UniVL: Unified Vision-Language Embedding for Spatially Grounded Contextual Image Generation

Jiayun Wang **Yu Wang** **Weijie Gan** **Zhenting Wang** **Wei Wei**
Center for Advanced AI, Accenture

Abstract

We introduce *spatially grounded contextual image generation*, a controllable image generation task in which each user-specified mask region carries a short text label, and propose UNIVL (Unified Vision-Language image generation) for it. Instead of supplying a reference image and a global text prompt through two separate encoders (vision and language), UNIVL takes a *single unified visual input* in which each label is rendered as pixels inside its mask, so that *what* to generate and *where* are specified by the same signal. The UNIVL encoder, adapted from an optical-character-recognition (OCR)-pretrained vision backbone, reads this condition and produces a single token sequence, the UNIVL embedding f_{VL}. A two-stage pipeline first aligns f_{VL} with the VAE latent space and then fine-tunes a pretrained diffusion transformer conditioned only on f_{VL}. For short-label, object-level spatial contextual generation, this eliminates the standalone text encoder (e.g., T5); the user still supplies a mask and a short label per region. On UNIVL-ImgGen, a benchmark of 477K mask-annotated images that we construct for training and evaluation, UNIVL without a text encoder improves image quality over the text-prompted OminiControl baseline, both on the full image (FID: $13.5 \rightarrow 11.1$, PSNR: $16.1 \rightarrow 19.6$) and inside the masks (PSNR: $17.6 \rightarrow 18.6$), while reducing inference TFLOPs by up to 52% and runtime by up to 44%. Zero-shot on COCO, it has the best FID, SSIM, and PSNR among methods without a text prompt, and the same recipe works with an SD3.5-Medium backbone. Our code is available at `https://github.com/Accenture/UniVL`.

1 Introduction

Text-to-image diffusion models generate high-fidelity images from natural-language prompts [25, 26, 43], and controllable generation methods add spatial signals such as sketches, depth maps or bounding boxes [15, 33, 41]. In most of these systems, semantics and location arrive through separate inputs: *text prompts* carry rich semantics but no explicit spatial grounding, while *structural controls* carry precise geometry but little semantic content. We study how to specify *what* should appear *where* with a single conditioning signal.

A signal of this kind already exists in visual *understanding*. Optical context compression [35, 36] renders text as pixels and reads it with a vision encoder; DeepSeek-OCR reports about 97% decoding precision at text-to-vision-token compression ratios below $10\times$ [35]. Text rendered at a pixel location carries both its meaning and its position, which is the property controllable generation needs. We therefore ask: can an encoder trained for optical compression serve as the conditioning interface for controllable generation? We answer it with a new task, *spatially grounded contextual image generation*. The user provides a reference image, draws one or more mask regions, and supplies a short text label for each region. Our pipeline renders each label as pixels inside its mask, producing a single *contextual condition* image, and the model must generate content matching each label inside its mask while preserving the rest of the image. This formulation has two properties. First, it *co-locates semantics and position in one input*: each label sits at the coordinates of its region, so several labeled regions can be edited in a single forward pass (Fig. 3, row 2). Second, for short object-level labels it

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

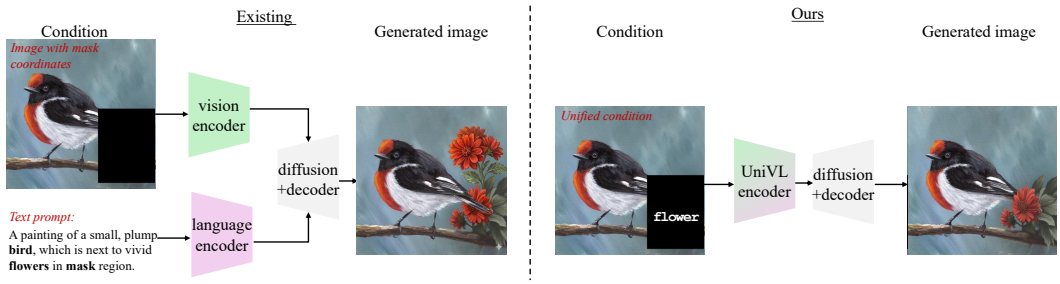


Figure 1: Conditioning paradigm for contextual image generation: existing (*left*) vs. our UNIVL (*right*). The user's intent is the same: inpaint a target region with a desired object. Existing methods take two inputs from two domains, a masked image (vision) and a global text prompt (language), each processed by its own encoder before the diffusion model and decoder. In UNIVL, the same intent is captured by a *single unified visual condition*: our pipeline renders the text label as pixels inside the mask, and the UNIVL encoder (an OCR-pretrained vision backbone adapted to our task) reads the spatial layout and the label in one pass. No text encoder is used, and the generated image contains the requested object ("flower") at the specified location.

eliminates the standalone text encoder: one OCR-pretrained vision encoder reads both the visual context and the rendered labels, so the text encoder (T5 [24] in FLUX) is no longer needed, which reduces inference cost (Section 5.4). We target short, object-level labels (e.g., "car", "a wooden bench"), which make up most of our training data. Simple attribute phrases work, whereas style, negation, and relational labels do not (Section 5.6).

We instantiate this task with UNIVL (Unified Vision-Language image generation), built on the FLUX.1-dev diffusion transformer [2] with OminiControl's inpainting LoRA [29]. UNIVL has two training stages: (1) UNIVL *embedding alignment*, which fine-tunes the UNIVL encoder (initialized from the OCR-pretrained encoder of DeepSeek-OCR [35]) so that its output f_{VL} matches the VAE latent of the target image (Fig. 2, Stage 1); and (2) *diffusion fine-tuning*, which conditions the diffusion model only on f_{VL} (Fig. 2, Stage 2). To support training and evaluation, we construct the UNIVL-*ImgGen* benchmark of $\sim$477K mask-annotated images ($\sim$710K mask-text records). Without a text encoder, UNIVL improves image quality over the text-prompted OminiControl [29] (FID: 13.50 $\rightarrow$ 11.13), including inside the masks (region-PSNR: 17.6 $\rightarrow$ 18.6), while reducing inference TFLOPs by up to 52% and runtime by up to 44%. Zero-shot on COCO, UNIVL has the best FID, SSIM, and PSNR among methods without a text prompt, and the same training recipe works with SD3.5-Medium [9] as the backbone (Section 5.5). An optional UNIVL +T5 hybrid narrows the remaining text-alignment gap to prompted baselines (Table 1), so the unified condition can be combined with standard text conditioning when the cost of a text encoder is acceptable.

We summarize our contributions as follows: (1) **A controllable generation task with a unified condition**: we formalize *spatially grounded contextual image generation*, which replaces the two-domain conditioning interface (vision + language) with a single visual input in which short text labels are rendered inside their masks, and we construct the UNIVL-*ImgGen benchmark* ($\sim$477K images with mask-text annotations). (2) **UNIVL, a unified vision-language image generation framework**: we show that an OCR-pretrained vision encoder can be *adapted* into a conditioning encoder for diffusion through a two-stage alignment-and-fine-tuning pipeline, ablate its design choices (Section 5.5), and measure its operating range in mask size and label type (Section 5.6). (3) **Efficiency without loss of image quality**: removing the text encoder reduces conditioning token size by up to 98%, encoder parameters by 92%, TFLOPs by up to 52%, and runtime by up to 44% (Table 2; savings are largest at 256^2 and shrink to 8% runtime at 1024^2), while UNIVL outperforms the prompted baselines on all four image-quality metrics (Table 1).

2 Related Work

Diffusion-based image generation and editing. Diffusion models [8, 11] and their latent variants [26] are the dominant text-to-image foundation; region-aware editing has been built on top through inpainting and instruction-tuned variants [4, 10, 19]. *Our difference*: rather than supply a separate text prompt alongside a mask, we render the textual instruction directly inside the mask and remove the standalone text encoder.

Unified vision-language representation learning. Large-scale vision-language pretraining yields aligned image and text embeddings that transfer broadly across tasks [14, 23]. CLIPPO [31] and PIXEL [27] process text rendered as pixels with a vision model, showing that rendered text can replace a tokenizer for understanding tasks. Another line builds a single backbone for both multimodal understanding and generation, including the Janus series [7, 37], Ming-Omni [1], and Omni-Video [30]. Closest to our work is DeepSeek-OCR [35], which compresses long text into a small number of vision tokens. *Our difference*: these works use rendered text or unified representations for understanding, or keep separate input encoders for each modality (e.g., Janus decouples visual encoding for understanding and generation). UNIVL instead uses a single OCR-pretrained vision encoder, which reads rendered text and image context together, as the conditioning interface for image synthesis.

Spatially grounded controllable generation. Before diffusion, GANs explored spatial control via image-to-image translation [44] and semantic-layout-to-image generation [21], using auxiliary spatial signals to constrain *where* content appears. Diffusion models extended these ideas: ControlNet adds structured spatial signals such as edges and depth [41], GLIGEN grounds generation with layout boxes and class labels [15], and inpainting methods (BrushNet [12], PowerPaint [45], FLUX.1 Fill [2]) complete masked regions. Subject-driven variants such as AnyDoor [6] and Paint-by-Example [39] condition on a reference image, and OmniGen [38] handles multiple generation tasks in one transformer that takes interleaved text and image inputs. *Our difference*: to our knowledge, UNIVL is the first to use a single vision encoder that reads both spatial layout and rendered textual semantics from one visual input as the conditioning interface for diffusion-based controllable generation.

Glyph-aware diffusion generation. A complementary line tackles *rendering* legible text inside generated images: AnyText [32] introduces an auxiliary text-control branch on top of latent diffusion to synthesize multilingual glyphs at user-specified positions; GlyphDraw [20] conditions generation on glyph images and their positions; TextDiffuser [5] adds a layout transformer that predicts keyword bounding boxes and conditions on character-level masks. *Our difference*: these methods treat the rendered text as the *output target*, i.e., the goal is an image with that text legibly drawn in it. UNIVL treats rendered text as a *conditioning interface*: the instruction is rendered into pixels and read by a vision encoder to drive non-textual content generation inside the masked region.

3 The UNIVL-ImgGen Benchmark

To support training and evaluation of spatially grounded contextual image generation, we construct UNIVL-ImgGen, a benchmark comprising 477K unique images and 710K mask-text records. We first define the task, then describe how the benchmark is built and used for training and evaluation.

Task definition. We introduce *spatially grounded contextual image generation*, an inpainting-style controllable task with a unified visual interface. The user provides a source image $X \in \mathbb{R}^{H\times W\times 3}$, draws one or more mask regions $\{m_i\}_{i=1}^{K}$, and supplies a short text label ℓ_i per region. Our pipeline produces a single *contextual condition* C_I by masking each region (filling it with black) and rendering its label as pixels inside the mask. The model must then generate an output $\hat{X}$ that (a) matches ℓ_i within each m_i and (b) preserves the source content outside all masks. In existing paradigms, semantics and location are supplied through separate inputs: global text prompts [25, 26], layout boxes paired with labels [15], or structural cues such as edges and depth that carry little semantics [41]. Our task places both in a single visual input that the UNIVL encoder reads jointly, and supports multiple regions ($K \geq 1$) in one forward pass.

Benchmark overview. UNIVL-ImgGen comprises 710K mask-text records spanning 477K unique images and 28K unique class-name phrases, drawn from four sources: a *mask* category built from LAION-5B [28] images via Grounding DINO [18] detection followed by size and CLIP-similarity filtering and augmentation (4,675 images, 238K records), and three image-editing categories (*add*, *replace*, *extract*) taken from the released ImgEdit data [40]. The mask category provides images with many labeled boxes for multi-region training; the editing categories provide targets that depend on the label, which discourages the model from reconstructing the image condition and ignoring the rendered text. The benchmark uses rectangular masks; we evaluate free-form masks zero-shot in Appendix E.1. Construction details, augmentation, class-name statistics, and the training/evaluation split are in Appendix F. Before splitting the rest into training and validation data, we remove 3,000 samples for evaluation: 1,500 single-mask samples and 1,500 multi-mask samples with $N\in[1, 5]$ masks per image. Because the annotations are generated automatically, two volunteer annotators (not

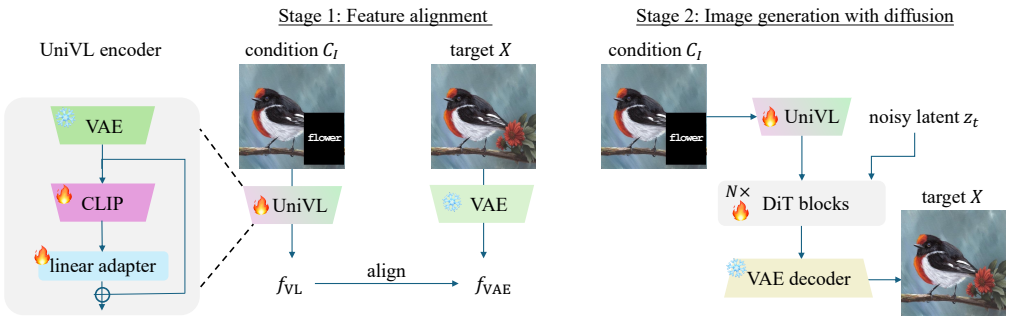


Figure 2: UNIVL training pipeline. *Left:* the UNIVL encoder consists of a frozen VAE encoder, which replaces SAM as the low-level stream of DeepSeek-OCR's encoder, a CLIP backbone (DeepSeek-OCR's OCR-pretrained CLIP with its frozen SAM stem, fine-tuned with LoRA), and a trainable linear adapter; the VAE features bypass CLIP via a skip connection and are fused with the adapted CLIP features by the mask-aware operator of Eq. 2. **Stage 1 (feature alignment):** the UNIVL encoder takes the contextual condition C_I and is trained so that its output f_{VL} matches the VAE latent $z_0 = \mathcal{E}(X)$ of the target image X (labeled f_{VAE} in the figure). **Stage 2 (image generation with diffusion):** the UNIVL encoder maps C_I to f_{VL}, which conditions the DiT together with the noisy latent z_t; the VAE decoder maps the denoised latent to the output image. Both the DiT (LoRA) and the UNIVL encoder (CLIP-LoRA and adapter) are trained in Stage 2.

authors) audited 400 records (100 per category). Their mean scores are 0.95 for whether the mask covers the labeled object, 0.80 for whether the label matches the masked region, and 4.1/5 for overall quality (Appendix F.4).

4 UNIVL: Unified Vision-Language Embedding for Contextual Generation

4.1 Architecture overview

We build UNIVL on a pretrained DiT-based diffusion model: FLUX.1-dev [2] with the inpainting LoRA of OminiControl [29]. The inpainting prior handles *where* to generate, and UNIVL conditioning supplies *what* to generate. We do not train the inpainting stage ourselves; we focus on the UNIVL encoder and the two training stages that produce and use the UNIVL embedding (Fig. 2).

Diffusion transformers (DiT). Diffusion Transformers [22] replace the U-Net of latent diffusion models with a transformer operating on patchified latent tokens. FLUX.1 is trained with rectified flow matching [9, 17]. Given a latent $z_0 = \mathcal{E}(X)$ from a VAE encoder $\mathcal{E}$ and Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, the noisy latent is $z_t = (1-t)\, z_0 + t\, \epsilon$ for $t \in [0, 1]$, and a transformer v_θ is trained to predict the velocity $\epsilon - z_0$:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0, \epsilon, t}\left[\|v_\theta(z_t, t, c) - (\epsilon - z_0)\|^2\right], \tag{1}$$

where c is the conditioning signal and t is sampled from a logit-normal distribution. At inference, the learned velocity field is integrated from pure noise (t=1) to t=0, and the result is decoded by the VAE. Recent methods inject spatial control into DiTs by concatenating condition tokens with the noisy latent tokens; OminiControl [29] places condition and image tokens in a single sequence processed by the same transformer.

From two conditioning encoders to one. In standard conditional DiT frameworks [2, 29], the DiT consumes three inputs: image-condition tokens (the VAE encoding of a reference or masked image), text-prompt tokens C_T from a separate text encoder such as T5, and the noisy latent z_t. The DiT iteratively denoises z_t conditioned on the image and text signals, and the final latent is decoded by the VAE into the output image. UNIVL changes the conditioning interface: it removes C_T and lets the contextual condition C_I (the RGB image with labels rendered inside the masks; Section 3) carry both the visual context and the textual instruction. The UNIVL encoder (Section 4.2) maps C_I to a unified embedding f_{VL} that the DiT consumes alongside z_t. Removing the text-encoder branch yields the efficiency gains in Section 5.4, while the DiT architecture and training objective stay the same.

4.2 UNIVL encoder architecture

The UNIVL encoder (left of Fig. 2), adapted from the DeepSeek-OCR encoder [35], takes the contextual condition C_I and produces a token sequence f_{VL} that conditions the DiT. DeepSeek-OCR's encoder pairs a low-level visual encoder (SAM) with a high-level semantic encoder (CLIP). UNIVL keeps the CLIP part and replaces SAM with the FLUX VAE encoder as the low-level stream, because VAE features lie in the latent space of the VAE decoder and the DiT, which makes alignment much easier. The encoder has three components. (1) A *frozen VAE encoder* extracts low-level visual features $\mathcal{E}(C_I)$, which preserve the source content outside the masks. (2) A *CLIP backbone* extracts high-level semantic features f_{CLIP}. To reuse DeepSeek-OCR's OCR-pretrained weights, we keep this backbone exactly as it was pretrained: a SAM stem encodes C_I, the CLIP transformer reads the SAM features, and its output tokens (without the class token) are concatenated with the SAM features. The SAM stem is therefore part of the CLIP backbone rather than a separate low-level stream; it stays frozen, and the CLIP transformer is fine-tuned with LoRA. DeepSeek-OCR's projector and language decoder are not used. (3) A small *trainable linear adapter* A projects f_{CLIP} to the channel dimension of $\mathcal{E}(C_I)$. The CLIP backbone weights come from a released checkpoint trained on a large OCR corpus (30M PDF pages, 20M scene-OCR images, plus chart/formula/geometry parsing data; Appendix C), so it already reads text rendered as pixels. We do not re-run OCR pretraining; instead, we adapt the encoder to generation conditioning in Stage 1.

Mask-aware fusion. The two feature streams are combined by a mask-aware additive operator. Let $M \in \{0,1\}^{H\times W}$ be the binary mask aggregating all user-specified regions ($M_{ij} = 1$ inside any masked region, 0 outside), resized to the token grid. The fused embedding is

$$f_{\text{VL}} \;=\; \mathcal{E}(C_I) \odot (1 - M) \;+\; A(f_{\text{CLIP}}) \odot M, \tag{2}$$

i.e., positions outside the masks take the VAE features (preserving background context), and positions inside the masks take the projected CLIP features (carrying the semantics from rendered text).

4.3 Two-stage training

Stage 1: UNIVL embedding alignment. We adapt the UNIVL encoder to produce an embedding f_{VL} suitable for conditioning diffusion generation. As shown in Fig. 2 (Stage 1), the UNIVL encoder maps the contextual condition C_I to f_{VL}, while the frozen VAE $\mathcal{E}$ maps the target image X to the latent $z_0 = \mathcal{E}(X)$. We minimize the feature alignment objective

$$\mathcal{L}_{\text{align}} = \|f_{\text{VL}} - z_0\|_2^2, \tag{3}$$

which aligns the embedding of the text-on-mask input with the latent of the desired output. Stage 1 uses only $\mathcal{L}_{\text{align}}$; the CLIP losses below require a generated image and are used only in Stage 2 (Table 5). The VAE encoder is frozen; the CLIP backbone is updated through LoRA, and the linear adapter is trained from scratch (kept in full precision, while the rest of training uses bfloat16). This keeps the number of trainable parameters small and leaves most OCR-pretrained CLIP weights unchanged, while letting the encoder adapt to natural images with rendered text.

Stage 2: Diffusion fine-tuning with UNIVL conditioning. The UNIVL encoder (initialized from Stage 1) maps C_I to f_{VL}, which is the only conditioning signal of the DiT; no text-prompt embedding is used. As shown in Fig. 2 (Stage 2), f_{VL} is fed into the DiT blocks together with the noisy latent z_t. The model is trained with the flow-matching objective of Eq. 1 with $c = f_{\text{VL}}$, plus the feature-alignment loss and two auxiliary CLIP losses:

$$\mathcal{L}_{\text{stage-2}} = \mathcal{L}_{\text{diff}} + \mathcal{L}_{\text{align}} + \lambda_{\text{clip-img}}\,\mathcal{L}_{\text{clip-img}} + \lambda_{\text{clip-txt}}\,\mathcal{L}_{\text{clip-txt}}. \tag{4}$$

Both CLIP losses are computed on the predicted image $\hat{X}$, obtained by decoding the clean-latent estimate $\hat{z}_0 = z_t - t\,v_\theta(z_t, t, c)$ with the VAE decoder, using a separate, frozen OpenAI CLIP ViT-L/14 [23] that is not part of the UNIVL encoder. The CLIP image loss $\mathcal{L}_{\text{clip-img}}$ is the cosine distance between the CLIP image embeddings of the masked region in $\hat{X}$ and in the target image X. The CLIP text loss $\mathcal{L}_{\text{clip-txt}}$ is the cosine distance between the CLIP image embedding of the masked region in $\hat{X}$ and the CLIP text embedding of its label (formulas in Appendix A.2). These losses supervise the semantics of the generated content directly; without the two CLIP losses, FID rises from 11.13 to 28.11 (Appendix D.1). In Stage 2 we update the DiT through LoRA (r=4, α=4) and continue to update the CLIP-LoRA and linear adapter of the UNIVL encoder; the VAE encoder stays

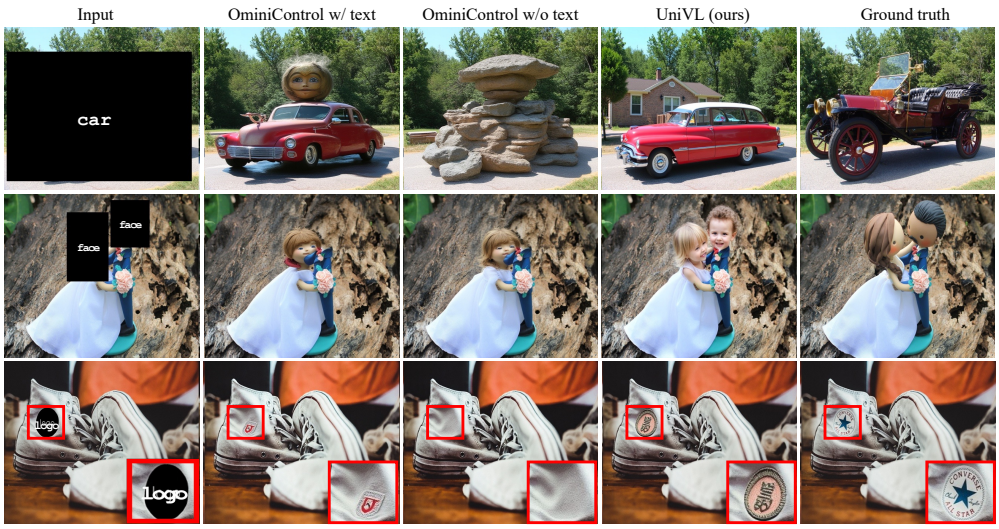


Figure 3: Qualitative comparison with OminiControl [29]. The input column shows the contextual condition given to UNIVL (see Fig. 1); "OminiControl w/ text" is the only method using a text encoder. *Row 1* (single mask, "car"): OminiControl with text generates a car with a doll-like head on top, OminiControl without text generates a rock formation, and UNIVL generates a car. *Row 2* (two masks, "face"): both OminiControl variants generate a single doll face, while UNIVL generates a face in each mask (photographic rather than doll-like). *Row 3* (zero-shot, free-form elliptical mask, "logo"): UNIVL generates a round badge that fills the mask, OminiControl with text generates a smaller shield-shaped badge, and OminiControl without text leaves the region blank (more examples in Appendix E.1).

frozen. Training uses condition dropout, which independently drops the entire condition image, the VAE-stream tokens, or the CLIP-stream tokens, so that classifier-free guidance can be applied to each stream at inference (Appendix A.4).

The two-stage pipeline adds *training* cost relative to end-to-end joint fine-tuning, but gives better results (Table 3, R4: FID 13.05 without Stage 1 vs. 11.13). Stage 1 and Stage 2 wall-clock times are reported in Section 5.1. The efficiency results in Table 2 concern *inference* only, where the UNIVL encoder runs once and Stage 1 adds no cost; the training overhead is a one-time cost.

5 Experimental Results

We evaluate UNIVL along five axes: (i) generation quality and alignment against controllable-generation baselines on the UNIVL-ImgGen benchmark (Section 5.2), (ii) multi-region conditioning (Section 5.3), (iii) computational efficiency (Section 5.4), (iv) ablations and generalization to other datasets and backbones (Section 5.5), and (v) the operating range in mask size and label type (Section 5.6).

5.1 Experimental Setup

Training, evaluation and metrics. We train on the UNIVL-ImgGen benchmark (Section 3), which contains 710K mask-text records from four sources: a mask category derived from LAION-5B [28] via Grounding DINO [18] detection and filtering, and three image-editing categories (add, replace, extract) following ImgEdit [40]. The evaluation set has 3,000 samples: 1,500 single-mask samples (375 from each of the four source categories: mask, object addition, replacement, and extraction) and 1,500 multi-mask samples ($N \in [1, 5]$ masks per image). See Appendix F for details. We report metrics in two groups. For image quality, we use FID (lower is better), SSIM, PSNR, and MUSIQ [13] (higher is better) on the full image, and region-SSIM and region-PSNR computed inside the masks. For alignment, we report CLIP_{txt} and CLIP_{img} (global text-image and image-image similarity) and their region-level counterparts region-CLIP_{txt} and region-CLIP_{img}, i.e., the CLIP similarity between the cropped generated region and the text label or the ground-truth crop (higher is better).

Table 1: Comparison on the UNIVL-ImgGen benchmark (single and multiple masks), split by whether the method uses a text encoder. The *w/o text encoder* group (top) is our main comparison; the *w/ text encoder* group (bottom) reports prompted methods for reference. All baselines are fine-tuned on UNIVL-ImgGen. Image quality is measured on the full image (FID, SSIM, PSNR, MUSIQ) and inside the masks (region-SSIM, region-PSNR); outputs are not composited with the source image. Full-image and alignment metrics are means over 5 inference seeds (standard deviations in Table 14). Best per group in **bold**, second-best underlined; –: not computed.

	Image Quality						Alignment			
Method	FID↓	SSIM↑	PSNR↑	MUSIQ↑	region-SSIM↑	region-PSNR↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
Without text encoder (primary focus)										
ControlNet-FLUX	13.98	0.734	18.22	70.50	0.49	17.9	0.250	0.856	0.249	0.689
OminiControl [29]	20.50	0.719	16.45	72.05	0.47	17.7	0.249	0.849	0.271	0.742
UNIVL (ours)	**11.13**	**0.754**	**19.61**	**73.36**	**0.53**	**18.6**	**0.262**	**0.886**	**0.279**	**0.768**
With text encoder (reference)										
BrushNet [12]	18.33	0.611	17.38	**73.21**	0.45	17.5	0.272	0.836	0.247	0.757
PowerPaint [45]	29.4	0.519	14.17	63.6	0.44	16.9	0.266	0.766	0.256	0.699
ControlNet [41]	23.93	0.684	15.30	69.82	0.45	17.2	0.264	0.859	0.282	0.763
OminiControl [29]	13.50	0.718	16.13	72.81	0.47	17.6	0.270	0.871	**0.298**	0.770
ControlNet-FLUX	13.77	0.731	18.02	70.89	0.48	17.8	0.273	0.863	0.252	0.691
FLUX.1 Fill [2]	12.67	0.734	17.23	68.31	0.47	17.5	0.280	0.892	0.277	0.767
UNIVL (ours) + T5 [24]	**10.49**	**0.752**	**19.44**	73.13	**0.52**	**18.4**	**0.295**	**0.904**	0.294	**0.793**

Implementation. All experiments use FLUX.1-dev [2] with OminiControl's inpainting LoRA [29] as the base diffusion model, trained with AdamW in bfloat16. Stage 1 (alignment) takes $\approx$12.6 hours for 30K steps on 2×H100 80GB GPUs. Stage 2 (diffusion fine-tuning, LoRA r=4) takes $\approx$97.3 hours for 40K steps on 6×H100 80GB GPUs. The auxiliary CLIP-image and CLIP-text losses ($\lambda_{\text{clip-img}}$=1.0, $\lambda_{\text{clip-txt}}$=0.8) are applied during Stage 2 and ablated in Appendix D.1. At inference we use 28 denoising steps with classifier-free guidance, which combines the conditional prediction with the prediction under a dropped condition; all guidance scales (full condition, VAE stream, and CLIP stream) are 1.0. At scale 1.0 the guided prediction equals the conditional prediction, so each step needs a single forward pass. Scales from 1.5 to 3.0 slightly raise FID and double the per-step compute (Appendix B, Table 7). The full latent, inside and outside the masks, is denoised and decoded by the VAE; we do not composite the unmasked region from the source image, so full-image metrics also measure how well the background is regenerated. Per-stage hyperparameters and loss details are in Appendix A.1 and Appendix A.2.

Baselines. Our primary baseline is OminiControl [29], a controllable generation method for FLUX that uses token-sequence conditioning for spatial and subject-driven control. We evaluate it (1) with its standard T5 text prompt, as a reference, and (2) without a text prompt, where, like UNIVL, it uses no text encoder. We also compare with ControlNet-FLUX with and without a text prompt, and with the text-prompted methods BrushNet [12], PowerPaint [45], ControlNet [41], and FLUX.1 Fill [2]. All baselines are fine-tuned on the UNIVL-ImgGen training split with the same number of steps (40K), LoRA rank, and optimizer as UNIVL; OminiControl and FLUX.1 Fill use the same flow-matching objective (Eq. 1). Off-the-shelf versions perform worse (e.g., FLUX.1 Fill: FID 18.9 off-the-shelf vs. 12.67 fine-tuned). Because the baselines are not trained to read rendered text, they receive the masked image without the rendered label; prompted baselines receive the text prompts of their original setting, mostly full captions. In the setting without a text encoder, the baselines therefore receive no label.

5.2 Main Results

Table 1 compares UNIVL with all baselines, grouped by whether the method uses a text encoder. The group without a text encoder is our main comparison, since its methods take only an image as input, like UNIVL.

Without a text encoder, UNIVL outperforms both baselines. In the setting without a text encoder (top of Table 1), UNIVL outperforms OminiControl and ControlNet-FLUX on all eight full-image and alignment metrics. The largest gains are in image quality (vs. OminiControl, FID: $20.50 \rightarrow 11.13$, PSNR: $16.45 \rightarrow 19.61$). The gains also hold inside the masks (region-PSNR: $17.7 \rightarrow 18.6$; region-SSIM: $0.47 \rightarrow 0.53$), so they are not explained by background reconstruction alone. Since these baselines receive no label (Section 5.1), the alignment gains mainly show that UNIVL reads the label rendered in the condition. UNIVL and OminiControl build on the same FLUX.1-dev transformer and are trained on the same data; they differ in how the condition is encoded. The gains therefore reflect the UNIVL encoder combined with our training pipeline (Section 5.5 ablates each ingredient). Removing the text encoder also yields the efficiency savings in Section 5.4.

Table 2: Computational cost of UNIVL vs. OminiControl with its text encoder [29]. Replacing the 512 T5 text tokens with a single text-stream token (of FLUX's fixed text-token dimension, 4096) reduces tokens per step by 6–50% and runtime by 8–44%. Encoder parameters shrink by $\sim$92% (4.92B $\rightarrow$ 401.6M). Relative savings are larger at lower resolutions, where the text tokens make up a larger fraction of the DiT sequence. At 512×512, the noisy/image latent has 32×32 tokens. Runtimes are measured on one H100 with 28 steps.

	256×256			512×512			1024×1024		
Metric	OminiControl	UNIVL	Impr.	OminiControl	UNIVL	Impr.	OminiControl	UNIVL	Impr.
Tokens/step	1,024	513	49.9%	2,560	2,049	20.0%	8,704	8,193	5.9%
⌞ noise tokens	256	256	–	1,024	1,024	–	4,096	4,096	–
⌞ image C_I tokens	256	256	–	1,024	1,024	–	4,096	4,096	–
⌞ text C_T tokens	512	1	99.8%	512	1	99.8%	512	1	99.8%
Token size (count×dim)	2,130,688	36,864	98.3%	2,228,992	135,168	93.9%	2,622,208	528,384	79.8%
Params (encoder + transformer)	16.82B	12.30B	26.9%	16.82B	12.30B	26.9%	16.82B	12.30B	26.9%
⌞ encoder params	4.92B	401.6M	91.8%	4.92B	401.6M	91.8%	4.92B	401.6M	91.8%
⌞ transformer params	11.90B	11.90B	0.0%	11.90B	11.90B	0.0%	11.90B	11.90B	0.0%
TFLOPs ($\times1$ enc. + $\times28$ DiT)	792.8	382.1	51.8%	2,121.0	1,648.4	22.3%	9,285.1	8,567.1	7.7%
⌞ encoder FLOPs ($\times1$)	10.3	0.5	95.4%	12.0	1.9	83.9%	19.5	10.3	47.2%
⌞ transformer FLOPs ($\times28$)	782.6	381.6	51.2%	2,109.0	1,646.5	21.9%	9,265.7	8,556.8	7.6%
Runtime (s, $\times1$ enc. + $\times28$ DiT)	1.678	0.936	44.2%	3.620	2.873	20.6%	14.366	13.215	8.0%

Compared with text-prompted baselines, UNIVL has better image quality and a small alignment gap; the optional UNIVL +T5 hybrid narrows the gap. UNIVL without T5 is better than every prompted baseline on all four full-image quality metrics (FID: 11.13 vs. 12.67 for FLUX.1 Fill and 13.50 for OminiControl; PSNR: 19.61 vs. 18.02 for ControlNet-FLUX), and inside the masks it has higher region-PSNR and region-SSIM than OminiControl with text (18.6 vs. 17.6; 0.53 vs. 0.47). Every prompted baseline has higher global text alignment (CLIP_{txt}: 0.262 vs. 0.264–0.280), and OminiControl and ControlNet also have higher region-CLIP_{txt} (0.298 and 0.282 vs. 0.279). This is expected, since they receive the label through a dedicated text encoder. The UNIVL +T5 hybrid (last row) is best on 6 of the 8 full-image and alignment metrics among prompted methods and second on MUSIQ (behind BrushNet) and region-CLIP_{txt} (behind OminiControl). We include UNIVL +T5 only as a reference point against methods that also use a text encoder. Our primary configuration is UNIVL without T5, since adding T5 gives up the efficiency gains in Section 5.4.

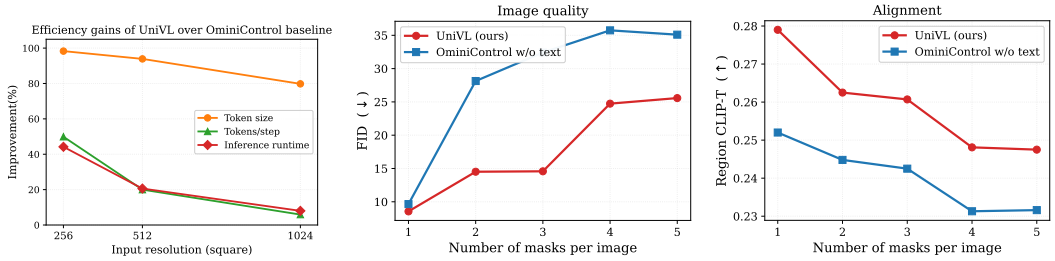
5.3 Multi-Region Composability

In our task, multiple labeled masks are encoded jointly and generated in a single forward pass. The main results in Table 1 include the 1,500 multi-mask test samples ($N\in[1,5]$ masks per image). Fig. 4(b)(c) breaks results down by mask count. UNIVL has lower FID and higher region-CLIP_{txt} than OminiControl without text at every count. The FID gap is 1.1 with one mask and 9.5–17.8 with two to five masks. Both methods degrade as the number of masks grows. Qualitative examples of single-step multi-region edits and sequential multi-step edits are in Appendix E.5 (Fig. 8).

5.4 Computational Efficiency

Removing the text encoder reduces token count, token size, FLOPs, and wall-clock runtime. Table 2 compares UNIVL with OminiControl with its text encoder [29]. Both use the same FLUX.1-dev transformer and differ only in the conditioning encoders: the T5 and CLIP text encoders plus the VAE encoder for OminiControl, and the UNIVL encoder for ours.

Token size drops by 80–98% because T5's 4096-dimensional embeddings dominate the total element count. Encoder parameters drop by $\sim$92% (4.92B $\rightarrow$ 401.6M), a 26.9% reduction in total parameters. Relative TFLOPs and runtime savings shrink as resolution grows, from 52%/44% at 256^2 to 7.7%/8% at 1024^2, because the number of removed text tokens (511) is fixed while the number of image tokens grows with resolution. Most of the TFLOPs saving comes from the shorter DiT sequence rather than from the encoder itself (at 256^2, 401.0 of the 410.7 TFLOPs saved are in the transformer). Fig. 4(a) plots these trends.



(a) Efficiency gains (b) FID vs. # masks (c) region-CLIP$_{\text{txt}}$ vs. # masks

Figure 4: (a) Efficiency gains of UniVL over OminiControl with text encoder [29] across resolutions: reductions in token size, tokens per step, and runtime are largest at 256×256 and smallest at 1024×1024. (b)–(c) Results by number of masks per image for UniVL (red) and OminiControl without text (blue): image quality (FID, lower is better) and region-level alignment (region-CLIP$_{\text{txt}}$, higher is better. no additional text encoder.)

Table 3: Component ablation of the UniVL encoder. Each row (R1–R4) changes one component; the first row (our full model) is from Table 1. Among R1–R4, the best is underlined.

	Image Quality				Alignment			
Configuration	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP$_{\text{txt}}$ ↑	CLIP$_{\text{img}}$ ↑	region-CLIP$_{\text{txt}}$ ↑	region-CLIP$_{\text{img}}$ ↑
UniVL (ours)	**11.13**	**0.754**	**19.61**	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**
(R1) UniVL encoder → VAE encoder	13.60	0.710	18.34	72.46	0.254	0.810	0.261	0.690
(R2) OCR-pretrained CLIP → SigLIP ViT-L	13.13	0.723	18.80	72.64	0.254	0.852	0.271	0.701
(R3) OCR-pretrained CLIP → OpenAI CLIP	14.33	0.701	16.33	70.22	0.259	0.814	0.268	0.687
(R4) UniVL encoder w/o Stage 1 feature align.	13.05	0.698	18.02	73.29	0.257	0.864	0.274	0.763

5.5 Ablations and Analysis

We ablate the key design choices of UniVL. All ablations are evaluated on the UniVL-ImgGen benchmark at 512×512 resolution unless otherwise noted.

(a) Component ablation. We run four ablations (Table 3), each changing one design choice. *(R1)* Replacing the UniVL encoder (VAE + CLIP) with the bare VAE encoder tests whether the CLIP backbone matters. *(R2)* Swapping the OCR-pretrained CLIP backbone for SigLIP (other components unchanged) tests an alternative image-text-aligned backbone. *(R3)* Replacing the OCR-pretrained CLIP with the original OpenAI CLIP ViT-L/14 [23] isolates the contribution of OCR pretraining. *(R4)* Skipping Stage 1 and training the encoder and DiT jointly with the unchanged Stage-2 objective (Eq. 4, which includes $\mathcal{L}_{\text{align}}$) tests the two-stage recipe against end-to-end training.

(R1) **The CLIP backbone helps** (ΔFID $+2.5$): the VAE encoder alone is a weaker conditioning encoder than VAE + CLIP. *(R2)* **Substituting SigLIP yields a smaller gap** (ΔFID $+2.0$). This SigLIP checkpoint is not OCR-pretrained; we did not test an OCR-pretrained SigLIP. *(R3)* **Removing OCR pretraining hurts the most** (ΔFID $+3.2$). R3 cannot fully separate the OCR task from the additional pretraining data. The smaller R2 gap (SigLIP is also not OCR-pretrained) is consistent with a contribution from the OCR task itself, but does not prove it. *(R4)* **Two-stage training outperforms end-to-end training** (ΔFID $+1.9$), indicating that aligning the encoder before training the diffusion model on its output helps.

(b) Other ablations and analysis. Appendix D reports additional ablations: auxiliary CLIP losses (Appendix D.1), CLIP-loss weights, LoRA rank, per-resolution generation quality, 5-seed inference variability, and the diffusion backbone (Appendix D.6). Appendix B reports a classifier-free guidance sweep, and Appendix E evaluates UniVL on free-form masks (best on 5 of 8 metrics; Appendix E.1) and overlapping masks (Appendix E.2).

(c) Generalization beyond UniVL-ImgGen. Zero-shot on COCO val2017 (Appendix E.3), compared with baselines fine-tuned on UniVL-ImgGen and the proprietary gpt-image-1, UniVL has the best FID (16.99 vs. 21.10 for FLUX.1 Fill), SSIM, and PSNR among methods without a text prompt, whereas gpt-image-1 and FLUX.1 Fill have higher region-CLIP scores. Zero-shot on ADE20K (Appendix E.4), UniVL reaches FID 12.01, compared with 11.13 on UniVL-ImgGen; we did not evaluate baselines on ADE20K. With SD3.5-Medium [9] (2.5B parameters) as the backbone and the same recipe, UniVL reaches FID 11.93, compared with 11.13 with FLUX.1-dev and 20.50 for OminiControl without text (Appendix D.6).

Table 4: Operating range of UNIVL (r-: region-level metric). (a) Results by mask area as a fraction of the image, on 252 evaluation samples. (b) Labels outside the training distribution, 30 samples per type; no ground-truth images exist, so only region-CLIP$_{txt}$ is reported. For reference, UNIVL reaches region-CLIP$_{txt}$ 0.279 on UNIVL-ImgGen (Table 1).

(a) Mask area

Mask area	n	r-CLIP$_{txt}$ ↑	r-CLIP$_{img}$ ↑	r-PSNR↑
<0.5%	42	0.154	0.531	10.82
0.5–5%	56	0.294	**0.829**	17.10
5–15%	59	**0.297**	0.804	**19.80**
15–30%	46	0.270	0.763	16.90
≥30%	49	0.254	0.695	13.73

(b) Label type

Label type	Example	r-CLIP$_{txt}$ ↑
Attribute	"red sports car"	**0.258**
Style	"watercolor style dog"	0.151
Negation	"dog not black"	0.163
Compound attribute	"vintage red sports car with open-top soft roofs"	0.136
Relational	"dog under table"	0.123
Multi-clause	–	0.115

5.6 Operating Range: Mask Size and Label Type

Because each label is rendered inside its mask, performance depends on the mask area and on what the label says. Table 4 measures both.

Mask size. Masks covering less than 0.5% of the image (about 36×36 pixels at 512×512) fail: region-CLIP$_{txt}$ drops to 0.154 and region-PSNR to 10.82, compared with at least 0.294 and 17.10 for masks covering 0.5–15%. At this size the rendered label spans only a few pixels (Fig. 9, row 2). Scores also decrease for masks covering 30% or more of the image (region-CLIP$_{img}$ 0.695, region-PSNR 13.73). In this sample, all three metrics are highest for masks covering 0.5–15% of the image.
Label type. Attribute phrases reach region-CLIP$_{txt}$ 0.258, close to the 0.279 that UNIVL reaches on UNIVL-ImgGen. Style, negation, compound-attribute, relational, and multi-clause labels reach 0.115–0.163, close to or below the 0.154 of the failed small-mask bucket. UNIVL therefore supports short object labels with simple attributes, but not style, negation, or relations expressed in the label text. Spatial relations between objects can be specified only by placing separate labeled masks (Section 5.3).

6 Summary

We introduced *spatially grounded contextual image generation*, a controllable generation task that replaces the two-domain conditioning interface (vision + language) with a single visual input in which short text labels are rendered inside their masks. Our framework, UNIVL, replaces the text encoder with an OCR-pretrained vision encoder for short-label, object-level generation. On UNIVL-ImgGen, it improves image quality over text-prompted baselines, on the full image and inside the masks, while reducing token size by up to 98%, encoder parameters by 92%, TFLOPs by up to 52%, and runtime by up to 44%. Zero-shot on COCO, it also has the best FID, SSIM, and PSNR among methods without a text prompt, and the training recipe works with a second DiT backbone (SD3.5-Medium).

Limitations. *Label scope*: UNIVL is designed and evaluated for short, object-level labels; 96% of training labels have 1–3 words (Fig. 10). Simple attribute phrases work, but style, negation, compound-attribute, relational, and multi-clause labels reach region-CLIP$_{txt}$ of only 0.115–0.163 (Table 4), so UNIVL does not replace a text encoder for such prompts. The UNIVL +T5 hybrid accepts free-form prompts, but we have not evaluated it on these prompt types. Spatial relations between objects can be specified only by the placement of separate masks, and the length of text that can be rendered legibly is bounded by the mask area. *Mask size*: below 0.5% of the image area, the rendered label spans only a few pixels and generation fails; scores also decrease for masks above 30% (Table 4). *Mask shape and overlap*: training uses rectangular masks with little overlap; free-form masks are evaluated only zero-shot (Appendix E.1), and almost fully overlapping masks make the condition ambiguous (Appendix E.2). *Pseudo-labels and evaluation data*: the mask category relies on Grounding DINO detections and CLIP filtering, and the editing categories on model-generated edits, so some labels are wrong; in our audit, human annotators gave a mean label-match score of 0.80 (Appendix F.4). Our evaluations use benchmark masks and labels rather than inputs from real users. *Backbones*: we test two DiT backbones (FLUX.1-dev and SD3.5-Medium) but no U-Net-based model. *Standardized text rendering*: our pipeline renders text with a fixed procedure (Courier, white on black, font size scaled to the mask width; Appendix A.5), so the encoder is not trained for arbitrary fonts or styles. This is a design choice: training and inference inputs come from same renderer.

Broader impact. Controllable image generation can support design, accessibility, and creative tools, but it also lowers the cost of misleading or non-consensual image manipulation. We encourage watermarking and provenance tracking for generated content, and any release of our models will follow the license and usage restrictions of the underlying FLUX.1-dev model.

Acknowledgment

We thank Minghao Liu, Md Amirul Islam, Gyuhak Kim, Yuyang Deng and Shanka Subhra Mondal for insightful discussions and feedback on the paper.

References

[1] Inclusion AI. Ming-Omni: A unified multimodal model for perception and generation, 2025. URL `https://arxiv.org/abs/2506.09344`.

[2] Black Forest Labs. FLUX. `https://github.com/black-forest-labs/flux`, 2024.

[3] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.

[4] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[5] Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. TextDiffuser: Diffusion models as text painters. In *NeurIPS*, 2023.

[6] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. AnyDoor: Zero-shot object-level image customization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[7] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-Pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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

[10] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *International Conference on Learning Representations (ICLR)*, 2023.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Xuan Ju, Xian Liu, Xintao Wang, Yuxuan Bian, Ying Shan, and Qiang Xu. BrushNet: A plug-and-play image inpainting model with decomposed dual-branch diffusion. In *European Conference on Computer Vision (ECCV)*, 2024.

[13] 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*, pages 5148–5157, 2021.

[14] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, 2023.

[15] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22511–22521, 2023.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

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

[18] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision (ECCV)*, 2024.

[19] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, 2022.

[20] Jian Ma, Mingjun Zhao, Chen Chen, Ruichen Wang, Di Niu, Haonan Lu, and Xiaodong Lin. GlyphDraw: Seamlessly rendering text with intricate spatial structures in text-to-image generation. *arXiv preprint arXiv:2303.17870*, 2023.

[21] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[22] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[23] 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 *International Conference on Machine Learning (ICML)*, 2021.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[25] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

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

[27] Phillip Rust, Jonas F. Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. Language modelling with pixels. In *International Conference on Learning Representations (ICLR)*, 2023.

[28] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[29] Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. OminiControl: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[30] Zhiyu Tan, Hao Yang, Luozheng Qin, Jia Gong, Mengping Yang, and Hao Li. Omni-Video: Democratizing unified video understanding and generation. *arXiv preprint arXiv:2507.06119*, 2025.

[31] Michael Tschannen, Basil Mustafa, and Neil Houlsby. CLIPPO: Image-and-language understanding from pixels only. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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

[33] Jiayun Wang, Sangryul Jeon, Stella X Yu, Xi Zhang, Himanshu Arora, and Yu Lou. Unsupervised scene sketch to photo synthesis. In *European Conference on Computer Vision*, pages 273–289. Springer, 2022.

[34] Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. General OCR theory: Towards OCR-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*, 2024.

[35] Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

[36] Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR 2: Visual causal flow. *arXiv preprint arXiv:2601.20552*, 2026.

[37] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024.

[38] Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. OmniGen: Unified image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13294–13304, 2025.

[39] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[40] Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. ImgEdit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.

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

[42] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[43] Wentao Zhou, Weijie Gan, and Jiayun Wang. Mutually adversarial self-training with evolving data for unified multimodal models. *arXiv preprint arXiv:2609.36224*, 2026.

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[45] Junhao Zhuang, Yanhong Zeng, Wenran Liu, Chun Yuan, and Kai Chen. A task is worth one word: Learning with task prompts for high-quality versatile image inpainting. In *European Conference on Computer Vision (ECCV)*, 2024.

Appendix

We provide details omitted from the main text in appendix:

- **Section A(Implementation Details):** full training hyperparameters, multi-loss formulas, mask-jittering, condition-dropout schedule, text-rendering details, and multibox training data (extends Section 4).
- **Section B (Inference Details):** denoising and guidance settings, a classifier-free guidance sweep, and a clarification on how CLIP is used for both training and evaluation (extends Section 5.1).
- **Section C (OCR Encoder Choice):** rationale for adapting DeepSeek-OCR's encoder, methodological comparison against alternative OCR methods, and a summary of the underlying OCR-pretraining corpus.
- **Section D (Additional Ablations):** auxiliary CLIP-loss ablation, CLIP-loss-weight sensitivity sweeps, LoRA rank, per-resolution generation quality, the SD3.5-Medium backbone, and 5-seed inference-stability table (extends Section 5.5).
- **Section E (Additional Visualizations and Experiments):** zero-shot free-form masks, overlapping-mask behavior, zero-shot generalization to COCO and ADE20K, multi-step / single-step multibox edits, and failure cases.
- **Section F (UNIVL-ImgGen Benchmark Details):** per-category breakdown, augmentation pipeline, class-name vocabulary statistics, the training/evaluation split, and a human audit of the automatic annotations (extends Section 3).

A Implementation Details

A.1 Training Hyperparameters

Table 5 reports the full training configuration for both stages. Training is performed in bfloat16, with the linear adapter kept in full precision (Stage 1 on 2×H100 80GB GPUs, Stage 2 on 6×H100 80GB GPUs with gradient checkpointing); Stage 1 takes ≈12.6 hours (30K steps) and Stage 2 takes ≈97.3 hours (40K steps). The training pool is the UNIVL-ImgGen benchmark described in Section 3 (∼477K unique images, ∼710K mask-text records); both stages share the same data and 512×512 resolution.

Table 5: Training hyperparameters for the two UNIVL stages.

Hyperparameter	Stage 1 (alignment)	Stage 2 (diffusion)
Base model	FLUX.1-dev + inpainting LoRA (frozen DiT)	FLUX.1-dev + inpainting LoRA (DiT + LoRA)
Trainable params	UNIVL encoder (CLIP-LoRA + linear adapter)	DiT LoRA + UNIVL encoder (continued from Stage 1)
Optimizer	AdamW	AdamW
Learning rate	2×10^{-3}	1×10^{-4}
Weight decay	10^{-3}	10^{-3}
Batch size	24	42
Max steps	30K	40K
Resolution	512×512	512×512
Precision	bfloat16 (adapter: fp32)	bfloat16 (adapter: fp32)
Gradient checkpointing	off	on
λ_{align} (feature alignment)	1.0	1.0
$\lambda_{\text{clip-img}}$ (CLIP image)	0.0 (off)	1.0
$\lambda_{\text{clip-txt}}$ (CLIP text)	0.0 (off)	0.8
λ_{diff} (diffusion MSE)	N/A	1.0
Multibox batch ratio	20%	20%
Condition dropout (image / VAE / CLIP)	0.1 / 0.1 / 0.1	0.1 / 0.1 / 0.1
LoRA rank (r, α)	4, 4 (UNIVL encoder)	4, 4 (DiT) + 4, 4 (UNIVL encoder)

A.2 Multi-Loss Supervision

Alongside the primary objective of each stage ($\mathcal{L}_{\text{align}}$ in Stage 1 and the flow-matching loss $\mathcal{L}_{\text{diff}}$ of Eq. 1 in Stage 2), we apply auxiliary losses on the predicted image $\hat{X} = \mathcal{D}(\hat{z}_0)$, where $\mathcal{D}$ is the VAE decoder and $\hat{z}_0 = z_t - t\, v_\theta(z_t, t, c)$ is the clean-latent estimate from the DiT's velocity prediction. These losses use a separate, frozen OpenAI CLIP ViT-L/14 [23], distinct from the CLIP backbone inside the UNIVL encoder; the same frozen model computes the CLIP-based evaluation metrics. Stage 1 uses only $\mathcal{L}_{\text{align}}$. In Stage 2, both CLIP losses ($\lambda_{\text{clip-img}}$=1.0, $\lambda_{\text{clip-txt}}$=0.8) and the

feature-alignment loss are active alongside $\mathcal{L}_{\text{diff}}$. Let ϕ_I and ϕ_T denote the CLIP image and text encoders, and let $\hat{X}_m$ and X_m denote the masked regions of $\hat{X}$ and of the target image X:

1. **Feature alignment** ($\mathcal{L}_{\text{align}}$, λ=1.0): per-token distance between the UNIVL embedding and the target VAE latent, $\|f_{\text{VL}} - z_0\|_2^2$ where $z_0 = \mathcal{E}(X)$.
2. **CLIP image loss** ($\mathcal{L}_{\text{clip-img}}$, λ=1.0): $1 - \cos\big(\phi_I(\hat{X}_m),\, \phi_I(X_m)\big)$, the cosine distance between the CLIP image embeddings of the generated and target regions.
3. **CLIP text loss** ($\mathcal{L}_{\text{clip-txt}}$, λ=0.8): $1 - \cos\big(\phi_I(\hat{X}_m),\, \phi_T(\ell)\big)$, the cosine distance between the CLIP image embedding of the generated region and the CLIP text embedding of its label ℓ.

Feature alignment supervises a per-token reconstruction of the specific target, whereas the CLIP text loss supervises only the category of the masked region.

A.3 Mask Jittering

During training, bounding boxes are randomly perturbed:

- Scale range: $(0.85, 1.5)\times$ centered on the bounding box.
- Translation: up to 5% of bounding box size in each direction.
- Applied to both text rendering coordinates and the normalized bounding box mask.

The goal is robustness to variations in mask size at inference; Table 4(a) reports how performance varies with mask area.

A.4 Condition Dropout

Three independent dropout mechanisms (each with probability p=0.1) enable classifier-free guidance at inference:

Table 6: Condition dropout modes during training.

Dropout mode	What is zeroed	Inference use
Full image dropout	Entire condition image	`guidance_scale`
VAE dropout	VAE tokens (keep CLIP)	`image_guidance_scale`
CLIP dropout	CLIP tokens (keep VAE)	`clip_guidance_scale`

CLIP dropout trains the model both with and without the CLIP-stream semantic signal, which enables classifier-free guidance on that stream at inference. All experiments in the paper use guidance scale 1.0 for every stream, at which the guided prediction equals the conditional prediction.

A.5 Text Rendering on Mask

Text rendering is fully automated and standardized: the user provides only the textual instruction, and our pipeline renders it with a fixed font (Courier), white-on-black, with the font size scaled so the rendered text spans the mask width (shrink until text_width $\approx$ bbox_width). The label is rendered once, as a single line. Because training and inference use the same renderer, the UNIVL encoder does not need to generalize across fonts or styles.

A.6 Multibox Training Data

20% of training batches use multibox data: images with 2–5 non-overlapping masked regions, each with a different class label. Mask sampling rules:

- Each mask area: 4%–40% of image area.
- Pairwise IoU < 0.05 between all masks.
- If the non-overlapping constraint fails after 5 retries: fall back to 2 masks, then 1.

B Inference Details

At inference, we use 28 denoising steps with `guidance_scale=1.0`, `image_guidance_scale=1.0`, and `clip_guidance_scale=1.0`. These are the classifier-free guidance scales for the full condition, the VAE stream, and the CLIP stream; each combines the conditional prediction with the prediction under the corresponding dropped condition (Table 6). At 1.0 the guided prediction equals the conditional prediction, so each step needs a single forward pass. Classifier-free guidance applies to any conditioning modality, and the condition dropout of Appendix A.4 trains the model for it. We therefore swept the image and CLIP guidance scales from 1.5 to 3.0 (Table 7). Larger scales raise FID by 0.06–0.29, which is within one standard deviation of the 5-seed FID (0.33; Table 14), and change region-CLIP_{img} by at most 0.006. Since each doubles the per-step compute, all reported experiments use 1.0.

Table 7: Classifier-free guidance sweep on the UNIVL-ImgGen benchmark. Any scale above 1.0 requires two forward passes per denoising step.

Image scale	CLIP scale	FID↓	region-CLIP_{img} ↑	Forward passes / step
1.0 (ours)	**1.0 (ours)**	**11.13**	0.768	1
1.5	1.0	11.24	0.767	2
2.0	1.0	11.31	0.765	2
3.0	1.0	11.42	0.762	2
1.0	1.5	11.19	**0.769**	2
1.0	2.0	11.28	0.766	2
1.0	3.0	11.41	0.762	2

Metrics clarification. We use CLIP in three places: dataset filtering (Section 3), auxiliary training losses ($\mathcal{L}_{\text{clip-img}}$, $\mathcal{L}_{\text{clip-txt}}$), and evaluation metrics (CLIP_{txt}, CLIP_{img}, region-CLIP_{txt}, region-CLIP_{img}). All three use the original OpenAI CLIP ViT-L/14 [23], kept frozen and separate from the CLIP block inside the UNIVL encoder, and the metrics apply it identically to every method. Because the loss CLIP and the metric CLIP are the same model, and the CLIP losses optimize the same image–image and image–text similarities that the CLIP metrics measure, CLIP-based metrics could favor UNIVL. We therefore also report four metrics that do not use CLIP (FID, SSIM, PSNR, MUSIQ), and UNIVL leads on these as well (Table 1).

C OCR Encoder Choice

We provide additional rationale for using the DeepSeek-OCR encoder [35] as the backbone of UNIVL, and discuss why we did not adopt other OCR-capable vision models.

Why DeepSeek-OCR. The encoder of DeepSeek-OCR (called *DeepEncoder*) combines a SAM-base block with window attention and a CLIP ViT-L block with global attention, connected by a 16× convolutional token compressor [35]. We use it for two reasons: (i) its CLIP block produces image-text-aligned semantic features, which we need in order to read rendered labels and bind them to locations; and (ii) it has been pretrained on a large and diverse OCR corpus (∼30M PDF pages, scene OCR, and general vision data), so it already reads text rendered as pixels. As described in Section 4.2, we replace SAM with the FLUX VAE encoder as the low-level stream, because VAE features lie in the latent space of the VAE decoder. To reuse the pretrained CLIP weights unchanged, the CLIP backbone keeps its SAM stem: CLIP reads the SAM features, and its output is concatenated with them, as in DeepSeek-OCR. The other components of DeepSeek-OCR (its projector, multiple resolution modes, and the DeepSeek-3B-MoE decoder) are not used. DeepSeek-OCR 2 [36] is a successor with a redesigned encoder. We expect the conditioning approach of UNIVL to apply to other OCR-pretrained encoders, but have not tested this.

Summary of OCR encoder pretraining. For completeness, we summarize the data used to pretrain DeepSeek-OCR, whose encoder weights we use as initialization before our feature-alignment stage; the original training was performed by Wei et al. [35]. The training mix of the full DeepSeek-OCR

model is roughly *70% OCR / 20% general vision / 10% text-only*. The OCR portion comprises two regimes:

- **Document OCR**: $\sim$30M PDF pages spanning $\sim$100 languages ($\sim$25M in Chinese and English and $\sim$5M in other languages), labeled with both *coarse* text extracted via `fitz` and *fine* layout-and-text annotations (2M Chinese + 2M English pages from PP-DocLayout/MinerU/GOT-OCR2.0; $\sim$600K minority-language samples). An additional $\sim$3M Word documents cover formulas and HTML tables without layout.
- **Natural-scene OCR** ($\sim$20M images): 10M Chinese + 10M English from LAION and Wukong, labeled with PaddleOCR.

The 20% general-vision data (captioning, detection, grounding) and the 10% text-only data serve the full DeepSeek-OCR model, including its language decoder; DeepEncoder itself is pretrained on the OCR data together with 100M general images sampled from LAION [35]. We adapt the resulting encoder to generation conditioning in Stage 1 (Section 4).

Why not other OCR models. We considered two other public OCR-capable models:

- **Nougat** [3]: an end-to-end OCR model for academic documents, trained on a narrower domain (scientific PDFs) than DeepSeek-OCR. We expect this narrower domain to transfer less well to labels rendered on natural images.
- **GOT-OCR2.0** [34]: an end-to-end OCR model covering documents, charts, and formulas, with a scope comparable to DeepSeek-OCR.

We chose DeepSeek-OCR mainly for the scale and diversity of its pretraining data. This choice is not ablated in the paper.

D Additional Ablations

We report additional ablations that complement Section 5.5 in the main paper. All experiments use the default configuration (resolution 512×512, LoRA $r=4$, $\lambda_{\text{clip-img}}=1.0$, $\lambda_{\text{clip-txt}}=0.8$, 40K Stage-2 training steps) unless otherwise noted, and the reference row ("ours") uses our final model as reported in Table 1.

D.1 Auxiliary CLIP Loss

In stage 2, we study the effect of adding CLIP-based auxiliary losses on top of the feature alignment objective. We compare four configurations: feature alignment only ($\mathcal{L}_{\text{align}}$), adding a CLIP image-space loss ($\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{clip-img}}$), adding only a CLIP text loss ($\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{clip-txt}}$), and adding both ($\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{clip-img}} + \mathcal{L}_{\text{clip-txt}}$, ours). Results on the UNIVL-ImgGen benchmark are shown in Table 8.

Table 8: Auxiliary CLIP loss ablation on top of the feature alignment objective. The full configuration ($\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{clip-img}} + \mathcal{L}_{\text{clip-txt}}$, ours) leads on all 8 metrics. Best in **bold**, second-best underlined.

	Image Quality				Alignment			
Configuration	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
No CLIP loss ($\mathcal{L}_{\text{align}}$ only)	28.11	0.665	17.97	69.36	<u>0.256</u>	0.791	<u>0.277</u>	0.709
$+\mathcal{L}_{\text{clip-img}}$	21.77	0.663	17.12	<u>73.03</u>	0.251	0.840	0.273	0.746
$+\mathcal{L}_{\text{clip-txt}}$	<u>12.72</u>	<u>0.724</u>	<u>18.80</u>	72.61	0.253	<u>0.876</u>	<u>0.277</u>	<u>0.751</u>
$+\mathcal{L}_{\text{clip-img}} + \mathcal{L}_{\text{clip-txt}}$ **(ours)**	**11.13**	**0.754**	**19.61**	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**

Each CLIP loss improves FID over feature alignment alone, and combining both is best on all metrics. Adding $\mathcal{L}_{\text{clip-img}}$ lowers FID by 6.3 and raises MUSIQ by 3.7 and region-CLIP_{img} by 0.04 over no CLIP loss, but slightly lowers SSIM, PSNR, and the two text-alignment metrics. Adding $\mathcal{L}_{\text{clip-txt}}$ has a larger effect: FID drops by 15.4 (to 12.72), and SSIM, PSNR, CLIP_{img}, and region-CLIP_{img} all improve.

D.2 CLIP Image Loss Weight

We complement Appendix D.1 (which studies the *presence* of the auxiliary CLIP losses) with a sensitivity analysis on the weight of the CLIP image loss, $\lambda_{\text{clip-img}}$, holding the CLIP text loss weight fixed at $\lambda_{\text{clip-txt}}=0.8$.

Table 9: Sensitivity to the CLIP image loss weight $\lambda_{\text{clip-img}}$. Best in **bold**.

	Image Quality				Alignment			
$\lambda_{\text{clip-img}}$	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
0.1	13.81	0.732	19.42	72.64	0.238	0.851	0.255	0.736
1.0	**11.13**	**0.754**	**19.61**	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**
2.0	15.72	0.665	17.53	**73.36**	0.255	0.854	0.274	0.756

Both a smaller (0.1) and a larger (2.0) weight are worse than $\lambda_{\text{clip-img}}=1.0$ on every metric except MUSIQ, where 2.0 ties. The larger weight mainly hurts pixel-level fidelity (SSIM $0.754 \rightarrow 0.665$, PSNR $19.61 \rightarrow 17.53$), while the smaller weight mainly hurts alignment (CLIP_{txt} $0.262 \rightarrow 0.238$).

D.3 CLIP Text Loss Weight

We additionally sweep the CLIP text loss weight $\lambda_{\text{clip-txt}}$ across three values, holding $\lambda_{\text{clip-img}}=1.0$ fixed.

Table 10: Sensitivity to the CLIP text loss weight $\lambda_{\text{clip-txt}}$. Best in **bold**.

	Image Quality				Alignment			
$\lambda_{\text{clip-txt}}$	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
0.1	14.43	0.737	**19.78**	71.75	0.246	0.868	0.261	0.748
0.8 (ours)	**11.13**	**0.754**	19.61	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**
2.0	14.28	0.739	19.74	71.89	0.242	0.859	0.262	0.745

Both 0.1 and 2.0 are worse than $\lambda_{\text{clip-txt}}=0.8$ on FID, SSIM, MUSIQ, and all alignment metrics, though slightly better on PSNR (19.78 and 19.74 vs. 19.61).

D.4 LoRA Rank

We compare LoRA ranks $r=4$ and $r=8$ for the DiT fine-tuning stage (Stage 2).

Table 11: LoRA rank ablation. Doubling rank from 4 to 8 doubles trainable parameters without improvement across any metric. Best in **bold**.

	Params	Image Quality				Alignment			
LoRA rank	(rel.)	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
$r=4$ **(ours)**	1×	**11.13**	**0.754**	**19.61**	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**
$r=8$	2×	15.77	0.669	17.62	73.20	0.255	0.855	0.275	0.754

$r=8$ is worse than $r=4$ on every metric (e.g., FID 15.77 vs. 11.13), so a larger DiT LoRA does not help in our setting.

D.5 Generation Quality Across Resolutions

Table 2 (main paper) quantifies the efficiency gains of UniVL at 256×256, 512×512, and 1024×1024. To check generation quality at each of these resolutions, we train separate UniVL models at each resolution and report their generation metrics in Table 12. These rows are *not* directly comparable, since FID, PSNR, SSIM, MUSIQ, and CLIP-based metrics all depend on image resolution. The table is a per-resolution sanity check, *not* a cross-resolution comparison.

The 512^2 and 1024^2 models give similar metrics. The 256^2 model is lower on most metrics, notably MUSIQ (54.66 vs. 73.36 at 512^2). The largest efficiency gains in Table 2 (at 256×256) therefore come with lower absolute image quality than at 512×512. We did not evaluate baselines at 256^2 or 1024^2, so this table does not compare methods at those resolutions.

Table 12: Per-resolution generation quality for UNIVL. Each row is evaluated at its own resolution; metrics are not directly comparable across rows (FID, PSNR, and SSIM all depend on input resolution). The purpose of this table is to validate that UNIVL maintains reasonable generation quality at every scale reported in the efficiency analysis (Table 2).

	Image Quality				Alignment			
Resolution	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
256×256	17.55	0.632	17.15	54.66	0.254	0.830	0.277	0.761
512×512 (default)	11.13	0.754	19.61	73.36	0.262	0.886	0.279	0.768
1024×1024	10.87	0.749	20.57	73.83	0.254	0.855	0.271	0.757

D.6 Diffusion Backbone: SD3.5-Medium

To test whether the recipe depends on FLUX.1-dev, we retrain UNIVL with Stable Diffusion 3.5 Medium [9] (`stabilityai/stable-diffusion-3.5-medium`, 2.5B parameters, about $5\times$ smaller than the 12B FLUX.1-dev transformer) as the diffusion backbone. Only the backbone changes; the two training stages are the same as for FLUX.1-dev, with the same UNIVL encoder design, LoRA rank and α for the backbone and the encoder, optimizer, learning-rate schedule, and step counts (30K for Stage 1, 40K for Stage 2). The SD3.5 VAE replaces the FLUX VAE, both inside the UNIVL encoder and as the Stage-1 alignment target. Table 13 reports the results on UNIVL-ImgGen.

Table 13: UNIVL with two diffusion backbones on the UNIVL-ImgGen benchmark.

	Image Quality				Alignment			
Backbone	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
FLUX.1-dev (12B, default)	**11.13**	**0.754**	**19.61**	**73.36**	**0.262**	**0.886**	**0.279**	**0.768**
SD3.5-Medium (2.5B)	11.93	0.714	19.03	71.32	0.251	0.879	0.271	0.716

With SD3.5-Medium, all eight metrics are worse than with FLUX.1-dev, e.g., FID $+0.8$ and region-CLIP_{img} -0.05. The SD3.5-Medium model still has lower FID than OminiControl without text on FLUX.1-dev (20.50; Table 1). Both backbones are DiTs; we have not tested a U-Net backbone.

D.7 Robustness to Inference Seed

To assess statistical significance of the headline comparison in Table 1, we re-run inference 5 times with different seeds (using the same trained checkpoint and the same 3,000-sample evaluation set) and report the per-cell standard deviations alongside the means in Table 14. The dispersion is small ($\sigma_{\text{FID}} \leq 0.53$ across all rows). UNIVL's margins over the baselines on FID, SSIM, PSNR, and MUSIQ are larger than the standard deviations. Some alignment differences between methods fall within one standard deviation (e.g., region-CLIP_{img} of OminiControl w/ text, 0.770 ± 0.031, vs. UNIVL +T5, 0.793 ± 0.016). These deviations reflect inference randomness only, not training variance.

Table 14: Mean ± standard deviation across 5 inference seeds on the UNIVL-ImgGen benchmark. Means are identical to Table 1 (reproduced for convenience).

	Image Quality				Alignment			
Method	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
Without text encoder								
ControlNet-FLUX	13.98±0.40	0.734±0.003	18.22±0.14	70.50±0.08	0.250±0.005	0.856±0.005	0.249±0.009	0.689±0.011
OminiControl [29]	20.50±0.36	0.719±0.003	16.45±0.19	72.05±0.15	0.249±0.005	0.849±0.013	0.271±0.006	0.742±0.021
UNIVL (ours)	11.13±0.33	0.754±0.002	19.61±0.15	73.36±0.09	0.262±0.001	0.886±0.010	0.279±0.006	0.768±0.017
With text encoder								
ControlNet [41]	23.93±0.53	0.684±0.003	15.30±0.25	69.82±0.13	0.264±0.007	0.859±0.011	0.282±0.010	0.763±0.021
OminiControl [29]	13.50±0.42	0.718±0.003	16.13±0.20	72.81±0.11	0.270±0.005	0.871±0.014	0.298±0.009	0.770±0.031
ControlNet-FLUX	13.77±0.31	0.731±0.002	18.02±0.19	70.89±0.07	0.273±0.004	0.863±0.009	0.252±0.011	0.691±0.021
FLUX.1 Fill	12.67±0.36	0.734±0.002	17.23±0.11	68.31±0.08	0.280±0.004	0.892±0.003	0.277±0.007	0.767±0.010
UNIVL (ours) + T5	10.49±0.41	0.752±0.003	19.44±0.16	73.13±0.11	0.295±0.002	0.904±0.009	0.294±0.008	0.793±0.016

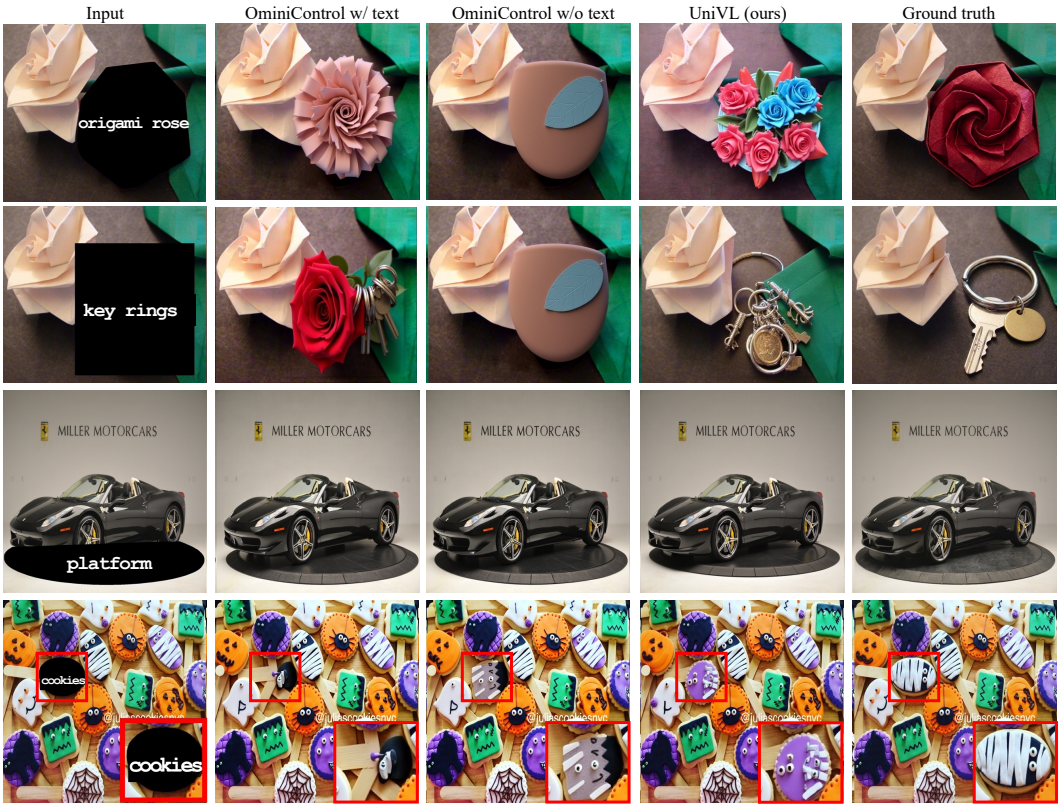


Figure 5: Zero-shot results on free-form masks (UNIVL is trained only on rectangular masks). *Row 1* ("origami rose", polygonal mask): UNIVL fills the mask with a bouquet of roses (not origami), OminiControl with text generates a paper flower, and OminiControl without text generates an unrelated object. *Row 2* ("key rings", rectangular mask, for reference): UNIVL generates key rings, while OminiControl with text adds a rose next to keys. *Row 3* ("platform", elliptical mask): all methods generate a platform. *Row 4* ("cookies", round mask): UNIVL generates a round cookie inside the mask, while OminiControl without text fills the region with a more rectangular cookie.

E Additional Visualizations and Experiments

E.1 Non-Square (Free-Form) Masks

UNIVL is trained only on axis-aligned rectangular masks, but the task formulation does not restrict mask shape: the contextual condition C_I is an image with a region masked and a text label rendered inside it. We test UNIVL on free-form masks at inference time *without retraining or fine-tuning* (zero-shot evaluation).

In these examples, UNIVL reads labels rendered inside non-rectangular regions, and in rows 3–4 its output follows the mask outline rather than the bounding rectangle. Row 1 shows that the label can still be followed only loosely (roses rather than origami roses).

Quantitative evaluation on free-form masks. We evaluate UNIVL and both OminiControl variants on a held-out free-form mask test set (irregular polygonal masks; same image distribution and prompts as the in-domain set). Table 15 reports the metrics.

UNIVL is best on 5 of 8 metrics and second on the two text-alignment metrics, so the ranking from the rectangular benchmark (Table 1) largely holds without training on free-form masks; OminiControl is better on MUSIQ. The absolute image-quality numbers are higher than on the rectangular benchmark (e.g., FID 8.64 vs. 11.13), but the two test sets differ, so these numbers are not directly comparable. One possible factor is that a free-form mask covers less area than its bounding rectangle, so a larger fraction of the pixels lies outside the mask, where the target equals the source image.

Table 15: Zero-shot quantitative results on free-form (non-rectangular) masks. UNIVL is best on 5/8 metrics (FID, SSIM, PSNR, CLIP_{img}, region-CLIP_{img}) and second-best on the two text-alignment metrics. Best in **bold**, second-best underlined.

	Image Quality				Alignment			
Method	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
UNIVL (ours)	**8.64**	**0.8376**	**23.34**	72.82	0.2619	**0.9486**	0.2872	**0.8177**
OminiControl w/o text	10.44	0.7833	21.52	**73.20**	0.2599	0.9298	0.2867	0.7883
OminiControl w/ text	9.97	0.7838	21.21	73.08	**0.2686**	0.9375	**0.2904**	0.8018

E.2 Overlapping Masks

We test UNIVL on multiple labeled masks that overlap, a setting not seen in training (training masks have pairwise IoU < 0.05; Appendix A.6). Fig. 6 shows three cases. When masks overlap partially (rows 1 and 3), UNIVL generates the labeled content in each region in these examples. When masks overlap almost completely (row 2), two labels are rendered into nearly the same pixels and the condition becomes ambiguous. Resolving this would require extra cues, such as an explicit ordering of masks, that the current interface does not provide.

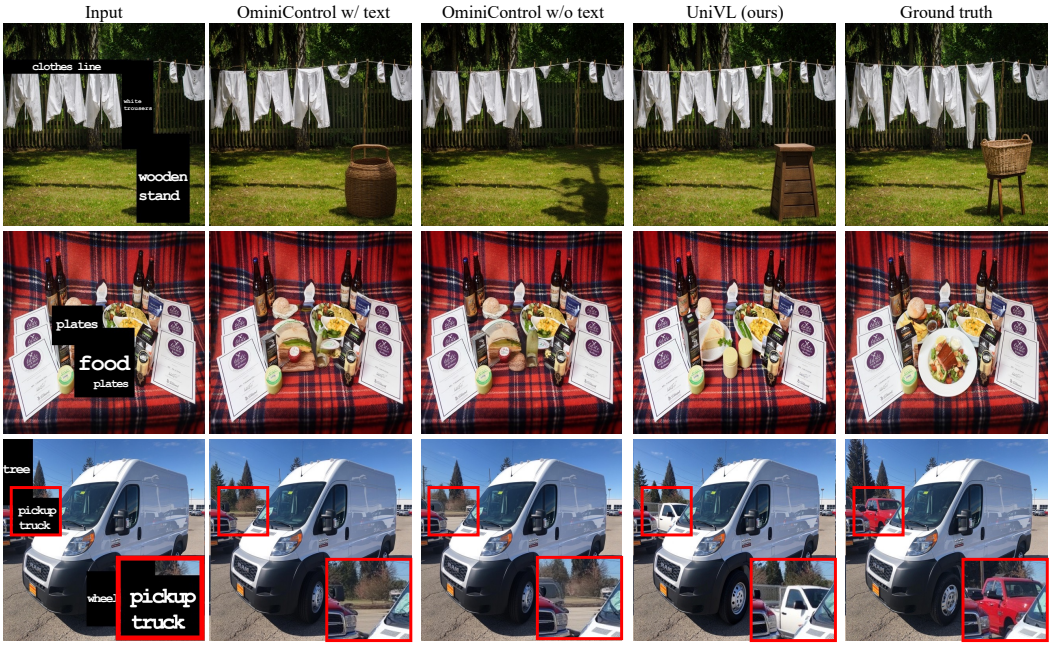


Figure 6: Overlapping masks. *Row 1:* UNIVL works when two masks do not overlap too much—each labeled region is generated as instructed. *Row 2 (failure case):* when the "food" and "plate" boxes totally overlay, UNIVL cannot disambiguate which label belongs to which region and produces a single fused object. *Row 3:* UNIVL generates a pickup truck in the requested region, while both OminiControl variants fill it with background (trees).

E.3 Results on COCO

To assess generalization beyond the UNIVL-ImgGen training distribution, we evaluate UNIVL zero-shot on COCO val2017 [16]. We randomly sample 3,000 images, use ground-truth objects and their class names as the spatial-text condition, with either the bounding box or the convex hull of the segmentation mask as the mask, and run UNIVL with no fine-tuning. Table 16 compares UNIVL with OminiControl, ControlNet-FLUX, and FLUX.1 Fill, all FLUX.1-dev-based and fine-tuned on UNIVL-ImgGen, and with the proprietary gpt-image-1 model (OpenAI; not fine-tuned), each with and without a text prompt. As on UNIVL-ImgGen, the baselines receive the masked image without the rendered label. Fig. 7 shows representative qualitative results.

Table 16: Zero-shot results on COCO val2017 (3,000 randomly sampled images; no fine-tuning on COCO). Best per group in **bold**, second-best underlined.

	Image Quality				Alignment			
Method	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
Without text prompt								
UniVL (ours)	**16.99**	**0.759**	**20.69**	71.03	**0.264**	**0.879**	0.271	0.731
OminiControl [29]	21.37	0.711	19.50	**71.06**	0.261	0.858	0.264	0.713
ControlNet-FLUX	29.12	0.729	19.43	66.92	0.251	0.821	0.261	0.619
FLUX.1 Fill [2]	21.10	0.717	20.42	67.82	0.255	0.878	0.274	0.739
gpt-image-1	35.86	0.431	15.04	66.55	0.256	0.832	**0.284**	**0.755**
With text prompt (reference)								
UniVL (ours) + T5	**15.61**	**0.736**	**20.23**	71.32	0.267	0.875	0.299	0.744
OminiControl [29]	17.83	0.714	19.11	**71.73**	0.281	0.868	0.281	0.732
ControlNet-FLUX	27.40	0.721	18.51	67.54	0.277	0.821	0.292	0.634
FLUX.1 Fill [2]	16.73	0.717	20.21	68.67	0.277	**0.891**	0.304	0.765
gpt-image-1	33.56	0.421	14.59	69.01	**0.285**	0.843	**0.313**	**0.787**

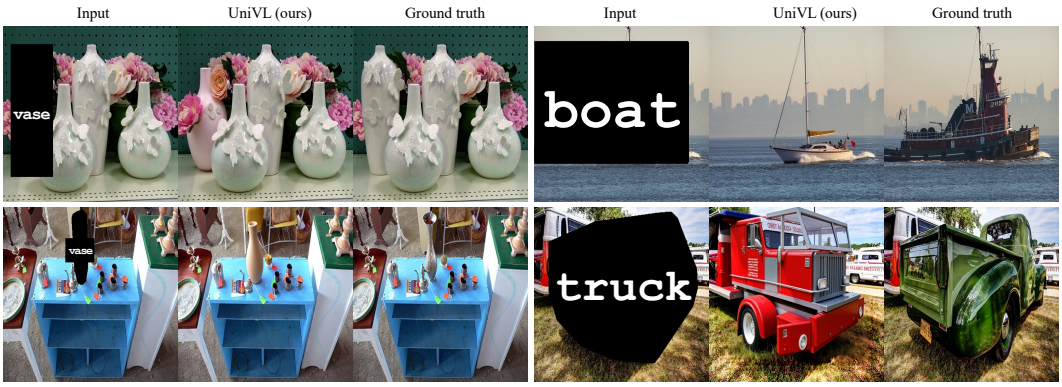


Figure 7: Examples of zero-shot UniVL on COCO val2017 (no fine-tuning on COCO).

Without a text prompt, UniVL is best on 5 of 8 metrics (FID, SSIM, PSNR, CLIP_{txt}, CLIP_{img}); its FID is 4.1 lower than the next-best method (FLUX.1 Fill). OminiControl is best on MUSIQ (71.06 vs. 71.03), and gpt-image-1 is best on both region-CLIP metrics but worst on FID, SSIM, and PSNR (35.86, 0.431, 15.04), i.e., its outputs deviate most from the ground-truth images. With a text prompt, UniVL +T5 is best on FID, SSIM, and PSNR, while the baselines lead on MUSIQ and all four alignment metrics. Against OminiControl without text, UniVL leads on all metrics except MUSIQ, which matches the pattern on UniVL-ImgGen (Table 1).

E.4 Results on ADE20K

We also evaluate UniVL zero-shot on the 2,000 ADE20K [42] validation images, with one annotated object per image as the labeled mask. Table 17 compares the result with the in-domain result on UniVL-ImgGen. We did not run baselines on ADE20K, so this experiment measures the transfer of UniVL itself rather than its ranking against other methods.

Table 17: Zero-shot results of UniVL on ADE20K (2,000 validation images, one mask per image), with the in-domain result on UniVL-ImgGen for reference.

	Image Quality				Alignment			
Evaluation set	FID↓	SSIM↑	PSNR↑	MUSIQ↑	CLIP_{txt} ↑	CLIP_{img} ↑	region-CLIP_{txt} ↑	region-CLIP_{img} ↑
UniVL-ImgGen (in-domain)	11.13	0.754	19.61	73.36	0.262	0.886	0.279	0.768
ADE20K (zero-shot)	12.01	0.721	17.07	68.79	0.254	0.863	0.274	0.748

On ADE20K, FID rises by 0.9, region-CLIP_{img} drops by 0.02, and PSNR drops from 19.61 to 17.07 relative to UniVL-ImgGen.

E.5 Multi-Step and Single-Step Multibox Edits

UNIVL handles two complementary multi-region workflows in a single architecture: (i) *sequential multi-step edits*, where the user applies one mask at a time and the model's output from a previous step becomes the source image for the next; and (ii) *single-step multibox edits*, where multiple labeled masks are placed on the same image and processed jointly in one forward pass. Fig. 8 demonstrates both modes on the same source.



Figure 8: Multi-region edit modes. The source image is shown at the bottom-left. *Columns 2–3 (multi-step edit):* the user applies a "pillow" mask (column 2) and then, on top of that result, a "dresser" mask (column 3); each step is a separate UNIVL forward pass. *Column 4 (single-step multibox edit):* both "pillow" and "dresser" masks are applied jointly in a single forward pass. UNIVL handles both modes without architectural changes, producing spatially accurate edits while preserving the rest of the scene.

E.6 Failure Cases

Fig. 9 illustrates two failure modes. For the very small mask, the rendered text has too few pixels to be read; Table 4(a) shows that masks below 0.5% of the image fail in general (region-CLIP$_{\text{txt}}$ 0.154). For the uncommon phrase, one possible cause is limited coverage of the category in the training data.

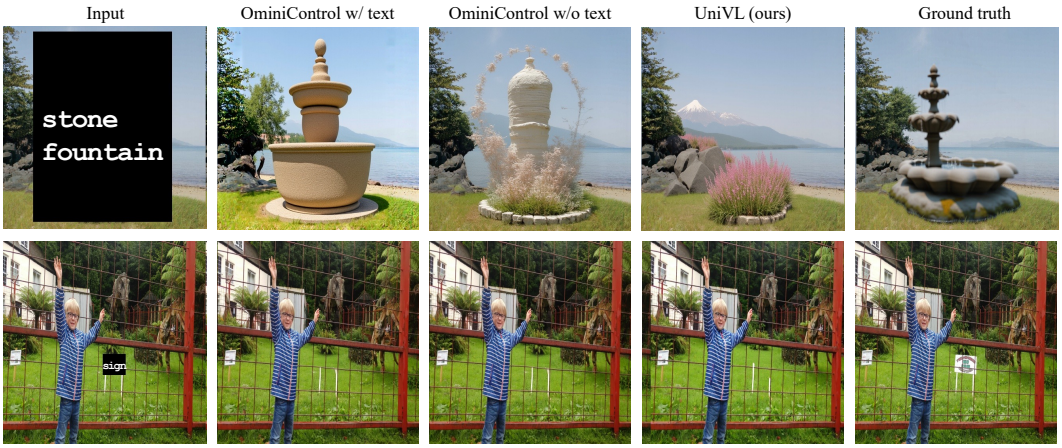


Figure 9: Failure cases of UNIVL. *Row 1, uncommon phrase* ("stone fountain"): UNIVL generates rocks and a flower bed instead of a fountain, while OminiControl with text generates a fountain-like object. *Row 2, very small mask* ("sign"): the rendered text covers only a few pixels, and UNIVL, like both baselines, leaves the region close to the background.

F UNIVL-ImgGen Benchmark Details

F.1 Dataset categories and sources

UNIVL-ImgGen comprises four source categories: the mask category provides images with many labeled boxes for spatial localization, and the three editing categories provide label-dependent edited targets, which discourage the model from reconstructing the image condition and ignoring the rendered text. Table 18 summarizes the per-category statistics.

Table 18: UNIVL-ImgGen breakdown by source category. The *mask* category provides multibox spatial training; the three editing categories provide label-dependent edited targets. *Boxes* are the labeled masks before augmentation; *records* additionally count the augmented instances of the mask category. The 3,000 evaluation samples are drawn from these records (Appendix F.3).

Category	Unique images	Boxes	Boxes / image	Records
mask	4,675	56,180	12.02	238,425
add	122,594	122,594	1.00	122,594
replace	217,606	217,606	1.00	217,606
extract	131,996	131,996	1.00	131,996
Overall	476,871	528,376	1.11	710,621

(1) *mask* category (multibox spatial training). Following the construction sketched in Section 3, we start from LAION-5B [28] images, apply Grounding DINO [18] to detect objects and obtain bounding boxes with class labels, and apply two filters: (i) a *size filter* that discards detections whose mask area is less than 4% of the total image area; and (ii) a *semantic filter* that retains pairs with normalized CLIP similarity >0.85 between the cropped mask region and the class name. For each retained sample, the contextual condition is constructed by masking the detected region and rendering the class label inside it (Appendix A.5). After filtering, the mask category has 4,675 images with 56,180 boxes (12.02 per image), which we exploit through an *augmentation* step: random crop and color jittering, with each augmented instance retaining 2–10 boxes. Augmentation adds 233,750 instances, so the mask category contributes 238,425 records and the benchmark has 710,621 records in total (Table 18).

(2–4) *add*, *replace*, and *extract* categories (image editing). These three categories are taken directly from the released ImgEdit dataset [40], without changes. In ImgEdit's pipeline, captions and edit instructions are written by GPT-4o, and the edited images are produced with FLUX- and SDXL-based generators:

- **Add** (122,594 pairs): inserting a new object into a designated region.
- **Replace** (217,606 pairs): editing the masked region with a new text-described object (text-prompt-driven replacement).
- **Extract** (131,996 pairs): an image-reference variant in which the masked region is filled by transferring content from a reference image; following ImgEdit [40], the class label of the rendered text in this category is the reference image's label.

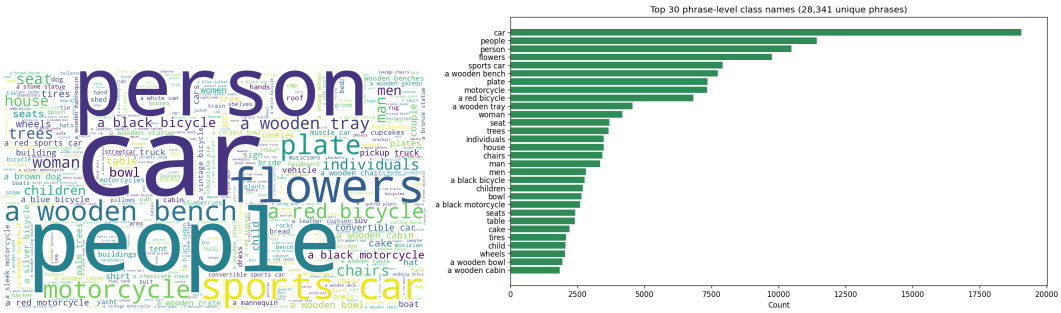
Each editing-category record corresponds to a single image with one labeled mask. Because the edited target depends on the label, these records encourage the model to use the rendered text rather than reconstruct the image input.

F.2 Class-name vocabulary statistics

The class labels rendered onto masks span a long-tail vocabulary of natural-image object descriptions. Table 19 reports the distribution of label phrase length (in words) over the 528,376 labeled boxes (before augmentation; Table 18). Most labels are short (1–3 words: 96.0%); 4.0% have four or more words. There are 28,341 unique phrases. Fig. 10 visualizes the 30 most frequent class-name phrases.

Table 19: Phrase-length distribution of class-name labels in UNIVL-ImgGen. Counts are over the 528,376 labeled boxes before augmentation. Most labels are short (1–3 words: 96.0%), but the benchmark also includes longer multi-word phrases.

# words	Labels	% labels	Unique phrases	% unique
1	270,607	51.21%	3,221	11.37%
2	89,675	16.97%	9,687	34.18%
3	146,959	27.81%	10,778	38.03%
4	19,486	3.69%	3,937	13.89%
5	1,253	0.24%	536	1.89%
6	327	0.06%	149	0.53%
7	47	0.01%	23	0.08%
8	13	0.00%	7	0.02%
9	6	0.00%	2	0.01%
11	3	0.00%	1	0.00%



(a) Word cloud over all class-name phrases.

(b) Top-30 most frequent phrases.

Figure 10: Class-name vocabulary in UNIVL-ImgGen. (a) Word cloud where font size is proportional to frequency: the most prominent labels are everyday objects and people (*person*, *car*, *people*, *flowers*, *bench*, *trees*), with a long tail of attribute-modified phrases (*a wooden tray*, *a black bicycle*, *a red bicycle*). (b) Frequency chart of the 30 most frequent phrases. The vocabulary is dominated by natural-image object categories, consistent with our object-level scope.

F.3 Training and evaluation splits

We first remove 3,000 evaluation samples from the 710K records and then split the remaining records into training and validation sets. To support multi-region training, 20% of training batches use multibox data drawn from the mask category (2–5 non-overlapping masks per image; mask sampling rules in Appendix A.6). The 3,000 evaluation samples are evenly split into:

- **1,500 single-mask samples**: 375 per task type (mask, object addition, replacement, extraction).
- **1,500 multi-mask samples**: $N \in [1, 5]$ non-overlapping masks per image, testing composable multi-region generation.

Metrics in Table 1 are computed on the union of these 3,000 samples; per-mask-count breakdowns appear in Fig. 4(b)(c).

F.4 Annotation audit

The mask category relies on Grounding DINO detections and CLIP filtering, and the editing categories on model-generated edits. To measure annotation quality, we audit a stratified sample of 400 mask–label–image records (100 per category, at most 3 records per phrase). Each record is rated on three questions: **Q1**, does the mask cover the labeled object (Yes/Partial/No); **Q2**, does the label match the masked region (Yes/Close/No); and **Q3**, the overall quality of the pair for training (1–5). Q1 and Q2 are scored 1 for Yes, 0.5 for Partial or Close, and 0 for No. Two volunteer annotators, who are not

authors, rate all records. Three VLMs (GPT-4o, GPT-5-mini, and Qwen3-VL-32B) rate the same records as additional automated raters. Table 20 reports the results.

Table 20: Audit of 400 UNIVL-ImgGen annotations (mean $\pm$ standard deviation over records). Q1: mask covers the labeled object; Q2: label matches the masked region (both scored 1/0.5/0 for Yes/Partial/No); Q3: overall quality (1–5).

Rater	Q1↑	Q2↑	Q3↑
Human annotator 1	0.985 ± 0.08	0.762 ± 0.38	4.00 ± 1.54
Human annotator 2	0.905 ± 0.27	0.830 ± 0.33	4.25 ± 1.32
GPT-4o	0.764 ± 0.42	0.752 ± 0.43	4.03 ± 1.70
GPT-5-mini	0.837 ± 0.37	0.827 ± 0.38	4.32 ± 1.47
Qwen3-VL-32B	0.855 ± 0.35	0.855 ± 0.35	4.40 ± 1.41

Averaged over the two annotators, the scores are 0.95 (Q1), 0.80 (Q2), and 4.1/5 (Q3). The annotators agree on 85.6% of records for Q1 and 81.2% for Q2; on Q3 they agree exactly on 65.8% and within one point on 84.7%. The VLMs give lower Q1 scores than both annotators (0.764–0.855 vs. 0.905–0.985) and similar Q2 and Q3 scores (0.752–0.855 vs. 0.762–0.830; 4.03–4.40 vs. 4.00–4.25). Q2 is lower than Q1 for both annotators. In a manual inspection of low-scoring records, the errors mainly fall into three groups: Grounding DINO selects the wrong instance in crowded scenes, the label of an *extract* record does not fully describe the transferred content, and the mask is offset from the object boundary, so the crop includes background and the label is judged only a "Close" match.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 state the three contributions (the task with a unified visual condition, the UNIVL framework, and the efficiency gains), restrict the claims to short-label, object-level generation, and give numbers that match Section 5.2 (quality), Section 5.4 (efficiency), and Section 5.6 (operating range).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6 (Limitations) covers the label scope (style, negation, and relational labels fail; Table 4), small and large masks, rectangular training masks and overlapping masks, pseudo-label noise (Appendix F.4), the tested backbones, and the fixed text renderer. Section 5.2 reports the alignment gap to text-prompted baselines.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper presents an empirical method and benchmark; it contains no theoretical results, theorems, or formal proofs.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 5.1 and Appendix A.1 provide complete training and inference configuration (optimizer, learning rate, batch size, step counts, loss weights per stage, LoRA rank, dropout); the dataset construction pipeline (Grounding DINO + filtering thresholds) is described in Section 3; the encoder, fusion, losses, and per-stage trainable parameters are described in Section 4 and Appendix A.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code, model weights, and the UniVL-ImgGen benchmark are released at `https://github.com/Accenture/UniVL`. Training and inference hyperparameters are in Appendix A.1 and Appendix B.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5.1 reports the base model, optimizer, learning rate, batch size, step counts, LoRA configuration, multibox ratio, and inference settings; Appendix A.1 provides a complete per-stage hyperparameter table. The 3,000-sample evaluation split (1,500 single-mask + 1,500 multi-mask, $N \in [1, 5]$) is detailed in Section 3 and Appendix F.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: For the headline comparison (Table 1), we re-run inference 5 times with different seeds and report per-cell standard deviations in Appendix D.7 (Table 14). Dispersion is small ($\sigma_{\text{FID}} \leq 0.53$ across all rows); UniVL's image-quality margins exceed one standard deviation, while some alignment differences do not. The deviations reflect inference randomness only, not training variance. Ablation tables report single-run point estimates due to compute cost, and the generalization and operating-range results are reported without error bars.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section 5.1 and Appendix A.1 report the GPUs (2× and 6× H100 80GB for Stages 1 and 2), precision, batch sizes, step counts, and wall-clock time per stage. Wall-clock inference times across resolutions are reported in Table 2.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics. Human involvement is limited to two volunteer annotators (not authors) who rated dataset records (Appendix F.4); no user-facing system is deployed, and the broader-impact paragraph (Section 6) discusses misuse risks and mitigations.

Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The broader-impact paragraph in Section 6 discusses positive applications (design, accessibility, creative tools) and negative risks (misleading or non-consensual image manipulation), and points to watermarking and provenance tracking as mitigations.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The released UniVL weights (LoRA and encoder adapters) work only together with FLUX.1-dev, which its authors distribute under their own license and usage terms; the benchmark is built from the public LAION-5B and ImgEdit data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Third-party assets are credited by citation: FLUX.1-dev and FLUX.1 Fill [2] (FLUX.1 [dev] non-commercial license), SD3.5-Medium [9], DeepSeek-OCR [35], Omini-Control [29], ControlNet [41], BrushNet [12], PowerPaint [45], CLIP [23], Grounding DINO [18], LAION-5B [28], ImgEdit [40], COCO [16], and ADE20K [42]; gpt-image-1 is used through the OpenAI API under its terms of use. We use these assets for research only.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The UniVL-ImgGen benchmark and the trained weights are released with this paper (`https://github.com/Accenture/UniVL`); their construction, statistics, and annotation audit are documented in Section 3 and Appendix F.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: The paper does not involve research with human subjects. Two volunteer annotators (not authors) rated 400 dataset records for the annotation audit, using the three questions given in Appendix F.4; the evaluation itself uses automated metrics on held-out UniVL-ImgGen samples, COCO, and ADE20K.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve research with human subjects, so IRB review does not apply.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The editing categories of UniVL-ImgGen are taken from ImgEdit, whose pipeline uses GPT-4o for captions and instructions (Appendix F.1); three VLMs (GPT-4o, GPT-5-mini, Qwen3-VL-32B) serve as auxiliary raters in the annotation audit (Appendix F.4); and gpt-image-1 is a baseline (Appendix E.3). LLM use in writing was limited to editing and formatting.

Guidelines:

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