RECODE: RE-CONDITIONED DECODING FOR FAITHFUL CONTROLLABLE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Controllable image generation conditions a diffusion model on spatial conditions, but only while the latent is being generated. The decoder that renders that latent never sees the condition, so structure is lost in the pixels: boundaries blur, thin lines break, and small objects fade. In this work, we propose to apply the condition *twice*, once to shape the latent and once in the decoder. Our method, dubbed Re-Conditioned Decoding (ReCoDe), denoises in pixel space under the original condition, and attaches to any conditioned generator without retraining it. Across three condition types, two datasets, and three controllers of a FLUX generator, ReCoDe overall outperforms condition-blind decoding of the fully denoised latent, improving pixel-space condition adherence on most metrics.

1 INTRODUCTION

In controllable image generation, the user supplies a spatial condition, and a diffusion model produces an output that follows it [34, 45, 68, 86]. The paradigm is now standard for interactive editing [5, 20, 42], layout-driven design [25, 56, 74, 89, 91, 92], controllable data generation for downstream vision models [11], and scientific and remote-sensing imagery where spatial layout carries meaning [30]. These uses share one requirement: the structure and content the user specifies must survive to the pixel level. However, effective control over the generative process does not guarantee that the condition survives the decode into the pixels.

Existing methods inject the condition during *latent* generation [34, 56, 68, 86], and the resulting latent reflects the condition only imperfectly [32]. That latent is then mapped to pixels by an autoencoder decoder that is trained only to reconstruct and never sees the condition [2, 14, 15, 54, 60]. A well-conditioned latent is therefore rendered by a module that does not know what the user asked for, and spatially precise structure degrades exactly here: boundaries blur, thin lines break, and small objects fade. The autoencoder that defines the latent space already loses part of the condition [6]: encoding a real image and decoding it back, with no generation involved, forfeits fine structure before any controller is asked to do anything. This loss is largest at the native output resolution, where most fine detail is not read from the latent but synthesized by the decoder.

Recent generative decoders can synthesize missing detail, but they read only from the latent, so they may invent structure that contradicts the condition [39, 67, 90]. A few works do condition the decoder, but the condition is itself an image: the unmasked region in inpainting, or the low-resolution input in super-resolution [16, 73, 93]. The appearance is therefore given, and the decoder reproduces detail rather than inventing it, which is why a deterministic decoder suffices in those settings. A spatial condition, by contrast, supplies no appearance: an edge map marks where a boundary belongs and says nothing about what it looks like, so the detail the latent lost has to be synthesized. A deterministic decoder cannot supply that detail.

In this work, we propose to apply the condition *twice*, once to shape the latent and once in the decoder. Our method, dubbed Re-Conditioned Decoding (ReCoDe), is a generative decoder that denoises in pixel space under the original condition, and attaches to any conditioned generator without retraining it (Figure 1). ReCoDe preserves the condition the latent retained, restores the detail the latent lost, and corrects the output where the latent departs from the condition. Because it reads the condition while it finishes the denoising, ReCoDe also lets the generator stop early.

The contributions of our work are as follows:

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

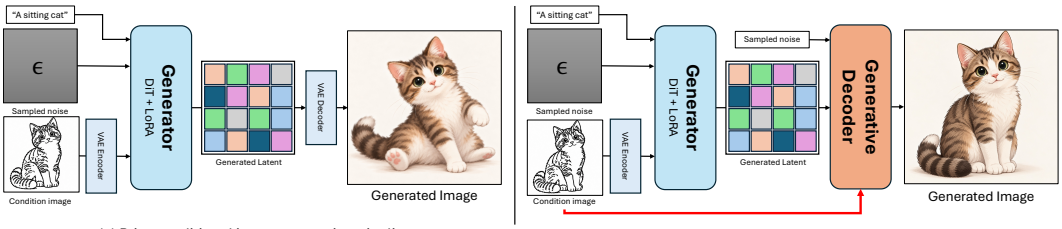

Figure 1: **Re-Conditioned Decoding.** Prior decoders read only the latent $\mathbf{z}_0$.

- We identify the latent-to-pixel decoder as the condition-blind stage that caps how faithfully a spatial condition reaches the pixels, and propose ReCoDe, which applies the condition *twice*, at generation and again at decoding, without retraining the generator.
- We show that a decoder must be both generative and condition-aware: a deterministic decoder only repositions existing structure, and a condition-blind one invents detail that contradicts the condition.
- Across three conditions, two datasets, and three FLUX controllers, ReCoDe overall outperforms both alternatives while the generator runs only 8 of its 28 steps, with the clearest gains at the native 2048×2048 output.

2 Related Work

Generative Diffusion Models. Early deep generative modeling was dominated by generative adversarial networks [4, 18, 26, 27] and variational autoencoders [31], alongside autoregressive and token-based models [8, 14, 57, 70, 72, 84]. Denoising diffusion replaced adversarial training with a stable regression objective [64], maturing into denoising diffusion probabilistic models [22], deterministic samplers [65], and the continuous score-based formulation [66]. Refinements to the objective, sampling schedule, and design space followed [28, 48]. Guidance methods sharpened the trade-off between sample fidelity and conditioning, first through an auxiliary classifier [13] and then through classifier-free guidance [21]. Text-to-image systems scaled this recipe [47, 58, 61], and latent diffusion moved the process into a compressed latent space, making large-scale training practical [60]. The modern era is defined by diffusion transformers [9, 52], by rectified-flow and flow-matching objectives [15, 37, 38], and by few-step distillation for fast sampling [40, 82], as in current systems [2, 15, 54]. A recent line moves generation back to raw pixel space, where diffusion transformers synthesize high-resolution detail directly without a latent autoencoder [33, 85].

In this work, we act on the decoding stage that every latent diffusion model above shares and leaves condition-blind. Pixel-space generation [33, 85] shows fine detail can be synthesized directly in pixels, and we use that capacity at the last stage only.

Controllable Generation. Text prompts provide only coarse spatial control, which motivates structural conditioning of the generator. Adapters and branches attach control signals to a pretrained model, beginning with ControlNet [86] and continuing through consistency feedback, lightweight adapters, and unified routing [32, 35, 45, 81]. Grounded, layout, and instance methods place content at specified regions [1, 34, 74, 79, 91, 92], and OminiControl [68] brings spatial control to diffusion transformers. Further lines compose heterogeneous conditions within one model [25, 46, 56, 89] or steer sampling without extra parameters [17, 36, 44, 53, 77]. Editing methods condition on an existing image [5, 20, 42], low-rank adaptation reduces the cost of fitting any of these to a new task [24], and domain systems apply the same controls to scientific and remote-sensing imagery [30, 69]. All of these methods inject the condition in the generative latent phase, and none of them constrains the decoder that maps latents back to pixels.

In this work, we also apply the condition at the *latent-to-pixel* step these methods leave unconstrained. Latent-side adherence fixes [32, 44, 77] improve what the latent carries but cannot exceed what the decoder preserves, and are evaluated on downscaled outputs where the decode loss is least visible; we target native resolution, which latent-phase control alone does not guarantee.

Generative Decoding. The standard latent-to-pixel decoder in latent generative models is deterministic and trained for reconstruction [14, 60, 72]. Generative decoders replace this fixed mapping with a synthesis process, from early denoising diffusion decoders [63] to consistency models [67], pixel-space diffusion decoders such as PiD [39], and representation autoencoders that decode from semantic latents [90]. In parallel, autoencoder and latent design has been revisited for generation [10, 19, 55, 71, 80, 88]. A few works instead condition the decoder itself, yet each takes a single fixed condition, such as inpainting [93], a low-resolution image [73], a reference image [16], a text mask [49], or a compositing mask [3], and most keep a deterministic decoder that can only reproduce what the latent retained. Cascaded and super-resolution pipelines similarly re-apply a coarse condition at the pixel stage [23, 62]. This family is the closest to ours, since it also treats the decode as a place to act on the condition.

In this work, we introduce ReCoDe, which makes the generative decoder condition-aware and builds on the pixel-diffusion decoder [39]. Only a decoder that is both generative and condition-aware can restore the detail the latent lost and correct the output.

3 Preliminaries

3.1 Conditioned Generation with Spatial Conditions

We consider latent generative models, in which a pretrained autoencoder maps an image $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ to a latent $\mathbf{z} = E(\mathbf{x})$ and a decoder D reconstructs it as $\mathbf{x} \approx D(\mathbf{z})$ [14, 60]. Generation is carried out in the latent space under the flow-matching formulation [15, 37, 38]

$$\mathbf{z}_t = (1-t)\,\mathbf{z}_0 + t\,\boldsymbol{\epsilon}, \qquad t \in [0,1], \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

from the clean latent $\mathbf{z}_0$ at $t = 0$ to noise at $t = 1$. A velocity network v_θ regresses the path velocity from $(\mathbf{z}_t, t, c)$ under a condition c, and a sample is drawn by integrating $\mathrm{d}\mathbf{z}_t/\mathrm{d}t = v_\theta(\mathbf{z}_t, t, c)$ from $t = 1$ to $t = 0$. Here c is a spatial condition such as a segmentation mask, an edge map, or a depth map, optionally with a text prompt, injected into v_θ through a control adapter [45, 68, 86].

3.2 High-Resolution Decoding with Pixel Diffusion

The standard decoder D is deterministic and trained only for reconstruction, so it maps $\mathbf{z}_0$ to pixels in a single pass and does not synthesize detail beyond what the latent encodes [60]. Pixel-diffusion decoding replaces this fixed mapping with a generative model that denoises directly in pixel space, which both decodes and upsamples to a high resolution [39].

The same formulation applies over images, along $\mathbf{x}_s = (1-s)\,\mathbf{x}_0 + s\,\boldsymbol{\epsilon}'$ for a clean high-resolution image $\mathbf{x}_0$ and $\boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. A pixel velocity network u_ϕ is trained to decode a noisy latent,

$$\mathcal{L}_{\mathrm{dec}} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}', s, \tau}\Big[\, \big\| \, u_\phi(\mathbf{x}_s, s, \mathbf{z}_\tau, \tau) - (\boldsymbol{\epsilon}' - \mathbf{x}_0) \, \big\|_2^2 \Big], \tag{2}$$

where the training latent is the encoding of $\mathbf{x}_0$ downsampled by a factor r, noised along Eq. equation 1 to a sampled level τ [39]. The latent enters u_ϕ through a lightweight, noise-level-aware adapter, which lets the decoder read $\mathbf{z}_\tau$ at each step s and weight it by its noise level τ [39]. At inference, the high-resolution image is produced by integrating

$$\frac{\mathrm{d}\mathbf{x}_s}{\mathrm{d}s} = u_\phi(\mathbf{x}_s, s, \mathbf{z}_\tau, \tau) \tag{3}$$

from $s = 1$ to $s = 0$. The decoder produces an image at r times the generation resolution, with $r = 4$ in the released model, so a latent generated at 512×512 decodes to 2048×2048. The released decoder integrates Eq. equation 3 in four steps, distilled from an undistilled multi-step teacher that is also released [39]. Apart from the text prompt, the decoder is conditioned only on the latent $\mathbf{z}_\tau$ and its noise level, so the generation condition c that produced the latent is not available at this stage, and when $\tau > 0$ the decoder completes the generation without it.

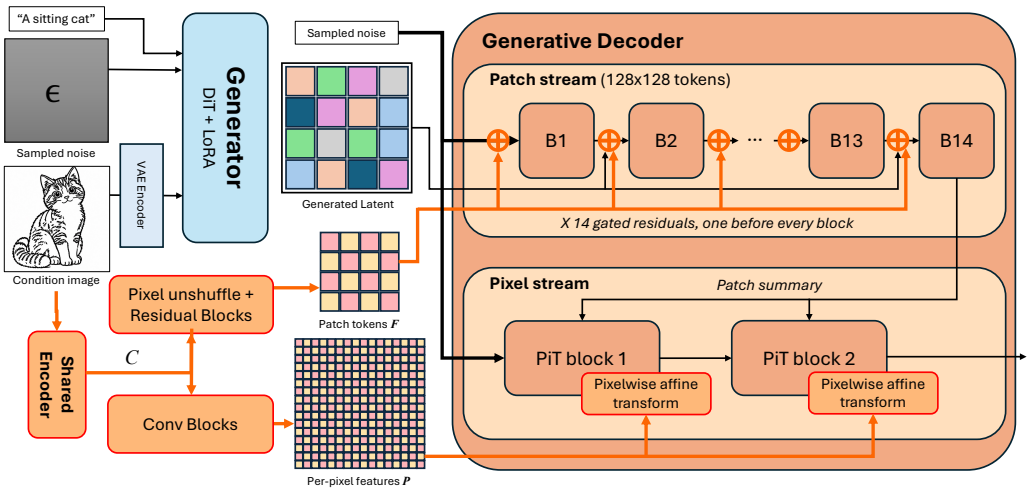

Figure 2: **Overview of Re-Conditioned Decoding.** The generator stops after $K = 8$ of its 28 steps and hands its latent $\mathbf{z}_\tau$ to the frozen pixel-diffusion decoder, which completes the image at 2048×2048. A shared encoder reads the same condition c at the output resolution and feeds it to both decoder streams: a gated residual enters the patch stream before every patch block, and a per-pixel modulation enters each pixel block (Section 4.2).

4 METHOD

4.1 RE-CONDITIONING A FROZEN DECODER

The pretrained decoder $u_\phi(\mathbf{x}_s, s, \mathbf{z}_\tau, \tau)$ of Eq. equation 2 reads the truncated latent, its noise level and the text prompt, but not the spatial condition that shaped the latent. ReCoDe adds it, giving $u_\phi(\mathbf{x}_s, s, \mathbf{z}_\tau, \tau, c)$, and integrates Eq. equation 3 with c present at every step (Figure 2). The decoder and the generator stay frozen, and only a condition path with under one percent of the decoder's parameters is trained, so ReCoDe composes with any conditioned generator that supplies a latent.

4.2 INJECTING THE CONDITION AT TWO GRANULARITIES

The patch stream fixes the global arrangement, but each token spans a $p \times p$ region ($p = 16$), so it can place a boundary only within that region. The pixel stream holds a state for every output pixel, but the adaptive normalisation of each pixel block is predicted from a single summary vector per patch, so any condition detail finer than a patch must pass through that summary. We therefore feed the condition to both streams: a patch path carries the layout, and a light pixel path gives the pixel stream condition features at its own resolution.

Condition encoder. A single convolutional encoder reads the condition once, at the native output resolution, which a latent-space controller cannot do, and returns one read-out per stream. A convolutional stem (a learned label embedding for segmentation) lifts c to a native feature map $\mathbf{C}$. A pixel unshuffle by p, a projection and two residual blocks turn $\mathbf{C}$ into patch tokens $\mathbf{F} = \mathrm{Res}\big(\mathrm{Proj}(\mathrm{PU}_p(\mathbf{C}))\big) \in \mathbb{R}^{L \times d}$ with $L = (rH/p)(rW/p)$, and two further convolutions on $\mathbf{C}$ give per-pixel features $\mathbf{P}$ at the full output resolution. $\mathbf{F}_\ell$ and the patch state $\mathbf{S}_\ell$ address the same $p \times p$ region and no positional embedding is needed.

Patch path. The condition enters the patch stream as a gated residual before each of its blocks, after the decoder's latent adapter (Section 3.2). Before block k, for each token ℓ,

$$\mathbf{S}_\ell \leftarrow \mathbf{S}_\ell + \lambda\, \mathbf{g}_{k,\ell} \odot h_k\big([\mathbf{S}_\ell, \mathbf{F}_\ell]\big), \qquad \mathbf{g}_{k,\ell} = \mathrm{sigmoid}\big(q_k([\mathbf{S}_\ell, \mathbf{F}_\ell])\big) \in (0,1)^{d_s}, \tag{4}$$

where h_k and q_k are two-layer perceptrons with a narrow SiLU bottleneck that read the current patch state together with the condition token. The residual therefore depends on what the decoder has already formed at that location, and q_k gates it per channel.

Pixel path. In each of the two pixel blocks j, the condition modulates the activation: the pretrained adaptive layer normalization is left unchanged, and ReCoDe applies a per-pixel affine modulation,

in the manner of SPADE [50], to its output $\tilde{\mathbf{x}}_j$, the normalised input of the feed-forward sublayer,

$$\tilde{\mathbf{x}}_j \leftarrow \tilde{\mathbf{x}}_j \odot \big(1 + \lambda\, \boldsymbol{\gamma}_j(\mathbf{P})\big) + \lambda\, \mathrm{sg}\big(\rho(\tilde{\mathbf{x}}_j)\big)\, \boldsymbol{\beta}_j(\mathbf{P}), \tag{5}$$

where $\boldsymbol{\gamma}_j$ and $\boldsymbol{\beta}_j$ are zero-initialised per-pixel linear maps of $\mathbf{P}$, and $\mathrm{sg}(\rho(\cdot))$ is the root mean square of the activation with the gradient stopped, which scales the shift to the activation. Unlike the pretrained normalisation, which shares one scale and shift per patch, this modulation varies per pixel and so carries condition detail finer than a patch.

4.3 Training the Condition Path

Supervision. No high-resolution ground truth exists, so we decode the clean latent $\mathbf{z}_0 = E(\mathbf{x})$ of each training image with the released pixel decoder, without condition, and use the result $\hat{\mathbf{x}}$, at r times the generation resolution, as the target. Edge and depth maps are extracted from $\hat{\mathbf{x}}$, following adapter training [32, 86], while segmentation masks are the ground-truth annotations (Appendix C).

Input latents. The input latent is $\mathbf{z}_0$ noised along equation 1 to the level τ reached after $K = 8$ of the generator's $N = 28$ steps. A truncated generator latent can already disagree with c, so for 20% of the images we replace $\mathbf{z}_0$ by the clean estimate of two text-only steps of the base generator; where latent and condition disagree, the loss then rewards following the condition.

Objective and inference. We train the undistilled multi-step decoder with its own flow-matching loss of equation 2, now with the condition present,

$$\mathcal{L}_{\mathrm{ReCoDe}}(\psi) = \mathbb{E}_{(\hat{\mathbf{x}},c,\mathbf{z}_\tau),\boldsymbol{\epsilon}',s}\Big[\, \big\| u_{\phi,\psi}(\hat{\mathbf{x}}_s, s, \mathbf{z}_\tau, \tau, c) - (\boldsymbol{\epsilon}' - \hat{\mathbf{x}}) \big\|_2^2 \Big], \qquad \hat{\mathbf{x}}_s = (1-s)\,\hat{\mathbf{x}} + s\,\boldsymbol{\epsilon}'. \tag{6}$$

The target does not depend on τ while the input does, so the loss rewards recovering from c what truncation or drift removed from the latent. We first train the encoder and the patch path, and then freeze them and train only the pixel path for 1,500 updates (Appendix C).

5 Experiments

5.1 Experimental Setup

We follow the standard controllable-generation protocol: each condition is re-extracted from the generated image with a fixed network and compared to the input condition, reported alongside FID on the benchmark that condition is conventionally evaluated on [32, 68, 87].

Datasets and Benchmarks. We evaluate three conditions on their field-standard benchmarks. Segmentation uses the ADE20K validation set (2k images) with the captions released by ControlNet++ [32], and canny edge and depth use MultiGen-20M (5k images each) [56].

Conditions and Adherence Metrics. Each condition is re-extracted from the generated image with a frozen extractor and compared to the input condition [32]. The comparison is made at the resolution of the generated image, so adherence at the native output resolution is well defined. Segmentation uses Mask2Former (Swin-L, ADE20K semantic) and reports mIoU against the input mask [12]. Canny edge is re-extracted with OpenCV Canny at the ControlNet annotator thresholds and reported as F1 against the input edge map, under the matching rule below. Depth is predicted with DPT-Large, aligned to the input depth by a per-image least-squares scale and shift, and reported as MSE and RMSE on the 0 to 255 scale of the normalized input [59].[1]

Scoring Across Output Resolutions. The decoders emit at different resolutions, 512 from the VAE and 2048 from the pixel-diffusion decoders, so one rule fixes what every number in the paper means. Each score is computed at a declared resolution, either the matched 512 view or the native output, and scores are compared only within one of them. A decoder whose output is only 512 has no native score, so its 2048 cell is marked ‡ and gives the score of a bicubic upsample. For edges we count a match within one condition pixel, following edge-detection practice [41], which is one output pixel at the matched view and four at the native output. This tolerant score is our primary one, since it is the only score comparable across resolutions.

[1] Depth differences between the deterministic and generative families partly reflect the estimator responding to the sharper texture of a downsampled 2048 output, an effect every generative row shares.

Baselines. Every row is a decoder, a controller it is run under, or a decoder-only reference. The decoders are the four cells of Table 2: the VAE decode, vanilla PiD [39], a conditioned VAE decoder, and ReCoDe. The conditioned VAE decoder is our re-implementation of the decoder-conditioning archetype on the FLUX latent, since the published methods [16, 73, 93] cannot decode a FLUX latent. Each decoder is run under three FLUX-based controllers, OminiControl [68], EasyControl [87], and a FLUX ControlNet [86]. TokenConcat faithfully implements OminiControl [68]-like conditioning on top of vanilla PiD, without introducing any ReCoDe-specific modifications.

Resolution and Truncation Protocol. All latents are generated at 512×512, so only the pixel-diffusion decoders have a native output above that resolution. Each row carries three edge scores: against the 512×512 condition map c512, at the matched view and at the native output, and against a 2048×2048 condition map c2048 at the native output. The matched view scores the output downsampled to 512×512 against c512, and every decoder has it. At the native output the 2048×2048 edges are matched against the same c512, where one condition pixel covers four output pixels, so this score charges the detail a decoder synthesizes where the condition has no edge. The 2048×2048 map c2048 does not exist in the evaluation sets, which hold no pixels above 512, so it is re-extracted from a blind round trip of the real image at a held-out seed, and an independent decode of the same latent agrees with it only to 0.708.

The generative decoders also need not wait for a fully denoised latent. ReCoDe decodes the latent the generator reaches after $K = 8$ of its 28 steps, while the condition-blind decoders, the VAE decode and vanilla PiD, receive the fully denoised latent, their strongest setting.

5.2 Implementation Details

The generator is FLUX.1-dev [2] at 512×512, 28 steps, guidance 3.5, run under OminiControl [68], EasyControl [87], or the Union-Pro-2.0 FLUX ControlNet [86]. Canny and depth use each controller's released adapter; for segmentation we train an OminiControl adapter on ADE20K with its official recipe, which scores above EasyControl's released one. The condition-blind decoder is the released four-step PiD-FLUX student [39], which reaches 2048 from the 512 latent, and ReCoDe adds the condition path of Section 4 to the undistilled PiD teacher it was distilled from.

Training data for ReCoDe. The condition path is trained per condition on the training split of the corresponding evaluation dataset, a subset of MultiGen-20M train for canny and depth (36,122 images) and ADE20K train for segmentation (19,701 images), with the 2048×2048 targets built as described in Section 4.3.

5.3 Results

Table 1 is the core comparison: a decoder swap on the same FLUX latent, with every row run by us. Each adherence metric is reported at the matched 512 view and at the native 2048 output. ReCoDe decodes the latent after 8 of the 28 generator steps, yet matches or improves the pixel-space condition adherence of both the deterministic VAE decode and the condition-blind PiD on the fully denoised latent, with the lowest FID and pFID under every controller, so adherence is not bought by a loss in image quality. Figure 3 shows the same effect qualitatively. The gains concentrate where the latent is uncertain, such as small objects and thin structures, since there the latent under-specifies the pixels and the condition supplies the target it lacks, as the close-ups in Appendix Figure 5 show.

5.4 Comparison and Analysis

Effect of conditioning the decoder. The headline comparison is ReCoDe against vanilla PiD, which decodes from the latent alone. Vanilla PiD receives the fully denoised latent after all 28 generator steps, while ReCoDe decodes the latent after only 8, so any advantage of ReCoDe is obtained against a baseline that has more of the generation already done. Re-injecting the condition improves depth under every controller, edge adherence at the native output, and segmentation under EasyControl, while tolerant edge F1 at the matched view and OminiControl segmentation stay within 0.05 and 0.3 of vanilla PiD; FID and pFID improve under every controller, and ReCoDe scores above vanilla PiD on seven of eight no-reference metrics (Table 4).

Table 1: Core decoder-swap comparison. PiD denotes the vanilla condition-blind decoder, and bold marks the best decoder per column within each controller block. Within each controller block all rows decode the same latent under the same seed and prompt, so the table isolates the decode stage. Adherence is reported at the matched 512×512 view, where the 2048 outputs are area-downsampled, and edge adherence also at the native 2048 output. The three edge columns are the matched view and the native output against the 512 condition c512, and the native output against the 2048 condition c2048; since c2048 is the edge map of a blind round trip of the real image, the PiD round trip is its reference at 1.0. Rows that are 512-native have no native output, and their 2048 cells, marked ‡, give the score of a bicubic upsample, which is never read as a native output.

	Seg mIoU↑	Canny F1↑ (view / reference map)			Depth MSE↓	Depth RMSE↓	FID↓	pFID↓
Method	512	512 / c512	2048 / c512	2048 / c2048	512	512	512	512
Real image	53.7	1.0	0.547‡	0.306‡	0	0	n/a	n/a
VAE round trip	53.6	0.943	0.529‡	0.296‡	10.8	2.54	0.64	1.56
PiD round trip	53.5	0.917	0.800	1.0	62.4	6.18	1.88	3.78
OminiControl [68], VAE and PiD at $K = 28$, ReCoDe at $K = 8$								
+ VAE decode	**43.9**	0.771	0.276‡	0.114‡	717	23.8	17.2	20.4
+ PiD	43.7	**0.785**	0.733	0.510	696	23.5	16.1	17.6
+ TokenConcat	26.2	0.391	0.415	0.287	2197	44.1	24.3	23.0
+ ReCoDe (Ours)	43.4	0.763	**0.799**	**0.862**	**677**	**23.2**	**14.8**	**15.9**
EasyControl [87], VAE and PiD at $K = 28$, ReCoDe at $K = 8$								
+ VAE decode	39.3	0.810	0.439‡	0.156‡	622	22.5	20.9	23.0
+ PiD	39.3	**0.819**	0.770	0.405	612	22.3	20.0	22.1
+ TokenConcat	26.8	0.481	0.493	0.337	1398	34.8	18.6	19.4
+ ReCoDe (Ours)	**43.2**	0.778	**0.802**	**0.862**	**602**	**22.0**	**12.1**	**13.3**
FLUX ControlNet (Union-Pro-2.0) [86], VAE and PiD at $K = 28$, ReCoDe at $K = 8$								
+ VAE decode	n/a	0.656	0.305‡	0.130‡	1103	29.9	16.5	18.0
+ PiD	n/a	0.659	0.654	0.433	1094	29.8	16.2	16.2
+ TokenConcat	n/a	0.426	0.445	0.315	2467	46.9	23.0	21.7
+ ReCoDe (Ours)	n/a	**0.759**	**0.799**	**0.862**	**762**	**24.7**	**14.7**	**15.5**

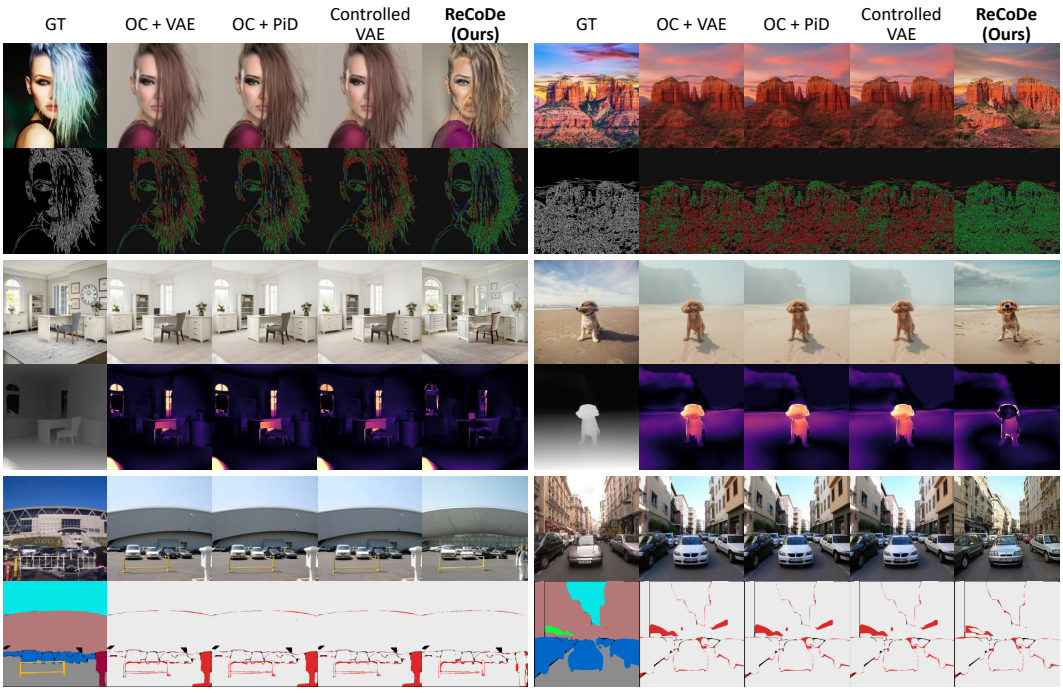

Figure 3: **Main qualitative comparison at native resolution.** ReCoDe preserves fine structure and sharp boundaries where the deterministic and condition-blind decoders lose them. Best viewed on electronics.

Modularity across controllers. We repeat the decoder swap under three controllers with different conditioning mechanisms: two inject condition tokens into the transformer, and one adds a ControlNet branch. The condition path is trained without any controller, yet ReCoDe improves on each



Figure 4: What the condition path does. (a) The magnitude of the residual ReCoDe injects, averaged over its injection sites. (b) Samples whose generator latent departs from the condition, decoded from the same $K = 8$ latent by the VAE, by vanilla PiD, and by ReCoDe.

Table 2: The four decoder types on the same OminiControl latents, matched 512×512 view: condition-blind or condition-aware, deterministic or generative. *Latent* is the noise level handed at evaluation; $K = 28$ is the fully denoised latent. Canny F1 is the tolerant score of our protocol, $\mathrm{F1}_{\mathrm{exact}}$ the pixel-exact score of the standard one.

Decoder	Latent	Seg mIoU↑	Canny F1↑	$\mathrm{F1}_{\mathrm{exact}}$ ↑	Depth RMSE↓	MANIQA↑	FID↓	pFID↓
Condition-blind, deterministic								
VAE decode	$K = 28$	43.9	0.771	0.358	23.8	0.503	17.2	20.4
Condition-blind, generative								
vanilla PiD	$K = 28$	43.7	0.785	0.357	23.5	0.622	16.1	17.6
vanilla PiD	$K = 16$	37.5	0.548	0.194	30.8	0.682	26.3	27.0
Condition-aware, deterministic (the published archetype, SPADE-style modulation)								
Cond. VAE, modulated	$K = 28$	43.8	0.766	0.442	23.8	0.471	17.0	18.9
Cond. VAE, modulated	$K = 16$	15.9	0.848	0.353	33.6	0.242	201.9	251.8
Condition-aware and generative (ours)								
ReCoDe	$K = 8$	43.4	0.763	0.404	**23.2**	0.598	**14.8**	**15.9**

controller's own vanilla PiD in depth, native-resolution edges, and FID under all three (Table 1), so the gain belongs to the decoder, which attaches to a conditioned generator without retraining.

Condition-aware versus condition-blind, generative versus deterministic. Table 2 compares the four decoder types on the same latents. Conditioning the deterministic VAE decoder, as the published methods do [50, 73, 93], only moves existing edges into place: the pixel-exact F1 rises from 0.36 to 0.44, while the tolerant F1 stays at 0.77. Trained on re-noised latents, it decodes a truncated latent only by giving up content, tracing edges onto a blurred image at $K = 16$ (edge F1 0.85, but 15.9 mIoU and an FID of 202). ReCoDe, generative and condition-aware at once, decodes an even more truncated latent ($K = 8$) with the best depth RMSE and FID in the table (Appendix F).

Where the condition path acts. Figure 4 (a) overlays the magnitude of the residual that ReCoDe injects, averaged over its injection sites, on the decoded image. The injection concentrates on edges, small objects, and depth discontinuities, where the latent under-specifies the pixels, and stays small in flat regions that the latent already determines. The condition path therefore acts as a targeted correction where the latent is uncertain, not as a global change to the decoder output.

Correcting a latent that departs from the condition. Figure 4 (b) shows samples whose generator latent departs from the condition, with an object missing or displaced. The VAE decode reproduces the error, since a deterministic decoder can only render the latent it is handed, and vanilla PiD keeps it as well, since it never reads the condition. ReCoDe decodes the same latent and restores the structure the condition asks for, because it reads the condition while it finishes the denoising.

Resolution sweep. At the native output the interpolation route, the VAE decode upsampled to 2048 (‡ in Table 1), scores 0.28 to 0.44 edge F1 across the three controllers, where vanilla PiD on the fully

Table 3: **Efficiency across output resolutions**. Per-image latency and peak memory on one NVIDIA H200. Every baseline runs the generator for all $K = 28$ steps, and ReCoDe stops it after $K = 8$.

Output	Pipeline	Gen. res.	K	Dec. steps	Latency (s)↓	Mem. (GB)↓
Output 512 (matched resolution)						
512	FLUX + VAE	512	28	1	2.32	34.4
Output 2048 (native target)						
2048	FLUX + VAE	2048	28	1	36.2	44.0
2048	FLUX + PiD	512	28	4	3.34	34.4
2048	FLUX + PiD teacher	512	28	25	15.0	34.4
2048	FLUX + ReCoDe (Ours)	512	8	25	14.7	34.0
2048	FLUX + ReCoDe, distilled (projected)[§]	512	8	4	3.11[§]	34.0

Table 4: **No-reference image quality on the canny split of MultiGen-20M under the default controller OminiControl**. We follow the PiD metric suite [39]. ReCoDe scores above vanilla PiD on seven of the eight metrics.

Method	MUSIQ↑	NIQE↓	DeQA↑	MANIQA↑	Q-Align↑	UniP-IAA↑	UniP-IQA↑	VQ-R1↑
Real image	72.8	3.62	3.86	0.498	4.31	71.4	69.8	4.48
VAE decode	71.8	4.82	3.70	0.503	4.12	67.6	66.7	4.29
vanilla PiD	72.3	4.88	4.01	**0.622**	4.34	73.1	71.6	4.45
ReCoDe (Ours)	**73.3**	**3.96**	**4.15**	0.598	**4.51**	**74.8**	**73.4**	**4.56**

denoised latent scores 0.65 to 0.77, so the detail a generative decoder synthesizes is worth 0.3 to 0.4 in edge F1 before it consults the condition at all.

Early termination and efficiency. A decoder that does not read the condition needs the generator to finish: the VAE decode requires the fully denoised latent, and vanilla PiD accepts a partially denoised one [39] but degrades as the generator stops earlier, its edge F1 under OminiControl falling from 0.79 at 28 steps to 0.55 at 16 and its FID rising from 16.1 to 26.3. ReCoDe reads the condition, so the generator can stop after $K = 8$ of its 28 steps and ReCoDe still beats both on depth under every controller (Table 1). Following PiD, Table 3 compares routes to a 2048 output: stopping at $K = 8$ offsets the cost of the condition path, so ReCoDe (14.7 s) is slightly faster than the teacher it builds on (15.0 s) without raising peak memory, and a distilled ReCoDe is projected at 3.1 s, while generating at 2048 directly is slower (36.2 s) and loses the condition (Appendix E).

No-reference image quality. Conditioning the decoder should raise adherence without lowering general image quality, so we follow PiD [39] and score the decoded images with eight no-reference quality metrics (Table 4): MUSIQ [29], NIQE [43], DeQA-Score [83], MANIQA [78], Q-Align [75], UniPercept for aesthetics and for quality [7], and VisualQuality-R1 [76]. We report them on the canny split under OminiControl at the matched 512 view, where the real source images, the VAE decode and the pixel-diffusion decoders are all comparable. The comparison of interest is ReCoDe against vanilla PiD, since both are pixel-diffusion decoders of the same family and ReCoDe adds the condition, so any drop would be the price of adherence. No such drop appears: ReCoDe scores above vanilla PiD on seven of the eight metrics and is within 0.03 on MANIQA, and it scores above the VAE decode on all eight, while decoding after 8 generator steps rather than 28.

6 Conclusion

We proposed Re-Conditioned Decoding (ReCoDe), which applies the condition twice, once to shape the latent and once more in the decoder, the stage that produces the pixels a spatial condition refers to. ReCoDe attaches to any conditioned generator without retraining it, and it improves pixel-space condition adherence at matched perceptual quality across condition types and generators, even when the generator stops early. We hope this work encourages treating the decoder as the last stage of conditional generation, and extending condition-aware decoding to video and 3D generation.

AI USE STATEMENT

In this work, we used generative AI tools to design the evaluation methodology, implement methods and evaluation code, and prepare evaluation subsets. We have not used generative AI tools to generate synthetic data sets or for translation, and mathematical claims, proofs, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to draft and edit the paper, identify and summarize literature, format references, create figures, and draft the necessary code and artifacts. We have reviewed all AI-assisted work: every citation was checked against its source, every lifted number was verified against the published paper, all AI-written code was run and tested by the authors, and all claims were examined by the authors against the literature. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects and no personal data. All evaluation data are public benchmarks, ADE20K, MultiGen-20M, COCO 2017, and DreamBench, and all pretrained models, the FLUX.1-dev generator, its control adapters, and the PiD decoder, are used under their research licenses, which restrict commercial use. Our method improves how faithfully a generated image follows a user-specified structure, so it could lower the cost of producing misleading imagery; this risk is shared with the controllable generators we build on, and our contribution adds no identity or liken ess synthesis. The outputs inherit any biases of the pretrained generator and decoder, which we do not retrain. We document the evaluation protocol in full and will release our code and weights upon acceptance so that the claims can be scrutinized.

REPRODUCIBILITY STATEMENT

The evaluation protocol is specified in Section 5.1: the datasets and splits, the frozen condition extractors with their checkpoints and thresholds, the adherence metrics, the matched-resolution and native-resolution views, and the truncation point. Section C lists the generator, the four controllers, the adapters we train and their data, the decoder checkpoints, the sampling steps, and the seeds. Every component is publicly available, the generator, the controllers and their released adapters, the decoder checkpoints, the datasets, and the frozen scorers, each identified by name and version, and the adapters we train are described with their data, rendered-condition format, and recipe. The evaluation subsets, seeds, and the cached-latent protocol under which every decoder reads identical inputs are specified in the supplementary, together with the paper, table, and setting from which each lifted number was taken, so that every row of every table can be regenerated from the description. Our code and weights will be released upon acceptance.

REFERENCES

[1] Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. SpaText: Spatio-textual representation for controllable image generation. In *CVPR*, 2023.

[2] Black Forest Labs. FLUX.1. https://github.com/black-forest-labs/flux, 2024.

[3] Rowan Bradbury and Dazhi Zhong. Your latent mask is wrong: Pixel-equivalent latent compositing for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2512.05198.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[5] 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*, 2023. URL https://arxiv.org/abs/2211.09800.

[6] Pu Cao, Yiyang Ma, Feng Zhou, Xuedan Yin, Qing Song, and Lu Yang. A tilted seesaw: Revisiting autoencoder trade-off for controllable diffusion. *arXiv preprint arXiv:2601.21633*, 2026.

[7] Shuo Cao, Jiayang Li, Xiaohui Li, Yuandong Pu, Kaiwen Zhu, Yuanting Gao, Siqi Luo, Yi Xin, Qi Qin, Yu Zhou, Xiangyu Chen, Wenlong Zhang, Bin Fu, Yu Qiao, and Yihao Liu. Unipercept: Towards unified perceptual-level image understanding across aesthetics, quality, structure, and texture. *arXiv preprint arXiv:2512.21675*, 2025.

[8] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *CVPR*, 2022.

[9] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.00426.

[10] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10733.

[11] Kai Chen, Enze Xie, Zhe Chen, Yibo Wang, Lanqing Hong, Zhenguo Li, and Dit-Yan Yeung. GeoDiffusion: Text-prompted geometric control for object detection data generation. In *ICLR*, 2024.

[12] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2112.01527.

[13] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

[14] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[15] 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 *ICML*, 2024.

[16] Xiang Fan, Yuheng Wang, Bohan Fang, Zhongzheng Ren, and Ranjay Krishna. Refdecoder: Enhancing visual generation with conditional video decoding. *arXiv preprint arXiv:2605.15196*, 2026.

[17] Yutong Feng, Biao Gong, Di Chen, Yujun Shen, Yu Liu, and Jingren Zhou. Ranni: Taming text-to-image diffusion for accurate instruction following. In *CVPR*, 2024.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[19] Jonathan Heek, Emiel Hoogeboom, Thomas Mensink, and Tim Salimans. Unified latents: How to train your latents. *arXiv preprint arXiv:2602.17270*, 2026. URL https://arxiv.org/abs/2602.17270.

[20] 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*, 2023. URL https://arxiv.org/abs/2208.01626.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[23] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 2022. URL https://arxiv.org/abs/2106.15282.

[24] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[25] Lianghua Huang, Di Chen, Yu Liu, Yujun Shen, Deli Zhao, and Jingren Zhou. Composer: Creative and controllable image synthesis with composable conditions. In *ICML*, 2023.

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[27] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

[28] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

[29] 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*, 2021. URL https://arxiv.org/abs/2108.05997.

[30] Samar Khanna, Patrick Liu, Linqi Zhou, Chenlin Meng, Robin Rombach, Marshall Burke, David Lobell, and Stefano Ermon. DiffusionSat: A generative foundation model for satellite imagery. In *ICLR*, 2024.

[31] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[32] Ming Li, Taojiannan Yang, Huafeng Kuang, Jie Wu, Zhaoning Wang, Xuefeng Xiao, and Chen Chen. Controlnet++: Improving conditional controls with efficient consistency feedback. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2404.07987.

[33] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025. URL https://arxiv.org/abs/2511.13720.

[34] 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 *CVPR*, 2023.

[35] Han Lin, Jaemin Cho, Abhay Zala, and Mohit Bansal. Ctrl-adapter: An efficient and versatile framework for adapting diverse controls to any diffusion model. *arXiv preprint arXiv:2404.09967*, 2024.

[36] Kuan Heng Lin, Sicheng Mo, Ben Klingher, Fangzhou Mu, and Bolei Zhou. Ctrl-x: Controlling structure and appearance for text-to-image generation without guidance. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.07540.

[37] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

[38] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

[39] Yifan Lu, Qi Wu, Jay Zhangjie Wu, Zian Wang, Huan Ling, Sanja Fidler, and Xuanchi Ren. Pid: Fast and high-resolution latent decoding with pixel diffusion. *arXiv preprint arXiv:2605.23902*, 2026.

[40] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[41] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004. doi: 10.1109/TPAMI.2004.1273918.

[42] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2108.01073.

[43] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013.

[44] Sicheng Mo, Fangzhou Mu, Kuan Heng Lin, Yanli Liu, Bochen Guan, Yin Li, and Bolei Zhou. Freecontrol: Training-free spatial control of any text-to-image diffusion model with any condition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2312.07536.

[45] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2I-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *AAAI*, 2024.

[46] Kiet T. Nguyen, Chanhyuk Lee, Donggyun Kim, Dong Hoon Lee, and Seunghoon Hong. Universal few-shot spatial control for diffusion models. *arXiv preprint arXiv:2509.07530*, 2025. URL https://arxiv.org/abs/2509.07530.

[47] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022.

[48] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

[49] Shubham Paliwal, Arushi Jain, Monika Sharma, Vikram Jamwal, and Lovekesh Vig. Customtext: Customized textual image generation using diffusion models. *arXiv preprint arXiv:2405.12531*, 2024.

[50] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[51] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2104.11222.

[52] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2212.09748.

[53] Quynh Phung, Songwei Ge, and Jia-Bin Huang. Grounded text-to-image synthesis with attention refocusing. In *CVPR*, 2024.

[54] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024.

[55] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2111.15640.

[56] Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, Stefano Ermon, Yun Fu, and Ran Xu. UniControl: A unified diffusion model for controllable visual generation in the wild. In *NeurIPS*, 2023.

[57] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

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

[59] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2103.13413.

[60] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[61] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

[62] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. URL https://arxiv.org/abs/2104.07636.

[63] Jie Shi, Chenfei Wu, Jian Liang, Xiang Liu, and Nan Duan. Divae: Photorealistic images synthesis with denoising diffusion decoder. *arXiv preprint arXiv:2206.00386*, 2022. URL https://arxiv.org/abs/2206.00386.

[64] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

[65] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[66] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[67] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2303.01469.

[68] Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. OminiControl: Minimal and universal control for diffusion transformer. In *ICCV*, 2025.

[69] Datao Tang, Xiangyong Cao, Xingsong Hou, Zhongyuan Jiang, Junmin Liu, and Deyu Meng. CRS-Diff: Controllable remote sensing image generation with diffusion model. *IEEE Trans. Geosci. Remote Sens.*, 2024.

[70] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

[71] Théophane Vallaeys, Jakob Verbeek, and Matthieu Cord. Ssdd: Single-step diffusion decoder for efficient image tokenization. *arXiv preprint arXiv:2510.04961*, 2025. URL https://arxiv.org/abs/2510.04961.

[72] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[73] Jianyi Wang, Zongsheng Yue, Shangchen Zhou, Kelvin C.K. Chan, and Chen Change Loy. Exploiting diffusion prior for real-world image super-resolution. *International Journal of Computer Vision*, 2024.

[74] Xudong Wang, Trevor Darrell, Sai Saketh Rambhatla, Rohit Girdhar, and Ishan Misra. InstanceDiffusion: Instance-level control for image generation. In *CVPR*, 2024.

[75] Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Liang Liao, Chunyi Li, Yixuan Gao, Annan Wang, Erli Zhang, Wenxiu Sun, Qiong Yan, Xiongkuo Min, Guangtao Zhai, and Weisi Lin. Q-align: Teaching lmms for visual scoring via discrete text-defined levels. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2312.17090.

[76] Tianhe Wu, Jian Zou, Jie Liang, Lei Zhang, and Kede Ma. Visualquality-r1: Reasoning-induced image quality assessment via reinforcement learning to rank. *arXiv preprint arXiv:2505.14460*, 2025.

[77] Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. BoxDiff: Text-to-image synthesis with training-free box-constrained diffusion. In *ICCV*, 2023.

[78] Sidi Yang, Tianhe Wu, Shuwei Shi, Shanshan Lao, Yuan Gong, Mingdeng Cao, Jiahao Wang, and Yujiu Yang. Maniqa: Multi-dimension attention network for no-reference image quality assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022. URL https://arxiv.org/abs/2204.08958.

[79] Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. ReCo: Region-controlled text-to-image generation. In *CVPR*, 2023.

[80] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2501.01423.

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

[82] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.18828.

[83] Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2501.11561.

[84] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Trans. Mach. Learn. Res. (TMLR)*, 2022.

[85] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2511.20645.

[86] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[87] Yuxuan Zhang, Yirui Yuan, Yiren Song, Haofan Wang, and Jiaming Liu. Easycontrol: Adding efficient and flexible control for diffusion transformer. In *IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2503.07027.

[88] Long Zhao, Sanghyun Woo, Ziyu Wan, Yandong Li, Han Zhang, Boqing Gong, Hartwig Adam, Xuhui Jia, and Ting Liu. Epsilon-vae: Denoising as visual decoding. *arXiv preprint arXiv:2410.04081*, 2024.

[89] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K. Wong. Uni-ControlNet: All-in-one control to text-to-image diffusion models. In *NeurIPS*, 2023.

[90] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

[91] Guangcong Zheng, Xianpan Zhou, Xuewei Li, Zhongang Qi, Ying Shan, and Xi Li. LayoutDiffusion: Controllable diffusion model for layout-to-image generation. In *CVPR*, 2023.

[92] Dewei Zhou, You Li, Fan Ma, Xiaoting Zhang, and Yi Yang. MIGC: Multi-instance generation controller for text-to-image synthesis. In *CVPR*, 2024.

[93] Zixin Zhu, Xuelu Feng, Dongdong Chen, Jianmin Bao, Le Wang, Yinpeng Chen, Lu Yuan, and Gang Hua. Designing a better asymmetric vqgan for stablediffusion. *arXiv preprint arXiv:2306.04632*, 2023.