layers behind occlusions. Existing native RGBA methods adopt different transparency representations. LayerDiffuse learns a latent transparency representation and decoder through reconstruction training (Zhang & Agrawala, 2024), while AlphaVAE and Qwen-Image-Layered train dedicated four-channel autoencoders (Wang et al., 2025b; Yin et al., 2026). These approaches introduce an additional reconstruction-trained representation to support transparency generation. Zippo takes a different route by reusing a pretrained RGB VAE and representing alpha through replicated RGB channels (Xie et al., 2024); however, color and matte are still modeled with separate latent states. These works show that transparency can be incorporated through substantially different representation choices. ALPHAANYTHING instead learns RGBA generation in the original RGB latent space while retaining the pretrained RGB autoencoder.

3 ALPHAANYTHING

Our central insight is that RGBA generation can be learned in the original RGB latent space. ALPHAANYTHING realizes this with DiT LoRA modules and a lightweight alpha branch embedded in the frozen RGB VAE decoder. Within the alpha branch, features are shared unidirectionally from RGB to alpha, preserving the original RGB decoding process for any fixed latent, while gradients from the alpha loss propagate through the decoder to update the DiT LoRA parameters. Our Endpoint-Aware Alpha Loss (EA Loss) further provides supervision for both exact endpoints and continuous opacity. Figure 2 summarizes the framework.

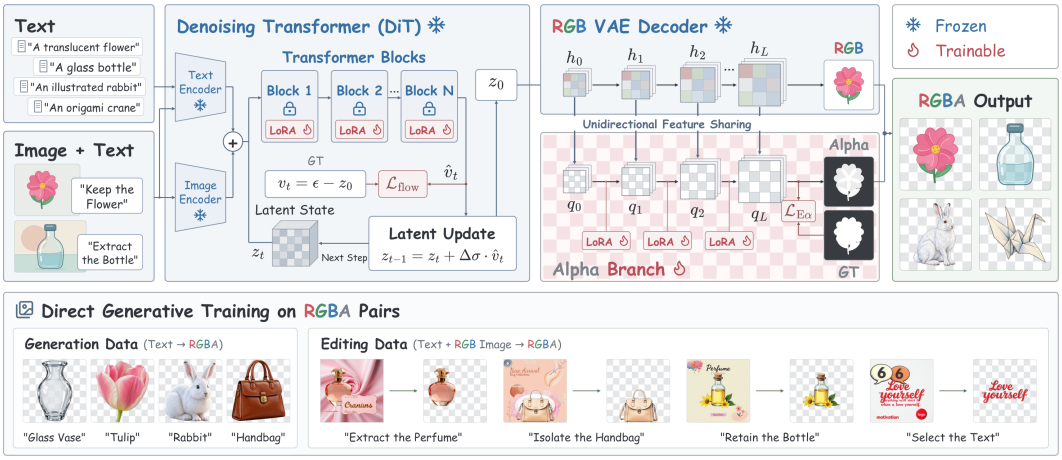

Figure 2: Overview of ALPHAANYTHING. Transformer LoRA and a VAE-embedded alpha branch learn RGBA generation through generative training. Unidirectional feature sharing keeps RGB decoding for any fixed latent.

3.1 PRELIMINARIES

RGBA image. A transparent image is represented as an RGBA layer $I_t = (I_c, I_\alpha)$, where $I_c \in [0,1]^{N\times 3}$ is the *straight* foreground color and $I_\alpha \in [0,1]^N$ is opacity, with $N = HW$. Given a background $B \in [0,1]^{N\times 3}$, compositing in a common linear color space is

$$C_B(I_t) = I_\alpha \odot I_c + (1 - I_\alpha) \odot B = I_p + (1 - I_\alpha) \odot B, \tag{1}$$

where $I_p = I_\alpha \odot I_c$ is the *premultiplied* foreground and opacity is broadcast over color channels. The endpoints $I_{\alpha,i} = 0$ and 1 represent complete transparency and opacity; intermediate values represent partial transparency. For brevity, we denote I_α by a below.

RGB training targets. As implied by the compositing equation in (1), a composite does not uniquely determine its foreground color–opacity pair. In particular, when opacity vanishes, the stored RGB values are physically undefined and do not affect the composite (Zhang & Agrawala, 2024). However, generative training in the pretrained RGB latent space requires a well-defined RGB target over the entire image, including transparent regions; leaving these values unconstrained would make the training target underdetermined. We therefore use the multiscale color-padding convention of LayerDiffuse to define a dense RGB target:

$$\widetilde{I}_c = \mathrm{Pad}(I_c, a), \qquad z_0 = E(\widetilde{I}_c). \tag{2}$$

The padding propagates foreground color from coarse to fine into transparent regions, providing a structured RGB target from which the frozen RGB encoder E defines the target latent z_0.

3.2 EXTENDING RGB GENERATION WITH OPACITY

Our goal is to endow a pretrained RGB generator with opacity generation without introducing a separate transparency latent or altering its pretrained RGB decoding function. Our key insight is that opacity can be learned directly in the original RGB latent space: alpha supervision is propagated back to the generator, so the sampled latent is jointly shaped by appearance and opacity. Opacity is therefore learned as part of generation rather than estimated post hoc from a completed RGB image.

Let θ_0 denote the pretrained generator parameters, ψ its trainable LoRA parameters, and D the frozen RGB decoder. The extended model produces

$$z = G_{\theta_0,\psi}(\epsilon, c), \qquad \widehat{I}_c = D(z), \qquad s = H_\phi(z; D), \tag{3}$$

where $\epsilon \sim \mathcal{N}(0, I)$ is sampling noise and H_ϕ is the trainable alpha branch producing opacity scores $s \in \mathbb{R}^N$. RGB and opacity thus originate from the same sampled latent z, with no separately encoded or sampled alpha latent. Because gradients from the alpha objective pass through the frozen decoder to ψ, opacity supervision also shapes the latent produced by the generator.

The alpha branch runs alongside the spatial hierarchy of D and reuses the RGB features produced during the same decoder traversal. Let h_ℓ denote the RGB feature at spatial stage ℓ, and let q_ℓ denote the spatially aligned alpha state. At the decoder bottleneck, we initialize $q_0 = A_\phi(h_0)$ and refine it with cross-attention using alpha queries and RGB keys and values. At each subsequent stage,

$$\bar{q}_\ell = q_\ell + \gamma_\ell B^{3\times3}_{\phi,\ell}\left(A^{1\times1}_{\phi,\ell}(h_\ell)\right), \qquad q_{\ell+1} = U_{\phi,\ell}(\bar{q}_\ell), \tag{4}$$

where $A^{1\times1}_{\phi,\ell}$ reduces the C_ℓ RGB channels to a low-rank bottleneck of width $r_\ell \ll C_\ell$, and $B^{3\times3}_{\phi,\ell}$ spatially projects the compressed feature into the alpha state. The learned gate γ_ℓ controls the contribution of the RGB feature at each stage, while $U_{\phi,\ell}$ contains the alpha-only residual refinement, channel transition, and learned upsampling that advance the alpha state to the next spatial scale. A final spatial head maps the last alpha state to s; Section 3.3 defines the corresponding opacity mapping. This reuse keeps the extension lightweight: only the alpha stream and low-rank RGB-to-alpha projections are added, with no additional RGB-decoder forward.

Crucially, all cross-branch dependencies are directed from RGB to alpha. The RGB hierarchy remains

$$h_{\ell+1} = d_\ell(h_\ell), \tag{5}$$

where d_ℓ is the original frozen RGB operation and has no dependence on q_ℓ or ϕ.

Proposition 1 (Fixed-latent RGB preservation)**.** *For every fixed latent z and alpha parameter setting ϕ, the augmented decoder's RGB output satisfies*

$$D^{\mathrm{RGB}}_{\mathrm{aug}}(z;\phi) = D(z), \qquad \frac{\partial D^{\mathrm{RGB}}_{\mathrm{aug}}(z;\phi)}{\partial \phi} = 0. \tag{6}$$

This follows directly from (5): RGB decoding is independent of the alpha state, so for any fixed latent its output is identical to that of the pretrained decoder. Meanwhile, D remains differentiable, allowing gradients from the alpha loss to propagate through it and supervise the generator.

3.3 Endpoint-Aware Alpha Loss

Alpha determines foreground–background mixing through the complementary weights $(a, 1-a)$. Our central idea is to construct both the pixelwise discrepancy and its spatial aggregation from this compositing pair. We use the Fisher–Rao geometry of the pair to define an opacity coordinate, and reuse its two components to weight the pixelwise gaps before separate root reductions.

Compositing-pair geometry. Equation (1) associates each opacity value a with the background–foreground weight pair $(1-a, a)$. This pair lies on the Bernoulli simplex, with $a = 0$ and $a = 1$ at its two vertices and partial opacity in its interior. Its Fisher information is $\mathcal{I}(a) = 1/[a(1-a)]$ for $0 < a < 1$. The corresponding normalized Fisher–Rao arc-length coordinate (Miyamoto et al., 2024) and inverse are

$$u = \chi(a) = \frac{2}{\pi}\arcsin\sqrt{a}, \qquad a = f(u) = \sin^2\!\left(\frac{\pi u}{2}\right). \tag{7}$$

Both extend continuously to the two simplex vertices. This geometry is non-uniform in physical alpha: near transparency, $\chi(\delta)^2 = 4\delta/\pi^2 + O(\delta^2)$ as $\delta \downarrow 0$, with symmetric behavior near opacity, while it remains locally smooth in the interior. Consequently, small residual foreground mass near transparency and residual background mass near opacity receive greater separation in the supervision space, while genuine partial opacity remains represented continuously.

Score supervision in the pair coordinate. To predict this coordinate while covering both the interior and the vertices of the simplex, the alpha branch outputs an unconstrained score $s \in \mathbb{R}$. We map it to the Fisher–Rao coordinate by $p(s) = \mathrm{clip}((s+1)/2, 0, 1)$ and predict $\widehat{a} = f(p(s))$. Finite scores $s \le -1$ and $s \ge 1$ therefore represent the two compositing vertices exactly. Direct regression through this clipped mapping, however, would have zero gradient when a prediction saturates at the wrong vertex. We instead supervise s with a Fenchel–Young loss (Blondel et al., 2020) for the target coordinate $u = \chi(a)$:

$$g(s,u) = \begin{cases} u(u-1-s), & s \le -1, \\ ((s+1)/2 - u)^2, & -1 < s < 1, \\ (1-u)(s-u), & s \ge 1. \end{cases} \tag{8}$$



Figure 3: Qualitative text-to-RGBA comparison. ALPHAANYTHING generates detailed lettering and feathers while capturing glass transparency.

Table 1: Quantitative results of text-to-RGBA generation on RGBA-Text.

Method / backbone		CLIP Score ↑	Aesthetic ↑	Image Reward ↑
LayerDiffuse		0.8542	5.4505	0.5921
AlphaVAE		0.8306	5.2679	0.5159
OmniAlpha		0.8508	5.2850	0.7535
Qwen-Image-2.1		0.8465	5.3011	0.6189
ALPHAANYTHING	FLUX.1	0.8353	5.5256	0.7588
	FLUX.2	0.8656	5.5125	0.9752
	Qwen Image	**0.8684**	5.5240	0.9477
	LongCat	0.8657	**5.5295**	**1.0013**

Table 2: Quantitative results of text-to-RGBA generation on AlphaTest.

Method / backbone		FID ↓	CLIP Score ↑	Aesthetic ↑	Image Reward ↑
LayerDiffuse		81.2141	**0.8487**	5.5169	1.1365
AlphaVAE		74.3047	0.8214	5.5919	1.1098
OmniAlpha		72.7426	0.8384	5.3542	1.2072
Qwen-Image-2.1		**72.2720**	0.8285	5.4672	1.1578
ALPHAANYTHING	FLUX.1	77.3089	0.8096	5.7105	1.2880
	FLUX.2	75.1796	0.8442	5.6352	1.4464
	Qwen Image	75.8076	0.8342	5.6799	1.3758
	LongCat	74.9846	0.8413	**5.7171**	**1.5050**

Within the active interval $-1 < s < 1$, g is exactly the squared coordinate error, while its gradient satisfies $\partial g/\partial s = p(s) - u$ everywhere. Thus an incorrectly transparent prediction receives gradient $-u$, and an incorrectly opaque prediction receives $1 - u$, allowing either to leave the saturated region. Correct endpoint predictions remain exactly representable, and every partial-opacity target has the finite exact solution $s = 2u - 1$.

Pair-weighted aggregation. Let $g_i = g(s_i, \chi(a_i))$ denote the per-pixel gap over N pixels. We use the target compositing weights to separate foreground- and background-associated contributions of the same pixelwise gap. Applying a square-root reduction to each contribution gives:

$$R_0 = \sqrt{\frac{1}{N}\sum_i (1 - a_i) g_i}, \qquad R_1 = \sqrt{\frac{1}{N}\sum_i a_i g_i}, \qquad \ell_{\mathrm{E}\alpha}(s, a) = R_0 + R_1. \tag{9}$$

The two weights assign each error according to its target background and foreground mass: transparent pixels contribute to R_0, opaque pixels to R_1, and partial-opacity pixels continuously to both. For a binary target, suppose nonzero gaps are confined to a fraction ρ of pixels in one endpoint group, with mean gap $\bar{g}$ over that region. The corresponding root contribution is $\sqrt{\rho\bar{g}}$, whereas direct averaging gives $\rho\bar{g}$. Thus, for a fixed mean gap, the root contribution decreases with the square root of the region's area fraction. We select a zero generalized gradient for an exactly zero root. Since $(1 - a_i) + a_i = 1$ and each gap vanishes exactly at its target, $\ell_{\mathrm{E}\alpha} = 0$ if and only if $\widehat{a} = a$.

3.4 JOINT GENERATIVE TRAINING AND INFERENCE

We jointly optimize the transformer LoRA ψ and alpha branch ϕ, while keeping the pretrained transformer parameters θ_0 and the RGB VAE encoder and decoder frozen. We follow the backbone's standard flow-matching objective. Given the clean target latent z_0 in (2), noise $\epsilon \sim \mathcal{N}(0, I)$, and

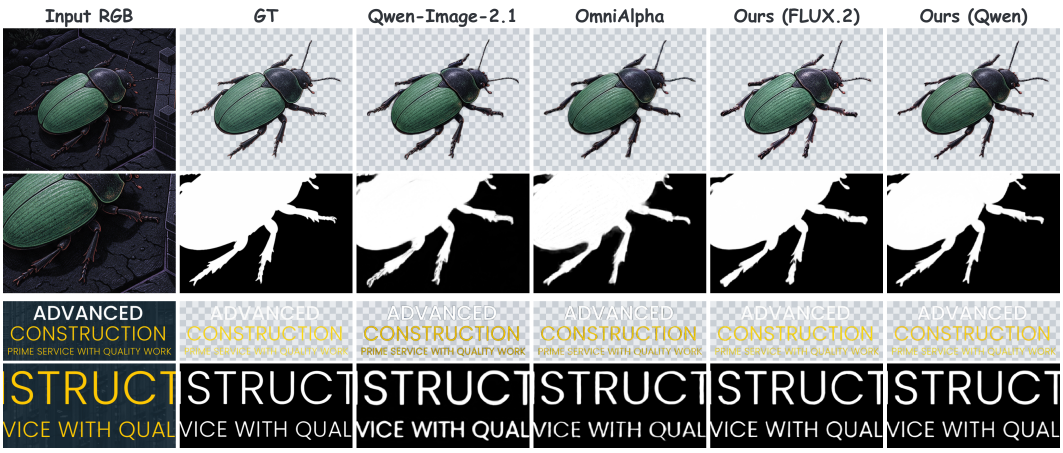

Figure 4: Image-to-RGBA comparison. ALPHAANYTHING produces cleaner alpha boundaries.

Table 3: Quantitative results of image-to-RGBA evaluation on RGBA-Image.

Method / backbone		RGB L1↓	Soft IoU↑	SAD↓	LPIPS↓	Alpha Grad↓	Alpha Conn↓
OmniAlpha		0.1052	0.6881	178.4325	0.0911	70.8791	176.9433
Qwen-Image-2.1		0.0868	0.6932	181.5245	0.0830	81.6191	166.0596
ALPHAANYTHING	FLUX.2	0.1155	0.8296	46.3572	0.0207	69.5947	46.0840
	Qwen Image Edit	**0.0746**	**0.9223**	**30.3820**	**0.0136**	**27.5221**	**30.0584**

noise level $\sigma \in [0, 1]$, the noisy latent and target velocity are

$$z_\sigma = (1-\sigma)z_0 + \sigma\epsilon, \qquad v^\star = \epsilon - z_0. \tag{10}$$

The model predicts

$$\widehat{v} = v_{\theta_0,\psi}(z_\sigma, \sigma, c), \qquad \widehat{z}_0 = z_\sigma - \sigma\widehat{v}, \tag{11}$$

where $\widehat{z}_0$ is the corresponding one-step clean-latent estimate. The alpha branch is applied to $\widehat{z}_0$ and supervised against the target opacity a.

The complete objective is

$$\mathcal{L} = \mathbb{E}\left[\|\widehat{v} - v^\star\|_2^2 + \lambda_{\mathrm{E}\alpha} w(\sigma)\ell_{\mathrm{E}\alpha}(H_\phi(\widehat{z}_0; D), a)\right], \qquad w(\sigma) = \min\left\{1, \frac{0.5^2}{\sigma^2 + 10^{-4}}\right\}. \tag{12}$$

The bounded weight $w(\sigma)$ reduces alpha supervision at high noise levels, where the one-step clean-latent estimate is less reliable.

At inference, the backbone produces the final clean latent z_0, which is passed directly to the augmented VAE decoder:

$$\widehat{I}_c = D(z_0), \qquad \widehat{a} = f(p(H_\phi(z_0; D))). \tag{13}$$

4 EXPERIMENTS

We evaluate whether pretrained RGB generators can acquire RGBA capability through generative training alone. All ALPHAANYTHING models keep the RGB VAE encoder and decoder frozen and jointly train the transformer LoRA and alpha head with flow matching and EA Loss, without any RGBA reconstruction training. For text-to-RGBA, we construct and curate 8,963 RGBA training samples using an image generation model and evaluate ALPHAANYTHING on FLUX.1-dev (Black Forest Labs, 2024), FLUX.2 Klein Base 9B (Black Forest Labs, 2025), Qwen-Image (Wu et al., 2025), and LongCat-Image (Meituan LongCat Team et al., 2025). For image-to-RGBA, we use PrismLayersPro (Chen et al., 2025) with corrected prompt annotations, and evaluate on FLUX.2 Klein Base 9B (Black Forest Labs, 2025) and Qwen-Image-Edit (Wu et al., 2025).

All ALPHAANYTHING variants are trained for 20K optimizer updates with an effective global batch size of 16, a learning rate of 10^{-4} for both the DiT LoRA and alpha branch, and DiT LoRA rank



Figure 5: RGB reconstruction with pretrained RGB VAEs and their RGBA counterparts. Original RGB VAEs preserve small lettering and colors more faithfully.

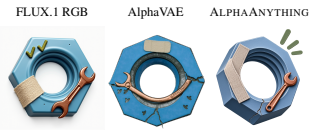

Figure 6: RGB and RGBA generation with FLUX.1. ALPHAANYTHING retains the RGB backbone's fine material details and volumetric shape, while AlphaVAE produces a flatter shape.



Decoder	RGB L1 ↓	Alpha Grad ↓	Alpha Conn ↓
ALPHAANYTHING FLUX.2	**0.1155**	**69.5947**	**46.0840**
w/ Coupled VAE	0.1349	70.1856	59.9648
ALPHAANYTHING Qwen	**0.0746**	**27.5221**	30.0584
w/ Coupled VAE	0.2435	27.8330	**28.1564**

Figure 7: RGB detail preservation (left) and decoder ablation (right). ALPHAANYTHING retains clearer letter shapes than OmniAlpha and the coupled decoder.

64. Trainable parameter counts for each backbone are reported in Appendix B.6; detailed training configurations, evaluation protocols, and metric definitions are provided in Appendix B.

4.1 MAIN RESULTS

We compare ALPHAANYTHING with existing text-to-RGBA and image-to-RGBA methods. We evaluate on two test sets constructed for this study: **RGBA-Text**, containing 294 text prompts, and **RGBA-Image**, containing 735 image-conditioned examples. We also evaluate text-to-RGBA generation on the 402 prompts in AlphaTest (Wang et al., 2025b). Bold and underlined table entries mark the best and second-best values.

Text-to-RGBA generation. We evaluate text-to-RGBA generation on RGBA-Text and AlphaTest using CLIPScore (Hessel et al., 2021), Aesthetic Score (Schuhmann et al., 2022), and ImageReward (Xu et al., 2023), averaging results on black and white composites. On AlphaTest, we additionally report FID (Heusel et al., 2017) against the reference images. We compare four ALPHAANYTHING backbones with AlphaVAE (based on FLUX.1-dev), LayerDiffuse, OmniAlpha (based on Qwen-Image-Edit), and Qwen-Image-2.1 (Qwen Team, 2026). As shown in Tables 1 and 2, ALPHAANYTHING achieves improvements in visual quality and text–image alignment metrics. Additional results on AIM-500 (Li et al., 2021) are reported in Appendix C.1. For qualitative comparisons in Figure 3, ALPHAANYTHING shows higher visual quality. Further matched examples are provided in Appendix D.

Image-to-RGBA generation. Table 3 compares the two ALPHAANYTHING backbones with OmniAlpha (Yu et al., 2026) and Qwen-Image-2.1 on RGBA-Image, using RGB L1, Soft IoU, SAD, LPIPS (Zhang et al., 2018), Alpha Grad, and Alpha Conn (Rhemann et al., 2009). Figure 4 compares the predicted RGBA outputs and enlarged alpha details.

4.2 PRESERVING RGB DETAILS

On the 295 source images in RGBA-Image, we find that RGBA-trained VAEs exhibit lower RGB reconstruction fidelity than their corresponding pretrained backbone VAEs. Compared with the original Qwen-Image-Edit RGB VAE, OmniAlpha's RGBA VAE exhibits visible deviations in fine details, as illustrated in Figure 5. Quantitatively, LPIPS increases from 0.03807 to 0.05197 and ΔE_{00}

from 1.917 to 2.078. Figure 7 (left) further shows that ALPHAANYTHING better preserves RGB details during RGBA generation.

We further ablate cross-branch feature sharing using a coupled decoder with additional alpha-to-RGB feedback while freezing all pretrained RGB parameters. As shown in Figure 7 (right), foreground RGB L1 increases from 0.1155 to 0.1349 on FLUX.2 and from 0.0746 to 0.2435 on Qwen-Image-Edit. These results show that alpha-to-RGB feedback could degrade foreground RGB fidelity, even when the pretrained RGB parameters remain frozen. Our unidirectional RGB-to-alpha design preserves the RGB decoding result for any fixed latent (Proposition 1).

Figure 6 complements these decoder comparisons with a qualitative generation example using FLUX.1-dev. The original RGB model and ALPHAANYTHING both render a volumetric nut with clear fabric texture and metallic highlights, while AlphaVAE produces a flatter shape. This example illustrates that ALPHAANYTHING can retain fine material and structural details exhibited by the original RGB backbone during RGBA generation.

4.3 EFFECT OF PROPAGATING ALPHA SUPERVISION TO THE DIT

We ablate gradient propagation from the alpha loss to the DiT by detaching $\widehat{z}_0$ before alpha decoding while keeping all other training settings fixed. As shown in Table 4, allowing alpha-loss gradients to update the DiT leads to more accurate opacity prediction, better alpha boundary and connectivity structure, and improved composite fidelity on both backbones, reflected by higher Soft IoU and lower SAD, LPIPS, Alpha Grad, and Alpha Conn. These results support our central design that opacity supervision should train the generator, rather than being confined to the decoder-side alpha branch.

4.4 EFFECTIVENESS OF THE ENDPOINT-AWARE ALPHA LOSS

On FLUX.2 and Qwen-Image-Edit, we compare the complete EA Loss with direct alpha L_1 regression, retaining the same alpha-head architecture and the remaining optimization settings. As shown in Table 5, EA Loss consistently achieves higher Soft IoU and lower SAD, LPIPS, Alpha Grad, and Alpha Conn on both backbones, with larger gains on Qwen-Image-Edit, demonstrating more effective alpha supervision.

Table 4: Ablation on alpha supervision of DiT.

	Soft IoU↑	SAD↓	LPIPS↓	Alpha Grad↓	Alpha Conn↓
ALPHAANYTHING Flux.2	**0.8296**	**46.3572**	**0.0207**	**69.5947**	**46.0840**
Stop-grad	0.8146	59.8725	0.0248	72.3753	59.5553
ALPHAANYTHING Qwen	**0.9223**	**30.3820**	**0.0136**	**27.5221**	**30.0584**
Stop-grad	0.9014	37.0486	0.0158	35.8853	36.5627

Table 5: Ablation on EA loss and Alpha L1 loss.

	Soft IoU↑	SAD↓	LPIPS↓	Alpha Grad↓	Alpha Conn↓
ALPHAANYTHING Flux.2 (EA)	**0.8296**	**46.3572**	**0.0207**	**69.5947**	**46.0840**
ALPHAANYTHING Flux (L_1)	0.8179	53.2542	0.0226	69.9201	52.3830
ALPHAANYTHING Qwen (EA)	**0.9223**	**30.3820**	**0.0136**	**27.5221**	**30.0584**
ALPHAANYTHING Qwen (L_1)	0.8830	49.2356	0.0222	38.0080	48.2089

5 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

ALPHAANYTHING extends pretrained RGB generators to RGBA through generative training alone, without RGBA reconstruction pretraining or a separate alpha latent. Alpha supervision guides RGBA generation in the original RGB latent space, while unidirectional RGB-to-alpha feature sharing preserves the original RGB decoding process for any fixed latent. EA Loss jointly handles exact opacity endpoints, soft transparency, and localized errors within a single objective, outperforming standard alpha L_1 loss in our experiments. Experiments across text-to-RGBA and image-to-RGBA tasks and multiple pretrained backbones show competitive RGBA performance, with improvements in aesthetic quality and text-image alignment.

High-quality RGBA data, widely accepted evaluation benchmarks and metrics are still scarce, which limits how broadly we can evaluate the method. Broader datasets and more comprehensive metrics that jointly assess appearance, opacity, and compositing quality are needed to further characterize data efficiency, generation quality, and generalization across RGBA generation tasks.

AI USE STATEMENT

Generative AI tools were used to create synthetic training and evaluation data, construct reference RGBA layers, select semantic targets, and draft or revise descriptions and extraction instructions. AI tools also assisted with method implementation, manuscript organization, and language editing. Human researchers performed spot checks of AI-assisted data and annotations and reviewed all AI-assisted implementation code and manuscript revisions in full. The dataset review procedures are described in Appendix B.3. We take responsibility for the final text, code, data, claims, and results.

ETHICS STATEMENT

ALPHAANYTHING is intended to support the creation and editing of reusable visual assets. Potential misuse includes deceptive compositing and unauthorized manipulation of visual content. Responsible use requires respect for privacy, intellectual-property rights, and the usage conditions of source data and pretrained models. Our constructed datasets include synthetic images and model-generated reference layers; their provenance and manual review procedures are described in Appendix B.3. Human review checks opacity, color fidelity, and agreement with instructions, but cannot eliminate biases inherited from source assets and generative models. The dataset release will document synthetic content, source provenance, and applicable usage conditions.

REPRODUCIBILITY STATEMENT

The model and training objective are specified in Section 3, with theoretical assumptions and derivations in Appendix A. Appendix B documents the architecture and optimization settings, data preparation, evaluation metrics, inference configurations, and controlled ablations. Appendices C and D provide supplementary numerical results, the inference-seed and bootstrap protocols, and matched visual comparisons. We will release our code, the constructed datasets, annotations, and data splits used in our experiments to facilitate independent training and evaluation of ALPHAANYTHING.

REFERENCES

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020. URL http://jmlr.org/papers/v21/19-021.html.

Junwen Chen, Heyang Jiang, Yanbin Wang, Keming Wu, Ji Li, Chao Zhang, Keiji Yanai, Dong Chen, and Yuhui Yuan. Prismlayers: Open data for high-quality multi-layer transparent image generative models, 2025. URL https://arxiv.org/abs/2505.22523.

Alessandro Fontanella, Petru-Daniel Tudosiu, Yongxin Yang, Shifeng Zhang, and Sarah Parisot. Generating compositional scenes via text-to-image rgba instance generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 43864–43893. Curran Associates, Inc., 2024. doi: 10.52202/079017-1392. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4d90363e96781894b1327b87d1ade17e-Paper-Conference.pdf.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.595. URL https://aclanthology.org/2021.emnlp-main.595/.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5148–5157, October 2021.

Jizhizi Li, Jing Zhang, and Dacheng Tao. Deep automatic natural image matting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, pp. 800–806, 2021. URL https://www.ijcai.org/proceedings/2021/111.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, Xunliang Cai, Yayong Guan, and Jie Hu. Longcat-image technical report, 2025. URL https://arxiv.org/abs/2512.07584.

Henrique K. Miyamoto, Fábio C. C. Meneghetti, Julianna Pinele, and Sueli I. R. Costa. On closed-form expressions for the fisher–rao distance. *Information Geometry*, 7(2):311–354, Nov 2024. ISSN 2511-249X. doi: 10.1007/s41884-024-00143-2. URL https://doi.org/10.1007/s41884-024-00143-2.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/081b08068e4733ae3e7ad019fe8d172f-Abstract-Conference.html.

Qwen Team. Qwen-Image-2.1, 2026. URL https://github.com/QwenLM/Qwen-Image-2.1. Released September 20, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1826–1833, 2009. doi: 10.1109/CVPR.2009.5206503. URL https://alphamatting.com/.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 25278–25294. Curran Associates, Inc., 2022. doi: 10.52202/068431-1833. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a1859debfb3b59d094f3504d5ebb6c25-Paper-Datasets_and_Benchmarks.pdf.

Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research and Application*, 30(1):21–30, 2005. doi: 10.1002/col.20070. URL https://hajim.rochester.edu/ece/sites/gsharma/ciede2000/ciede2000noteCRNA.pdf.

Haoran Wang, Bo Zhao, Jinghui Wang, Hanzhang Wang, Huan Yang, Wei Ji, Hao Liu, and Xinyan Xiao. Sega: A stepwise evolution paradigm for content-aware layout generation with design prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19321–19330, October 2025a.

Zile Wang, Hao Yu, Jiabo Zhan, and Chun Yuan. Alphavae: Unified end-to-end rgba image reconstruction and generation with alpha-aware representation learning, 2025b. URL https://arxiv.org/abs/2507.09308.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Keming Wu, Zuhao Yang, Kaichen Zhang, Shizun Wang, Haowei Zhu, Sicong Leng, Zhongyu Yang, Qijie Wang, Sudong Wang, Ziting Wang, Zili Wang, Hui Zhang, Haonan Wang, Hang Zhou, Yifan Pu, Xingxuan Li, Fangneng Zhan, Bo Li, Lidong Bing, Yuxin Song, Ziwei Liu, Wenhu Chen, Jingdong Wang, Xinchao Wang, Xiaojuan Qi, Shijian Lu, and Bin Wang. Visual generation in the new era: An evolution from atomic mapping to agentic world modeling, 2026. URL https://arxiv.org/abs/2604.28185.

Kangyang Xie, Binbin Yang, Hao Chen, Meng Wang, Cheng Zou, Hui Xue, Ming Yang, and Chunhua Shen. Zippo: Zipping color and transparency distributions into a single diffusion model, 2024. URL https://arxiv.org/abs/2403.11077.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 15903–15935. Curran Associates, Inc., 2023. doi: 10.52202/075280-0700. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/33646ef0ed554145eab65f6250fab0c9-Paper-Conference.pdf.

Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, Lionel M. Ni, Junyang Lin, and Chenfei Wu. Qwen-image-layered: Towards inherent editability via layer decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16196–16205, June 2026.

Hao Yu, Jinglin Wang, Jiabo Zhan, Rui Chen, Zile Wang, Huaisong Zhang, Hongyu Li, Xinrui Chen, Yongxian Wei, and Chun Yuan. Omnialpha: Aligning transparency-aware generation via multi-task unified reinforcement learning, 2026. URL https://arxiv.org/abs/2511.20211.

Lvmin Zhang and Maneesh Agrawala. Transparent image layer diffusion using latent transparency. *ACM Transactions on Graphics*, 43(4):1–15, July 2024. ISSN 1557-7368. doi: 10.1145/3658150. URL http://dx.doi.org/10.1145/3658150.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL https://arxiv.org/abs/1801.03924.

A Theoretical Analysis

This appendix establishes the RGB preservation, gradient transmission, and opacity-supervision properties of the method in Section 3. We use $a = I_\alpha$, $u = \chi(a)$, $r = p(s) - u$, and $N = HW$. All square roots denote their nonnegative real values.

A.1 RGB Preservation and Gradient Flow

Proof of Proposition 1. For a fixed latent z, the augmented and original RGB decoders start from the same feature h_0. If their RGB states agree at stage ℓ, both apply the same frozen function d_ℓ in (5), without reading the alpha state or its parameters ϕ. Induction gives identical RGB outputs at every stage. The final RGB output is therefore $D(z)$ for every ϕ, and its derivative with respect to ϕ is zero wherever the derivative exists. □

The joint feature update has the block lower-triangular Jacobian

$$\frac{\partial(h', q')}{\partial(h, q)} = \begin{pmatrix} J_{hh} & 0 \\ J_{qh} & J_{qq} \end{pmatrix}. \tag{14}$$

Products of these Jacobians retain the zero upper-right block. The lower-left blocks allow opacity supervision to reach the shared decoder input while preserving the RGB feature updates.

For the actual training path in (11), the one-step estimate is $\widehat{z}_0 = z_\sigma - \sigma\widehat{v}$, followed by $\overline{z}_0 = \mathrm{clip}(\widehat{z}_0, -20, 20)$. Let $K = \partial\overline{z}_0/\partial\widehat{z}_0$ be the diagonal clipping Jacobian and $J_s = \partial H_\phi(\overline{z}_0; D)/\partial\overline{z}_0$. Holding z_σ, σ, and conditioning fixed, the chain rule gives

$$\nabla_{\widehat{v}}\big[w(\sigma)\ell_{\mathrm{E}\alpha}\big] = -\sigma w(\sigma) K^\top J_s^\top \nabla_s \ell_{\mathrm{E}\alpha}. \tag{15}$$

Unclipped latent coordinates have unit derivative through K, while clipped coordinates have zero derivative away from the clipping boundaries. The frozen decoder retains its derivatives with respect to its input, so the gradient in (15) propagates through the predicted velocity to the transformer LoRA parameters. This is the gradient path retained by the full model and removed by the stop-gradient ablation.

The sampled latent can change as LoRA is trained; Proposition 1 preserves the original RGB decoding function at each fixed latent.

A.2 Fisher–Rao Geometry of Opacity

For $0 < a < 1$, the Bernoulli pair $(1 - a, a)$ has log-likelihood score $(Y - a)/[a(1 - a)]$. Its second moment, using $\mathrm{Var}(Y) = a(1 - a)$, gives

$$\mathcal{I}(a) = \frac{1}{a(1-a)}, \qquad \mathrm{d}\ell^2 = \frac{\mathrm{d}a^2}{a(1-a)}. \tag{16}$$

Integrating this line element and normalizing by the total arc length gives the coordinate used in (7):

$$\chi(a) = \frac{1}{\pi}\int_0^a \frac{\mathrm{d}t}{\sqrt{t(1-t)}} = \frac{2}{\pi}\arcsin\sqrt{a}. \tag{17}$$

For any differentiable interior path, its length is $\pi \int |\mathrm{d}\chi(a(t))/\mathrm{d}t|\,\mathrm{d}t$. A monotone path attains the minimum, so

$$d_{\mathrm{FR}}(a, b) = \pi|\chi(a) - \chi(b)|. \tag{18}$$

Endpoint limits extend this distance continuously to $[0, 1]$, with $d_{\mathrm{FR}}(0, 1) = \pi$. The inverse $f(u) = \sin^2(\pi u/2)$ is a continuous bijection on $[0, 1]$.

For an interior target, $\chi'(a) = 1/[\pi\sqrt{a(1-a)}]$. At zero, expanding $\arcsin x = x + O(x^3)$ gives

$$[\chi(a + \delta) - \chi(a)]^2 = \frac{\delta^2}{\pi^2 a(1-a)} + o(\delta^2), \qquad 0 < a < 1,$$
$$\chi(\delta)^2 = \frac{4}{\pi^2}\delta + O(\delta^2), \qquad \delta \downarrow 0. \tag{19}$$

The identity $\chi(1-a) = 1 - \chi(a)$ gives the corresponding expansion near opacity one. Thus the squared coordinate distance has first-order cost for small deviations from either opacity endpoint and remains locally quadratic at interior targets, as used in the endpoint-sensitive supervision of Section 3.3.

A.3 Exact Endpoints and Corrective Gradients

For $\Omega(u) = u^2 - u + \iota_{[0,1]}(u)$, the convex conjugate is

$$\Omega^*(s) = \sup_{v\in[0,1]} \{sv - v^2 + v\} = \begin{cases} 0, & s \le -1, \\ (s+1)^2/4, & -1 < s < 1, \\ s, & s \ge 1. \end{cases} \tag{20}$$

The unconstrained maximizer is $(s+1)/2$; restricting it to $[0,1]$ gives the map $p(s)$ defined in Section 3.3. The conjugate values and first derivatives agree at both joins, and $(\Omega^*)'(s) = p(s)$. Substitution into the Fenchel–Young gap

$$g(s,u) = \Omega^*(s) + u^2 - u - su \tag{21}$$

recovers exactly the three branches used in (8).

Proposition 2 (Endpoint-compatible coordinate supervision)**.** *For $u \in [0,1]$ and finite s,*

$$g(s,u) \ge [p(s) - u]^2 \ge 0, \qquad g(s,u) = 0 \iff p(s) = u, \tag{22}$$

and $\partial g(s,u)/\partial s = p(s) - u$.

Proof. For $s \in [-1,1]$, the gap is exactly $(p-u)^2$. Outside this interval,

$$g(s,u) - (p-u)^2 = \begin{cases} u(-1-s), & s \le -1, \\ (1-u)(s-1), & s \ge 1, \end{cases} \tag{23}$$

which is nonnegative. Differentiating the three branches gives $-u$, $(s+1)/2 - u$, and $1-u$; these derivatives agree at the joins and equal $p-u$. The lower bound makes $p = u$ necessary for zero gap, and substitution into the corresponding branch proves sufficiency. □

A transparent target has zero gap at every $s \le -1$, and an opaque target has zero gap at every $s \ge 1$. Every soft target has an exact fit at $s = 2u - 1$. At an incorrect transparent prediction the score derivative is $-u < 0$, and at an incorrect opaque prediction it is $1 - u > 0$. The loss therefore supports finite-score exact endpoints while retaining a corrective derivative under incorrect saturation.

The implementation evaluates the nonnegative branches in (8) directly and computes targets and reductions in FP32. The equivalent expression $\chi(a) = (2/\pi)\,\mathrm{atan2}(\sqrt{a}, \sqrt{1-a})$ evaluates the target coordinate at both endpoints.

A.4 Root Aggregation and Error Bounds

Proposition 3 (Root aggregation and corrective gradients)**.** *For the actual roots in* (9)*, let $M = N^{-1}\sum_i g(s_i, u_i)$. Then*

$$\sqrt{M} \le \ell_{\mathrm{E}\alpha} \le \sqrt{2M}, \qquad \ell_{\mathrm{E}\alpha} = 0 \iff \widehat{a} = a. \tag{24}$$

Where both roots are positive,

$$\frac{\partial \ell_{\mathrm{E}\alpha}}{\partial s_i} = \frac{r_i}{2N}\left(\frac{1-a_i}{R_0} + \frac{a_i}{R_1}\right). \tag{25}$$

Selecting the zero generalized gradient for an exactly zero root gives

$$r_i \frac{\partial \ell_{\mathrm{E}\alpha}}{\partial s_i} \ge 0. \tag{26}$$

Proof. The identity $R_0^2 + R_1^2 = M$ gives $(R_0 + R_1)^2 = M + 2R_0R_1 \in [M, 2M]$. A zero sum of the roots implies every gap is zero, since their per-pixel weights sum to one. Proposition 2 and the bijection f then give $\widehat{a} = a$; the converse follows by substitution.

Write $b_{0i} = 1 - a_i$ and $b_{1i} = a_i$. For $R_k > 0$, differentiation gives

$$\frac{\partial R_k}{\partial s_i} = \frac{b_{ki} r_i}{2N R_k}. \tag{27}$$

Summing the two root derivatives proves (25). The weights are nonnegative, so multiplication by r_i establishes (26).

At a zero root, all positively weighted gaps and residuals vanish. Its local expression is a norm of affine or one-sided endpoint residuals, or is identically zero. It is locally Lipschitz, and zero is an admissible generalized gradient at its minimum. The selected zero contribution preserves the corrective sign without evaluating $0/0$. □

Contribution of localized errors. For a binary target, suppose all nonzero gaps lie in a set S within one endpoint group $k \in \{0, 1\}$, so $a_i = k$ for $i \in S$. Let $m = |S| > 0$, $\rho = m/N$, and $\bar{g} = m^{-1} \sum_{i \in S} g_i$. Then

$$R_k^2 = \frac{1}{N} \sum_{i \in S} g_i = \rho \bar{g}, \qquad R_{1-k} = 0, \qquad \ell_{\mathrm{E}\alpha} = \sqrt{\rho \bar{g}}, \qquad M = \rho \bar{g}. \tag{28}$$

For fixed $\bar{g}$, the root contribution therefore decreases as $\sqrt{\rho}$, while direct averaging of the gaps decreases as ρ. This identity applies to both saturated and unsaturated scores and establishes the area dependence described in Section 3.3.

Opacity and compositing error. The inverse coordinate satisfies $|f'(u)| \le \pi/2$, giving $|\widehat{a}_i - a_i| \le (\pi/2)|r_i|$. Consequently,

$$\frac{1}{N} \sum_i (\widehat{a}_i - a_i)^2 \le \frac{\pi^2}{4N} \sum_i r_i^2 \le \frac{\pi^2}{4} M \le \frac{\pi^2}{4} \ell_{\mathrm{E}\alpha}^2. \tag{29}$$

Let $F = I_c$ and assume $F, \widehat{F}, B \in [0, 1]^{N \times 3}$ in the common linear color space of (1). The exact decomposition

$$C_B(\widehat{F}, \widehat{a}) - C_B(F, a) = a \odot (\widehat{F} - F) + (\widehat{a} - a) \odot (\widehat{F} - B) \tag{30}$$

and $|\widehat{F} - B| \le 1$ channelwise imply

$$\frac{\|C_B(\widehat{F}, \widehat{a}) - C_B(F, a)\|_2}{\sqrt{3N}} \le \sqrt{\frac{\sum_i a_i^2 \|\widehat{F}_i - F_i\|_2^2}{3N}} + \frac{\pi}{2} \ell_{\mathrm{E}\alpha}. \tag{31}$$

This bound separates foreground RGB error from the opacity contribution controlled by EA Loss and holds for every background in the stated range.

B Experimental Protocol

This appendix specifies the implementation, training, evaluation, and ablation settings used in Section 4. All AlphaAnything variants start from a pretrained RGB generator, keep the RGB VAE encoder and decoder frozen, and use the objective in (12), with the stated substitution for the alpha-L_1 ablation. Supplemental numerical results and visual comparisons are reported in Appendices C and D.

B.1 Implementation Details

RGB targets and native backbone interfaces. The color-padding operation in Section 3.1 follows the multiscale convention of LayerDiffuse (Zhang & Agrawala, 2024). We form a pyramid of premultiplied color $I_p = aI_c$ and alpha, initialize the coarsest color from an alpha-weighted mean,

and propagate color from coarse to fine. Each scale adds its premultiplied color to the upsampled coarse color multiplied by remaining transparency. The procedure supplies a dense three-channel training target without relying on arbitrary stored RGB values in fully transparent regions. The frozen encoder uses its posterior mode for FLUX and Qwen, while LongCat follows its native training convention of posterior sampling. Each backbone retains its native convolution dimensionality, feature widths, latent scaling, and token packing. Conditioning tokens remain clean when required by its image-conditioning interface.

Spatial feature injections. At decoder stage ℓ with m_ℓ RGB feature injections, feature $h_{\ell j}$ updates the aligned alpha state by

$$q \leftarrow q + \frac{\gamma_\ell}{\sqrt{m_\ell}} \frac{\eta_\ell}{r_\ell} B_{\ell j}^{3\times 3} A_{\ell j}^{1\times 1} h_{\ell j}. \tag{32}$$

The learned projection $A_{\ell j}$ reduces RGB channels to rank r_ℓ, and $B_{\ell j}$ performs spatial mixing into the alpha width. The factor η_ℓ / r_ℓ is the low-rank projection scaling; $1/\sqrt{m_\ell}$ normalizes the accumulated input by injection count; and the learned stage gate γ_ℓ controls its contribution. Channel transitions, residual blocks, and learned upsampling align the alpha state with each RGB stage.

Alpha-branch configuration. The five alpha stages use widths $(256, 256, 192, 128, 96)$ from the bottleneck toward the RGB output, with two residual blocks per stage. All RGB-to-alpha projections use rank $r_\ell = 64$ and scaling parameter $\eta_\ell = 64$. The learned stage gates are initialized to $(1, 0.75, 0.5, 0.25, 0.25)$, and injection-count normalization is enabled. Bottleneck cross-attention uses eight heads, each with 32 channels. The final spatial score head has 128 hidden channels. Dropout is zero throughout the alpha branch.

Initialization. Alpha residual outputs and the bottleneck cross-attention output projection are zero initialized. Spatial injection output projections and the final score convolution use random initialization with standard deviation 10^{-4}. This starts residual refinement from the identity while providing trainable feature and score paths.

B.2 Training and Inference Settings

Text-to-RGBA evaluation uses RGBA-Text, AIM-500 (Li et al., 2021), and AlphaTest from AlphaVAE (Wang et al., 2025b). For AIM-500, we use the text prompts released with AlphaVAE. Image-to-RGBA evaluation uses RGBA-Image. Prompt lists, source groups, aspect-ratio policy, conditioning protocol, and filtering rules are fixed within each backbone.

The text-to-RGBA comparison uses FLUX.1-dev, FLUX.2 Klein Base 9B, Qwen-Image-2512, and LongCat-Image; the image-to-RGBA comparison uses FLUX.2 Klein Base 9B and Qwen-Image-Edit-2511. Each backbone keeps its native conditioning interface, latent layout, encoding, scaling, token packing, and sampler.

Training data and preprocessing. Text-to-RGBA training uses 8,963 curated RGBA assets generated using an image generation model. Image-to-RGBA training uses $\sim$ 180K source-image-target-layer pairs from PrismLayersPro (Chen et al., 2025), with corrected English extraction instructions. Images are processed at a nominal resolution of 1024 using aspect-ratio buckets; the training examples in these runs occupy the 1024×1024 bucket. Center cropping and random horizontal flipping are disabled. Image-conditioned source and target pairs receive aligned spatial preprocessing. The RGB target uses color padding, while the original target alpha supplies opacity supervision.

Optimization. All models are trained for 20,000 optimizer updates with a per-device batch size of one and one gradient-accumulation step. The main experiments use 16 GPUs, giving an effective global batch size of 16. We use AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-4} for both the transformer LoRA and alpha branch, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0.01, and $\epsilon = 10^{-8}$. The learning rate follows cosine decay after 100 warmup updates, and the global gradient norm is clipped at 1.0. Training uses bfloat16 mixed precision and transformer activation checkpointing. The original transformer, RGB VAE, and text-encoder parameters remain frozen. The flow and alpha terms both have coefficient 1. Alpha supervision uses the time weight in (12), with reference noise level 0.5 and denominator stabilizer 10^{-4}; the one-step clean-latent estimate is clipped to $[-20, 20]$ before decoding.

Transformer LoRA modules. All transformer LoRA modules use rank 64, scaling parameter 128, Gaussian initialization, and zero dropout. Qwen and LongCat apply LoRA to the attention query, key, value, and output projections. FLUX.1 additionally applies LoRA to the context-stream attention projections and the input and output feed-forward projections in both image and context streams. FLUX.2 applies LoRA to the attention projections together with the fused query/key/value–MLP input projection and output projection of its single-stream blocks.

Conditioning and noise sampling. For text-to-RGBA, explicit and implicit caption variants are selected with equal probability; within either form, the short caption is selected with probability 0.2. Qwen, FLUX.2, and LongCat drop text conditioning with probability 0.1; image-conditioned training retains the reference image when its text is dropped. FLUX.1 uses its native guidance embedding with training guidance 1.0. Qwen and FLUX.2 uniformly sample valid entries of a 1,000-step flow schedule with shift 3, restricting scheduler timesteps to $[1, 999]$. FLUX.1 samples from its native shifted schedule over the same timestep range. LongCat samples $\sigma = \mathrm{sigmoid}(\xi)$ with $\xi \sim \mathcal{N}(0, 1)$ and then applies the backbone's resolution-dependent time shift. All backbones use the linear flow path and velocity target in (10). Controlled comparisons keep the sampler, guidance settings, inference seed, conditioning inputs, and sampling-step count fixed.

Inference settings. The text-to-RGBA comparisons use the corresponding 20,000-step ALPHAANYTHING checkpoints. FLUX.1 uses 28 sampling steps and guidance 3.5; FLUX.2, Qwen, and LongCat use 50 steps and guidance 4. The baselines use their recorded inference settings: 25 steps and guidance 7 for LayerDiffuse with SDXL (Podell et al., 2024), 28 steps and guidance 3.5 for AlphaVAE, 50 steps and guidance 4 for OmniAlpha, and 40 steps and guidance 1.0 for Qwen-Image-2.1. For Qwen-Image-2.1 only, we additionally wrap the shared content prompt in its official RGBA transparency template; using the content prompt alone produced largely opaque outputs ($\alpha \approx 1$). The main comparisons use inference base seed 42; Appendix C.4 additionally evaluates the fixed checkpoints with base seeds 43 and 44.

B.3 CONSTRUCTION OF RGBA-TEXT AND RGBA-IMAGE

We construct RGBA-Text and RGBA-Image for evaluation only. Neither set is used to train the ALPHAANYTHING models.

RGBA-Text. RGBA-Text contains 294 English prompts covering nine categories, with the distribution in Table 6. We follow the training set's category taxonomy while composing new combinations of subjects, materials, colors, poses, and visual styles. The set spans lettering and graphic assets, natural subjects with detailed boundaries, visual effects, and transparent or translucent materials. Prompts are selected for content coverage, clarity, and distinctiveness, without imposing equal category sizes or fixed description lengths. They range from 10 to 95 words, with a median of 53 words; prompts involving visible lettering use English text.

We check both the main prompts and their short summaries against all nonempty training caption variants. The curation combines normalized exact-text matching, shared 12-word sequence checks, and manual review of TF–IDF nearest neighbors. Descriptions that too closely resemble training captions are revised. The final set contains no normalized exact duplicates within the test prompts or against the training text, and no shared contiguous 12-word sequences with training captions. RGBA-Text contains prompts without paired reference images, so we report reference-free scores on this set.

RGBA-Image source images. RGBA-Image contains 735 image-conditioned examples from 295 source images. Each example includes a source image, a target RGBA layer, an English layer description, and an extraction instruction. The source images come from two pools: selected GenPoster100K posters and images synthesized using an image generation and editing model. Table 7 reports the final distribution by source and input-image category; multiple target layers can share one source image.

For the GenPoster100K pool (Wang et al., 2025a), we first filter the released metadata for adequate resolution, valid layouts, and usable text. Candidates must have a short side of at least 600 pixels, at least 0.7 megapixels, an aspect ratio between 0.45 and 2.20, and 2–24 annotated layout elements.

Table 6: Category distribution of RGBA-Text.

Category	Content	Prompts
Advertising assets	Lettering, titles, labels, buttons, and promotional decorations	33
Animals	Animal subjects	33
Visual effects	Light, electricity, smoke, particles, and liquids	32
Graphics	Decorative shapes, print motifs, and craft textures	27
Objects	Utensils, tools, and crafted objects	27
People	People, actions, and interactions	40
Plants	Flowers, foliage, fruit, and seeds	29
Products	Product and device presentations	32
Translucent materials	Transparent or translucent materials, including combinations with opaque parts	41
Total		**294**

We reject candidates with malformed layouts, CJK text, or template placeholders, and retain at most one candidate per original design file. Remaining candidates are ranked using layout metadata, CLIP-based scores (Radford et al., 2021), LAION aesthetic scores, MUSIQ (Ke et al., 2021), and image-quality statistics, followed by visual near-duplicate filtering and OCR screening. Full-resolution OCR removes remaining CJK text and placeholder copy, leaving 143 posters for layer extraction.

For the synthetic-image pool, we generate 200 source images with the image generation and editing model: 100 stylized design scenes and 100 photorealistic scenes. Scene prompts vary subject combinations, human presence, spatial layout, camera viewpoint, palette, lighting, and rendering style. The final retained inputs comprise 99 stylized scenes and 86 photorealistic scenes, contributing 261 and 217 target layers, respectively.

Layer construction, annotation, and manual review. We use the same image generation and editing model to construct RGBA reference layers for both source pools. Posters are decomposed into at most six layers. For the generated scenes, a vision-language model first identifies one to three visually unambiguous semantic targets per image, and the image model extracts these targets jointly as separate layers. Reference layers are placed on the source image's canvas, preserving its dimensions and the targets' spatial alignment. These are generated reference layers that undergo manual filtering.

We use the vision-language model to annotate or re-annotate English descriptions and extraction instructions, followed by manual review against the source images and target RGBA layers. Descriptions capture the target's visible appearance and spatial relationships; instructions identify the same target and its intended semantic grouping. Wording and length vary with target complexity. Review corrects ambiguous targets, mismatched descriptions, and incomplete grouping instructions, including text panels, frames, and attached objects when they belong to the accepted layer. Each candidate layer is inspected alongside its source image and alpha matte for alpha accuracy, alignment with the description and extraction instruction, and color fidelity. Layers with inaccurate opacity, prompt mismatches, or visible color deviations are rejected. The final evaluation set contains 257 poster layers from 110 posters and 478 generated-scene layers from 185 images, for a total of 735 layers from 295 source images. We use these manually screened reference layers for evaluation, although generative extraction does not guarantee exact pixelwise reconstruction of the source content.

Table 7: Sources and input categories of RGBA-Image.

Source	Input category	Images	Examples
GenPoster100K	Graphic-design posters	110	257
Synthetic images	Stylized design scenes	99	261
Synthetic images	Photorealistic scenes	86	217
Total		**295**	**735**

B.4 Evaluation Protocols and Metrics

B.4.1 Text-to-RGBA Evaluation

We compare against AlphaVAE (Wang et al., 2025b), LayerDiffuse (Zhang & Agrawala, 2024), OmniAlpha (Yu et al., 2026), and Qwen-Image-2.1. For AIM-500 (Li et al., 2021) and AlphaTest, we use the benchmark prompts provided by AlphaVAE. Evaluation uses FID (Heusel et al., 2017), CLIPScore (Hessel et al., 2021), Aesthetic Score (Schuhmann et al., 2022), and ImageReward (Xu et al., 2023). The main-text results are in Tables 1 and 2; the additional AIM500 comparison is in Appendix C.1.

The four appearance metrics are computed separately on lossless black-background and white-background composites, then averaged across the two backgrounds. CLIPScore uses CLIP ViT-B/32 (Radford et al., 2021) with the caption prefix "A photo depicts" and a weight of 2.5 on the nonnegative text–image cosine similarity. Aesthetic Score uses the LAION improved aesthetic predictor[1] with CLIP ViT-L/14 and the released `sac+logos+ava1-l14-linearMSE` weights, and ImageReward uses the original prompt. Each benchmark is evaluated separately: sample-level scores are averaged uniformly, and FID is computed for each dataset and background before averaging the two values. Table 1 evaluates all 294 RGBA-Text prompts using CLIPScore, Aesthetic Score, and ImageReward. This set has no reference images for FID.

B.4.2 Image-to-RGBA Evaluation

Table 3 evaluates RGBA-Image, whose 735 examples each pair a source image and target description with a target RGBA layer. The RGBA-Image comparison includes OmniAlpha, Qwen-Image-2.1, and the two AlphaAnything backbones. Within each inference run, each method contributes one prediction per example, and all six metrics are averaged uniformly over the 735 examples.

For a fair comparison, AlphaAnything uses exactly the same prompts as OmniAlpha. Each prompt consists of the prefix "Extract the object described by the text, preserving fine details and transparency:" followed by the target description. We use these prompts for Qwen-Image-2.1 as well. Figure 10 displays the full shared prompt for each selected example.

For the six reference-based metrics, predictions are resized to the GT resolution with RGBA LANCZOS resampling, using the paired target RGBA layers in RGBA-Image. RGB L1 is the mean RGB-channel error at pixels with nonzero GT alpha. Soft IoU and SAD use normalized alpha values in $[0, 1]$ over the whole image. Soft IoU is $\sum_i \min(\widehat{a}_i, a_i) / \sum_i \max(\widehat{a}_i, a_i)$, with a value of zero for an empty union; SAD is $\sum_i |\widehat{a}_i - a_i| / 1{,}000$.

Alpha Grad and Alpha Conn follow the matting benchmark of Rhemann et al. (2009), using implementations equivalent to MMEval 0.2.1.[2] Alpha Grad measures squared differences between gradient magnitudes after independently min–max normalizing each alpha map. We use Gaussian-derivative gradients with $\sigma = 1.4$ and divide the summed squared error by $1{,}000$. Alpha Conn measures connectivity disagreement using opacity thresholds in steps of 0.1, the largest four-connected component of each prediction–reference intersection, and a 0.15 opacity-difference cutoff in the connectivity map. The summed absolute difference between the connectivity maps is divided by $1{,}000$. Both metrics use float64 arithmetic over the whole image at GT resolution.

LPIPS is computed separately on black and white composites using the stored RGB values, without intermediate image quantization, and then averaged across backgrounds. It uses the official AlexNet network with version 0.1 and bilinear antialiased resizing to a maximum image side of 512 pixels, without upscaling. Higher Soft IoU and lower RGB L1, SAD, LPIPS, Alpha Grad, and Alpha Conn indicate better agreement with the reference.

B.4.3 RGB Reconstruction Evaluation

We compare the RGBA VAEs of OmniAlpha and AlphaVAE with their original Qwen-Image-Edit-2509 and FLUX.1-dev RGB VAEs, respectively. The evaluation uses all 295 distinct source images in RGBA-Image, counting each input once regardless of how many target layers it has. Inputs are

[1] `https://github.com/christophschuhmann/improved-aesthetic-predictor`

[2] `https://github.com/open-mmlab/mmeval/tree/v0.2.1`

resized with LANCZOS to a maximum side length of 1024 pixels while preserving aspect ratios, without upscaling. All models receive the same inputs, edge-padded to multiples of 64; the padding is removed before scoring. We use deterministic float32 encode–decode with the posterior mode and supply fully opaque alpha to the RGBA encoders. Decoded RGB values are clipped to $[0, 1]$ and evaluated directly, discarding the reconstructed alpha without compositing.

We evaluate RGB reconstruction using LPIPS (Zhang et al., 2018) and CIEDE2000 color difference, ΔE_{00} (Sharma et al., 2005). LPIPS uses AlexNet v0.1 on floating-point reconstructions before PNG quantization, at the full evaluation resolution. Both metrics are computed over complete images and averaged equally across inputs. The enlarged regions in Figure 5 illustrate local details within these full-image reconstructions. Results are reported in Appendix C.2.

Computation of ΔE_{00}. We read the saved 8-bit reference and reconstruction PNGs, retain their RGB channels, and divide by 255 to obtain sRGB values in $[0, 1]$. The images retain the evaluation resolution above; no additional resizing, region selection, or alpha weighting is applied. We convert each pixel to CIELAB using `skimage.color.rgb2lab` with the D65 illuminant and the 2° standard observer, then apply `deltaE_ciede2000` in float64 with $k_L = k_C = k_H = 1$. For a reference–reconstruction color pair, CIEDE2000 is

$$d_{00} = \left[\left(\frac{\Delta L'}{k_L S_L} \right)^2 + \left(\frac{\Delta C'}{k_C S_C} \right)^2 + \left(\frac{\Delta H'}{k_H S_H} \right)^2 + R_T \frac{\Delta C'}{k_C S_C} \frac{\Delta H'}{k_H S_H} \right]^{1/2}. \tag{33}$$

Here $\Delta L'$, $\Delta C'$, and $\Delta H'$ are the lightness, adjusted chroma, and adjusted hue differences. The weighting functions S_L, S_C, S_H and rotation term R_T follow the standard CIEDE2000 definitions, including the prescribed hue-angle wraparound and zero-chroma cases (Sharma et al., 2005). The score accounts for perceptual differences in lightness, chroma, and hue; smaller values indicate closer reconstructed colors.

For image j, let Ω_j be its unpadded pixel domain, and let $\mathbf{c}_{jp}$ and $\widehat{\mathbf{c}}_{jp}$ be the reference and reconstructed CIELAB colors at pixel p. The reported dataset score is

$$\overline{\Delta E}_{00} = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{|\Omega_j|} \sum_{p \in \Omega_j} d_{00}(\mathbf{c}_{jp}, \widehat{\mathbf{c}}_{jp}), \qquad N = 295. \tag{34}$$

Thus we first average pixelwise color differences within each image, then average the image scores with equal weight. Unlike LPIPS in this experiment, ΔE_{00} is computed from the saved 8-bit RGB images.

B.4.4 RGB AND RGBA GENERATION EXAMPLE

Figure 6 compares the original FLUX.1-dev RGB model, AlphaVAE, and ALPHAANYTHING with FLUX.1 using the following shared prompt:

> A matte blue hexagonal nut has a cream fabric bandage crossing one cracked corner. A copper wrench curves around its lower edge, and two olive check marks float above the open center. Woven threads, brushed metal, and rounded painted forms create a friendly text-free repair-service emblem.

All three outputs use seed 69, 28 sampling steps, guidance scale 3.5, and 1024×1024 resolution. The original RGB image retains its generated background; RGBA outputs are displayed on white. Each complete output canvas is shown without cropping.

B.5 ABLATION SETTINGS

Shared settings on RGBA-Image. Figure 7 (right) and Table 4 compare the full FLUX.2 Klein Base 9B and Qwen-Image-Edit-2511 models with their corresponding coupled-decoder and stop-gradient variants. Table 5 compares EA Loss and alpha L_1 on both backbones using their corresponding full-model controls. Each setting is evaluated on the same 735 target layers after 20K optimizer updates. Inference uses the same input images and target descriptions, base seed 42, 50 sampling steps, guidance 4, and aspect-ratio-preserving resizing to approximately 1024^2 pixels without cropping.

Table 8: Trainable parameters of ALPHAANYTHING, in millions. Counts include only the DiT LoRA modules and alpha branch.

Backbone	Task	DiT LoRA	Alpha branch
FLUX.1-dev	T2RGBA	179.306	12.436
FLUX.2 Klein Base 9B	T2RGBA, I2RGBA	111.149	12.436
Qwen-Image-2512	T2RGBA	94.372	12.253
LongCat-Image	T2RGBA	39.322	12.436
Qwen-Image-Edit-2511	I2RGBA	94.372	12.253

Unidirectional feature sharing. The coupled RGBA decoder adds zero-initialized alpha-to-RGB feedback at the bottleneck and upsampling stages while keeping the pretrained RGB parameters frozen. We report RGB L1 over pixels with nonzero target alpha to measure the effect on foreground color fidelity, together with Alpha Grad and Alpha Conn for opacity structure.

Alpha supervision of the DiT latent state. The stop-gradient setting detaches the predicted clean latent before alpha decoding; flow matching still trains the transformer LoRA, and the alpha head continues to train. Soft IoU and SAD measure how preventing alpha supervision from reaching the DiT affects generated opacity; black/white-averaged composite LPIPS measures the corresponding perceptual fidelity. Alpha Grad and Alpha Conn measure gradient and connectivity errors.

Standard alpha L_1 loss. The alpha-L_1 setting replaces the complete EA Loss with the mean absolute error on directly regressed physical opacity:

$$\ell_{L_1} = \frac{1}{N} \sum_i |\widehat{a}_i^{\mathrm{raw}} - a_i| \,. \tag{35}$$

Here $\widehat{a}^{\mathrm{raw}}$ is the linear output of the alpha head, with its final bias initialized to 0.5. Training uses this output directly, without the Fisher-coordinate mapping, Fenchel–Young gap, endpoint-group weighting, root aggregation, sigmoid, or output clipping. Predicted alpha is clipped to $[0, 1]$ only when exporting images. The head's convolutional structure, flow-matching objective, time weight $w(\sigma)$, alpha-loss coefficient, and alpha-to-DiT gradient path are retained. We report Soft IoU and SAD for opacity accuracy, Alpha Grad and Alpha Conn for gradient and connectivity errors, and black/white-averaged composite LPIPS for perceptual fidelity under the two objectives.

B.6 EFFICIENCY ACCOUNTING

Table 8 reports the trainable DiT LoRA and alpha-branch parameters for each backbone.

The trainable parameter count is

$$P_{\mathrm{train}} = P_{\mathrm{LoRA}} + P_{\mathrm{alpha}}, \tag{36}$$

where the two terms count the transformer LoRA and alpha-branch parameters. The original transformer weights, RGB VAE, and text encoders remain frozen. At inference, RGB and alpha share the backbone's denoising trajectory and one traversal of the augmented decoder. The added decoding computation consists of the alpha branch and its RGB-to-alpha feature injections, as specified in Appendix B.1.

C ADDITIONAL QUANTITATIVE RESULTS

The metric definitions and evaluation settings for these supplementary results are given in Appendix B.

C.1 TEXT-TO-RGBA RESULTS ON AIM500

Table 9 evaluates the eight text-to-RGBA methods on the 500 AIM500 prompts released with AlphaVAE (Wang et al., 2025b). All four ALPHAANYTHING backbones obtain higher Aesthetic Score and ImageReward than the compared baselines. FLUX.1 achieves the highest Aesthetic Score, and FLUX.2 achieves the highest ImageReward. FID and CLIPScore are reported under the same black/white-composite protocol as the AlphaTest comparison.

Table 9: Text-to-RGBA generation on AIM500.

Method / backbone		FID↓	CLIPScore↑	Aesthetic↑	ImageReward↑
LayerDiffuse		76.8254	**0.8493**	5.5621	1.0830
AlphaVAE		74.7061	0.8334	5.5631	0.9565
OmniAlpha		<u>72.2913</u>	0.8426	5.3879	1.0430
Qwen-Image-2.1		**69.5794**	0.8409	5.4272	1.0541
ALPHAANYTHING	FLUX.1	80.3795	0.8189	**5.7184**	1.2475
	FLUX.2	75.9445	<u>0.8484</u>	5.6102	**1.3638**
	Qwen Image	77.8268	0.8337	5.6646	1.2680
	LongCat	77.4516	0.8428	<u>5.6834</u>	<u>1.3593</u>

C.2 RGB RECONSTRUCTION FIDELITY

Table 10 reports reconstruction errors on the distinct source images in RGBA-Image, under the protocol in Appendix B.4.3. Both RGBA VAEs have higher LPIPS and CIEDE2000 errors than their corresponding original RGB VAEs. For the Qwen-Image-Edit backbone, LPIPS increases from 0.03807 to 0.05197 and ΔE_{00} from 1.917 to 2.078; for FLUX.1-dev, the corresponding values increase from 0.01216 to 0.04762 and from 2.109 to 3.240. These measurements complement the reconstruction examples in Figure 5 and the motivation for preserving the original RGB decoding function in Proposition 1.

Table 10: RGB reconstruction fidelity on RGBA-Image.

RGB backbone	VAE	LPIPS↓	ΔE_{00}↓
Qwen-Image-Edit	Original RGB	**0.03807**	**1.917**
	OmniAlpha RGBA	0.05197	2.078
FLUX.1-dev	Original RGB	**0.01216**	**2.109**
	AlphaVAE RGBA	0.04762	3.240

C.3 Robustness to Prompt Templates

We evaluate how instruction wording affects image-to-RGBA extraction with the FLUX.2 Klein Base 9B and Qwen-Image-Edit-2511 variants of AlphaAnything on all 735 target layers in RGBA-Image. Each target description is kept verbatim, while the surrounding instruction follows one of the four templates in Table 11: the standard OmniAlpha extraction template, a minimal request, a background-removal instruction, or explicit RGBA output requirements. Within each backbone, we fix the 20,000-step checkpoint, source images, resizing policy, 50 sampling steps, and guidance scale 4. Each target retains its seed from the corresponding backbone's standard run with base seed 42; standard predictions are reused from the main comparison.

Table 11 reports the mean metrics for each template. For FLUX.2, Soft IoU ranges from 0.8276 to 0.8296 and LPIPS from 0.0206 to 0.0214; for Qwen Image Edit, the corresponding ranges are 0.9223–0.9299 and 0.0126–0.0138. All three alternative templates improve Qwen Image Edit's RGB L1 and Soft IoU over the standard template, with the background-removal instruction achieving the best RGB L1, Soft IoU, SAD, and LPIPS among the four templates. SAD shows greater variation, spanning 46.3572–52.8804 for FLUX.2 and 27.8815–33.1917 for Qwen Image Edit. These results support robustness of overlap and perceptual fidelity to the tested instruction changes on both backbones, with greater sensitivity in absolute alpha error.

The table uses the same six metrics as the main image-to-RGBA evaluation: RGB L1, Soft IoU, SAD, LPIPS, Alpha Grad, and Alpha Conn, defined in Appendix B.4.2. Alpha Grad and Alpha Conn for the standard template are taken from the corresponding main-comparison runs.

Table 11: Prompt-template robustness of AlphaAnything with FLUX.2 Klein Base 9B and Qwen-Image-Edit-2511 on RGBA-Image. Each reported metric is averaged over 735 target layers. The placeholder {description} contains the same target description in every template.

Template	Complete instruction
Standard (OmniAlpha)	`Extract the object described by the text, preserving fine details and transparency: {description}`
Minimal	`Extract the object described below: {description}`
Remove background	`Remove the background and all other objects, keeping only the target described below: {description}`
RGBA output	`Target layer: {description} Output requirements: RGBA format; transparent background; original position and scale.`

Template	RGB L1 ↓	Soft IoU ↑	SAD ↓	LPIPS ↓	Alpha Grad ↓	Alpha Conn ↓
FLUX.2 Klein Base 9B						
Standard	0.1155	0.8296	46.3572	0.0207	69.5947	46.0840
Minimal	0.1154	0.8276	52.8804	0.0214	69.6647	52.6148
Remove background	0.1163	0.8289	48.5944	0.0206	69.9193	48.3572
RGBA output	0.1160	0.8285	52.3816	0.0213	69.8098	52.1291
Qwen Image Edit						
Standard	0.0746	0.9223	30.3820	0.0136	27.5221	30.0584
Minimal	0.0710	0.9262	33.1917	0.0138	27.0243	32.8460
Remove background	0.0691	0.9299	27.8815	0.0126	26.0230	27.5284
RGBA output	0.0731	0.9250	31.6661	0.0134	26.8173	31.3408

C.4 Robustness Across Inference Seeds

We evaluate inference base seeds 42, 43, and 44 using the trained checkpoints and evaluation settings of the main comparisons. Table 12 reports the mean and sample standard deviation across the three runs on RGBA-Text, AlphaTest, and RGBA-Image. Seed 42 is the main-table run. FID is computed separately within each run and background before averaging over backgrounds and summarizing across seeds.

All four AlphaAnything text-to-RGBA backbones outperform the compared baselines in ImageReward for every evaluated seed on both text benchmarks. LongCat obtains ImageReward scores of 1.0040 ± 0.0023 on RGBA-Text and 1.5146 ± 0.0094 on AlphaTest. On RGBA-Image, Qwen Image Edit ranks first on all six metrics in each run, with mean Soft IoU 0.9236 ± 0.0023 and LPIPS 0.0132 ± 0.0009. The per-seed comparisons therefore retain these main observations across the three inference runs.

Table 12: Inference-seed robustness. Entries give the mean and sample standard deviation.

RGBA-Text

Method / backbone		CLIPScore ↑	Aesthetic ↑	ImageReward ↑
LayerDiffuse		0.8505 ± 0.0032	5.4525 ± 0.0032	0.5684 ± 0.0217
AlphaVAE		0.8314 ± 0.0007	5.2758 ± 0.0070	0.5305 ± 0.0128
OmniAlpha		0.8515 ± 0.0024	5.2981 ± 0.0114	0.7268 ± 0.0233
Qwen-Image-2.1		0.8492 ± 0.0026	5.3023 ± 0.0092	0.6287 ± 0.0093
AlphaAnything	FLUX.1	0.8354 ± 0.0035	5.5108 ± 0.0128	0.7667 ± 0.0187
	FLUX.2	0.8667 ± 0.0018	5.5136 ± 0.0031	0.9741 ± 0.0143
	Qwen Image	0.8681 ± 0.0005	5.5287 ± 0.0114	0.9738 ± 0.0255
	LongCat	0.8658 ± 0.0026	5.5284 ± 0.0038	1.0040 ± 0.0023

AlphaTest

Method / backbone		FID ↓	CLIPScore ↑	Aesthetic ↑	ImageReward ↑
LayerDiffuse		80.8024 ± 0.9294	0.8494 ± 0.0007	5.5161 ± 0.0009	1.1457 ± 0.0119
AlphaVAE		74.2638 ± 0.7549	0.8227 ± 0.0014	5.5810 ± 0.0110	1.1074 ± 0.0226
OmniAlpha		72.7440 ± 0.0383	0.8364 ± 0.0027	5.3523 ± 0.0023	1.2086 ± 0.0057
Qwen-Image-2.1		71.9380 ± 0.3099	0.8313 ± 0.0027	5.4760 ± 0.0083	1.1776 ± 0.0184
AlphaAnything	FLUX.1	77.4103 ± 0.1232	0.8097 ± 0.0009	5.7074 ± 0.0097	1.2975 ± 0.0094
	FLUX.2	74.8920 ± 0.3339	0.8439 ± 0.0013	5.6347 ± 0.0045	1.4422 ± 0.0086
	Qwen Image	75.8810 ± 0.2563	0.8340 ± 0.0003	5.6853 ± 0.0088	1.3707 ± 0.0094
	LongCat	74.9678 ± 0.5422	0.8425 ± 0.0012	5.7198 ± 0.0084	1.5146 ± 0.0094

RGBA-Image

Method / backbone		RGB L1 ↓	Soft IoU ↑	SAD ↓
OmniAlpha		0.1032 ± 0.0024	0.6886 ± 0.0005	175.7390 ± 2.3482
Qwen-Image-2.1		0.0909 ± 0.0037	0.6853 ± 0.0069	185.0934 ± 7.4971
AlphaAnything	FLUX.2	0.1155 ± 0.0001	0.8270 ± 0.0022	52.6253 ± 6.4488
	Qwen Image Edit	0.0739 ± 0.0008	0.9236 ± 0.0023	29.3979 ± 3.0715

Method / backbone		LPIPS ↓	Alpha Grad ↓	Alpha Conn ↓
OmniAlpha		0.0902 ± 0.0010	70.8178 ± 0.8268	174.2699 ± 2.3331
Qwen-Image-2.1		0.0855 ± 0.0022	81.3381 ± 1.8358	169.7300 ± 7.6814
AlphaAnything	FLUX.2	0.0218 ± 0.0010	70.0745 ± 0.6561	52.3602 ± 6.4494
	Qwen Image Edit	0.0132 ± 0.0009	27.5225 ± 0.4195	29.0806 ± 3.0700

C.5 BOOTSTRAP CONFIDENCE INTERVALS

We estimate input-sampling uncertainty with 10,000 paired bootstrap resamples and 95% bias-corrected and accelerated (BCa) intervals computed with SciPy. Scores are first averaged over the three inference runs and, where applicable, the black/white backgrounds. We resample prompts for RGBA-Text and AlphaTest, and source-image clusters for RGBA-Image, retaining all associated layers and the layer-weighted mean. Every method uses the same resamples and bootstrap seed 42. Table 13 reports the resulting intervals.

Paired differences compare each ALPHAANYTHING variant with the baseline having the best observed mean for that metric. LongCat's ImageReward gains over OmniAlpha are 0.2771 with interval $[0.2282, 0.3296]$ on RGBA-Text and 0.3060 with interval $[0.2715, 0.3435]$ on AlphaTest. All six Qwen Image Edit differences on RGBA-Image favor ALPHAANYTHING; its Soft IoU gain over OmniAlpha is 0.2350 with interval $[0.2115, 0.2605]$. These pointwise 95% intervals quantify input-sampling uncertainty for the mean across the saved runs.

Table 13: Bootstrap results. Entries give the mean and 95% BCa interval.

RGBA-Text

Method / backbone		CLIPScore ↑	Aesthetic ↑	ImageReward ↑
LayerDiffuse		0.8505 [0.8427, 0.8585]	5.4525 [5.4035, 5.4958]	0.5684 [0.4780, 0.6580]
AlphaVAE		0.8314 [0.8226, 0.8399]	5.2758 [5.2279, 5.3239]	0.5305 [0.4351, 0.6247]
OmniAlpha		0.8515 [0.8428, 0.8599]	5.2981 [5.2519, 5.3415]	0.7268 [0.6385, 0.8117]
Qwen-Image-2.1		0.8492 [0.8409, 0.8578]	5.3023 [5.2570, 5.3482]	0.6287 [0.5379, 0.7144]
ALPHAANYTHING	FLUX.1	0.8354 [0.8270, 0.8436]	5.5108 [5.4614, 5.5584]	0.7667 [0.6735, 0.8559]
	FLUX.2	0.8667 [0.8582, 0.8755]	5.5136 [5.4640, 5.5614]	0.9741 [0.8911, 1.0532]
	Qwen Image	0.8681 [0.8603, 0.8763]	5.5287 [5.4800, 5.5751]	0.9738 [0.8920, 1.0515]
	LongCat	0.8658 [0.8574, 0.8745]	5.5284 [5.4784, 5.5768]	1.0040 [0.9244, 1.0803]

AlphaTest

Method / backbone		CLIPScore ↑	Aesthetic ↑	ImageReward ↑
LayerDiffuse		0.8494 [0.8436, 0.8554]	5.5161 [5.4870, 5.5436]	1.1457 [1.0848, 1.2025]
AlphaVAE		0.8227 [0.8168, 0.8290]	5.5810 [5.5473, 5.6127]	1.1074 [1.0348, 1.1781]
OmniAlpha		0.8364 [0.8304, 0.8426]	5.3523 [5.3219, 5.3818]	1.2086 [1.1426, 1.2676]
Qwen-Image-2.1		0.8313 [0.8251, 0.8375]	5.4760 [5.4434, 5.5089]	1.1776 [1.1140, 1.2379]
ALPHAANYTHING	FLUX.1	0.8097 [0.8032, 0.8165]	5.7074 [5.6745, 5.7396]	1.2975 [1.2368, 1.3543]
	FLUX.2	0.8439 [0.8374, 0.8506]	5.6347 [5.6016, 5.6672]	1.4422 [1.3865, 1.4915]
	Qwen Image	0.8340 [0.8276, 0.8404]	5.6853 [5.6529, 5.7176]	1.3707 [1.3137, 1.4208]
	LongCat	0.8425 [0.8363, 0.8491]	5.7198 [5.6893, 5.7497]	1.5146 [1.4670, 1.5559]

RGBA-Image

Method / backbone		RGB L1 ↓	Soft IoU ↑	SAD ↓
OmniAlpha		0.1032 [0.0942, 0.1159]	0.6886 [0.6608, 0.7145]	175.7390 [145.8230, 220.1509]
Qwen-Image-2.1		0.0909 [0.0855, 0.0974]	0.6853 [0.6638, 0.7056]	185.0934 [153.2073, 224.3758]
ALPHAANYTHING	FLUX.2	0.1155 [0.1070, 0.1260]	0.8270 [0.8050, 0.8457]	52.6253 [41.4052, 79.6911]
	Qwen Image Edit	0.0739 [0.0698, 0.0796]	0.9236 [0.9109, 0.9325]	29.3979 [19.0433, 60.7663]

Method / backbone		LPIPS ↓	Alpha Grad ↓	Alpha Conn ↓
OmniAlpha		0.0902 [0.0801, 0.1025]	70.8178 [62.3785, 82.2091]	174.2699 [144.4615, 218.6509]
Qwen-Image-2.1		0.0855 [0.0741, 0.0985]	81.3381 [70.3746, 95.0267]	169.7300 [138.1622, 208.3698]
ALPHAANYTHING	FLUX.2	0.0218 [0.0189, 0.0265]	70.0745 [60.2413, 81.8434]	52.3602 [41.1351, 79.3873]
	Qwen Image Edit	0.0132 [0.0109, 0.0183]	27.5225 [24.0329, 33.1854]	29.0806 [18.7263, 60.4309]

C.6 HUMAN EVALUATION

We conducted a blinded pairwise human evaluation of I2RGBA outputs on RGBA-Image with three experienced annotators. Each annotator evaluated randomly assigned comparisons, viewing the input image, extraction instruction, and two anonymized predictions with randomized left–right positions. Annotators selected the preferred output or a tie based on instruction consistency, visual quality, and the naturalness of transparency and object boundaries. Each ALPHAANYTHING backbone was compared against Qwen-Image-2.1 and OmniAlpha, without comparisons between our backbone variants.

For each backbone–baseline pair, let W, T, and L denote wins, ties, and losses for ALPHAANYTHING, respectively. We report two complementary metrics: the win rate excluding ties,

$$\mathrm{WR}_{\mathrm{excl}} = \frac{W}{W + L}, \tag{37}$$

and the tie-adjusted win rate,

$$\mathrm{WR}_{\mathrm{adj}} = \frac{W + 0.5T}{W + T + L}. \tag{38}$$

The results are reported in Table 14.

Table 14: Pairwise human evaluation of ALPHAANYTHING on RGBA-Image for I2RGBA. Both win rates are reported as percentages.

ALPHAANYTHING backbone	Baseline	Win rate (%)	
		Excluding ties	Tie-adjusted
FLUX.2 Klein Base 9B	Qwen-Image-2.1	90.48	72.37
FLUX.2 Klein Base 9B	OmniAlpha	77.78	63.51
Qwen-Image-Edit-2511	Qwen-Image-2.1	96.00	79.49
Qwen-Image-Edit-2511	OmniAlpha	100.00	80.56

C.7 INFERENCE OVERHEAD

Since the DiT LoRA weights can be merged into the pretrained weights at inference, the additional computation of ALPHAANYTHING comes only from the alpha branch, including its RGB-to-alpha feature injections. We profile five backbones using a single NVIDIA GPU per run, with the same GPU model across all measurements. All runs use 1024×1024 resolution, batch size 1, BF16 precision, and 50 sampling steps. The RGB control uses the same decoder traversal and input latents as ours, with unnecessary block-input retention and Qwen's temporal feature cache removed. Table 15 reports VAE decoding time, peak allocated GPU memory, and end-to-end generation time.

VAE timings are medians of 30 measurements after 10 warmup iterations; generation timings are medians of three measurements after a three-step warmup. End-to-end generation time includes text encoding, sampling, and decoding, excluding model loading and image saving. We compute the end-to-end overhead by dividing the additional VAE decoding time by the matched control's end-to-end generation time.

Table 15: Inference overhead of ALPHAANYTHING across 4 backbones. Peak memory is measured for the VAE in isolation. End-to-end generation time is measured for the matched control, and end-to-end overhead is computed from unrounded timings.

Backbone	VAE decode time (s)		Peak memory (GiB)		End-to-end generation (s)	End-to-end overhead (%)
	RGB VAE	Ours VAE	RGB VAE	Ours VAE		
FLUX.1-dev	0.149	0.247	2.297	2.389	22.103	0.445
FLUX.2 Klein Base 9B	0.148	0.249	2.299	2.386	41.212	0.244
Qwen-Image-2512	0.138	0.239	2.118	2.331	44.206	0.228
LongCat-Image	0.148	0.247	2.297	2.389	23.154	0.427

Across all five evaluated backbones, ALPHAANYTHING adds less than 0.5% runtime overhead to end-to-end generation.

D ADDITIONAL QUALITATIVE RESULTS

This appendix extends the main-text showcases with additional matched comparisons from RGBA-Text and RGBA-Image, qualitative component ablations, and an exploratory study of RGBA input for editing. The generation, extraction, and ablation comparisons follow the evaluation settings in Appendix B. Prompts for the teaser examples are provided in `prompts_for_examples.json` in the supplementary material.

D.1 TEXT-TO-RGBA GENERATION

Figures 8 and 9 compare all eight text-to-RGBA methods on eleven additional RGBA-Text prompts. Each case displays the shared content prompt above two rows: four baselines followed by all four ALPHAANYTHING settings. Method-specific prompt wrappers follow Appendix B.2. All RGBA outputs retain their complete canvases and are displayed on the same checkerboard.

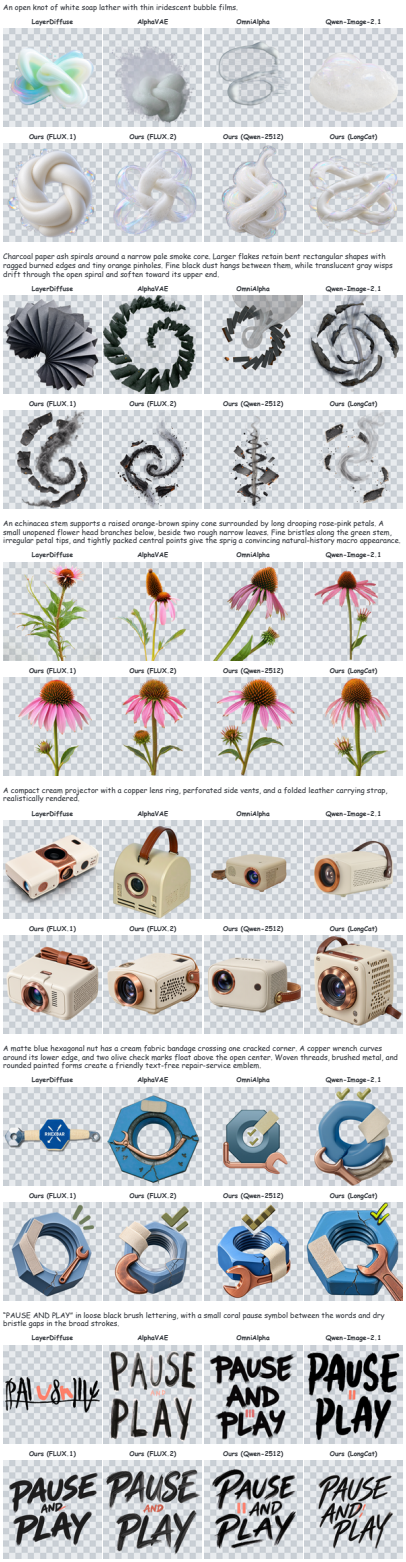

Figure 8: Additional qualitative text-to-RGBA comparisons on 6 RGBA-Text prompts (set 1 of 2). Each case shows four baselines in the upper row and all four ALPHAANYTHING settings in the lower row. Complete RGBA canvases are retained.

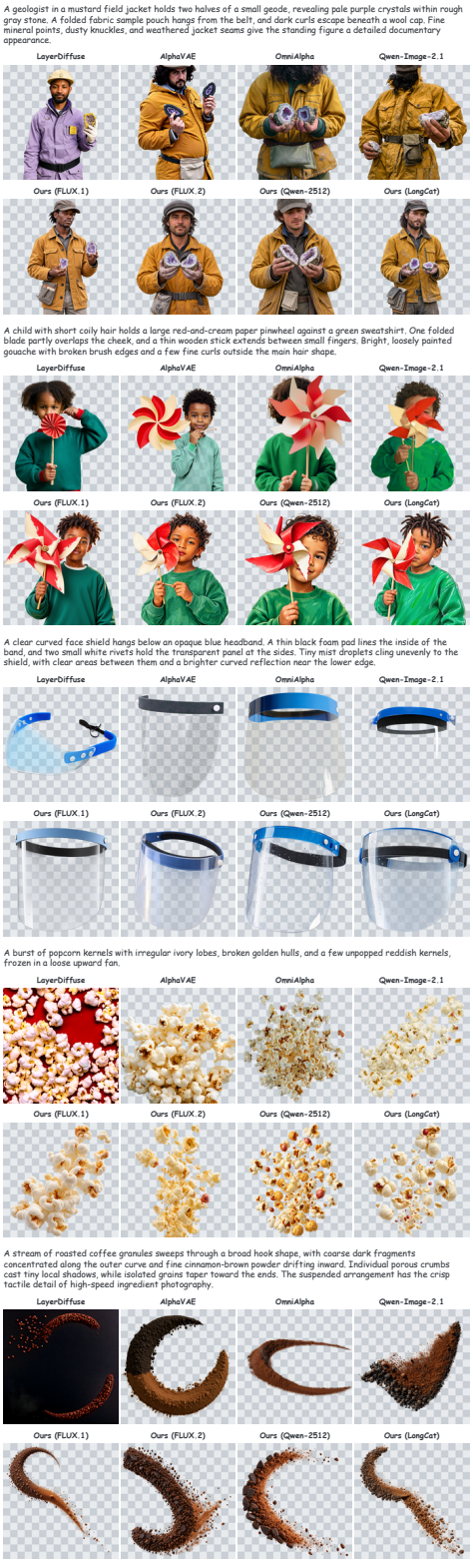

Figure 9: Additional qualitative text-to-RGBA comparisons on 5 RGBA-Text prompts (set 2 of 2). Each case shows four baselines in the upper row and all four ALPHAANYTHING settings in the lower row. Complete RGBA canvases are retained.

D.2 IMAGE-TO-RGBA EXTRACTION

Figures 10 and 11 compare the four image-to-RGBA methods on fourteen additional RGBA-Image examples. The full prompt shared by all four methods is printed above each example, followed by the complete input, target RGBA layer, and prediction canvases. Where informative, one additional row examines either RGBA details or alpha boundaries using the same crop coordinates for every method. These visual comparisons complement the full-dataset scores in Table 3.

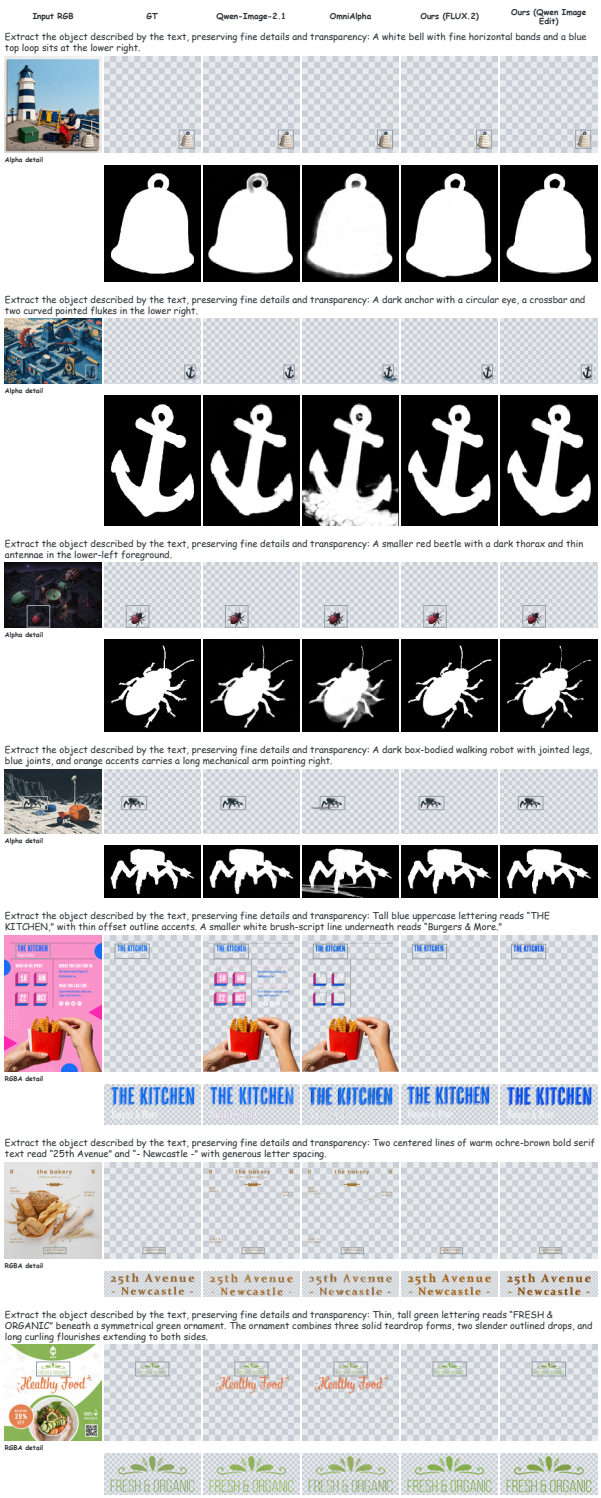

Figure 10: Additional qualitative image-to-RGBA comparisons on seven RGBA-Image examples (set 1 of 2). Inputs, target layers, and all four methods retain their complete canvases. Informative alpha or RGBA details appear below the corresponding full views.

D.3 UNIDIRECTIONAL FEATURE SHARING

Figure 12 compares ALPHAANYTHING with the coupled-decoder variant on RGBA-Image. Alpha-to-RGB feedback introduces visible foreground color shifts in these examples. The comparisons use the matched checkpoints and inference settings in Appendix B.5.

D.4 PROPAGATING ALPHA SUPERVISION TO THE DIT

Figure 13 compares ALPHAANYTHING with Stop-grad on Qwen-Image-Edit. The Stop-grad variant produces opacity artifacts around the price badge and wine bottle, omits the second line of lettering, and fills gaps between wheel spokes. The full model retains more accurate opacity structure in these cases.

D.5 ENDPOINT-AWARE ALPHA LOSS

Figure 14 compares EA Loss with standard alpha L_1 on Qwen-Image-Edit. The alpha-L_1 variant produces spurious opacity around the chocolate mold and craftsperson, and transparent holes inside the two opaque text panels. ALPHAANYTHING preserves the target opacity more closely in these examples.

D.6 EXPLORING RGBA UNDERSTANDING THROUGH EDITING

We view native RGBA image understanding as a potential advantage of jointly representing color and transparency in an RGBA VAE, while regarding high-quality RGBA generation as the more important capability. To explore this potential advantage, we conduct a qualitative editing experiment with Qwen-Image-2.1 (Qwen Team, 2026). For each image layer, we compare the original RGBA input with the same layer composited over white and supplied with an all-one alpha channel. Under matched editing instructions and sampling settings, the paired outputs appear largely similar in overall appearance and opacity structure (Figure 15). Future work could explore the distinctive value of RGBA understanding and the tasks in which it may offer practical benefits.

The six cases cover color modification, rotation, object removal, object addition, material modification, and style transfer. Each pair uses seed 42, 40 inference steps, a guidance scale of 1.0, and a 1024×1024 output canvas. We display the saved RGBA inputs and outputs on a common checkerboard.

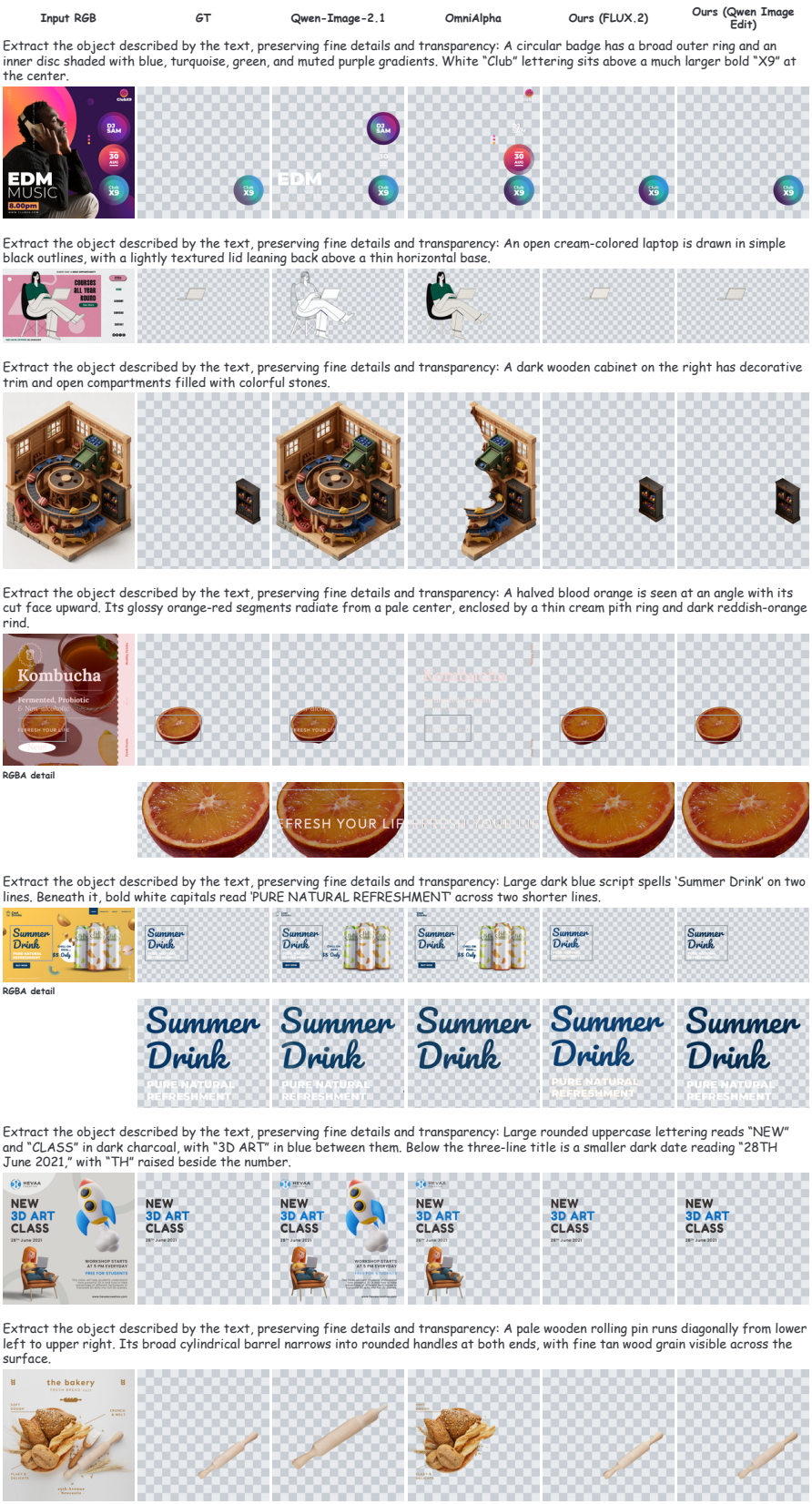

Figure 11: Additional qualitative image-to-RGBA comparisons on seven RGBA-Image examples (set 2 of 2). Inputs, target layers, and all four methods retain their complete canvases. Informative alpha or RGBA details appear below the corresponding full views.

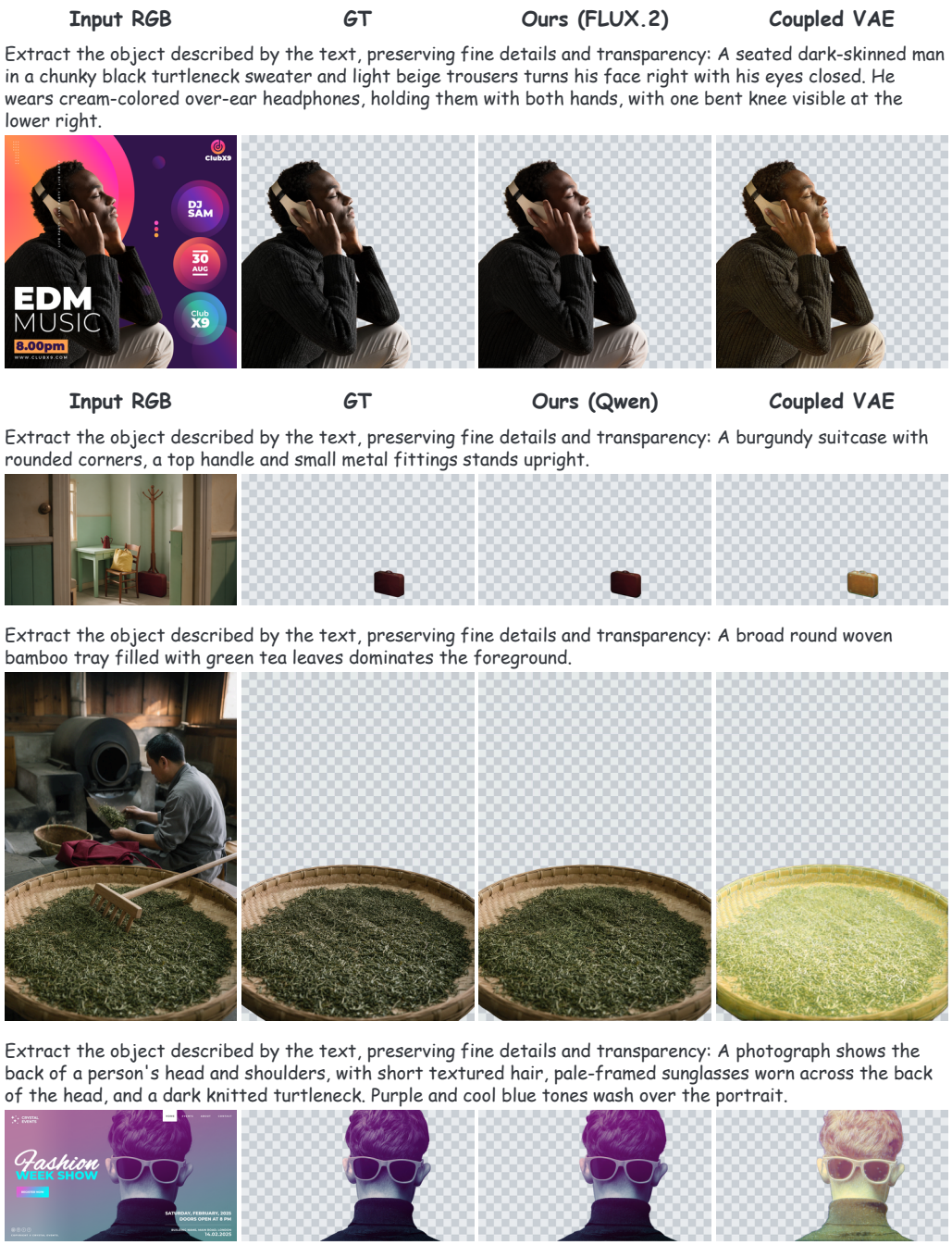

Figure 12: Qualitative coupled-decoder ablation on FLUX.2 and Qwen-Image-Edit. ALPHAANYTHING preserves foreground colors more faithfully than the coupled decoder.

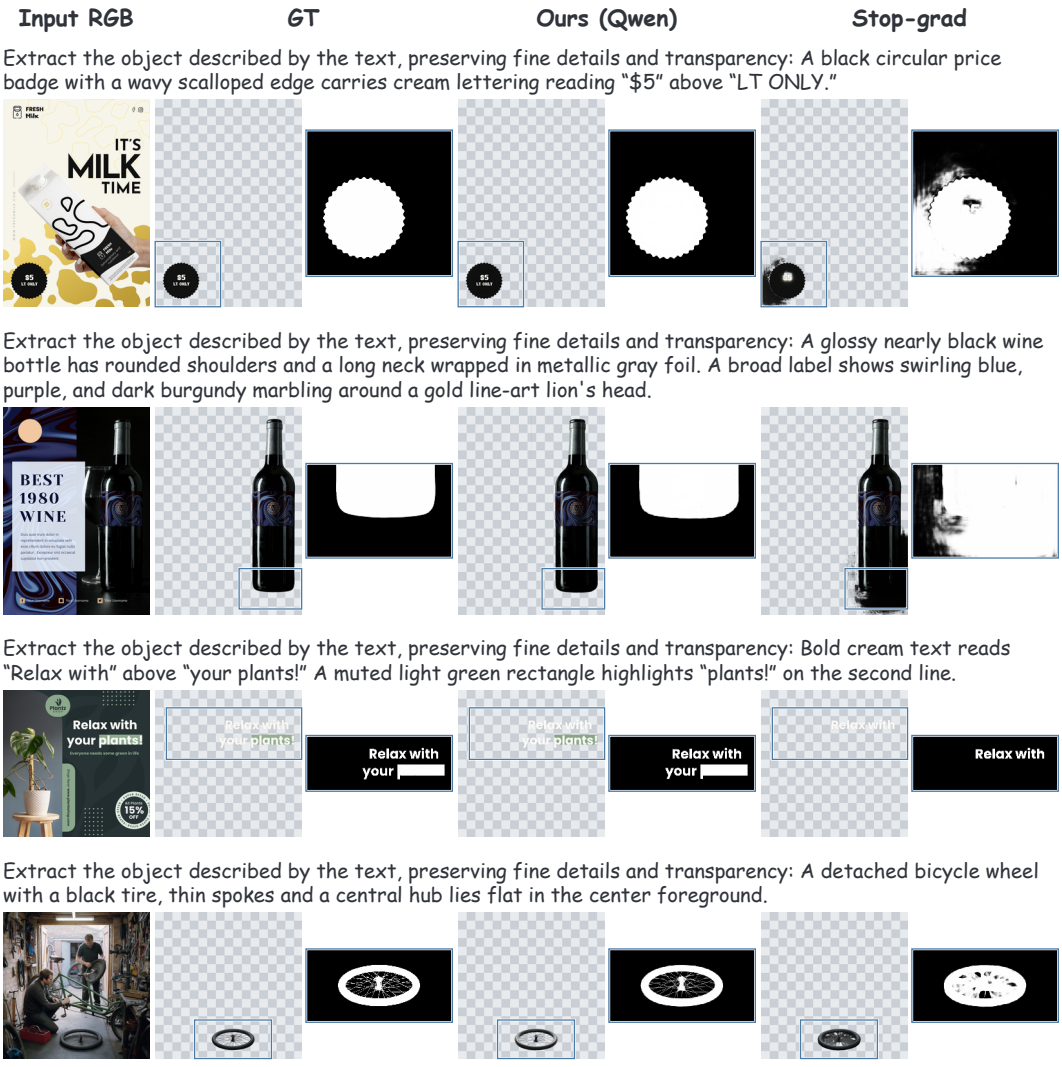

Figure 13: Qualitative ablation of alpha supervision to the DiT on Qwen-Image-Edit. ALPHAANYTHING better preserves lettering and wheel spokes, with less stray opacity than Stop-grad.

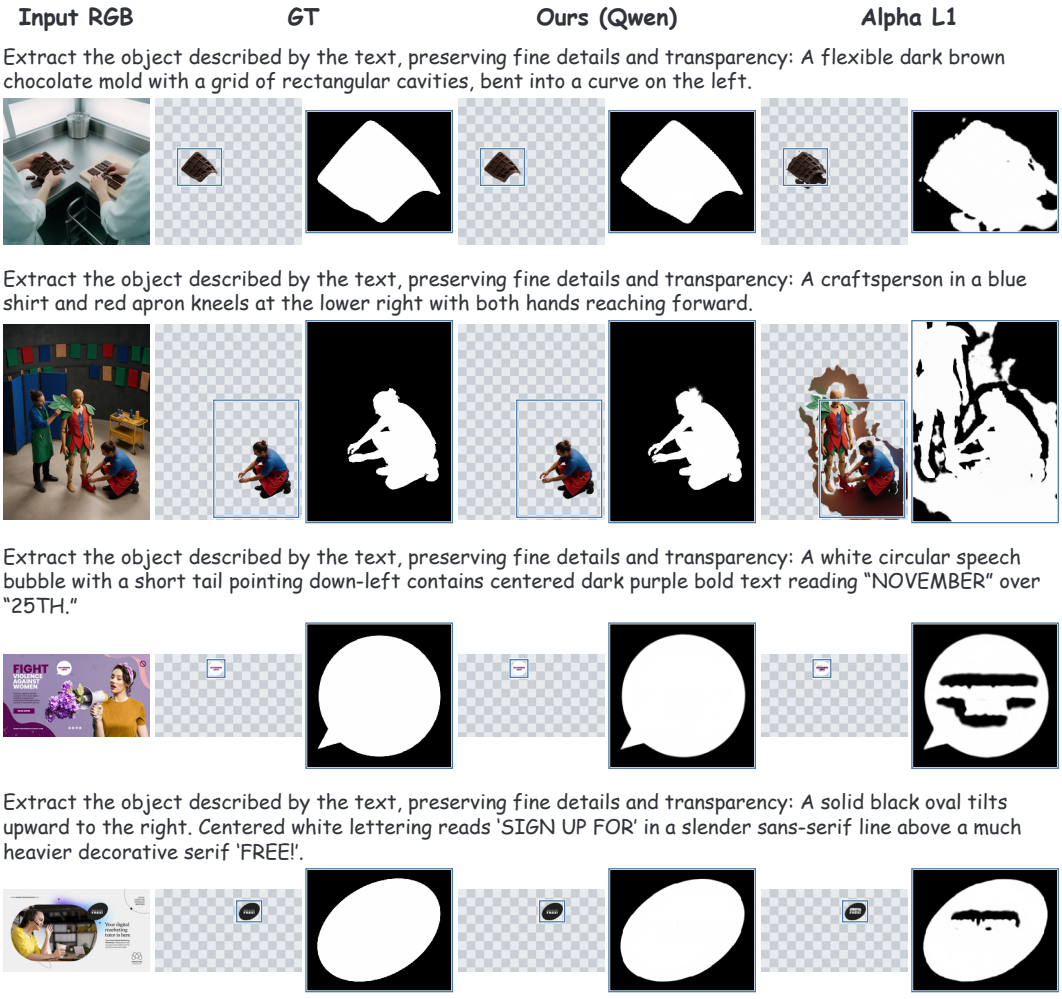

Figure 14: Qualitative comparison of EA Loss and alpha L_1 on Qwen-Image-Edit. EA Loss reduces stray background opacity and transparent holes in opaque text regions.

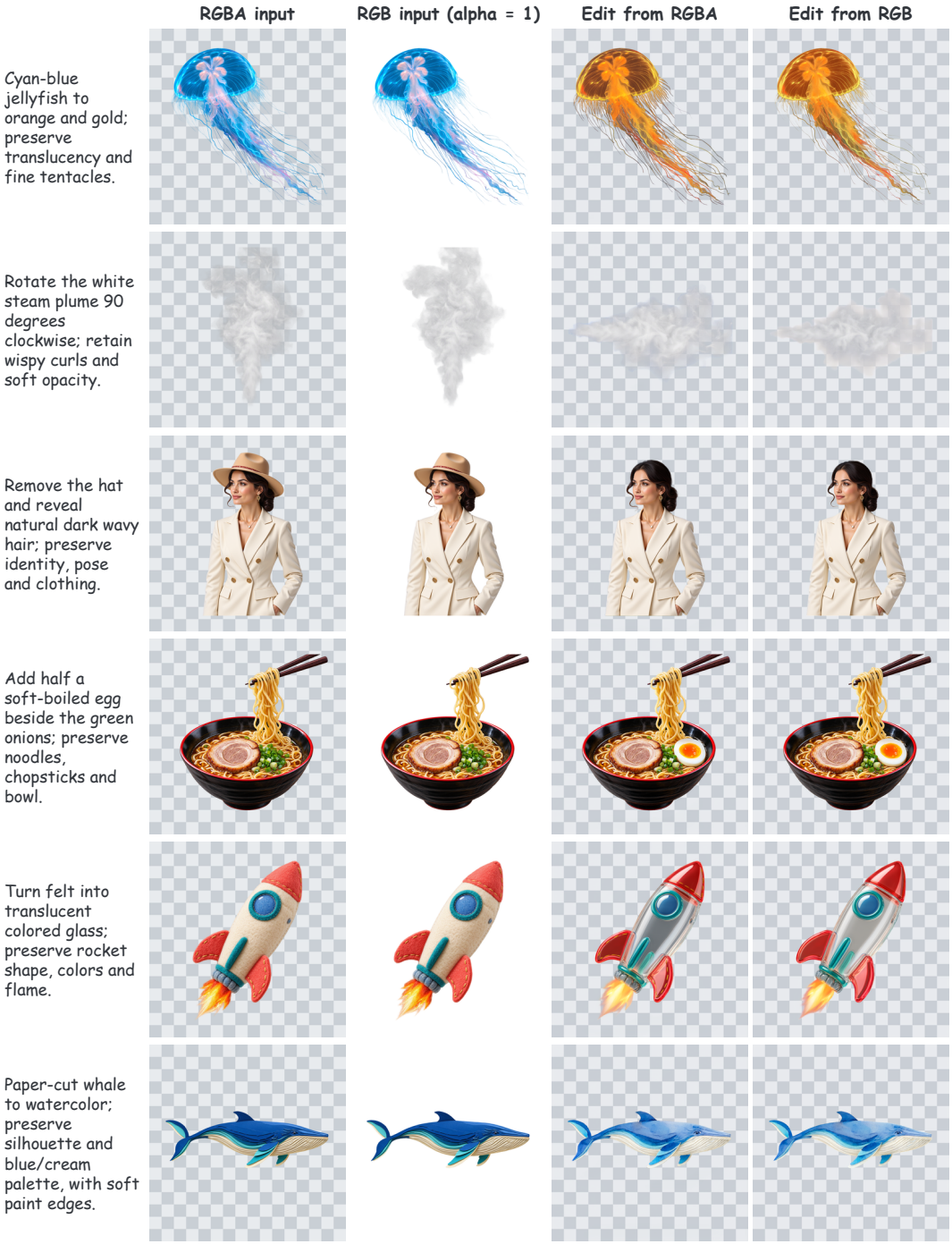

Figure 15: Paired editing with Qwen-Image-2.1. The first two columns show the original RGBA input and its white-background composite with alpha set to one. The last two columns show the corresponding edited RGBA outputs. Each row uses the same prompt and sampling settings; the instruction is summarized on the left. RGBA images are displayed on a checkerboard. The two input conditions yield visually similar results in these examples.