Logit-Conditioned Diffusion Decoding for Frozen Discrete-Token VLMs

Ji Woo Hong[1*] **Gwanhyeong Koo**[1] **Hee Suk Yoon**[1] **Eunseop Yoon**[1]
SooHwan Eom[1] **Qi Dai**[2] **Chong Luo**[2†] **Chang D. Yoo**[1†]
[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Microsoft Research Asia (MSRA)
{jiwoohong93,cd_yoo}@kaist.ac.kr
{cluo}@microsoft.com

Abstract

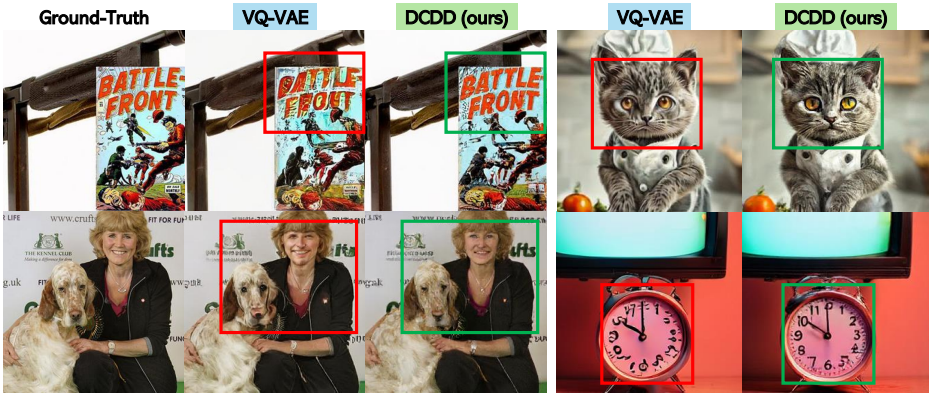
Pre-trained discrete-token vision-language models (VLMs) generate images by sampling sequences of codebook indices, but their visual fidelity is bounded by the VQ-VAE decoding stage. Prior work addresses this bottleneck by replacing the discrete tokenizer with continuous or hybrid visual token representations, or by integrating diffusion into the generation pipeline; both routes require foundation-scale retraining or realignment of VLM, which is undesirable when the VLM's broader capabilities should be preserved or when retraining is infeasible. We show that such retraining is unnecessary: before each token is sampled, discrete-token VLMs already compute a full logit distribution over the codebook, but standard decoding discards it after selecting a single index. Under Otsu's threshold, the pre-sampling logits of capable VLMs assign statistically meaningful probability to at least two codebook entries per token, and recovering this distribution can substantially close the fidelity gap. We propose DCDD, a post-hoc framework that conditions a diffusion decoder on this discarded distribution while leaving the VLM frozen. At inference, DCDD converts the VLM's pre-sampling logits into distribution-weighted code vectors; during training, it uses VQ-VAE encoder-derived proxy logits calibrated to match the distributional statistics of the VLM's inference-time logits. Our diffusion decoder then maps these representations to high-fidelity images as an optional alternative to the native VQ-VAE decoder. Trained on ImageNet-1K for 50K steps, DCDD reduces reconstruction FID by 74% and Janus-Pro 7B's generation FID on MJHQ-30K by 34%, while semantic alignment across GenEval, DPG-Bench, and WISE remain consistent. Controlled comparisons with diffusion models from the same backbone family, applied either as an image-to-image refiner or as an identically trained decoder on the sampled codes, confirm that the gain stems from conditioning on the VLM's pre-sampling logit distribution rather than from the diffusion model itself. Code is available at `https://github.com/jiwoohong93/dcdd_code`.

1 Introduction

Autoregressive vision-language models (VLMs) [39, 54, 2, 44] generate images by predicting discrete tokens from a learned VQ-VAE codebook [47, 40, 5]. Although these models exhibit strong semantic capabilities, their images often contain distortions in local structure and geometry (Figure 1); as VLMs scale, the bottleneck increasingly lies in the decoding stage rather than the transformer itself.

*This work was done while Ji Woo Hong was an intern at MSRA.
†Corresponding Authors.

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



(a) VQ-VAE reconstruction caps achievable fidelity. (b) The same bottleneck limits T2I generation.

Figure 1: **The VQ-VAE decoder is the fidelity bottleneck in discrete-token VLM image generation.** *(a) Reconstruction:* even when the input image is available and the VLM is bypassed entirely, the VQ-VAE encode–decode pipeline already loses fine structure relative to the ground truth (book cover text, facial details). *(b) Text-to-image generation:* the same bottleneck propagates to the deployed pipeline, where the native VQ-VAE decoder produces distorted local geometry on the VLM's predicted tokens. DCDD replaces only the decoding stage and recovers fine detail with the VLM left frozen.

Existing approaches to this fidelity bottleneck modify the system in one of two ways. One route replaces the original VQ-VAE tokenizer with a higher-capacity continuous or hybrid visual token representation (Figure 2, *left*(a)); another integrates diffusion directly into the generation pipeline (Figure 2, *left*(b)). In both cases, the generation is rebuilt around a new token, so the VLM must be retrained or realigned at foundation-scale. Both options are undesirable when the VLM's broader capabilities should be preserved, or when retraining is infeasible. We therefore ask: can the visual fidelity of a discrete-token VLM be improved post-hoc, with the VLM left untouched?

Our starting point is that discrete-token VLMs already compute, but discard, the information needed to answer this question. At each spatial position, the VLM produces logits over all codebook tokens, and standard decoding reduces this distribution to a single token index by argmax. The full distribution, however, is not one-hot: under Otsu's variance-maximizing threshold [36], the logits of Janus-Pro 7B [2], a publicly released discrete-token AR VLM, assign statistically meaningful probability to at least two codebook entries per token (Table 1), with top-1 probability 0.22 and top-2 probability 0.10. Recovering this distributional information can substantially close the fidelity gap.

Building on this observation, we introduce **DCDD (Distribution-Conditioned Diffusion Decoder)**, a post-hoc framework that conditions a diffusion decoder on the VLM's *logit distribution* rather than solely on the sampled token. DCDD comprises three components (Figure 2, *right*). *Logit-to-Code Distributional Mapping (LCDM)* converts each token's logit distribution into continuous distribution-weighted code vectors with complementary uncertainty features. *Logit Calibration (LC)* is a gradient-free procedure that aligns the VQ-VAE encoder-derived proxy logits used during training with the distribution statistics of the VLM's inference-time logits, so that the decoder sees matched conditioning at training and inference. *The Diffusion Decoder (DD)*, finetuned from FLUX.1-Fill, maps these representations to high-fidelity images and serves as an optional alternative to the VQ-VAE decoder, with the VLM left frozen.

DCDD is scoped to fidelity: the diffusion decoder is conditioned only on VLM-derived token distributions and excludes direct text conditioning, so it follows the VLM's original token belief rather than reinterpreting the scene. We primarily evaluate DCDD on Janus-Pro 7B [2], one of the most capable publicly released discrete-token VLMs. We also demonstrate that the decoder transfers to Janus-Pro 1B without retraining, as both models share the same tokenizer. Finally, we use LlamaGen-XL [44] to characterize when post-hoc decoding is effective. Trained on ImageNet-1K alone, DCDD substantially improves both reconstruction and generation fidelity while keeping semantic consistency stable. Comparisons with diffusion models from the same backbone family as DCDD's decoder, applied either as an image-to-image refiner (FLUX.1 Kontext [22]) or as an identically trained decoder on the sampled codes, further show that DCDD's gain stems from conditioning on the VLM's pre-sampling logit distribution rather than from the diffusion model itself.

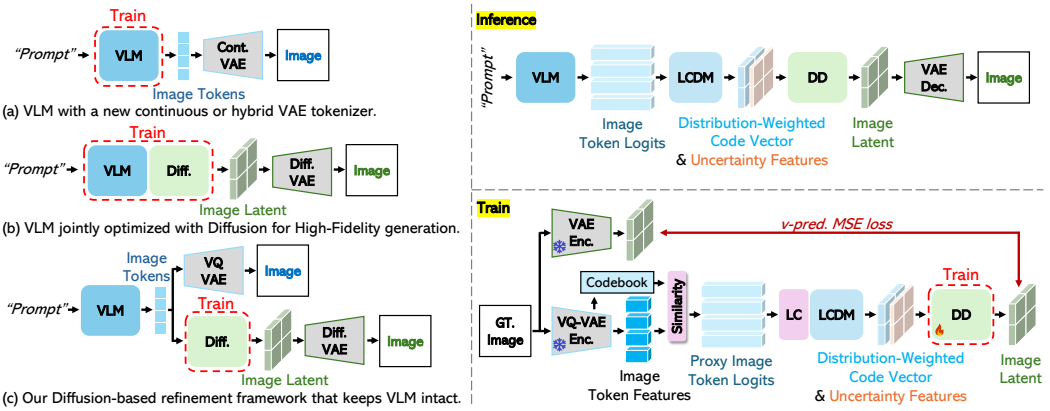


Figure 2: ***Left*: Strategies for improving visual fidelity in pre-trained discrete-token VLM image generation, characterized by which components are trained.** ***(a) Tokenizer replacement:*** the VLM is retrained on foundation-scale T2I data to accommodate a new continuous or hybrid tokenizer. ***(b) Joint optimization with diffusion:*** the VLM and a diffusion model are jointly trained, making diffusion a required part of generation. ***(c) DCDD (ours):*** only the diffusion decoder is trained; the VLM is left frozen, and the native VQ-VAE decoding path remains available. ***Right*: DCDD framework.** At inference, LCDM converts the VLM's image-token logits into conditioning for the Diffusion Decoder. At training, proxy logits from the VQ-VAE encoder are aligned to the VLM's inference-time statistics by LC, a gradient-free transform whose parameters are estimated beforehand.

Our contributions are: (i) We identify and quantify a previously underused signal in discrete-token VLMs: the pre-argmax logit distribution carries rank-2 or higher codebook information that standard decoding discards. (ii) We propose DCDD, a post-hoc framework that exploits this signal as a diffusion conditioning input, requires no VLM modification, and trains on ImageNet-1K alone. (iii) We provide controlled evidence that the gain originates from logit-level conditioning rather than from the diffusion prior or image-to-image refinement, that semantics are preserved, and that the method's effectiveness is governed by upstream token quality and logit structure.

2 Related Work

Approaches to improving the visual fidelity of discrete-token VLM image generation fall along three axes characterized by which components are trained (Figure 2, *left*).

Tokenizer replacement. One line of work replaces the discrete VQ-VAE with continuous or hybrid tokenization to reduce quantization artifacts and preserve fine-grained details. TokenBridge [48] and MAR [25] decouple quantization from training or eliminate the discrete codebook entirely. HART [45] and DC-AR [50] adopt hybrid schemes in which discrete tokens encode global structure while continuous residuals model fine details. These approaches achieve low reconstruction FID but require foundation-scale text-to-image retraining of the VLM.

Joint optimization with diffusion. A second line integrates diffusion directly into the autoregressive generation pipeline. X-Omni [6] aligns AR generation with diffusion-based decoding through reinforcement learning. UniFusion [24] treats a VLM as a unified encoder that conditions a diffusion model. Show-o [51] and ILLUME+ [15] unify multimodal understanding and generation in architectures that combine autoregressive modeling with diffusion-based decoding. These methods attain strong fidelity, but diffusion becomes a required part of generation: the VLM is jointly trained or realigned with the diffusion module, the lightweight VQ-VAE pathway is forfeited, and significant additional T2I-scale optimization is needed to preserve multimodal capabilities.

Post-hoc decoding. A third axis, the one occupied by our DCDD, leaves the VLM frozen and modifies only the decoding stage (Figure 2, *left*(c)). Prior work on this axis has been limited: text-conditioned image-to-image refiners [43, 31, 1] can be applied to VQ-VAE outputs without retraining the VLM, but because they are guided by their own pre-trained prior, they are prone to semantic drift from the VLM's intent. DCDD addresses this by conditioning the diffusion decoder directly on the VLM's pre-argmax logit distribution rather than on already-decoded images, so that the VLM's predictive belief guides the decoding (Section 5.4). Soft targets [10] and temperature

or label-smoothing calibration [8, 32] are established tools; DCDD's contribution is to identify the pre-sampling visual-token distribution of a frozen VLM as a recoverable signal, and to make it usable for decoder training through calibrated encoder proxies, without running the VLM during training. DCDD shows that a substantial fraction of the fidelity gap can be closed with decoder-only training on ImageNet-1K, without modifying the underlying VLM.

The three axes correspond to different deployment regimes. Tokenizer replacement and joint optimization with diffusion rebuild the generative system around a new decoding mechanism, which is appropriate when the VLM is being trained from scratch. Post-hoc decoding is appropriate when a pre-trained VLM is fixed by an upstream provider, when its broader capabilities should be preserved, or when foundation-scale retraining is not available. DCDD targets this third regime, showing that meaningful fidelity gains are achievable without foundation-scale retraining.

3 Conceptual Motivation

DCDD rests on two empirical observations about the logit distributions involved in discrete-token VLM image generation: VLM and VQ-VAE encoder logits share a structural codebook geometry, but differ sharply in their statistical concentration. We make these observations precise here, as they directly motivate the two non-trivial design choices in Section 4: using continuous logit distributions as conditioning (LCDM) and calibrating proxy logits before training (LC).

Logits over a shared codebook. At each generation step, the VLM transformer projects its hidden state into a vector of logits $l_t \in \mathbb{R}^K$, with one entry for each VQ-VAE codebook index. A discrete token index is sampled from this distribution, and only the selected index is passed to the VQ-VAE decoder. The VQ-VAE encoder, in turn, produces a latent feature $f_i \in \mathbb{R}^D$ at every spatial position and assigns it to the nearest codebook entry by L2 distance [28]. Thus, the VLM and the encoder operate over the same codebook-index space, although they arrive at it through different mechanisms: the VLM predicts a distribution over codebook indices, while the encoder assigns image-patch features to entries in the tokenizer latent space. We exploit this shared index space by converting the encoder's assignment scores into a continuous distribution over the codebook via cosine similarity (Section 4.2), yielding proxy logits that can be used during training in place of the VLM's inference-time logits.

Table 1: **Token-probability and Otsu thresholding statistics for Janus-Pro 7B.** VLM image-token logits versus VQ-VAE encoder proxy logits. The two distributions live in the same codebook space but differ by orders of magnitude in concentration: the VLM commits mass to roughly 2 entries per token, while the raw encoder spreads it over thousands. The third column shows the encoder proxy logits after Logit Calibration (Section 4.2), which restores sharpness comparable to the VLM.

Statistic	VLM	VQ-VAE	VQ-VAE (LC)
Probability Distribution Statistics			
Top-1 Probability	0.221627	0.000151	0.222283
Top-2 Probability	0.100532	0.000145	0.064229
Norm. Entropy	0.415303	0.993630	0.424155
Tail Entropy	0.560794	0.993661	0.520536
Otsu Thresholding Statistics			
Threshold (prob.)	6.24×10^{-2}	6.66×10^{-5}	6.38×10^{-2}
Threshold Rank	2	5767	2
Head Mass	0.322160	0.492722	0.286512

The two distributions are statistically incompatible. Although the VLM and encoder operate over the same codebook-index space, their probability distributions concentrate mass very differently. Table 1 reports token-probability and Otsu thresholding statistics for Janus-Pro 7B's image-token logits and the corresponding VQ-VAE encoder proxy logits. The VLM places concentrated mass on a small number of codebook entries (top-1 0.22, normalized entropy 0.42, Otsu threshold rank 2), reflecting a selective belief over plausible visual tokens. In contrast, the raw encoder proxy distributions are nearly uniform (top-1 1.5×10^{-4}, normalized entropy 0.99, Otsu threshold rank 5,767), spreading mass over thousands of codebook entries.

This flatness is not merely a numerical artifact; it follows from using bounded cosine similarities over a large codebook. The proxy logits $s_{i,k} = \cos(f_i, e_k)$ lie in $[-1, 1]$, and with $K = 16{,}384$ codebook entries, a top entry with cosine near 1 and most remaining entries near 0 yields a softmax probability of only approximately $\frac{e^1}{e^1+(K-1)e^0} \approx 1.6\times10^{-4}$, while the uniform probability is $1/K \approx 6.1\times10^{-5}$. Even under more favorable score separation, the top probability remains extremely small because of

the large vocabulary size. Consequently, the raw encoder distribution is only marginally sharper than uniform and does not match the selective probability structure of the VLM logits.

Implications. Two consequences follow. First, reducing each distribution to a single sampled token index discards rank-2 or higher information that the VLM has already computed (Table 1) [10]. Conditioning on a distribution-aware representation, rather than only the sampled index, provides a compact way to retain this additional codebook information. This motivates LCDM (Section 4.1). Second, proxy logits cannot be used directly during training, because their high-entropy, near-uniform shape differs sharply from the inference-time VLM logits that the diffusion decoder will receive at deployment. Bridging this gap requires aligning the encoder proxy distribution with the VLM's statistical regime, which motivates Logit Calibration (Section 4.2). As shown in the third column of Table 1, Logit Calibration substantially closes this gap, restoring the encoder's Otsu rank to 2 and bringing its top-1 probability and head mass close to those of the VLM by matching a small set of distributional statistics.

4 Method

We now describe the three components of the DCDD framework: Logit-to-Code Distributional Mapping (LCDM, Section 4.1), Logit Calibration (LC, Section 4.2), and the Diffusion decoder (DD, Section 4.3) built on the denoising generative modeling paradigm [11, 33]. An overview is shown in Figure 2, *right*. The detailed formulation, including the full set of equations, training objective, and architectural details, is provided in Appendix A.

4.1 Logit-to-Code Distributional Mapping (LCDM)

LCDM converts the VLM's per-token logits into the continuous representation that conditions the diffusion decoder. Given *image-token logits* $l_i \in \mathbb{R}^K$ at spatial position i, we form the probability distribution $p_{i,k} = \mathrm{softmax}(l_i)_k$ over codebook entries and compute the *distribution-weighted code vector* $v_i = \sum_{k=1}^{K} p_{i,k} e_k \in \mathbb{R}^D$, which is the expected codebook embedding under p_i. As established in Section 3, this provides a compact distribution-aware alternative to the standard argmax [10]: it retains the rank-2 or higher information that the VLM places on multiple codebook entries per token, rather than collapsing each token to a single index.

The expected embedding alone, however, hides the shape of p_i. Two tokens with very different sharpness can produce a similar v_i. To preserve this distinction, we additionally extract four scalar *uncertainty features* from p_i: the maximum probability, the top-1/top-2 confidence margin, an entropy-adaptive top-K_u probability mass, and the normalized tail entropy. These are concatenated into a per-token uncertainty vector $u_i \in \mathbb{R}^4$. Together, (v_i, u_i) form the conditioning input to the decoder. Full definitions are in Appendix A.1.

4.2 Logit Calibration (LC)

Ideally, the diffusion decoder would be trained on logits produced by the VLM itself, so that its training-time inputs match what it will see at inference. This is impractical at scale: producing VLM logits for every training image requires running the large VLM on each example, which dominates the cost of training. We instead obtain *proxy logits* from the VQ-VAE encoder: for a ground-truth image, we compute cosine similarity $s_{i,k} = \cos(f_i, e_k)$ between each image-token feature and every codebook entry, and apply the same LCDM transform.

As Section 3 establishes, however, these proxy logits are statistically incompatible with the VLM's inference-time logits: they are orders of magnitude flatter, with an Otsu threshold rank of 5,767 instead of 2. A decoder trained on raw proxy distributions and deployed on VLM distributions would therefore face a sharp train-inference mismatch. LC mitigates this mismatch by aligning four distributional statistics (mean entropy, mean confidence, 95th-percentile entropy, and 95th-percentile confidence) through a gradient-free scale, temperature, and label-smoothing transform (a, α, ϵ) applied to the proxy logits [8, 32]. Target statistics are estimated once from the VLM over 100 reference prompts from the GenEval benchmark [7]; the transform introduces no learnable parameters and is fixed before decoder training begins. The optimization procedure and full derivation are provided in Appendix A.2. Appendix E further analyzes sensitivity to the number and domain of reference prompts, showing that 100 GenEval prompts fall within a stable calibration regime.

Although LC matches only global statistics, the calibrated proxies agree with actual VLM logits position by position: their distribution-weighted code vectors reach 0.86 cosine similarity with those computed from the VLM, which drops to 0.04 when code identities are permuted while all global statistics are kept (Appendix C.1). The agreement therefore stems from the shared codebook geometry of Section 3, which LC preserves because its transform cannot reorder a distribution.

4.3 Diffusion Decoder (DD)

The conditioning representation (v_i, u_i) is consumed by a diffusion decoder [41, 37, 22] that operates in the latent space of a pre-trained VAE [19]. We finetune FLUX.1-Fill [22], a recent rectified-flow transformer [4, 29, 30] for image inpainting, well-suited to localized, token-aligned conditioning, by updating only its attention and adapter layers. The distribution-weighted code vector v_i is spatially reshaped, projected through a linear layer and a 3×3 convolution, and added to the noisy latent tokens; uncertainty features are bilinearly upsampled and concatenated channel-wise. Training uses the v-prediction objective [18] on ImageNet-1K only, without any text-to-image supervision. The decoder is conditioned only on VLM-derived token representations and excludes direct text conditioning by design, ensuring it follows the VLM's original token belief rather than reinterpreting the scene with its own prior. Full architecture and conditioning-injection details are in Appendix A.3.

5 Experiments

5.1 Setup

We train the diffusion decoder on the ImageNet-1K training set [42] and evaluate on ImageNet-1K validation (in-domain) and MJHQ-30K [23] (out-of-domain). For reconstruction, we measure fidelity by rFID [9], SSIM [49], and LPIPS [55]. For text-to-image generation, we measure fidelity by gFID on MJHQ-30K, and semantic preservation by GenEval [7], DPG-Bench [14], and WISE [34]. Semantic-preservation metrics are reported to verify that fidelity gains do not come at the cost of altering the VLM's intent. We finetune FLUX.1-Fill [22] as the decoder, updating only its attention and adapter layers, on 4×A100 GPUs with batch size 32 and 50K steps; inference uses 30 denoising steps. The default backbone is Janus-Pro 7B [2] at 384×384 image resolution; we additionally evaluate Janus-Pro 1B and a second backbone, LlamaGen-XL (256×256), in Section 5.7. Full hyperparameters and the calibration procedure are in Appendix B, and the main results are stable across diffusion-noise seeds (Appendix D.6).

5.2 Reconstruction Fidelity

Image reconstruction from VQ-VAE-encoded representations isolates the decoder's contribution from token-prediction error and reflects the fidelity ceiling of the decoder under correct token sequences. Table 2 reports the result. DCDD reduces rFID by 74% on ImageNet-1K and 60% on MJHQ-30K, with consistent improvements in SSIM and LPIPS. The MJHQ-30K result, where training is performed only on ImageNet-1K, indicates that the gain is not domain-locked. This confirms that the VQ-VAE decoder, rather than the upstream representation, is the dominant source of reconstruction-stage fidelity loss. Qualitative samples are provided in Appendix Figure 9.

Table 2: **Image reconstruction on ImageNet-1K and MJHQ-30K** (Janus-Pro 7B). Reconstruction from VQ-VAE-encoded representations, comparing the native VQ-VAE decoder against DCDD. The decoder is trained on ImageNet-1K only, so the MJHQ-30K result tests cross-domain generalization. DCDD substantially reduces rFID across both datasets while consistently improving SSIM and LPIPS. Superior results are highlighted.

ImageNet-1K Val.	rFID↓	SSIM↑	LPIPS↓
VQ-VAE Decoder	2.8511	0.5786	0.1541
DCDD (ours)	0.7535	0.5957	0.1374
MJHQ-30K	rFID↓	SSIM↑	LPIPS↓
VQ-VAE Decoder	4.4882	0.6564	0.1432
DCDD (ours)	1.8127	0.6782	0.1308

5.3 Text-to-Image: Fidelity and Semantic Preservation

We next apply DCDD to tokens generated by the VLM from text prompts. Table 3 reports per-category gFID on MJHQ-30K and semantic alignment scores across three independent benchmarks (GenEval, DPG-Bench, WISE). DCDD reduces gFID by 34% (14.29 → 9.42) on MJHQ-30K, with consistent

Table 3: **Text-to-image evaluation** (Janus-Pro 7B). DCDD is designed to improve fidelity while preserving the VLM's semantic intent. The fidelity benchmark (MJHQ-30K gFID) shows substantial gains across nearly all categories. We report GenEval, DPG-Bench, and WISE as semantic consistency checks: because DCDD is a decoder-only fidelity module conditioned on the VLM's own token belief, these scores are expected to remain stable rather than increase substantially. Best per metric is highlighted.

Fidelity Metric ↓											
MJHQ-30K	**Overall**	Animals	Art	Fashion	Food	Indoor	Landscape	Logo	People	Plants	Vehicles
VQ-VAE Decoder	14.2888	31.6866	35.6408	32.1410	33.1787	32.4016	30.5020	42.3565	37.6049	33.1228	27.4424
DCDD (ours)	9.4160	25.1060	28.9806	26.0182	28.1580	28.5172	26.7104	38.2032	30.2217	33.6617	25.0277

Semantic Consistency Checks							
GenEval	**Overall**	Single	Two	Counting	Colors	Position	Color Attr.
VQ-VAE Decoder	77.24	97.81	84.09	56.56	87.50	75.75	61.75
DCDD (ours)	78.31	99.06	88.89	59.38	88.30	72.50	61.75
DPG-Bench	**Overall**	Global	Relation	Entity	Attribute	Others	–
VQ-VAE Decoder	86.94	82.37	93.38	89.76	87.61	81.60	–
DCDD (ours)	87.64	83.59	93.42	90.05	87.15	82.00	–
WISE	**Consistency**	Realism	Aesthetic Quality	–	–	–	–
VQ-VAE Decoder	1.278	0.936	0.923	–	–	–	–
DCDD (ours)	1.320	1.107	1.173	–	–	–	–

improvements across nine of ten categories. By contrast, the semantic alignment metrics show only small movement: GenEval overall +1.07, DPG-Bench overall +0.70, and WISE Consistency +0.042. The larger sub-axis movements within each benchmark (GenEval Single +1.25, Two +4.80; WISE Realism +0.171, Aesthetic +0.250) are concentrated on axes that depend on object-level visual detectability and rendering quality rather than on alignment itself. Sharper boundaries, cleaner textures, and more realistic appearance make objects more reliably recognized by detection-based evaluators and rated as more visually plausible by model-based evaluators, even when the semantics are unchanged. Qualitative samples are provided in Appendix Figure 10.

5.4 Logit-Level Conditioning vs. Image-to-Image Refinement

A natural alternative to DCDD is to leave the VQ-VAE decoder in place and apply an off-the-shelf image-to-image (I2I) refiner to its output. We compare against FLUX.1-Kontext [22], which shares the same backbone family as our Diffusion Decoder, applied as an I2I refiner in two variants: a generic enhancement pass with a fixed quality-improvement prompt (*I2I (generic)*) and a pass that receives the original MJHQ-30K generation prompt (*I2I + prompt*). As shown in Figure 3, this approach either under-modifies or alters scene content in ways that drift from the VLM's intent.

Table 4: **Image-to-image refinement vs. logit-level conditioning on MJHQ-30K** (Janus-Pro 7B). gFID is computed against MJHQ-30K reference images; CLIP-I and DINOv2 measure semantic preservation of the refined output relative to the VQ-VAE-decoded image (the VLM's intended output). *I2I (generic)* uses a fixed quality-improvement prompt (detailed in Figure 3 caption); *I2I + prompt* uses the original MJHQ-30K generation prompt. Superior results are highlighted.

MJHQ-30K	gFID↓	CLIP-I↑	DINOv2↑
VQ-VAE	14.2888	–	–
I2I (generic)	10.6678	0.8752	0.7663
I2I + prompt	10.9767	0.8531	0.7586
DCDD (ours)	9.4160	0.9088	0.8692

To quantify this trade-off, Table 4 reports gFID alongside two semantic-preservation metrics measured against the VQ-VAE-decoded image (the VLM's intended output) on MJHQ-30K. CLIP-I [38] cosine similarity captures high-level semantic agreement such as scene category and salient objects; DINOv2 [35] cosine similarity is more sensitive to object identity and spatial layout, flagging cases where the broad category is preserved but the specific subject has drifted.

The I2I refiner also achieves a fidelity gain on gFID, but at the cost of lower CLIP-I and DINOv2 similarity to the VQ-VAE output, indicating semantic drift from the VLM's intent. Both I2I variants exhibit similar drift regardless of prompt strategy, with DINOv2 nearly identical (0.7663 vs. 0.7586), suggesting the refiner's structural rewrites are governed by its own pre-trained prior rather than by the conditioning image. DCDD, in contrast, attains the strongest fidelity improvement while preserving both similarity metrics at higher values, demonstrating that logit-level conditioning enhances visual quality without overwriting the VLM's semantic intent.



Figure 3: **Image-to-image refinement vs. our logit-conditioned decoding.** Qualitative comparison on Janus-Pro 7B generations from MJHQ-30K prompts. From left to right: native VQ-VAE decoding; FLUX.1-Kontext with a generic enhancement prompt (*"enhance to high quality, preserve all original details and composition, sharpen fine details, same scene, same content, same colors, best quality"*); FLUX.1-Kontext conditioned on the original MJHQ-30K generation prompt; and DCDD. FLUX-based I2I refinement either under-modifies the image or alters the semantics substantially, while DCDD improves visual fidelity while preserving the VLM's semantic intent.

5.5 Component Contributions

A strong pretrained diffusion decoder may account for part of the observed improvement. To separate this effect from that of distributional conditioning, we train a baseline diffusion decoder conditioned only on the code vectors of sampled tokens. This baseline uses the same FLUX.1-Fill initialization, trainable layers, training data, optimizer, batch size, 50K training steps, and inference schedule as DCDD, differing only in its conditioning. As shown in Table 5, the diffusion decoder alone

Table 5: **Isolating distributional conditioning** (Janus-Pro 7B). Both diffusion decoders share initialization, trainable layers, data, and the 50K-step schedule; only the conditioning differs. Superior results are highlighted.

	rFID↓		gFID↓	GenEval↑
Conditioning	IN-1K	MJHQ	MJHQ	Overall
VQ-VAE Decoder	2.8511	4.4882	14.2888	77.24
Diff. Dec. (argmax codes)	1.1842	2.4325	11.0837	78.08
DCDD (full distribution)	0.7535	1.8127	9.4160	78.31

Table 6: **Top-k truncation at inference** (Janus-Pro 7B, fixed decoder and tokens). gFID on the MJHQ-30K Animals category.

Condition	gFID↓	GenEval↑
VQ-VAE Decoder	31.6866	77.24
Top-1 only	33.2015	76.07
Top-2	27.0649	77.83
Top-8	25.5681	77.90
Full (DCDD)	25.1060	78.31

reduces gFID from 14.29 to 11.08, while full-distribution conditioning further reduces it to 9.42. This corresponds to a 15% improvement over the argmax-conditioned baseline, with a similar benefit in reconstruction (36% lower rFID on ImageNet-1K). Since the two decoders share the same architecture and training budget, the additional improvement reflects the contribution of LCDM and LC rather than increased decoder capacity. Most of the GenEval improvement is already achieved with argmax conditioning, consistent with the detectability effect discussed in Section 5.3. We also evaluate a lightweight learned adapter that feeds soft codes into the frozen native decoder (Appendix D.5). Although it improves reconstruction, it does not improve generation, suggesting that directly adapting the frozen native decoder is insufficient to fully exploit token distributions.

How much of the distribution is needed? To examine the contribution of lower-ranked tokens, we retain only the top-k probabilities at each position and recompute (v_i, u_i), keeping the decoder and generated token sequences fixed (Table 6). Including the second-ranked token recovers 76% of the improvement between top-1 and full-distribution conditioning. Tokens ranked 3–8 account for another 18%, while the remaining tail contributes the rest. This is consistent with the Otsu analysis in Section 3, which identifies rank 2 as the boundary between informative tokens and the noise floor. Since the decoder is trained on full distributions, this analysis serves as a diagnostic rather than an alternative inference configuration.

Finally, stepwise ablations (Appendix D.3) show that LCDM contributes most to the rFID reduction, while LC provides the largest SSIM gain. Figure 4 illustrates the distinct failure modes addressed by each component.

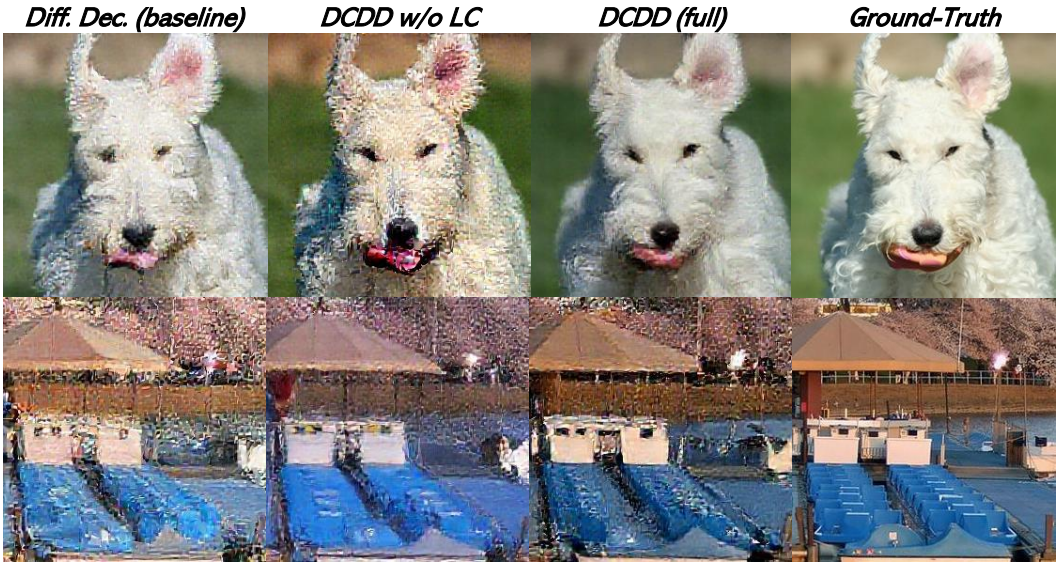


Figure 4: **Component ablation on ImageNet-1K reconstruction at 10K training steps.** *(a)* Discrete conditioning (***Diff. Dec. (baseline)***) produces low-frequency noise and broken textures. *(b)* Adding continuous distribution-weighted conditioning (***DCDD w/o LC***) recovers structure but inflates contrast and introduces residual color noise. *(c)* Adding calibration (***DCDD (full)***) stabilizes color tones and suppresses noise. Each transition addresses a specific failure mode rather than incrementally improving along a single axis.

Table 8: **Regime of applicability.** VLM logit statistics and the corresponding effect of DCDD on generation performance (VQ-VAE → DCDD). gFID is evaluated on MJHQ-30K Animals for all models. †Janus-Pro 1B uses the 7B-trained DCDD checkpoint without retraining.

Model	Otsu Rank	Top-1	Norm. Entropy	gFID↓ (Animals)	GenEval↑
Janus-Pro 7B	2	0.2216	0.4153	31.69 → **25.11**	77.24 → 78.31
Janus-Pro 1B †	2	0.2457	0.4033	35.14 → **26.09**	63.46 → 62.17
LlamaGen-XL	1	0.3794	0.3278	74.20 → 74.66	30.02 → 28.19

5.6 Computational Cost

Table 7: **End-to-end T2I cost on MJHQ-30K** (Janus-Pro 7B, batch size 1, A100 80GB). DCDD adds moderate inference latency for a substantial fidelity gain. The native VQ-VAE path remains available when efficiency is preferred.

Configuration	gFID ↓	Latency (s) ↓
VLM + VQ-VAE	14.28	11.86
VLM + DCDD (ours)	9.41	15.56
Trade-off	−34.12%	+31.19%

With 30 denoising steps, DCDD adds 31% end-to-end latency for a 34% gFID reduction on MJHQ-30K (Table 7). The end-to-end cost is dominated by the VLM rather than the decoder, and the native VQ-VAE path remains available when efficiency is preferred. On the training side, the decoder is trained on ImageNet-1K alone for 50K steps on 4×A100; the VLM and VAE are not modified, in contrast to tokenizer-replacement and joint-optimization alternatives that require foundation-scale T2I retraining. Even with 10 steps, DCDD outperforms the native decoder in reconstruction at only 10.7% additional latency (Appendix D.2).

5.7 Regime of Applicability

A post-hoc decoder can only exploit information available in its input. We therefore examine two additional models (Table 8). Janus-Pro 1B [2] shares the tokenizer with Janus-Pro 7B, allowing us to reuse the 7B-trained DCDD checkpoint without retraining. We align the 1B logits to the 7B statistics using the same gradient-free procedure as LC (Appendix F.3). With similar logit characteristics (Otsu rank 2, top-1 0.25), DCDD reduces gFID by 25.7% on MJHQ-30K Animals, compared with 20.8% for Janus-Pro 7B, while GenEval decreases by only 1.3 points. In contrast, LlamaGen-XL has a more concentrated distribution (Otsu rank 1) and produces weaker token sequences. DCDD provides no text-to-image improvement, although reconstruction gains remain (Appendix F). These results suggest that DCDD is most effective when upstream token sequences are semantically coherent and the logits retain meaningful probability mass beyond rank 1.

5.8 Additional Analyses

Several robustness and diagnostic analyses are deferred to the appendix. Appendix C.1 verifies that calibrated proxies and VLM logits induce compatible decoder conditioning, and Appendix C.2 analyzes the stability of that conditioning; Appendix D.1 and Appendix D.2 examine the number of denoising steps, latency, and memory; Appendix E verifies that Logit Calibration is stable across the size and content of its reference prompt set, ruling out benchmark-specific bias; Appendix C confirms that LC reshapes the encoder's probability decay to closely track the VLM's, as motivated in Section 3; and Appendix D.6 reports variation across diffusion-noise seeds.

6 Conclusion

We presented DCDD, a post-hoc decoder framework built on a simple observation about pre-trained discrete-token VLMs: before sampling, the VLM's logit distribution carries rank-2 or higher codebook information that standard decoding discards after selecting a single token index. DCDD combines Logit-to-Code Distributional Mapping, gradient-free Logit Calibration, and a Diffusion Decoder to recover this discarded signal as a high-fidelity image, serving as an optional alternative to the native VQ-VAE decoder while leaving the VLM frozen. Trained on ImageNet-1K alone, DCDD substantially improves reconstruction and generation fidelity while keeping semantic consistency stable. More broadly, our results show that useful distributional information can be recovered post-hoc from a frozen discrete-token model's logits.

Acknowledgments and Disclosure of Funding

This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the Global Research Support Program in the Digital Field program(RS-2024-00436680) supervised by the IITP(Institute for Information Communications Technology Planning Evaluation). This project is supported by Microsoft Research Asia. The authors declare no competing financial interests beyond those disclosed above.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[2] 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.

[3] SooHwan Eom, Hee Suk Yoon, Eunseop Yoon, Mark Hasegawa-Johnson, and Chang D. Yoo. Transcript-free flow-matching text-to-speech via speech feature conditioning. *arXiv preprint arXiv:2606.20266*, 2026.

[4] 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 *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[5] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[6] Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, Shuyang Gu, Zhao Zhong, Qinglin Lu, Han Hu, Xiaosong Zhang, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025.

[7] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

[8] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

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

[13] Ji Woo Hong, Tri Ton, Trung X. Pham, Gwanhyeong Koo, Sunjae Yoon, and Chang D. Yoo. ITA-MDT: Image-timestep-adaptive masked diffusion transformer framework for image-based virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[14] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

[15] Runhui Huang, Chunwei Wang, Junwei Yang, Guansong Lu, Yunlong Yuan, Jianhua Han, Lu Hou, Wei Zhang, Lanqing Hong, Hengshuang Zhao, et al. Illume+: Illuminating unified mllm with dual visual tokenization and diffusion refinement. *arXiv preprint arXiv:2504.01934*, 2025.

[16] Jaehyun Jang, Eunseop Yoon, Hee Suk Yoon, SooHwan Eom, Mark A. Hasegawa-Johnson, and Chang D. Yoo. Token-level response-visual attention guidance for multimodal LLMs knowledge distillation. *arXiv preprint arXiv:2607.02593*, 2026.

[17] Kyuwon Kim, Sunjae Yoon, and Chang D. Yoo. SPECSIA: Stylization dataset for novel-view enhancement in drawing-based 3D animation. *arXiv preprint arXiv:2607.00525*, 2026.

[18] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Gwanhyeong Koo, Hyunsu Kim, Youngji Kim, Taejae Lee, Siwoo Lim, Sunjae Yoon, Suyong Yeon, and Chang D. Yoo. InSpace: Structure-aware 3D indoor scene generation from a single 360° image. *arXiv preprint arXiv:2607.03990*, 2026.

[21] Gwanhyeong Koo, Sunjae Yoon, Younghwan Lee, Ji Woo Hong, and Chang D. Yoo. FlowDrag: 3D-aware drag-based image editing with mesh-guided deformation vector flow fields. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 31467–31485, 2025.

[22] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[23] Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation, 2024.

[24] Kevin Li, Manuel Brack, Sudeep Katakol, Hareesh Ravi, and Ajinkya Kale. Unifusion: Vision-language model as unified encoder in image generation. *arXiv preprint arXiv:2510.12789*, 2025.

[25] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[26] Siwoo Lim, Sunjae Yoon, Gwanhyeong Koo, and Chang D. Yoo. GADA: Geometry-aware deformable aggregation for image-based Gaussian splatting. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

[27] Siwoo Lim, Sunjae Yoon, Gwanhyeong Koo, Hyeonseo Yun, and Chang D. Yoo. TanGO: Training-free 3D editing via tangent-space guidance and optimization. *arXiv preprint arXiv:2607.14927*, 2026.

[28] Yoseph Linde, Andres Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

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

[30] 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.

[31] 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.

[32] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[33] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.

[34] Yuwei Niu, Munan Ning, Mengren Zheng, Weiyang Jin, Bin Lin, Peng Jin, Jiaqi Liao, Chaoran Feng, Kunpeng Ning, Bin Zhu, et al. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2025.

[35] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[36] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[37] 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.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 8748–8763. PMLR, 2021.

[39] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021.

[40] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[44] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[45] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812*, 2024.

[46] Tri Ton, Ji Woo Hong, and Chang D. Yoo. TARO: Timestep-adaptive representation alignment with onset-aware conditioning for synchronized video-to-audio synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14228–14237, 2025.

[47] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[48] Yuqing Wang, Zhijie Lin, Yao Teng, Yuanzhi Zhu, Shuhuai Ren, Jiashi Feng, and Xihui Liu. Bridging continuous and discrete tokens for autoregressive visual generation. *arXiv preprint arXiv:2503.16430*, 2025.

[49] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[50] Yecheng Wu, Han Cai, Junyu Chen, Zhuoyang Zhang, Enze Xie, Jincheng Yu, Junsong Chen, Jinyi Hu, Yao Lu, and Song Han. Dc-ar: Efficient masked autoregressive image generation with deep compression hybrid tokenizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18034–18045, 2025.

[51] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.

[52] Hee Suk Yoon, Eunseop Yoon, Ji Woo Hong, SooHwan Eom, Gwanhyeong Koo, Mark Hasegawa-Johnson, Qi Dai, Chong Luo, and Chang D. Yoo. PDCR: Perception-decomposed confidence reward for vision-language reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[53] Hee Suk Yoon, Eunseop Yoon, Jaehyun Jang, SooHwan Eom, Ji Woo Hong, Mark Hasegawa-Johnson, Qi Dai, Chong Luo, and Chang D. Yoo. Decomposed on-policy distillation for vision-language reasoning: Steering gradients for visual grounding. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

[54] 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. *arXiv preprint arXiv:2206.10789*, 2022.

[55] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Extended Method Details

This section provides the full formulation of DCDD's three components, expanding on the main-paper description in Section 4. Figure 5 provides a detailed view of the framework, showing the per-stage tensor shapes and the train/inference data flow. The remainder of the section presents Logit-to-Code Distributional Mapping (Appendix A.1), Logit Calibration (Appendix A.2), and the Diffusion Decoder (Appendix A.3) in turn.

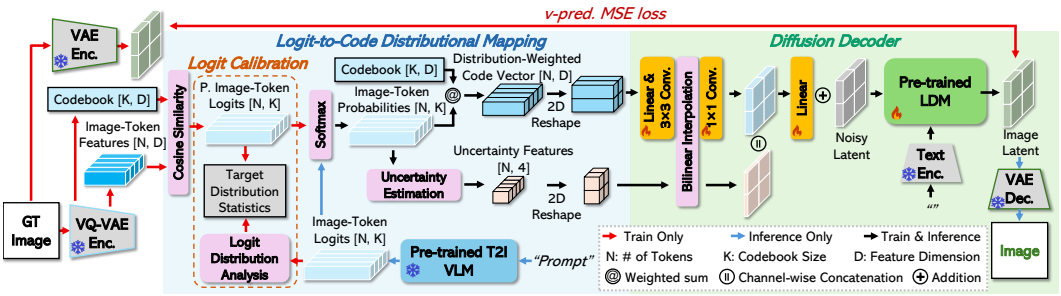


Figure 5: **Detailed view of the DCDD framework.** An expanded version of Figure 2 showing per-stage tensor shapes and explicit train/inference data flow. *Inference:* the frozen VLM produces image-token logits, which LCDM converts into continuous distribution-weighted code vectors and uncertainty features that condition the Diffusion Decoder. *Training:* ground-truth images pass through the frozen VQ-VAE encoder; cosine similarity to the codebook produces proxy image-token logits, which gradient-free Logit Calibration aligns to the VLM's logit statistics before feeding the same LCDM stage.

A.1 Logit-to-Code Distributional Mapping

This stage transforms the image-token logits generated from a pre-trained autoregressive VLM into continuous conditioning representations that enrich the diffusion decoder with dense, fine-grained visual information for high-fidelity image synthesis.

Given either the ***Image-Token Logits*** from the VLM during inference or the ***Proxy Image-Token Logits*** derived from the VQ-VAE encoder during training, this mapping converts these logits into two key representations: the ***Distribution-Weighted Code Vector*** capturing the expected semantic embedding per token, and the complementary ***Uncertainty Features*** quantifying confidence and ambiguity in token prediction.

Inference with VLM:

During inference, the autoregressive VLM produces a sequence of ***Image-Token Logits***

$$\mathbf{L} = [\mathbf{l}_1, \ldots, \mathbf{l}_N] \in \mathbb{R}^{N \times K}, \tag{1}$$

where N denotes the number of image tokens and K the size of the visual codebook. Each $\mathbf{l}_i = [l_{i,1}, \ldots, l_{i,K}]$ contains scores over the K **Code Vectors** $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ with $\mathbf{e}_k \in \mathbb{R}^D$, produced by the VLM's output projection from its hidden state. In practice, the logits may be obtained after applying classifier-free guidance and/or temperature scaling; for notational simplicity, we omit these operators and write the resulting distribution in a standard softmax form.

The logits are transformed into normalized ***Image-Token Probabilities*** via a standard softmax:

$$p_{i,k} = \frac{\exp(l_{i,k})}{\sum_{j=1}^{K} \exp(l_{i,j})}, \quad \mathbf{P} = [p_{i,k}] \in \mathbb{R}^{N \times K}. \tag{2}$$

Each row of $\mathbf{P}$ represents the probability distribution over the K codebook entries for a specific spatial token position. To convert these distributions into continuous embeddings, each probability vector $\mathbf{p}_i \in \mathbb{R}^K$ is used to compute a weighted combination of the corresponding **Code Vectors** in the codebook $\mathbf{E} = [\mathbf{e}_1; \ldots; \mathbf{e}_K] \in \mathbb{R}^{K \times D}$:

$$\mathbf{v}_i = \sum_{k=1}^{K} p_{i,k}\, \mathbf{e}_k, \quad \mathbf{v}_i \in \mathbb{R}^D. \tag{3}$$

Stacking all token embeddings yields the matrix form:

$$\mathbf{V} = \mathbf{P}\,\mathbf{E} \in \mathbb{R}^{N\times D}. \tag{4}$$

This operation performs a probability-weighted aggregation of the codebook features, projecting each token's probability distribution $\mathbf{p}_i$ into the continuous latent space of the VQ-VAE codebook. The resulting ***Distribution-Weighted Code Vectors*** $\mathbf{V}$ preserve the distributional information contained in the VLM's predictions.

However, this weighted aggregation inherently compresses each token's probability distribution into a single expectation, discarding information about its sharpness and variability. To compensate for this loss and explicitly capture the model's confidence and ambiguity, we derive four scalar statistics from $\mathbf{P}$ for each spatial token i:

$$\begin{aligned}
u_i^{(1)} &= \max_k p_{i,k} && \text{(confidence)},\\
u_i^{(2)} &= p_{i,(1)} - p_{i,(2)} && \text{(margin)},\\
u_i^{(3)} &= \textstyle\sum_{k\in \text{Top-}K_u} p_{i,k} && \text{(top-}K_u\text{ mass)},\\
u_i^{(4)} &= -\frac{\sum_{k\neq k^*} p_{i,k}\log p_{i,k}}{\log(K-1)} && \text{(tail entropy)},
\end{aligned} \tag{5}$$

where $p_{i,(1)}$ and $p_{i,(2)}$ denote the top-1 and top-2 probabilities, and $k^* = \arg\max_k p_{i,k}$. For the top-K_u mass, we employ an entropy-adaptive K_u constrained between 8 and 64 tokens, ensuring that the statistic reflects an appropriate portion of the probability mass across varying distribution sharpness. These measures are concatenated into ***Uncertainty Features*** $\mathbf{U} \in \mathbb{R}^{N\times 4}$, which complement the continuous code representations $\mathbf{V}$ by summarizing the dispersion characteristics of each token's probability distribution. Together, $(\mathbf{V}, \mathbf{U})$ provide the diffusion decoder with both semantic and confidence-aware information.

Training with VQ-VAE Encoder:

During training, we generate ***Proxy Image-Token Logits*** using the ***VQ-VAE encoder*** from the target VLM's tokenizer. Given a ground-truth image $x \in \mathbb{R}^{3\times H\times W}$, the VQ-VAE encoder downsamples the image through a series of convolutional and residual blocks, producing spatial latent features immediately before vector quantization. These latent features, referred to as ***Image-Token Features*** $\mathbf{F}$, capture compact yet semantically rich representations of local regions, encoding their color, texture, and structural patterns. Thus, we obtain

$$\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_N] \in \mathbb{R}^{N\times D}, \tag{6}$$

from the encoder, where N is the number of spatial tokens and D is the feature dimension.

The original VQ-VAE pipeline selects each $\mathbf{f}_i$'s nearest codebook entry by L2 distance. To produce a continuous distribution over the codebook rather than a single index, we instead compute cosine similarity between $\mathbf{f}_i$ and each entry in the learned ***Codebook*** $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ with $\mathbf{e}_k \in \mathbb{R}^D$, yielding the ***Proxy Image-Token Logits***:

$$s_{i,k} = \cos(\mathbf{f}_i, \mathbf{e}_k), \quad \mathbf{S} = [s_{i,k}] \in \mathbb{R}^{N\times K}. \tag{7}$$

We then apply the same LCDM transform as in inference (Eqs. (2)–(5)) to obtain the ***Distribution-Weighted Code Vectors*** and ***Uncertainty Features*** for diffusion conditioning:

$$\mathbf{V}^{\text{enc}} = \mathbf{P}^{\text{enc}}\mathbf{E} \in \mathbb{R}^{N\times D}, \qquad \mathbf{U}^{\text{enc}} \in \mathbb{R}^{N\times 4}.$$

For diffusion training, it is essential to establish explicit correspondence between each ground-truth image and its conditioning representation. The above process enables this by constructing ***Proxy Image-Token Logits*** that closely align with the inference-time outputs of the VLM. Although these proxy logits share the same codebook structure as the VLM's generated logits, their probability distributions may diverge, leading to a potential train-inference gap. To mitigate this mismatch, we apply **Logit Calibration**, as described in the following section.

A.2 Logit Calibration

Although the ***Proxy Image-Token Logits*** produced from the VQ-VAE encoder live in the same codebook space as the VLM's inference-time logits, their statistical behavior differs significantly from that of the pre-trained VLM. In particular, the encoder's cosine-similarity-based logits tend to exhibit substantially higher entropy and weaker confidence than the VLM's image-token logits, which encode a selective belief over a small set of semantically relevant codebook entries (Section 3). To bridge this discrepancy, we introduce a **Logit Calibration** procedure that aligns the encoder's token-probability distribution with the VLM's intrinsic confidence and entropy characteristics.

Target Distribution Statistics. We first estimate the ***Target Distribution Statistics*** of the VLM by sampling the pre-trained text-to-image model over a set of random prompts. Specifically, we draw $N = 100$ prompts from the GenEval benchmark and generate $T = 576$ image tokens per prompt using classifier-free guidance [12] with weight $w_{\text{cfg}} = 5.0$ and temperature $\tau_{\text{VLM}} = 1.0$. For each decoding step, we obtain the guided image-token distribution $p^{\text{VLM}}(\cdot)$ and compute its token-level confidence $\max_k p_k$ and normalized entropy

$$H_{\text{norm}}(p) = -\frac{1}{\log K} \sum_{k=1}^{K} p_k \log p_k, \tag{8}$$

where $K = 16{,}384$ is the size of the image-token vocabulary. Aggregating these quantities across all $N \times T$ tokens yields the VLM's target statistics

$$\mathcal{T}_{\text{VLM}} = \big(\mu_H,\ \mu_C,\ P_{95}(C),\ P_{95}(H)\big), \tag{9}$$

where μ_H and $P_{95}(H)$ denote the mean and 95th-percentile entropy, and μ_C and $P_{95}(C)$ denote the corresponding confidence statistics.

Encoder Logits and Calibration Transform. On the encoder side, we evaluate the VQ-VAE encoder on a set of $M = 100$ training images. For each latent image-token feature $\mathbf{f}_i \in \mathbb{R}^D$, we compute the cosine-similarity logits

$$s_{i,k} = \cos(\mathbf{f}_i, \mathbf{e}_k), \tag{10}$$

where $\mathbf{e}_k \in \mathbb{R}^D$ denotes the k-th codebook vector. These logits are then transformed via affine scaling and a temperature-controlled softmax:

$$\tilde{s}_{i,k} = a\, s_{i,k}, \qquad p^{\text{enc}}_{i,k} = \frac{\exp(\tilde{s}_{i,k}/\alpha)}{\sum_{j=1}^{K} \exp(\tilde{s}_{i,j}/\alpha)}, \tag{11}$$

where a denote the global logit scale, and α controls the sharpness of the resulting distribution. After softmax, we apply label smoothing

$$\hat{p}_{i,k} = (1-\epsilon)\, p^{\text{enc}}_{i,k} + \frac{\epsilon}{K}, \tag{12}$$

which mitigates overconfident predictions and stabilizes the tail behavior of the distribution.

Calibration Objective and Optimization. From the calibrated encoder distributions $\hat{p}_{i,k}$, we compute the corresponding entropy and confidence statistics $\mu^{\text{enc}}_H, \mu^{\text{enc}}_C, P_{95}(C)^{\text{enc}}, P_{95}(H)^{\text{enc}}$ and match them to the VLM's target statistics in Eq. (9). To quantify the mismatch, we define the weighted objective

$$\begin{aligned} \mathcal{L}(\theta) = {} & w_1(\mu^{\text{enc}}_H - \mu_H)^2 + w_2(\mu^{\text{enc}}_C - \mu_C)^2 \\ & + w_3(P_{95}(C)^{\text{enc}} - P_{95}(C))^2 \\ & + w_4(P_{95}(H)^{\text{enc}} - P_{95}(H))^2, \end{aligned} \tag{13}$$

where $\theta = (a, \alpha, \epsilon)$ and $(w_1, w_2, w_3, w_4) = (1.0,\ 0.25,\ 0.25,\ 0.35)$. We prioritize matching the mean entropy, which provides the most reliable characterization of distributional sharpness, while confidence-based terms serve as secondary constraints.

The calibration parameters are determined via a coordinated, gradient-free search. We first tune the global logit scale a over $[1,\ 60]$ using bisection until the encoder's mean entropy satisfies $|\mu^{\text{enc}}_H -$

$\mu_H| \leq 0.02$, fixing $(\alpha, \epsilon) = (1.0, 0)$. Next, we sweep $\alpha \in [0.5, 2.0]$ to refine the distribution's softness, followed by a search over $\epsilon \in [0.0, 0.05]$ to align the 95th-percentile confidence.

For Janus-Pro 7B, the final calibrated configuration is $a = 29.0381$, $\alpha = 1.0$, and $\epsilon = 0.01$. The calibration transform is applied to the encoder-derived proxy logits during training so that the diffusion decoder sees conditioning distributions that match the VLM's inference-time statistics. At inference, we use the VLM's image-token logits directly under the same classifier-free guidance and sampling settings used to estimate the target statistics ($w_{\text{cfg}} = 5.0$, $\tau_{\text{VLM}} = 1.0$). This ensures that the decoder is trained on calibrated proxy distributions that match the statistical regime of the VLM logits it receives at deployment.

A.3 Diffusion Decoder

The diffusion decoder is finetuned from FLUX.1-Fill, a rectified-flow transformer for image inpainting [22] built on the flow-matching and rectified-flow generative modeling paradigm [29, 30]. We update only its attention and adapter layers; all other parameters are frozen. The decoder is conditioned only on VLM-derived token representations and excludes direct text conditioning by design, ensuring it follows the VLM's original token belief rather than reinterpreting the scene with its own prior.

The decoder operates in the latent space of a pre-trained VAE. Given an image $x \in \mathbb{R}^{3\times H\times W}$, the VAE encoder produces a latent representation

$$\mathbf{z}_{\text{img}} = \text{VAE}_{\text{enc}}(x), \quad \mathbf{z}_{\text{img}} \in \mathbb{R}^{H'\times W'\times D_z}, \tag{14}$$

where H' and W' are the spatial dimensions of the latent feature map and D_z is the latent channel dimension. Noise is added according to a timestep-dependent variance schedule:

$$\tilde{\mathbf{z}}_{\text{img},t} = (1-\sigma_t)\,\mathbf{z}_{\text{img}} + \sigma_t\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{15}$$

where σ_t is the noise variance coefficient at timestep t.

For conditioning, the ***Distribution-Weighted Code Vectors*** $\mathbf{V}^{\text{enc}} \in \mathbb{R}^{N\times D}$ and the ***Uncertainty Features*** $\mathbf{U}^{\text{enc}} \in \mathbb{R}^{N\times 4}$ are spatially reshaped and interpolated to match the latent resolution. The code vectors are linearly projected and then refined through a lightweight convolutional layer:

$$\mathbf{H}_{\text{code}} = \phi_{\text{conv}}\big(\text{Linear}(\mathbf{V}^{\text{enc}})\big), \quad \mathbf{H}_{\text{code}} \in \mathbb{R}^{N\times D_h}, \tag{16}$$

where $\text{Linear}(\cdot)$ maps each code vector from $\mathbb{R}^D$ to a hidden dimension $\mathbb{R}^{D_h}$ aligned with the diffusion transformer input, and $\phi_{\text{conv}}(\cdot)$ is a 3×3 convolution with normalization and activation that enhances local spatial coherence among neighboring tokens. The uncertainty features $\mathbf{U}^{\text{enc}}$ are bilinearly upsampled to the same spatial resolution and concatenated channel-wise with the refined code features, forming the conditioning tensor:

$$\mathbf{C} = [\mathbf{H}_{\text{code}}, \mathbf{U}^{\text{enc}}], \quad \mathbf{C} \in \mathbb{R}^{N'\times(D_h+4)}, \tag{17}$$

where N' is the number of spatial tokens after interpolation.

The noisy latent $\tilde{\mathbf{z}}_{\text{img},t}$ is divided into patch tokens and projected into the transformer feature space:

$$\mathbf{H}_{\text{img}} = \text{Proj}_{\text{img}}(\text{Pack}(\tilde{\mathbf{z}}_{\text{img},t})), \quad \mathbf{H}_{\text{img}} \in \mathbb{R}^{N'\times D_\theta}, \tag{18}$$

$$\mathbf{H}_{\text{cond}} = \text{Proj}_{\text{cond}}(\mathbf{C}), \quad \mathbf{H}_{\text{cond}} \in \mathbb{R}^{N'\times D_\theta}, \tag{19}$$

where $\text{Pack}(\cdot)$ flattens the latent map into a sequence of N' spatial tokens, and $\text{Proj}_{\text{img}}(\cdot)$ and $\text{Proj}_{\text{cond}}(\cdot)$ are learned linear projections mapping the respective features into the transformer's internal feature dimension D_θ. The two representations are fused by element-wise addition:

$$\mathbf{H}_{\text{input}} = \mathbf{H}_{\text{img}} + \mathbf{H}_{\text{cond}}, \quad \mathbf{H}_{\text{input}} \in \mathbb{R}^{N'\times D_\theta}. \tag{20}$$

The fused tokens are processed by the diffusion transformer:

$$\mathbf{H}^t_{\text{out}} = \mathcal{D}_\theta(\mathbf{H}_{\text{input}}, t), \quad \mathbf{H}^t_{\text{out}} \in \mathbb{R}^{N'\times D_\theta}, \tag{21}$$

where $\mathcal{D}_\theta$ is the diffusion transformer parameterized by θ. The output features are projected back to the latent channel dimension and reshaped into the spatial latent form to predict the residual velocity:

$$\hat{\mathbf{v}}^t = \text{Unpack}\big(\text{Proj}_{\text{out}}(\mathbf{H}^t_{\text{out}})\big), \quad \hat{\mathbf{v}}^t \in \mathbb{R}^{H'\times W'\times D_z}, \tag{22}$$

where $\mathrm{Proj}_{\text{out}}(\cdot)$ projects to the latent dimension D_z and $\mathrm{Unpack}(\cdot)$ reshapes the token sequence into a spatial latent map of resolution $H' \times W'$.

The model is trained using the v-prediction objective [18]:

$$\mathcal{L}_{\text{v-pred}} = \mathbb{E}_t \left[\left\| \mathbf{w}_t \cdot (\hat{\mathbf{v}}^t - (\boldsymbol{\epsilon} - \mathbf{z}_{\text{img}})) \right\|_2^2 \right], \tag{23}$$

where $\mathbf{w}_t$ is a step-dependent weighting term that balances noise levels across timesteps. At inference, the same architecture operates on VLM-derived $\mathbf{V}$ and $\mathbf{U}$, and the final denoised latent is decoded through the VAE decoder to produce the output image.

B Extended Implementation Details

B.1 VQ-VAE Specifications

Our default backbone, Janus-Pro 7B, employs a VQ-GAN-style VQ-VAE as its image tokenizer for visual generation. The tokenizer encodes each RGB image into an 8-channel latent representation with a $16\times$ spatial downsampling factor, producing a 24×24 grid (576 spatial locations) for a 384×384 input resolution. Each latent vector is vector-quantized by selecting the nearest entry from a codebook of 16,384 learned embeddings, where each code vector is 8-dimensional. During VLM-based image generation, the multimodal transformer autoregressively predicts 576 discrete token IDs, each corresponding to one of the 16,384 codebook entries; the selected code vectors are arranged into an 8-channel 24×24 latent grid, which the VQ-VAE decoder maps back to the RGB image space through learned dequantization and upsampling layers.

B.2 Decoder Training and Inference Hyperparameters

We finetune the FLUX.1-Fill backbone with only its attention and adapter layers updated; all other parameters are frozen. Training is performed on the ImageNet-1K training set with standard resize and center-crop preprocessing, using AdamW with learning rate 2×10^{-5}, batch size 32, and 50,000 training steps on $4\times$ A100 80GB GPUs in mixed precision. At inference, the decoder uses 30 denoising steps; VLM-side generation settings ($w_{\text{cfg}} = 5.0$, $\tau_{\text{VLM}} = 1.0$) match those used during calibration (Appendix A.2).

C Extended Conceptual Motivation

This appendix provides the formal definition of Otsu's thresholding used in Section 3, reports the post-calibration analysis showing that Logit Calibration restores the encoder's distributional sharpness to match the VLM, verifies position by position that calibrated proxies and VLM logits provide compatible decoder conditioning (Appendix C.1), and analyzes the stability of that conditioning (Appendix C.2).

Otsu's thresholding. Otsu's method [36] is a classical variance-maximization technique that partitions a probability sequence into statistically meaningful "foreground" and "background" components by selecting the threshold that maximizes inter-class variance. Given sorted probabilities $\{p_{(1)}, \ldots, p_{(K)}\}$, the optimal threshold $\tau^\star$ is obtained by

$$\begin{aligned} \tau^\star &= \arg\max_{\tau} \ \sigma^2_{\text{between}}(\tau) \\ &= \omega_0(\tau)\, \omega_1(\tau)\, \big(\mu_0(\tau) - \mu_1(\tau)\big)^2, \end{aligned} \tag{24}$$

where ω_c and μ_c denote the total mass and mean of each partition, respectively. The corresponding threshold rank is the number of probabilities at or above $\tau^\star$, indicating how many codebook entries are statistically distinguishable from the noise floor.

Effect of Logit Calibration on encoder distributions. Section 3 establishes that raw proxy logits from the VQ-VAE encoder are nearly uniform (Otsu rank $\approx 5{,}767$), in stark contrast to the VLM's selective belief (Otsu rank 2). Table 1 reports the corresponding statistics after applying our Logit Calibration. The calibration substantially restores the desired sharpness: the Otsu threshold rank

becomes 2, identical to that of the VLM, and the top-1 probability and head mass closely match the VLM's values. This demonstrates that calibration reshapes the proxy-logit distribution into one that faithfully reproduces the semantic selectivity exhibited by the VLM.

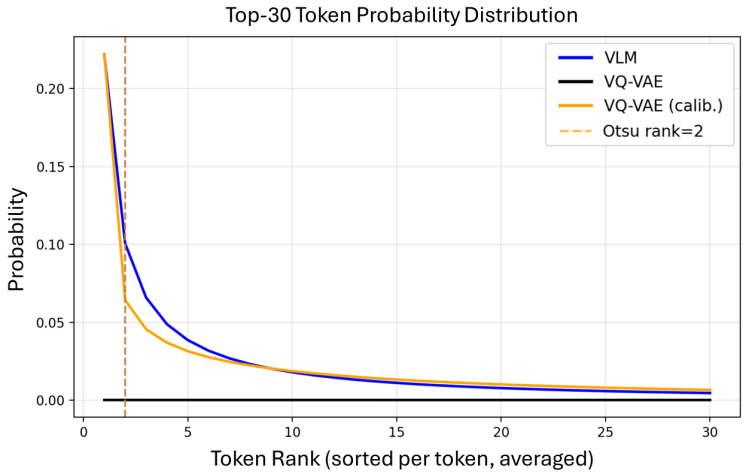


Figure 6: **Average top-30 token probability distributions** for Janus-Pro 7B (VLM), the raw VQ-VAE encoder, and the calibrated VQ-VAE encoder. Each curve shows per-token probabilities sorted by rank and averaged across 100 images. The dashed line marks the Otsu-derived threshold rank. The calibrated encoder closely follows the VLM's decay pattern, while the uncalibrated encoder produces a nearly flat distribution that no longer reflects semantic selectivity.

Figure 6 visualizes the same effect across the full top-30 distribution shape. The calibrated encoder distribution closely tracks the VLM's decay across all visible ranks, whereas the uncalibrated encoder remains nearly flat. This confirms that LC is essential for enabling LCDM to operate on probability distributions that faithfully reflect the VLM's semantic token structure, reducing the discrepancy between training-time conditioning via the VQ-VAE encoder and inference-time conditioning via the VLM.

C.1 Correspondence between Calibrated Proxies and VLM Logits

LC matches four global statistics, which alone do not guarantee that calibrated proxies and VLM logits place mass on the same codebook entries. We test this directly. For each of 200 MJHQ-30K Animals prompts, disjoint from the 100 GenEval prompts used to fit LC, we record the VLM's pre-sampling distribution and sampled token at all 576 positions, decode the sampled sequence with the native VQ-VAE decoder, re-encode the resulting image with the same VQ-VAE encoder used to construct training proxies, and form raw and calibrated proxies. The round trip is faithful: 87.92% of positions recover the exact token, and the sampled token lies in the proxy's top-8 at 98.47% of positions.

Since the 16,384-dimensional distribution never reaches the decoder, we compare what the decoder actually receives: the distribution-weighted code vector (Eq. (3)) and the four uncertainty features (Eq. (5)) at each position. As a control, we keep the calibrated proxy's probability values but reassign them to permuted codebook entries, which preserves every global statistic.

Table 9: **Position-wise agreement between proxy and VLM conditioning** (Janus-Pro 7B, 200 MJHQ-30K prompts, 576 positions each). Permuting code identities preserves the uncertainty features but destroys code-vector agreement.

Proxy vs. VLM logits	Code-vector cosine↑	Uncertainty MAE↓
Raw proxy	0.8125	0.4327
Calibrated proxy	0.8647	0.1220
Calibrated proxy, code identities permuted	0.0379	0.1220

Permutation leaves the uncertainty features unchanged but collapses the code-vector cosine from 0.8647 to 0.0379 (Table 9). The agreement therefore does not come from matched entropy and confidence alone; it comes from the VQ-VAE encoder and the VLM placing mass on the same codebook entries at the same position. LC then aligns their concentration: a positive scale, a positive temperature, and uniform label smoothing cannot reorder a distribution and only change how much weight each code carries, reducing the uncertainty-feature error by 72%.

Source substitution. We further test whether the two sources induce similar decoder outputs. For the same 200 token sequences, we decode with the same DCDD checkpoint while changing only the conditioning source, using the same initial diffusion noise, and compare against outputs conditioned on actual VLM logits. As a reference for natural variation, we also change only the diffusion noise.

Table 10: **Source substitution at inference** (Janus-Pro 7B). Similarity to the output conditioned on actual VLM logits (mean ± SD over 200 prompts).

Conditioning source	CLIP-I↑	DINOv2↑	LPIPS↓
Same source, different diffusion noise	0.9539 ± 0.0265	0.9396 ± 0.0405	0.1045 ± 0.0328
Calibrated encoder proxy	0.9403 ± 0.0401	0.9088 ± 0.0692	0.1380 ± 0.0509

Replacing VLM logits with the calibrated proxy changes the output only slightly more than changing the diffusion noise (Table 10). We do not claim that the two raw distributions are identical: the encoder observes a complete image, whereas the VLM predicts causally from the prompt and previous tokens. DCDD only requires them to induce compatible code-vector and uncertainty conditioning. Note also that all text-to-image results in the paper are computed with actual VLM logits; proxies are used only during training.

C.2 Stability of the Decoder Conditioning

Matching four global statistics cannot guarantee equivalence between two full distributions over $K = 16{,}384$ codes, and we do not claim a generalization bound. Instead, we show that the quantities the decoder receives change in a controlled way when the underlying distribution changes. For distributions $\mathbf{p}, \mathbf{p}'$ over the codebook, let $\delta = \mathrm{TV}(\mathbf{p}, \mathbf{p}') = \frac{1}{2}\|\mathbf{p}-\mathbf{p}'\|_1$, and let $R = \min_{\mathbf{c}} \max_k \|\mathbf{e}_k - \mathbf{c}\|_2$ be the radius of the codebook.

Proposition. *Let* $(\mathbf{v}, \mathbf{u})$ *and* $(\mathbf{v}', \mathbf{u}')$ *be the LCDM outputs of* $\mathbf{p}$ *and* $\mathbf{p}'$*. Then (i)* $\|\mathbf{v} - \mathbf{v}'\|_2 \le 2R\,\delta$*; (ii)* $|u^{(1)} - u'^{(1)}| \le \delta$*; (iii)* $|u^{(2)} - u'^{(2)}| \le 2\delta$*; and (iv) if both use the same* K_u*,* $|u^{(3)} - u'^{(3)}| \le \delta$*.*

Proof. (i) Since $\sum_k (p_k - p'_k) = 0$, for any $\mathbf{c}$ we have $\mathbf{v} - \mathbf{v}' = \sum_k (p_k - p'_k)(\mathbf{e}_k - \mathbf{c})$, so $\|\mathbf{v} - \mathbf{v}'\|_2 \le \max_k \|\mathbf{e}_k - \mathbf{c}\|_2 \, \|\mathbf{p} - \mathbf{p}'\|_1$; minimizing over $\mathbf{c}$ gives the claim. (ii) For every k, $|p_k - p'_k| \le \delta$, and the maximum is 1-Lipschitz in $\|\cdot\|_\infty$. (iii) Each order statistic $p_{(j)}$ is 1-Lipschitz in $\|\cdot\|_\infty$, so the margin changes by at most $2\|\mathbf{p} - \mathbf{p}'\|_\infty \le 2\delta$. (iv) The top-$K_u$ mass equals $\max_{|A|=K_u} P(A)$, and $|\max_A P(A) - \max_A P'(A)| \le \max_A |P(A) - P'(A)| \le \delta$. □

Scope. The tail entropy $u^{(4)}$ is continuous but not Lipschitz near zero probabilities, and the entropy-adaptive K_u is piecewise constant, so (iv) holds whenever both distributions select the same K_u; we therefore measure the uncertainty features empirically (Table 9). The proposition bounds the change in conditioning, not the distance δ between proxy and VLM distributions, nor the response of the decoder to a change in conditioning. In addition, the LC transform is order-preserving: for $a > 0$, $\alpha > 0$, and $\epsilon \in [0, 1)$, $\hat{p}_{i,k} > \hat{p}_{i,k'}$ if and only if $s_{i,k} > s_{i,k'}$. Calibration therefore changes only the concentration of each proxy distribution and leaves its ranking of codebook entries, and hence its top-k sets, unchanged; the agreement of those rankings with the VLM is an empirical property, measured in Appendix C.1.

D Additional Ablation Studies

D.1 Denoising Steps

We evaluate our diffusion decoder under different numbers of denoising steps while keeping all other components fixed, measuring reconstruction quality on the first 1,000 validation images of ImageNet-1K against the VQ-VAE decoder of Janus-Pro 7B. Table 11 summarizes the results.

Table 11: **Reconstruction performance across denoising steps** (Janus-Pro 7B, ImageNet-1K validation). All rows use the final 50K-step decoder; absolute rFID values are higher than those in Table 2 because evaluation here uses a 1,000-image subset rather than the full validation split. Best results are highlighted.

ImageNet-1K Val.	rFID↓	SSIM↑	LPIPS↓
VQ-VAE Decoder	23.1038	0.5756	0.1551
Ours (5 steps)	35.0169	0.5855	0.1918
Ours (10 steps)	20.8859	0.5995	0.1478
Ours (15 steps)	18.3279	0.6012	0.1417
Ours (20 steps)	17.6918	0.5993	0.1405
Ours (25 steps)	17.5659	0.5975	0.1404
Ours (30 steps)	17.6969	0.5954	0.1410
Ours (35 steps)	17.7818	0.5940	0.1414

Even with 10 denoising steps, our decoder surpasses the VQ-VAE baseline across all metrics. As the step count increases, rFID and LPIPS reach their best values at 25 steps before gradually degrading, while SSIM peaks earlier at 15 steps, suggesting that global structure is recovered in the initial denoising stages and subsequent steps refine texture and low-frequency detail. The trajectory thus rapidly establishes the core scene layout, with additional steps primarily polishing perceptual details rather than altering structural content.

Choice of inference denoising steps. Although 25 steps yields the lowest rFID and LPIPS, the gap to 30 steps is marginal in absolute terms (rFID 17.57 vs 17.70, LPIPS 0.1404 vs 0.1410, SSIM 0.5975 vs 0.5954). We adopt 30 denoising steps throughout the paper, matching the default sampling schedule of FLUX.1-Fill, our diffusion backbone. This choice avoids backbone-specific hyperparameter tuning and provides a more reproducible setting: practitioners finetuning FLUX.1-Fill on related tasks can use the model's standard sampling configuration without re-running this ablation on their setup.

D.2 Latency and Memory

Table 12 reports per-image latency across denoising steps, measured in BF16 at 384×384 with batch size 1, two warm-up runs, and seven synchronized runs on an A100 80GB. Denoising cost is linear at approximately 0.12 s per step, and VLM autoregression takes 11.84 s. Ten steps already outperform the native decoder at 10.7% additional end-to-end latency, and 15 steps recover 86% of the best fidelity gain using 51% of the additional latency of the 30-step setting.

Table 12: **Latency across denoising steps** (Janus-Pro 7B, A100 80GB, batch size 1). rFID is measured on the 1,000-image ImageNet-1K subset of Table 11.

Decoder	Steps	Decoder-only (s/img)↓	End-to-end (s/img)↓	rFID↓
VQ-VAE Decoder	–	0.02	11.86	23.1038
DCDD	10	1.10	13.13	20.8859
DCDD	15	1.71	13.74	18.3279
DCDD	20	2.32	14.35	17.6918
DCDD	25	2.93	14.96	17.5659
DCDD	30	3.53	15.56	17.6969

Peak allocated memory is 14.90 GiB for the VLM, 0.32 GiB for the native decoder, and 22.67 GiB for the DCDD decoder, or approximately 37.6 GiB for the full pipeline. This exceeds typical edge-device budgets; deployment in that setting would require a quantized or distilled decoder (Appendix H).

D.3 Training Dynamics of Component Ablations

Table 13 tracks three decoders that differ only in their conditioning across training checkpoints up to 10K steps: (i) discrete code vectors; (ii) DCDD without LC, with continuous distribution-weighted code vectors and uncertainty features but no calibration; and (iii) full DCDD. LCDM provides the

largest single rFID reduction, LC provides the largest single SSIM improvement, and LPIPS decreases monotonically across the three variants. The relative ranking emerges by 5K steps and remains stable, indicating that the contributions are driven by the components rather than by training-time noise. The argmax and full variants are compared at the full 50K-step schedule in Table 5.

Table 13: **Training-step ablation on ImageNet-1K reconstruction** (Janus-Pro 7B). We track three variants up to 10K training steps under matched optimization: *Diff. Dec. (baseline)*, with discrete code-vector conditioning; *DCDD w/o LC*, with continuous distribution-weighted code vectors and uncertainty features but no calibration; and *DCDD (full)*, with calibration applied to the proxy logits. The relative ranking among variants emerges by 5K steps and remains stable. Best value at each step for each metric is highlighted.

	Diff. Dec. (baseline)			***DCDD w/o LC***			***DCDD (full)***		
Steps	rFID ↓	SSIM ↑	LPIPS ↓	rFID ↓	SSIM ↑	LPIPS ↓	rFID ↓	SSIM ↑	LPIPS ↓
2.5K	173.7125	0.1940	0.7821	161.4700	0.1949	0.7732	158.7311	0.1965	0.7801
5.0K	106.0594	0.2279	0.6359	93.4573	0.2717	0.6051	91.8926	0.2750	0.5833
7.5K	45.3929	0.3080	0.5713	39.5000	0.3147	0.5558	37.8744	0.3222	0.5498
10.0K	33.5081	0.4244	0.4132	31.2049	0.4275	0.4009	30.5760	0.4360	0.3875

D.4 Comparison of Post-hoc Alternatives

Table 14 places the matched argmax-conditioned diffusion decoder of Section 5.5 alongside the image-to-image refiners of Section 5.4, with the VLM and tokenizer fixed. The argmax decoder attains the highest CLIP-I and DINOv2 similarity because its conditioning is exactly the code sequence that the VQ-VAE reference image is decoded from, so it stays closest to that reference by construction. DCDD departs slightly further from the reference as it uses the additional distributional information to recover detail, which is reflected in the lower gFID, while remaining well above both I2I refiners in similarity.

Table 14: **Post-hoc alternatives on MJHQ-30K** (Janus-Pro 7B, VLM and tokenizer fixed). CLIP-I and DINOv2 are measured against the VQ-VAE-decoded image.

Method	gFID↓	CLIP-I↑	DINOv2↑
VQ-VAE Decoder	14.2888	–	–
I2I (generic)	10.6678	0.8752	0.7663
I2I + prompt	10.9767	0.8531	0.7586
Diff. Dec. (argmax codes)	11.0837	0.9275	0.8933
DCDD (ours)	9.4160	0.9088	0.8692

D.5 Soft-Token Adapter without Diffusion

To test whether the distributional signal can be exploited without a diffusion decoder, we train a token-wise residual MLP ($8 \to 64 \to 8$, 1,096 parameters, zero-initialized) placed before the frozen native VQ-VAE decoder. Argmax and soft variants use identical architecture, data, and optimization, with an L1 + 0.5 LPIPS objective and code-displacement and norm penalties. Table 15 reports reconstruction on the first 1,000 ImageNet-1K validation images and generation on the MJHQ-30K Animals category.

Expected codes improve all reconstruction metrics with no trained parameters, and the soft adapter improves them further, so the distributional signal is useful independently of diffusion. In generation, however, the soft adapter is worse than the native decoder, while the argmax adapter is indistinguishable from it. In reconstruction the expected codes come from the VQ-VAE encoder, whereas in generation they come from the VLM; a frozen decoder trained on quantized codes cannot learn to read a distribution through a small residual. Exploiting the signal for generation therefore requires a decoder trained for distributional conditioning.

D.6 Variation across Diffusion-Noise Seeds

We re-run DCDD with two additional diffusion-noise seeds, reusing the same VLM token sequences so that only the initial diffusion noise changes. Table 16 reports the mean and standard deviation over

Table 15: **Soft-token adapter before the frozen native decoder** (Janus-Pro 7B). *Top:* reconstruction (ImageNet-1K, 1,000 images). *Bottom:* generation (MJHQ-30K Animals gFID and GenEval).

Reconstruction	Train steps	MSE↓	rFID↓	SSIM↑	LPIPS↓
Argmax, no adapter	–	0.0360	23.10	0.5756	0.1551
Expected code, no adapter	0	0.0285	22.71	0.6081	0.1418
Adapter on argmax	100	0.0345	23.00	0.5829	0.1550
Adapter on expected code	2K	0.0274	22.26	0.6164	0.1390

Generation	gFID↓ (Animals)	GenEval↑
VQ-VAE Decoder	31.68	77.24
+ Adapter on argmax	31.81	77.12
+ Adapter on expected code	34.28	75.41
DCDD (ours)	25.10	78.31

the three runs for the overall metrics of Tables 2, 3, and 4. All values reported in the main tables fall within the observed range.

Table 16: **DCDD across three diffusion-noise seeds** (Janus-Pro 7B, mean ± SD; VLM token sequences fixed).

Table	***Metric***	Mean ± SD
Table 2	ImageNet-1K rFID↓	0.7671 ± 0.0478
	ImageNet-1K SSIM↑	0.5957 ± <0.0001
	ImageNet-1K LPIPS↓	0.1374 ± 0.0001
	MJHQ-30K rFID↓	1.8174 ± 0.0480
	MJHQ-30K SSIM↑	0.6782 ± <0.0001
	MJHQ-30K LPIPS↓	0.1307 ± 0.0001
Table 3	MJHQ-30K gFID↓	9.3787 ± 0.2929
	GenEval Overall↑	78.32 ± 0.10
	DPG-Bench Overall↑	87.64 ± 0.03
	WISE Consistency↑	1.318 ± 0.012
Table 4	CLIP-I↑	0.9085 ± 0.0022
	DINOv2↑	0.8701 ± 0.0036

E Sensitivity of Logit Calibration to Reference Prompts

Logit Calibration estimates four target statistics from a sample of VLM-generated images and uses them to align proxy-logit distributions to the VLM's regime. Two natural concerns about this setup are: (i) whether the sample is large enough for the estimates to be stable, and (ii) whether the choice of prompt set biases the resulting calibration toward a particular benchmark's content. Figure 7 illustrates the calibration data flow. This appendix addresses both concerns.

E.1 Sensitivity to Sample Size

We vary the number of reference prompts from 1 to 200 and measure the resulting global logit scale a alongside the target statistics it is fit against. We focus on a because it is the primary calibration parameter, fit first against the VLM's mean entropy in our gradient-free search; the remaining parameters (b, α, ϵ) are estimated subsequently and are conditioned on the choice of a. Table 17 reports the results.

The estimated a stabilizes from 100 prompts onward (29.04, 29.09, 29.02 at 100, 150, and 200 prompts respectively), with mean entropy in $[0.412, 0.432]$ and mean confidence in $[0.174, 0.189]$ across the full range. The estimates show no monotonic trend with sample size, indicating that target statistics primarily reflect stable, model-level logit behavior rather than prompt-specific variation. Our default choice of 100 prompts is at this stable regime.

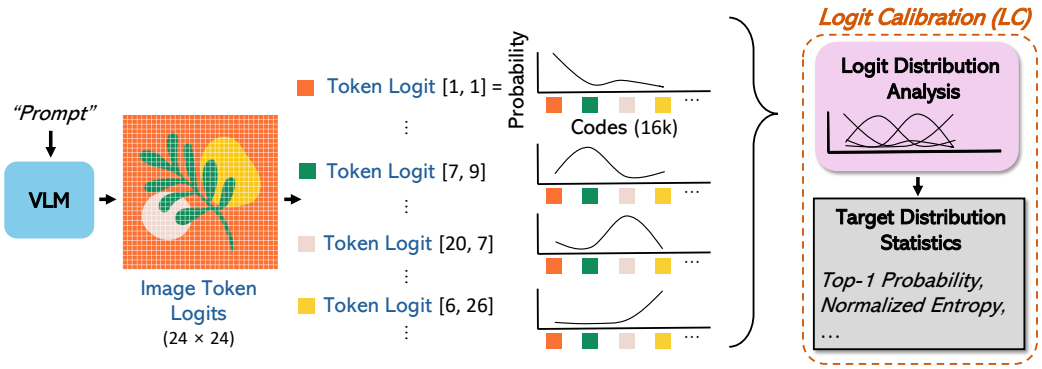


Figure 7: **Logit Calibration matches coarse global statistics of VLM logits.** Each 384×384 image produces a 24×24 grid of image-token logits, i.e., 576 patch-level codebook logit distributions. LC analyzes these VLM logits to estimate global statistics such as confidence and entropy, which are then used as calibration targets for the proxy logits derived from the VQ-VAE encoder.

Table 17: **Sensitivity of Logit Calibration to the number of reference prompts** (Janus-Pro 7B). The estimated global logit scale a and target statistics (mean normalized entropy, mean confidence) remain within a narrow range across sample sizes, with no clear trend as the number of prompts grows. Each prompt contributes 576 token-level distributions, so even a modest number of prompts provides sufficient coverage to estimate stable global statistics.

# Prompts	a	*Mean Entropy*	*Mean Confidence*
1	28.4570	0.4318	0.1807
5	30.2344	0.4123	0.1887
10	29.5679	0.4196	0.1846
20	28.3716	0.4328	0.1743
50	28.5938	0.4302	0.1766
100	29.0381	0.4254	0.1797
150	29.0894	0.4248	0.1801
200	29.0210	0.4255	0.1797

E.2 Sensitivity to Prompt Type

A separate question is whether calibration is sensitive to what the reference prompts are, not how many. This bears on the concern that calibrating on GenEval prompts could implicitly bias the decoder toward GenEval-style content. We measure target statistics across several qualitatively different prompt sets:

- **Natural-language benchmarks.** GenEval and MJHQ-30K prompts (average prompt lengths 32.2 and 185.7 characters, respectively).
- **Constructed prompts.** Three categories synthesized to stress the VLM in specific ways, all controlled to roughly 50 characters: *(1) Unnatural* (lexically implausible), *(2) Compositional overload* (very dense object specifications), and *(3) Self-contradiction* (prompts containing internal inconsistencies).
- **Length-controlled random tokens.** Empty prompts and random alphanumeric-with-space strings of varying lengths (50 to 1000 characters), which probe whether prompt length alone affects the VLM's logit regime in the absence of semantic content.

For a fair comparison, each prompt type is evaluated using 10 prompts. Sample prompts from each set are shown in Figure 8; results are in Table 18.

The data reveal two effects that disentangle semantic content from prompt length.

Semantically extreme prompts increase uncertainty, not decrease it. The constructed Compositional-overload and Self-contradiction prompts produce noticeably lower a (25.8 and 24.5)

Table 18: **Logit statistics across different prompt types** (Janus-Pro 7B, 10 prompts per type). Key values discussed in the text are highlighted.

Prompt type	a	*Mean Entropy*	*Mean Confidence*
GenEval	29.5679	0.4195	0.1846
MJHQ-30K	32.1826	0.3920	0.2127
(1) Unnatural	30.2856	0.4117	0.1901
(2) Comp. overload	25.8081	0.4632	0.1541
(3) Self-contradict.	24.4751	0.4800	0.1368
Blank prompt, ""	27.2949	0.4452	0.1639
Rand. char. 50	27.7905	0.4395	0.1645
Rand. char. 100	29.0381	0.4254	0.1821
Rand. char. 200	30.6104	0.4083	0.1986
Rand. char. 300	31.7383	0.3966	0.2100
Rand. char. 400	30.7983	0.4064	0.2018
Rand. char. 500	29.3799	0.4215	0.1866
Rand. char. 1000	29.5166	0.4201	0.1894

(1) Rare or unnatural composition prompts that combine unusual materials, geometry, or physical situations

"a cat made entirely of glass melting into liquid metal in space"
"a city built from flowing water structures under inverted gravity"
"a human face composed of fire and ice simultaneously"
"a tree growing upside down from the sky with glowing roots"
"a mountain made of transparent crystal floating above the ocean"
"an animal whose body is made of shifting geometric patterns"
"a landscape where the ground behaves like liquid and the sky is solid"
"a creature formed entirely from reflections and light distortions"
"a building that continuously changes shape in a single static image"
"a surreal environment where all materials behave opposite to physics"

(2) Counting/compositional overload prompts that require unusually dense object layouts or large object counts

"1000 tiny colorful objects arranged in a perfectly symmetric grid"
"a scene with hundreds of overlapping transparent spheres with detailed textures"
"thousands of small animals densely packed in a single image frame"
"a city made of millions of tiny buildings all visible at once"
"a complex pattern with countless repeating geometric shapes filling the image"
"a large crowd of people where every individual face is clearly visible"
"hundreds of objects floating randomly but arranged in perfect order"
"a highly detailed scene containing thousands of distinct elements"
"a massive library with infinite bookshelves all clearly readable"
"an image containing every type of object in existence at once"

(3) Self-contradictory prompts that impose incompatible visual requirements

"a completely black image filled with bright white details everywhere"
"a blank canvas densely packed with thousands of objects"
"a pure white image with intricate colorful patterns in every pixel"
"a silent explosion with no movement but intense motion blur"
"a perfectly empty room filled with furniture in every corner"
"a dark night scene illuminated with no light source but extremely bright"
"a transparent object that completely blocks all light"
"a still image capturing extreme motion in complete stillness"
"a frozen waterfall flowing rapidly with no movement"
"an invisible object clearly visible in the center of the image"

Random alphanumeric-and-space strings (50 char.)

"yWAcqGFzYtEwLnGisiWgNZqITZMjtgUe52RvEJgwBuNOnJECHq"
"dZJafUzTYQOaNF03vpUuT3em6KopZ9jZlCffu4G7FgtJsThJv0"
"a3OAHXKZGAF0OTfEquI2SyWbKZw39zFZdrEVSH8jHxCaCbXQ7 "
"8s8YGPI5g2q3UjrnvQ0Fp4dNZcui8kBRlg6QjcwIAcw58cwQP "
"NQ7zT5zU qM1C3R9KAV4 6h3C2nVBc6Zz 9F4vY9rfz1h9qKzS"
" U9yR0C3L0uXQy6 zJ3M 8P Zc8f3cZL8E5M VY6Z3E5gqf 9v"
"b5f8yNCl6xf3nV9 u 2i9yF7XvR6pW4l5Jf3n3Fv6s3Q1nV5Y8"
"8O 8C 6hU3tW9g2jQ2k3t4v5r7Z5 m3h1W0b0y5x9g6q0V7r 8"
"c3m7l5n9k2f6p4s8d1g 3h5j7k 9l2 z 4x6c8v1b3n5m7q9w2"
"5n7v9c3x1z4l6k8j2 h5g7f 9d3s1 a4q6w8e2r5t7y9u3i1o"

Figure 8: **Sample prompts from each prompt set used in the prompt-type analysis.** Three constructed prompt categories (Unnatural, Compositional overload, Self-contradiction) and length-controlled random alphanumeric-with-space prompts.

and higher entropy than GenEval, consistent with the VLM having more ambiguity about what to render. However, MJHQ-30K produces *higher* a (32.2) than GenEval, in the opposite direction. If semantic content drove the natural-prompt gap, we would expect MJHQ-30K (a more diverse, free-form benchmark) to produce higher uncertainty, not lower. Instead, MJHQ-30K is sharper than GenEval, ruling out semantic content as the dominant driver.

Length-controlled random prompts reproduce the natural-prompt gap. Random alphanumeric-and-space strings strip out semantic content entirely while varying length. Across these length-only controls, a rises from 27.8 (50 characters) to 30.6 (200 characters), a difference of 2.8. The gap between GenEval (about 30 characters, $a = 29.6$) and MJHQ-30K (about 180 characters, $a = 32.2$) is 2.6, comparable in magnitude. Since random prompts contain no benchmark-specific content, the comparable gap suggests that the GenEval-MJHQ-30K difference is driven primarily by prompt length rather than by GenEval-specific or MJHQ-specific semantic structure.

These two observations together address the concern that calibrating on GenEval introduces benchmark-specific bias. The calibration target is governed by length-dependent properties of the VLM's logit regime that hold across diverse prompt content; GenEval prompts are neither outliers nor uniquely informative for the calibration. We also note that LCDM accounts for the majority of the fidelity gain (Appendix D.3), so even if the calibrated scale of the LC shifted modestly under different reference prompts, the downstream impact on decoder performance would be minor at this scale of difference.

F Regime of Applicability: Results on LlamaGen-XL

A post-hoc decoder cannot recover information absent from its input token sequence. As summarized in Section 5.7, DCDD's effectiveness depends on the upstream model's logit structure; here we characterize that dependence empirically by evaluating DCDD on a backbone outside the regime where the main paper's quantitative claims hold: LlamaGen-XL at 256×256 resolution, with baseline T2I performance of GenEval 30.0 (vs. Janus-Pro 7B's 77.2) and MJHQ-30K gFID 59.0 (vs. 14.3). The two backbones thus span a wide range of upstream token quality, allowing us to characterize the regime in which post-hoc decoding is effective.

F.1 Image Reconstruction

We first evaluate reconstruction, where the token sequence comes from the VQ-VAE encoder and bypasses the VLM entirely. Quantitative results are in Table 19, with qualitative samples in Figure 11.

Table 19: **Reconstruction performance on LlamaGen-XL** (256×256). Best results per metric are highlighted.

ImageNet-1K Val.	rFID↓	SSIM↑	LPIPS↓
VQ-VAE Decoder	4.1919	0.5566	0.1388
DCDD (ours)	1.4087	0.5600	0.1404
MJHQ-30K	rFID↓	SSIM↑	LPIPS↓
VQ-VAE Decoder	4.9653	0.6140	0.1407
DCDD (ours)	1.9028	0.6060	0.1484

DCDD substantially reduces rFID on both datasets (66% on ImageNet-1K, 62% on MJHQ-30K), with SSIM and LPIPS remaining comparable to the VQ-VAE baseline. The lower input resolution provides fewer spatial samples for the diffusion decoder than the 384×384 Janus-Pro setting, yet substantial reconstruction gains still hold. This confirms that the decoder is not backbone-specific for reconstruction: when the upstream token sequence carries faithful information about the input image (as in reconstruction), DCDD recovers fidelity across backbones.

F.2 Text-to-Image Generation

The result differs for end-to-end text-to-image generation, where the token sequence is produced by the VLM from a text prompt rather than from an image. Table 20 reports the results.

Table 20: **Text-to-image generation on LlamaGen-XL** (256×256). DCDD inherits and amplifies the weaknesses of the upstream token sequence on this backbone. Best results per metric are highlighted.

					Fidelity Metric ↓						
MJHQ-30K	**Overall**	Animals	Art	Fashion	Food	Indoor	Landscape	Logo	People	Plants	Vehicles
VQ-VAE Decoder	59.0292	74.1959	82.9698	92.8457	86.4562	89.2189	82.0326	68.5263	105.9981	87.2854	82.0326
DCDD (ours)	61.0925	74.6621	87.1321	104.2776	93.6168	96.6309	90.0496	66.9729	114.4049	93.7683	90.0496

Semantic Consistency Checks							
GenEval	**Overall**	Single	Two	Counting	Colors	Position	Color Attr.
VQ-VAE Decoder	30.02	69.06	32.32	14.69	55.85	5.25	3.00
DCDD (ours)	28.19	70.94	25.76	19.38	47.07	4.25	1.75

In contrast to the strong reconstruction gains, DCDD's T2I metrics on LlamaGen-XL are essentially flat or mildly negative: gFID rises slightly (59.03 to 61.09 on MJHQ-30K) and GenEval shifts marginally (30.02 to 28.19), with sub-category movements scattered in both directions. This is the opposite pattern from Janus-Pro 7B, where DCDD reduces gFID by 34% with stable semantics.

Why DCDD does not help on LlamaGen-XL. The difference traces to two factors. First, LlamaGen-XL's baseline T2I quality is much weaker than Janus-Pro 7B's, indicating that the underlying token sequence already contains substantial semantic and perceptual errors. DCDD is designed to render the VLM's intended output faithfully, not to correct semantic errors; when the upstream token sequence is already semantically degraded, faithful decoding inherits the degradation rather than fixing it.

Second, LlamaGen-XL's logit distributions leave less distributional information for DCDD to exploit. Table 21 compares the two backbones' image-token logit statistics. LlamaGen-XL is more concentrated than Janus-Pro 7B: top-1 probability 0.38 vs. 0.22, normalized entropy 0.33 vs. 0.42, and crucially, an Otsu threshold rank of 1 vs. 2. Each LlamaGen-XL token already commits strongly to a single codebook entry, leaving little rank-2 or higher distributional information for LCDM to convert into conditioning. This is the precondition our method relies on, as established in Section 3: when the pre-argmax logits do not carry meaningful rank-2 mass, there is little discarded signal for a post-hoc decoder to recover.

Table 21: **Token-probability statistics and Otsu thresholding for Janus-Pro 7B vs. LlamaGen-XL.** LlamaGen-XL's distributions already commit more strongly to a single codebook entry (top-1 0.38, Otsu rank 1) than Janus-Pro 7B's (top-1 0.22, Otsu rank 2), leaving less distributional information beyond the top-1 entry for LCDM to exploit.

Statistic	**Janus-Pro 7B**	**LlamaGen-XL**
Probability Distribution Statistics		
Top-1 Probability	0.221627	0.379384
Top-2 Probability	0.100532	0.117179
Normalized Entropy	0.415303	0.327751
Tail Entropy	0.560794	0.539393
Otsu Thresholding Statistics		
Threshold (prob.)	6.24×10^{-2}	1.17×10^{-1}
Threshold Rank	2	1
Head Mass	0.322160	0.379384

Together, these factors describe a regime of applicability: post-hoc fidelity decoding via DCDD is most effective when (a) the upstream VLM produces token sequences that already carry coherent semantics, and (b) the per-token logit distribution retains non-trivial rank-2 or higher mass that LCDM can convert into continuous conditioning. Reconstruction satisfies both conditions on both backbones. T2I generation on Janus-Pro 7B satisfies both conditions; T2I generation on LlamaGen-XL satisfies neither. Because this diagnostic depends only on the upstream model's logit statistics, it provides a practical criterion for predicting when DCDD is likely to help.

F.3 Transfer to Janus-Pro 1B without Retraining

Janus-Pro 1B shares the VQ-VAE tokenizer of Janus-Pro 7B, so the DCDD checkpoint trained for the 7B model can be applied to it directly. Because the decoder was trained for the 7B conditioning regime, we align the 1B model's inference-time logits to the 7B target statistics (Eq. (9)) using the same gradient-free scale, temperature, and label-smoothing search as LC (Appendix A.2), applied at generation time; no parameters are trained. Measured with the same protocol as Table 21 (100 GenEval prompts, each model's own CFG and sampling settings, before top-k/top-p filtering), the 1B logits fall in the same regime as the 7B logits: Otsu rank 2, top-1 probability 0.2457, top-2 probability 0.0978, head mass 0.3435, and normalized entropy 0.4033.

Table 22: **Janus-Pro 1B with the Janus-Pro 7B DCDD checkpoint** (no retraining; mean ± SD over three seeds). gFID is measured on the MJHQ-30K Animals category.

Janus-Pro 1B	gFID↓ (Animals)	GenEval↑
VQ-VAE Decoder	35.14 ± 0.28	63.46 ± 0.28
DCDD (ours)	26.09 ± 0.26	62.17 ± 0.45

Fidelity improves by 25.7%, while GenEval changes by -1.29 consistently across seeds. This matches the intended behavior: DCDD improves fidelity while following the upstream model's token belief, so semantic scores are expected to remain close to those of the native decoder. Within a shared tokenizer, a single DCDD checkpoint therefore extends across model scales; models with a different tokenizer require a separately trained decoder.

G Qualitative Samples

This section provides additional qualitative samples for both backbones evaluated in the paper. Figures 9 and 10 show reconstruction and text-to-image results on Janus-Pro 7B at 384×384 resolution; Figures 11 and 12 show the corresponding results on LlamaGen-XL at 256×256. In each figure, the top two rows draw from ImageNet-1K (reconstruction) or MJHQ-30K (T2I), and the bottom two rows from MJHQ-30K (reconstruction) or GenEval (T2I). All comparisons are between the backbone's native VQ-VAE decoder and DCDD using the same input tokens.

H Limitations

Dependence on the upstream VLM. The effectiveness of DCDD is shaped by properties of the upstream VLM, as characterized in Section 5.7 and Appendix F: gains are most pronounced when the per-token logits retain meaningful distributional structure and when the baseline token sequence is already reasonably coherent. Since the decoder is conditioned only on VLM-derived token representations, it preserves the upstream model's semantic output rather than correcting it, which is a property of the post-hoc design choice consistent with our objective of improving fidelity without altering the VLM's intent. DCDD is therefore not a remedy for weak upstream generation.

Scope across tokenizers. A decoder is trained for one tokenizer. Within a shared tokenizer, the same checkpoint transfers across model scales (Appendix F.3); models with a different tokenizer require a separately trained decoder, and we have not evaluated such models end to end beyond LlamaGen-XL.

Distribution shift. The decoder is trained only on ImageNet-1K, yet improves MJHQ-30K reconstruction and generation, and the calibration targets remain stable under unnatural, compositionally overloaded, self-contradictory, blank, and random prompts (Appendix E). We have not, however, systematically evaluated semantic consistency on long-form or highly out-of-distribution prompts. The correspondence between training-time proxies and inference-time VLM logits is established empirically (Appendix C.1); Appendix C.2 bounds the sensitivity of the conditioning but does not provide a generalization guarantee.

Computational cost. Inference introduces additional latency from iterative denoising (31% end-to-end at 30 steps, 10.7% at 10 steps; Section 5.6, Appendix D.2), and the full pipeline requires approximately 37.6 GiB of peak memory, which exceeds typical edge-device budgets. The native VQ-VAE path remains available when efficiency is preferred, and a quantized or distilled decoder is a natural direction for constrained deployment.

I Impact Statement

Our work improves the visual fidelity of images generated by pre-trained text-to-image VLMs. As with other advances in image generation, this could contribute to creating more realistic synthetic content, with attendant concerns about deceptive use; these risks are not unique to our method and are addressed by established safeguards developed for generative AI more broadly. We note one property specific to our approach: because DCDD does not introduce its own semantic prior, it will faithfully

render biases or errors present in the underlying VLM. Responsible deployment therefore requires bias auditing and mitigation at the level of the upstream VLM rather than at the decoding stage.

J Future Directions

DCDD shows that the pre-sampling distribution of a frozen discrete-token model is a useful conditioning signal, which suggests several extensions. On the conditioning side, timestep-adaptive feature weighting and alignment, effective in diffusion and flow-matching transformers for virtual try-on and video-to-audio synthesis [13, 46], could replace our uniform injection across denoising steps, and the same principle applies to speech and audio generators that condition on rich continuous features [3]. On the downstream side, higher-fidelity decoding and per-token uncertainty can benefit 3D scene generation, image-based Gaussian splatting, novel-view enhancement, and geometry-aware or training-free editing [20, 26, 17, 21, 27]. Finally, visual-token distributions could serve as soft targets or confidence-aware rewards for training smaller or stronger multimodal models, in line with recent token-level distillation and confidence-based reinforcement learning for vision-language models [16, 53, 52].

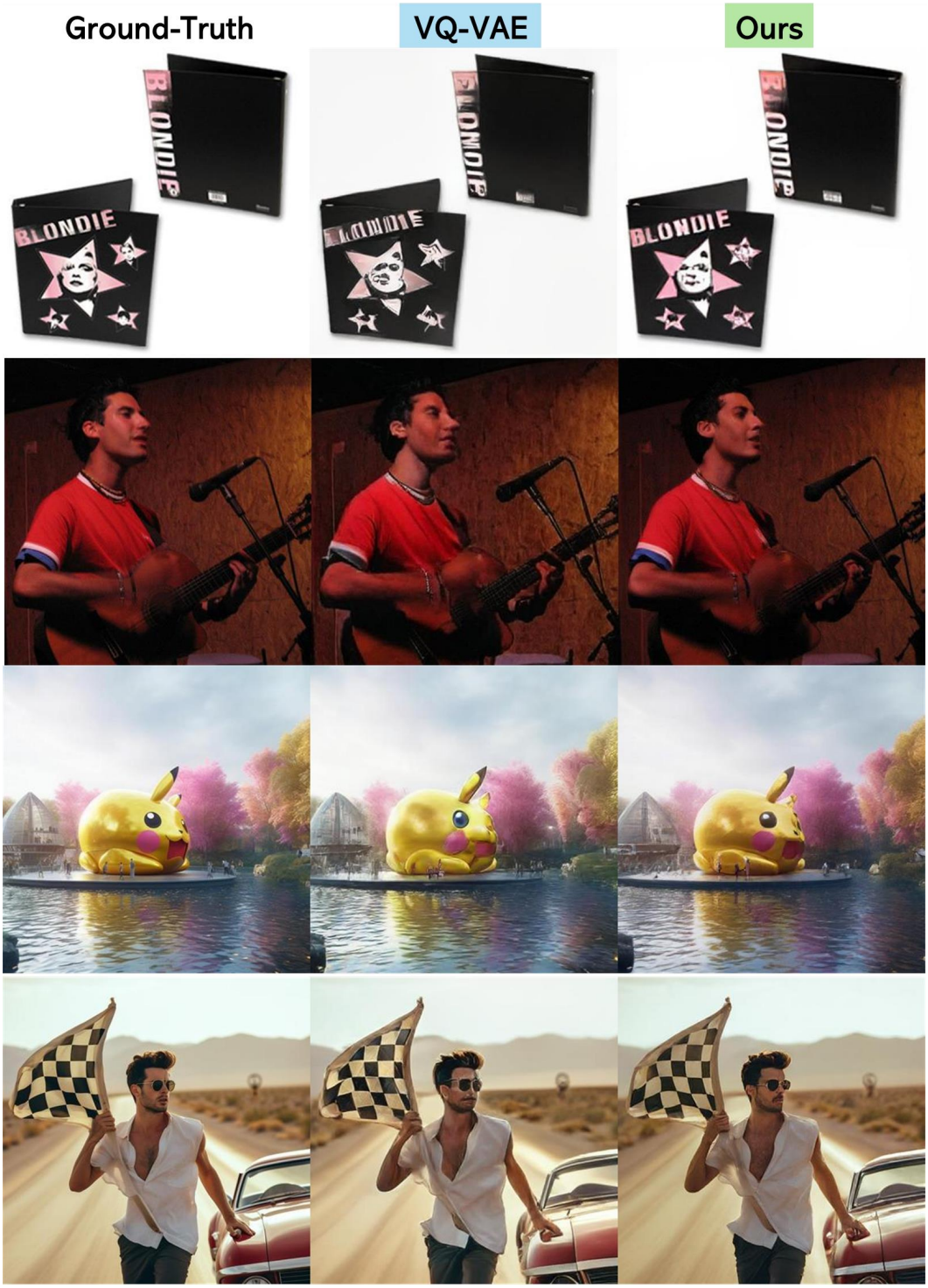


Figure 9: **Reconstruction samples on Janus-Pro 7B** (384×384). VQ-VAE decoder versus DCDD on identical encoded representations. Top two rows: ImageNet-1K. Bottom two rows: MJHQ-30K. DCDD recovers fine structure and texture detail that the native VQ-VAE decoder loses.

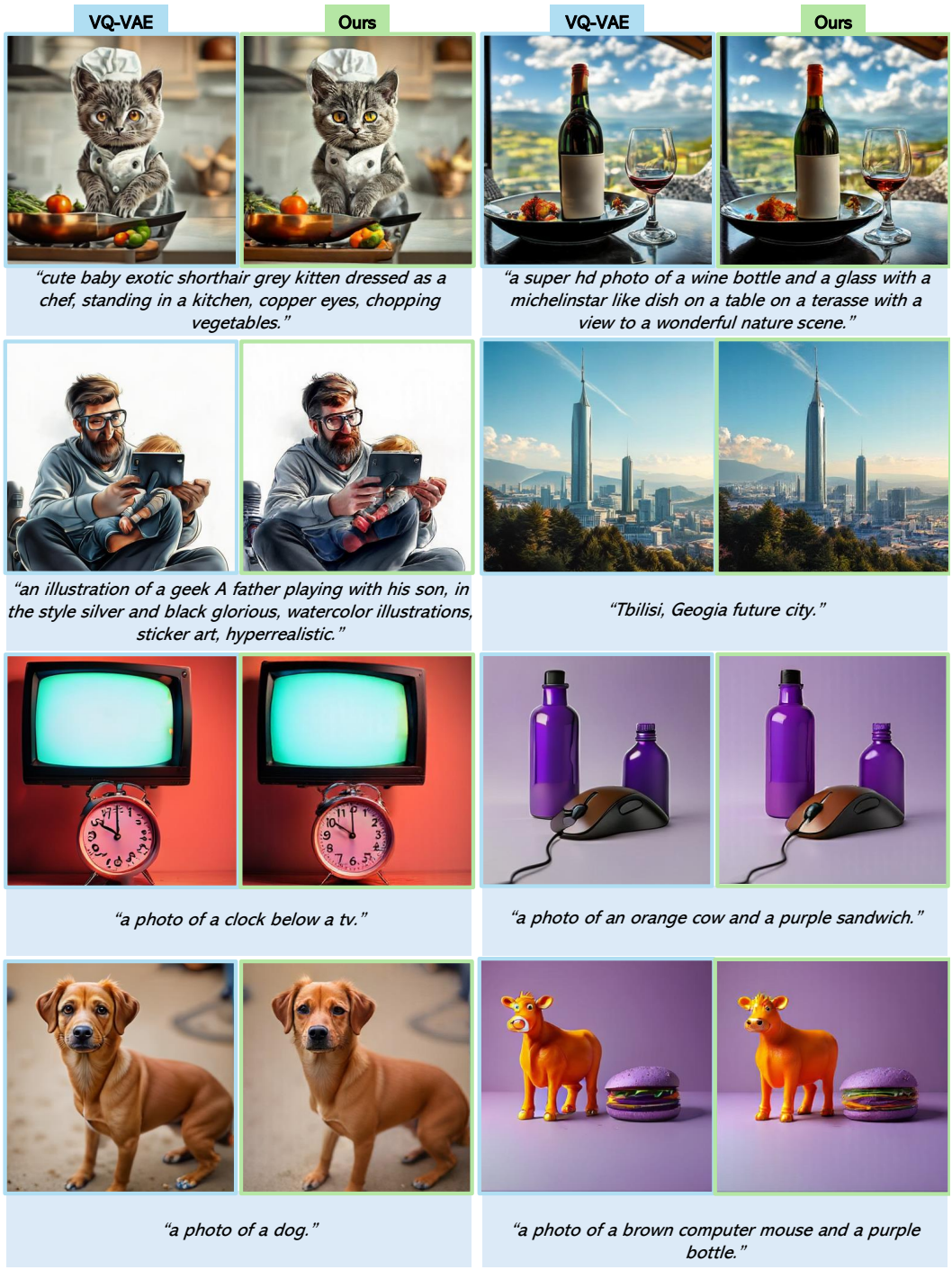


Figure 10: **Text-to-image samples on Janus-Pro 7B** (384×384). VQ-VAE decoder versus DCDD on identical VLM-predicted token sequences. Top two rows: prompts from MJHQ-30K. Bottom two rows: prompts from GenEval. DCDD improves visual fidelity while preserving the VLM's semantic intent.



Figure 11: **Reconstruction samples on LlamaGen-XL** (256×256). VQ-VAE decoder versus DCDD on identical encoded representations. Top two rows: ImageNet-1K. Bottom two rows: MJHQ-30K. As with Janus-Pro 7B, reconstruction fidelity improves substantially across both datasets.



Figure 12: **Text-to-image samples on LlamaGen-XL** (256×256). VQ-VAE decoder versus DCDD on identical VLM-predicted token sequences. Top two rows: prompts from MJHQ-30K. Bottom two rows: prompts from GenEval. Consistent with the regime-of-applicability analysis (Appendix F), T2I gains are limited on this backbone because the upstream token sequence is already semantically degraded.

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 introduction state DCDD's contributions, and the regime-of-applicability scope is explicitly stated in the contributions list and characterized in Section 5.7 and Appendix F.

   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: Appendix H discusses dependence on the upstream VLM (also characterized in Section 5.7 and Appendix F), scope across tokenizers, distribution shift including untested long-form and out-of-distribution prompts, and computational cost including latency and peak memory.

   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 contains no main theoretical claims; Appendix C.2 provides a supporting proposition with full proof.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper provides full method formulation in Section 4 and Appendix A, complete hyperparameter and inference settings in Section 5.1 and Appendix B, the calibration procedure with converged parameter values in Appendix A.2, and the computed evaluation metrics on standard public benchmarks (ImageNet-1K, MJHQ-30K, GenEval, DPG-Bench, WISE).

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 for DCDD training, generation, reconstruction, logit calibration, and evaluation is available at https://github.com/jiwoohong93/dcdd_code. All datasets (ImageNet-1K, MJHQ-30K) and pre-trained backbones (Janus-Pro 7B/1B, LlamaGen-XL) are publicly available.

   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: Training and inference settings are reported in Section 5.1 and detailed in Appendix B.

   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: Appendix D.6 reports the mean and standard deviation over three diffusion-noise seeds for the overall metrics of Tables 2, 3, and 4; the VLM token sequences are fixed, so the variation reflects only the decoder's initial noise. Janus-Pro 1B results (Appendix F.3) and the source-substitution analysis (Appendix C.1) are also reported as mean ± SD. Main tables report single runs, all of which fall within the observed range.

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 B state that the diffusion decoder is trained on 4xA100 80GB GPUs with batch size 32 for 50K steps in mixed precision. Inference latency at batch size 1 on a single A100 80GB is reported in Section 5.6 (Table 7). Appendix D.2 reports latency across denoising steps and peak memory (approximately 37.6 GiB for the full pipeline).

   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: Our research conforms with the NeurIPS Code of Ethics. The work uses only publicly available datasets and pre-trained models under their respective licenses, and we discuss societal impacts in Section I.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix I discusses both the benefit of reducing foundation-scale retraining cost for fidelity improvements and potential concerns regarding deceptive synthetic content and inherited bias from upstream VLMs, with deployment recommendations.

    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: Our work modifies the decoding stage of existing publicly released models; the upstream image-generation capability is provided by the original VLMs and governed by their release terms. DCDD does not unlock new generation capabilities beyond what the underlying VLM already supports, and outputs are subject to the same safeguards applicable to the underlying VLM.

    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: All used assets (Janus-Pro 7B, Janus-Pro 1B, LlamaGen-XL, FLUX.1-Fill, FLUX.1 Kontext, ImageNet-1K, MJHQ-30K, GenEval, DPG-Bench, WISE, CLIP, DINOv2) are cited in the paper and used consistent with their respective licenses and terms of use.

    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: [N/A]

    Justification: The paper does not release new datasets. Trained decoder weights, if released, would be documented at release time.

    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 crowdsourcing or research with human subjects.

    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.

    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: [N/A]

    Justification: LLMs were used only for incidental writing and editing assistance and are not part of the core methodology. The method, experiments, and analysis do not depend on LLMs as a research component.

    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.