End-to-End Training for Unified Tokenization and Latent Denoising

Shivam Duggal[1*] **Xingjian Bai**[1*] **Zongze Wu**[2] **Richard Zhang**[2]
Eli Shechtman[2] **Antonio Torralba**[1] **Phillip Isola**[1] **William T. Freeman**[1]
[1]Massachusetts Institute of Technology [2]Adobe

Abstract

Latent diffusion models (LDMs) enable high-fidelity synthesis by operating in learned latent spaces. However, training state-of-the-art LDMs requires complex staging: a tokenizer must be trained first, before the diffusion model can be trained in the frozen latent space. We propose UNITE – an autoencoder architecture for unified tokenization and latent diffusion. UNITE consists of a *Generative Encoder* that serves as both image tokenizer and latent generator via weight sharing. Our key insight is that tokenization and generation can be viewed as the same latent inference problem under different conditioning regimes: tokenization infers latents from fully observed images, whereas generation infers them from noise together with text or class conditioning. Motivated by this, we introduce a single-stage training procedure that jointly optimizes both tasks via two forward passes through the same Generative Encoder. The shared parameters enable gradients to jointly shape the latent space, encouraging a "common latent language". Across image and molecule modalities, UNITE achieves near state-of-the-art performance without adversarial losses or any pretrained encoders (e.g., DINO), reaching FID 2.12 and 1.75 for Base and Large models on ImageNet 256×256. Our experiments suggest that unification provides mutual benefits for tokenization and generation. We further analyze the Generative Encoder through the lenses of representation alignment and compression. These results show that single-stage joint training of tokenization and generation from scratch is feasible. **Project Page:** https://xingjianbai.com/unite-tokenization-generation/

1 Introduction

Modern foundation models [3, 29, 41, 7, 14, 40, 46]—from language models to video generators, vision–language systems, and scientific generative models—are built around two core operations: tokenization and generation. Tokenization maps high-dimensional observations into a compact latent space that enables both faithful reconstruction and efficient discrimination; generation learns a distribution over this space to synthesize plausible new samples. This division naturally suggests a sequential recipe: first learn a representation space that is easy to reconstruct from and useful for downstream computation; then learn a generative process that samples from that space. As a result, most systems treat tokenization & generation as *separate* design problems & train them in stages – learning a tokenizer, freezing it & only then fitting a generator on the induced latent distribution.

This separation is convenient, but it departs from the principle of end-to-end learning and leaves a basic question unresolved: *should tokenization and generation be trained jointly so that each objective can shape the learned latent space*? In a joint setting, generative pressure could sculpt the latent space toward regions that are easier to model, while reconstruction pressure could preserve instance-specific information and semantic structure. Understanding what emerges when these objectives are trained together—and whether their interaction helps or hurts—is the starting point of this work.

[*]Equal contribution.

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

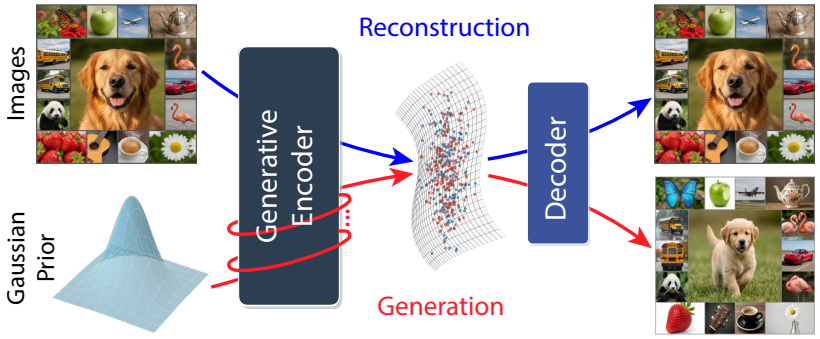

Figure 1: **Unified Tokenization & Generation via Generative Encoder (UNITE):** We propose a single-stage architecture that unifies tokenization and generation through shared parameters. The *Generative Encoder* operates in two modes: (top) as a tokenizer, it processes image patches and register tokens to produce a latent representation z_0; (bottom) as a generator/denoiser, it evolves latents along a flow-matching trajectory to synthesize z_0 from Gaussian noise. The latent space is jointly shaped by reconstruction & generative objectives from scratch, without external supervision.

A natural way to pursue this idea is to start from the standard latent generative pipeline. A tokenizer is typically learned as part of an autoencoder with an encoder E and decoder D: the encoder maps an image x to a latent sequence $z = E(x)$ and the decoder reconstructs $\hat{x} = D(z)$. A generator then models the latent distribution, most commonly by training a diffusion/flow denoiser [20, 43, 33] on noisy versions of z: sample a noise level t, form z_t by corrupting z, and learn a network that predicts the clean latent (or an equivalent parameterization) so that new samples can be generated by starting from noise and iteratively denoising in latent space. In a joint training setting, the same latent z must therefore serve two purposes: it must be decodable by D to preserve instance information, and it must be structured in a way that makes the denoising objective well-posed and easy to learn.

Prior works have explored fully end-to-end training of latent diffusion models (LDMs) by backpropagating the denoising objective through the tokenized latents and into the encoder. However, when the tokenizer and diffusion model are optimized primarily through the denoising objective, this can lead to degenerate solutions and poor performance, as observed in REPA-style methods [51, 30]. To address this, these works propose anchoring the tokenizer with an additional objective that aligns diffusion features to pretrained visual encoders. Although effective, this strategy introduces a third component, a pretrained teacher, to stabilize joint optimization. In contrast, we rely *only* on reconstruction & denoising objectives to jointly train the tokenizer & LDM, without any external supervision.

We propose an alternative perspective on end-to-end training. Our key insight is that tokenization and generation can be viewed as the same latent inference problem under different conditioning regimes (see Fig. 2). Tokenization can be viewed as a *generative process under strong observability*: given a data point x, the model induces a highly concentrated (near-single-point) distribution over latents, yielding a latent z that is consistent with and informative about x. Generation corresponds to a weak-observability regime, where z must be synthesized from noise (and optional conditions) using the learned prior. Under this view, these two operations differ mainly in how much information is available—from the full observation x in tokenization to only a prior in generation. Motivated by this view, we propose UNITE, which jointly trains tokenization and generation end-to-end without external supervision. UNITE ties tokenization and generation through a shared-parameter module, the Generative Encoder (GE), so gradients from both objectives shape the same weights and drive the model toward a representation jointly optimized for both tasks. Hence the name, **UNITE**: Unifying Tokenization & Latent Generation via shared Generative Encoder. See Fig. 1 for an overview.

Concretely, our system consists of only two modules: a *Generative Encoder* GE_θ and a decoder D_ϕ. The GE operates in two modes: (i) *tokenization*, mapping an input x to latent tokens $z = \mathrm{GE}_\theta(x)$, and (ii) *generation*, denoising corrupted latents to produce $\hat{z} = \mathrm{GE}_\theta(z_t, t)$ at noise level t. Thus, the same network serves both as the tokenizer and as the multi-step latent denoiser, with parameters θ shared across the two objectives. Training proceeds with two forward passes through GE_θ. First, we tokenize an input image to obtain clean latents z. We then corrupt z using a rectified-flow (flow-matching) process to obtain z_t, and pass z_t back through GE_θ to predict the corresponding denoising target. The full pipeline is trained end-to-end in a *single stage* by jointly optimizing a pixel-space reconstruction objective and a latent-space flow-matching objective.

We find that this end-to-end formulation yields a strong latent generative model, with near state-of-the-art generation & reconstruction FIDs, while training all modules from scratch rather than relying on large pretrained networks. To understand what drives this behavior, we study multiple alternatives to end-to-end training in Sec. 4 and Appendix Sec. E.

This includes an ablation that keeps the full training pipeline fixed but removes parameter tying between the encoder and denoiser. Interestingly, even without explicit weight sharing, the encoder and denoiser exhibit strong per-layer representational alignment, as measured by centered kernel alignment (CKA) [28] (See Fig. 6), suggesting that tokenization and latent denoising are intrinsically compatible tasks in our setting – thus, motivating weight-sharing. Further analysis (Tab. 8, Sec. 4 and Appendix Sec. E) indicates that the model differentiates the two modes primarily through normalization: the tokenization and denoising pathways occupy different norm/scale regimes, while attention and MLP sublayers remain highly reusable across both. In fact, recent concurrent work on Unified Latent [19] investigates a closely related two-module formulation. It can be interpreted as a special case of our end-to-end setting, aligning closely with our separate-weights ablation. While this separate-weights variant performs almost as competitively, we find that parameter tying yields the best overall reconstruction-generation trade-off in our experiments (Fig. 5 and Tab. 6). Overall, these results provide a concrete single-stage recipe in which reconstruction and denoising objectives can jointly shape the latent space, rather than being optimized in disjoint stages. Practically, this means **one training job & one model** to attain near state-of-the-art tokenizer and generator.

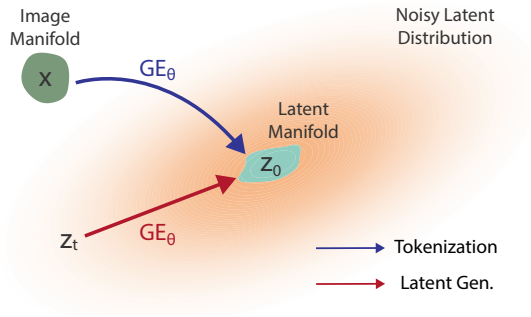

Figure 2: Tokenization and Generation can be viewed as the *same latent inference problem under different conditioning regimes.* In tokenization, the full observation x strongly constrains the clean latent, $z_0 \sim p_\theta(z \mid x)$; in generation, with noisy latent z_t as a weaker evidence, the same target latent z_0 is recovered by denoising. This view motivates using a **single shared Generative Encoder** (GE_θ) for both tokenization and latent denoising operations.

2 Related Work

Tokenization & Generation via Auto-Encoding: VAEs [27], VQ-VAE [45], and VQ-GAN [15] provide foundational frameworks for learning latent representations through reconstruction, with VQ-GAN-style tokenizers also relying on adversarial losses [16]. However, in modern latent diffusion pipelines [39, 35], these VAE/VQGAN-style models primarily serve as *tokenizers* for a downstream diffusion model trained in the resulting frozen latent space; their standalone generative capability is typically weaker and is therefore rarely used in practice. In standard downstream diffusion training, denoising gradients do not update the tokenizer, so the representation is not shaped for generation. We address this by coupling tokenization with latent denoising and *train a single model end-to-end for both tokenization and generation, without any adversarial losses* unless otherwise stated.

Self-supervised Visual Encoders for Generation: Self-supervised learning has produced visual encoders that go beyond naive reconstruction. MAE [18] learns strong representations via masked patch reconstruction, while DINO-style models [4, 38] learn semantic features through self-distillation. Recent works adapt MAE-style objectives to better train tokenizers for two-stage generative modeling: l-DeTok [49] corrupts MAE embeddings before decoding, while MAETok [6] adds auxiliary decoders to predict semantic target features. Building on these, REPA [51], REPA-E [30], VAVAE [50], and RAE [54] use pretrained SSL models to improve diffusion training, either by aligning diffusion features to SSL representations or by replacing/tuning the tokenizer around the SSL space. While effective, these methods require external pretrained encoders and pipeline staging. Importantly, their best performance typically relies on DINOv2 / SigLIP encoders trained on substantially more ($140\times$) data than ImageNet, making them less directly comparable to ImageNet-only methods. In contrast, we study single-stage joint tokenization and generation without any pretrained SSL encoders.

Pixel-Space Diffusion Models denoise directly in RGB space, avoiding learned latents but facing sharper scaling issues at high resolution. Prior work addresses changing denoising difficulty/SNR with resolution through noise scaling or loss reweighting [21, 10, 26]. Other methods introduce architectural adaptations for high-resolution pixel modeling: SiD2 [22] reduces high-resolution feature capacity, PixelFlow [9] alternates denoising with progressive upsampling, and PixNerd [47],

PixelDiT [52], DiP [11] use specialized heads for fine-grained inputs. JiT [31] trains a plain ViT generator directly on raw patches without tokenizers or auxiliary losses. Unlike pixel-space approaches for single-stage generation, UNITE is one of the first *single-stage latent diffusion approaches*.

Concurrent work: Please refer to Appendix Sec. C for relevant concurrent work [19, 2, 5].

3 Unifying Tokenization & Latent Denoising

Can we jointly train a tokenizer and a generator end-to-end in a single stage, such that gradients from one objective meaningfully shape the other? The answer is yes: we show that single-stage end-to-end training can learn a latent space that supports both high-fidelity reconstruction and iterative generation. While recent work has begun to explore end-to-end training, most approaches still rely on multi-stage pipelines (e.g., pretraining or freezing parts of the system) or introduce external supervision from pretrained representation models. These design choices can be effective, but they make it harder to isolate and study the intrinsic interaction between tokenization and generation. In this work, we take a step toward single-stage joint tokenization and generation without external supervision, using a single unified network trained simultaneously with reconstruction and latent denoising objectives.

In many ways, an early and elegant solution to this already exists: *variational autoencoders* (VAEs) jointly learn an encoder–decoder for reconstruction while also imposing a simple latent prior, typically $\mathcal{N}(0, I)$, that enables sampling and generation. This classical design suggests that tokenization and generation need not be separated into distinct stages.

3.1 From VAE to UNITE

In a VAE, the "tokenizer" is the encoder, E_θ: it maps a data point x to a conditional latent distribution $q(z \mid x)$ rather than a single code. The decoder, D_ψ, reconstructs by sampling $z \sim q(z \mid x)$ and mapping back to data space via $p(x \mid z)$. For any generative model, a central requirement is to map an easy-to-sample distribution into an expressive latent space that supports high-quality decoding. VAEs operate by regularizing the encoder, so that the latent distribution remains close to a simple prior $p(z) = \mathcal{N}(0, I)$ (via KL term), making generation as simple as sampling $z \sim p(z)$ and decoding.

$$\textbf{VAE:}\ \ z = E_\theta(x); \hat{x} = D_\psi(z);$$

Notably, VAE-family encoder–decoder tokenizers have become a standard building block in modern vision and video foundation-model pipelines: high-dimensional visual inputs are first compressed into latents via a VAE/VQ-style encoder, but generation is performed in latent space by a separate model. In this regime, autoencoders primarily serve as a tokenizer rather than final generative model, since a simple Gaussian prior typically does not reach the sample fidelity of modern diffusion generators.

Modern high-fidelity generative models therefore replace VAE-style Gaussian prior sampling with a learned iterative generative process, while retaining the VAE's role as the tokenizer. In latent diffusion and flow models, a VAE-style encoder first maps data into a compact latent space, and a separate denoising model, G_ϕ, is trained to transform Gaussian noise into samples from the latent data distribution via iterative denoising. In practice, this is often implemented as a staged pipeline: the tokenizer is trained and frozen; the denoiser is trained on top of the fixed latent space.

$$\textbf{LDM:}\ \ z = E_\theta(x); \hat{x} = D_\psi(z); \hat{z} = G_\phi(z_t, t);$$

UNITE replaces the separate tokenizer and latent denoiser with a shared set of parameters, the Generative Encoder, as demonstrated in Fig. 2. This shared module retains the simplicity of the autoencoder interface—an encoder and a decoder—while enabling *single-stage* learning of both tokenization and generation. Paired with a decoder D_ψ that maps latents back to image space, the Generative Encoder GE_θ operates in two modes. In *tokenization mode*, GE_θ maps an image x to latent tokens $z = \mathrm{GE}_\theta(x)$ optimized for reconstruction, without enforcing an explicit KL-to-Gaussian bottleneck. In *generation mode*, the same GE_θ is used as a latent denoiser: given a noisy latent z_t and noise level t, it predicts the corresponding denoising target, enabling iterative sampling from Gaussian noise at inference time. Sharing parameters across these two modes lets gradients from both objectives jointly shape the same weights in a single training job. This yields a minimal end-to-end pipeline with performance approaching modern latent generative models.

$$\textbf{UNITE:}\ \ z = GE_\theta(x); \hat{x} = D_\psi(z); \hat{z} = GE_\theta(z_t, t);$$

3.2 End-to-End Training for UNITE

Training Pipeline: We adopt a simple pure self-attention Vision Transformer (ViT) [12] backbone for both the Generative Encoder and the decoder, motivated by the strong empirical performance of Transformer architectures in diffusion / flow denoising. The generative encoder GE_θ must support two operating modes with compatible input/output types: *tokenization pathway*, ingesting image patch tokens to produce compact latent representation, and *generation (denoising) pathway*, ingesting noisy latents along a flow or diffusion trajectory that connects the latent distribution to a standard normal prior.

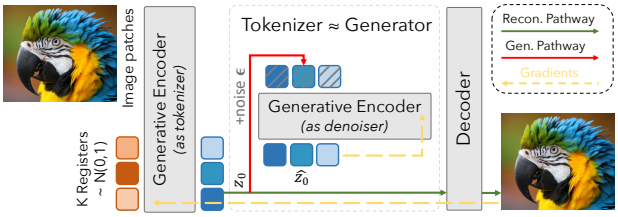

Figure 3: **UNITE Training Pipeline** uses two forward passes through the Generative Encoder. First, it maps image patches into latent registers; second, it denoises a noised version of those latents. The weights of Generative Encoder are shared across these two passes. Its training combines reconstruction losses (L1 loss and LPIPS loss) with a velocity-space denoising loss $\|\mathrm{toVelo}(\hat{z}_0; z_t, t) - \mathrm{sg}(\mathrm{toVelo}(z_0; z_t, t))\|_2^2$.

To unify the input format across pathways, we represent the latent z as a fixed set of K *register tokens*. In the tokenization pathway, we concatenate the image patch tokens with these K registers, initializing the registers as i.i.d. Gaussian noise, $\mathcal{N}(0, I)$, to match the input distribution at the maximum noise level. The concatenated sequence is processed with self-attention in a *first* forward pass through GE_θ. We then discard the patch tokens and retain only the updated registers. These updated registers serve as the image latents z_0, having absorbed the relevant information from patches through attention. The decoder D_ψ consumes z_0 & reconstructs the image using a ViT-style stack plus a lightweight unpatchification head to produce pixels. Note: (a) Both GE_θ, D_ψ use standard self-attention architectures, no specialized cross-attention for tokenization or denoising. (b) The size of GE_θ is comparable to the modern tokenizers, DINOv2 & MAE, in most experiments. (c) We study an extreme case of full weight sharing, one could have a small shared backbone & separate heads.

In the generation (denoising) pathway, we first detach and corrupt the clean latents z_0 to obtain a noisy latent z_t at noise level t (using our rectified-flow / flow-matching corruption process) and then use z_t to initialize the same K registers. *No image patches are concatenated in this pathway.* A *second* forward pass through GE_θ (now in generation mode, conditioned on t and optional class information) predicts an x-start / clean-latent estimate, i.e., $\hat{z}_0 = \mathrm{GE}_\theta(z_t, t)$, so the denoiser output lies in the same space as the tokenizer, with loss computed in velocity space.

The final layer of GE_θ is a normalization module. Empirically, we find that LayerNorm [1] with learnable scale and shift parameters performs best. As a result, the clean latents z_0 (from the tokenization pathway), the denoised predictions $\hat{z}_0$ (from the generation pathway), and the model outputs at each denoising step during inference are all normalized.

Overall, as in Fig. 3, each training iteration performs two forward passes through the shared GE_θ: an image-conditioned pass to produce clean latents for reconstruction, followed by a latent-only pass to denoise a corrupted version of those latents. The system is trained end-to-end in a *single stage* by jointly optimizing pixel-space reconstruction (via D_ψ) & latent-space denoising objective (via GE_θ).

Training Objectives: We optimize two losses computed from the two forward passes described above. For reconstruction, we encode the image into clean latents $z_0 = \mathrm{GE}_\theta(x)$, inject small Gaussian noise $\tilde{z}_0 = z_0 + \sigma\epsilon$ with reconstruction noise scale $\sigma = 0.3$ following [30, 51], and decode $\hat{x} = D_\psi(\tilde{z}_0)$. The reconstruction loss combines pixel-level and perceptual terms: $\mathcal{L}_{\text{recon}} = \|\hat{x} - x\|_1 + \mathrm{LPIPS}(\hat{x}, x)$. For generation, we apply rectified flow matching [34] on the latents. Given clean latents z_0, we construct noisy latents $z_t = tz_0 + (1-t)\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ and $t \sim \mathcal{U}[0, 1]$ (where t=0 corresponds to pure noise and t=1 to clean data), then train the generative encoder to predict clean latents via $\hat{z}_0 = \mathrm{GE}_\theta(z_t, t)$. We minimize $\mathcal{L}_{\text{flow}} = \mathbb{E}_{t,\epsilon}[\|\hat{v} - \mathrm{sg}(v)\|_2^2]$, where $\mathrm{sg}(\cdot)$ denotes stop-gradient to prevent degenerate solutions, $\hat{v} = \mathrm{toVelo}(\hat{z}_0; z_t, t)$ and $v = \mathrm{toVelo}(z_0; z_t, t)$. The function toVelo maps clean latent to flow velocity. The total objective is sum of reconstruction & generation losses.

Inference: At inference, the Generative Encoder can serve as the tokenizer by mapping an input image to its latent representation in a single forward pass. For generation, we start from a class label and noisy latent registers, and iteratively refine them through multiple passes of the GE into clean, decodable latents (shown as red loops in Fig. 1).

3.3 Understanding UNITE's Training Dynamics

The adversarial nature of joint training. Jointly training tokenization and generation under weight sharing induces non-trivial dynamics. In a standard LDM, the latent space is produced by a pretrained (and typically frozen) tokenizer, so the generative objective does not shape the latent interface. In UNITE, reconstruction and generative objectives are optimized jointly over the same parameters, so each objective can influence the representations used by the other.

This dynamic can be understood as the search for a latent space shaped by two complementary pressures. The reconstruction objective encourages the encoder to preserve sufficient information for faithful decoding, while generative objective encourages the shared network to organize this information so that corrupted latents are easier to denoise. Joint optimization therefore balances informativeness & denoisability: latent space must be rich enough for reconstruction, yet structured enough to support low-loss denoising/high-fidelity generation.

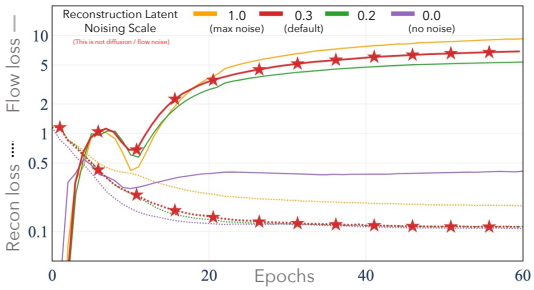

Figure 4: **UNITE training dynamics.** Joint training with reconstruction (dotted) and denoising (solid) objectives exhibit adversarial training behavior. Non-starred curves show reconstruction-noise ablations for decoder robustness; see Appendix Sec. G.3, Tab. 12.

Empirically, this interaction can resemble an "adversarial" game: the two losses do not necessarily decrease monotonically together. Improvements in generative fidelity can even coincide with an increase in denoising loss, as shown in Fig. 4 (see red curves with star markers). Crucially, a rising denoising loss does not imply worse generation. Instead, it often signals that the latent space is becoming richer and more informative to satisfy the reconstruction objective, making the denoising task harder but the resulting samples more realistic. During training, we often observe generation metrics (e.g., FID/IS) improving even as the denoising loss increases, until the system reaches a stable equilibrium. Similar to GAN-style training, the goal is therefore not to drive all losses to zero, but to reach *stable* training dynamics where the latent space balances information density with generative robustness. This perspective is consistent with modern diffusion models, where denoising loss typically stabilizes at a non-zero value.

4 Analyzing UNITE's Generative Encoder

In UNITE, we pursue end-to-end training by *sharing* parameters between the encoder and denoiser roles of a single network. This choice suggests a natural hypothesis: parameter tying encourages the model to develop a common latent "language"—shared internal features and transformations that simultaneously support reconstruction and iterative denoising.

To better understand this design choice, we study two alternative routes to end-to-end latent diffusion training that each relax a component of our Generative Encoder mechanism. First, we remove parameter tying, maintaining *separate* encoder and denoiser networks while still training both objectives jointly. Second, we remove the stop-gradient through clean latents, allowing denoising gradients to backpropagate into the tokenization pathway (presented in Appendix Sec. E). Together, these alternatives help isolate the role of weight sharing and gradient flow in our end-to-end formulation. Finally, we study these end-to-end training approaches through the lenses of representation alignment and compression via Fig. 6, Fig. 7, Fig. 8, Tab. 8 and Appendix Sec. E.

Weight-Shared vs. Separate Encoder – Denoiser Training: Our Generative Encoder ties the encod-

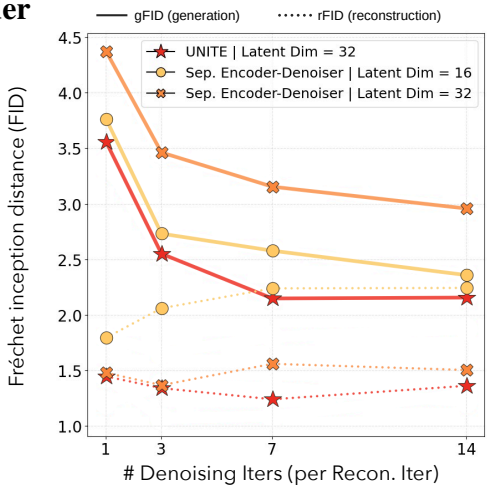

Figure 5: **Weight-shared vs. Separate Enc-Denoiser training.** Both UNITE (weight-shared setting) and separate encoder-denoiser ablation attain *competitive* performance, with UNITE benefiting from more denoising-to-recon. steps ratio during training, achieving best overall rFID–gFID trade-off. Also, see Appendix Tab. 6 for additional single-image reconstruction metrics – PSNR, SSIM.

ing and denoising roles by sharing parameters. As an ablation, we keep the entire end-to-end pipeline fixed—including the stop-gradient that prevents denoising gradients from flowing through the tokenization output into the encoder–but instantiate *separate* networks for the encoder and the denoiser. In this separate-networks ablation, the encoder & denoiser are optimized for their own objectives, with no gradient interaction involved.

If the weight-shared Generative Encoder matches or improves upon this separate-weights variant, it already offers a practical advantage: fewer parameters to store and update, and a shorter description length (MDL) for the learned model. Fig. 5 (and Tab. 6) shows that, while the separate-weights ablation is competitive, parameter tying yields the best overall reconstruction–generation trade-off. Specifically, we report rFID and gFID as a function of the number of denoising (flow) steps performed per reconstruction step during training. Table 7 reports a gFID of 2.12 with shared weights and 2.93 with separate weights, using 14 denoising updates per reconstruction and latent dimension 32. Even with more compressed latent dimension of $d = 16$, weight sharing achieves better gFID of 2.24 compared to the separate encoder-denoiser gFid of 2.58. Furthermore, Table 4 compares these weight-shared and separate encoder-denoiser joint-training approaches with two-stage alternatives for downstream generation capabilities. Training a fresh denoiser on a frozen reconstruction-only tokenizer yields gFID 2.99, while using a frozen UNITE tokenizer yields 2.60. Joint training with separate encoder and denoiser weights reaches 2.93 with stop-gradient and 2.24 without it (we study the role of stop gradient operator between denoiser and tokenizer in Appendix Sec. E.). The jointly trained shared-weight UNITE denoiser achieves the lowest gFID=2.12 among these configurations, *highlighting the benefit of single-stage, joint, weight-shared training of tokenizer & generator.*

Tokenization-Generation Representation Alignment Analysis: As shown in Fig. 2, tokenization can be viewed as a generative process under strong observability, $p_\theta(z \mid x)$, whereas generation corresponds to unconditional sampling from the induced prior, $z \sim p_\theta(z)$. This viewpoint suggests that the two tasks may be aligned, and motivates measuring representational alignment between the two modes. We test this by measuring alignment between tokenization & denoising-pathway activations using Centered Kernel Alignment (CKA / CKNNA) & Cosine Similarity (Fig. 6). Precisely, for an input image, we first record intermediate activations along the tokenization pathway, then corrupt the encoded latent & record the corresponding denoising-pathway activations.

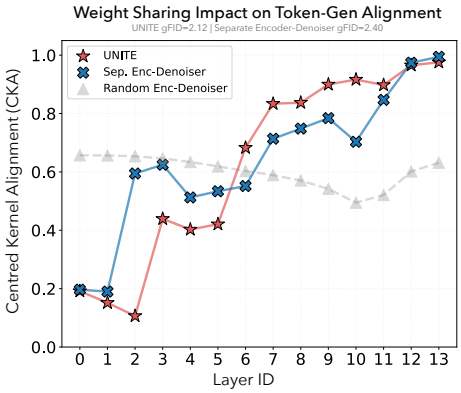

Figure 6: **Tokenization-Generation Alignment.** We measure CKA alignment between tokenization & denoising activations. Both weight-shared UNITE & separate encoder–denoiser ablation show strong alignment, especially in later layers, suggesting *tokenization & denoising are intrinsically aligned tasks, motivating weight sharing.*

Several aspects of our design encourage alignment. First, both modes are trained to operate in the same latent space: the denoiser is supervised to predict the corresponding clean latent for a corrupted version of the encoded latent. Second, the GE receives the same latent register parameterization in both modes; during tokenization these registers are initialized from $\mathcal{N}(0, 1)$, reducing input-domain mismatch between tokenization and generation. Finally, we adopt architectural and optimization choices that limit drift between modes: (i) consistent normalization throughout the network (within blocks and at the encoder output), (ii) matched conditioning interfaces across modes to avoid mode-specific shortcuts. Specifically, during tokenization we use the same conditioning interface as diffusion: the time embedding is set to the minimum-noise timestep, and class conditioning is omitted, corresponding to the no-CFG diffusion setting. (iii) conservative optimization (learning-rate warmup and schedules) to prevent one objective from dominating shared parameters early in training.

With these choices in place, we find that both the weight-shared Generative Encoder and the separate encoder-denoiser variant exhibit high CKA/CKNNA alignment (Fig. 6, left), indicating that tokenization and denoising are *intrinsically aligned tasks* in our setting. This also clarifies the role of weight sharing: when the two tasks already align, parameter tying becomes a principled way to remove redundancy (see Compression analysis in Appendix Sec. E and Tab. 8) – especially in the reusable functional layers (attention, MLPs) – while retaining strong reconstruction & generation fidelity.

5 Experimental Results

Can a single training job produce both a strong tokenizer & a strong generator? We show that UNITE achieves near–state-of-the-art reconstruction and generation performance across image and molecule modalities. See Appendix for more results, ablations, experiment details and training effeciency.

5.1 ImageNet-256 Results

Generation. Tab. 1 summarizes our main generation results on ImageNet-256. The results indicate that truly end-to-end training of tokenization and generation is not only feasible, but also competitive. In particular, **UNITE-B reaches an FID of 2.12**, substantially improving over the single-stage baseline JiT-B/16 [31] (FID 3.66) and even JIT-L (FID 2.36). Increasing the model capacity further improves performance: UNITE-L, UNITE-XL both reduce FID to **1.75** (XL is only trained for 120 eps), surpassing two-stage approaches such as DiT-XL/2 (FID 2.27) and SiT-XL/2 (FID 2.06), suggesting that the unified setup continues to benefit from scale. Unlike previous LDMs training VAEs w/ GAN-based adversarial losses, **UNITE uses no adversarial loss.**

Unlike RAE [54] and REPA [51], which fundamentally rely on pretrained vision encoders (trained on $140\times$ more data), our single-stage approach reaches comparable performance while training from scratch, without requiring an external pretrained representation model.[1] This simplicity—one encoder that serves both tokenization & generation via weight sharing — makes the system easier to train & deploy, reducing reliance on pretrained components.

Table 1: **ImageNet 256×256 generation.** UNITE outperforms recent single-stage pixel baselines and standard two-stage latent diffusion works by a large margin. Yellow rows use DINOv2, which is pretrained on 140M curated images ($140\times$ larger than ImageNet), and are therefore less directly comparable to ImageNet-only methods. See RAE Appendix G.1, Table 15a – gFID significantly increases when replacing DINOv2 with Imagenet-trained MAE.

Method	Aux. Token	Params	FID↓	IS↑
Single-stage Frameworks				
JiT-B/16	-	131M	3.66	275.1
UNITE-B (Ours)	**Joint**	217M	**2.12**	**294.1**
RIN	-	410M	3.42	182.0
JiT-L/16	-	459M	2.36	298.5
ADM-G	-	554M	4.59	186.7
UNITE-L (Ours)	**Joint**	589M	**1.75**	**296.0**
PixelFlow-XL/4	-	677M	1.98	282.1
PixelGen-XL/16	-	677M	1.83	293.6
PixNerd-XL/16	-	700M	2.15	297
UNITE-XL (Ours)	**Joint**	806M	**1.75**	**309.9**
JiT-H/16	-	953M	1.86	303.4
SiD	-	2B	2.44	256.3
VDM++	-	2B	2.12	267.7
JiT-G/16	-	2B	1.82	292.6
Two-stage Frameworks				
DiT-XL/2	SD-VAE	675M+49M	2.27	278.2
SiT-XL/2	SD-VAE	675M+49M	2.06	277.5
Two-stage Frameworks with Aux Supervision (DINOv2)				
REPA	SD-VAE	675M+49M	1.29	306.3
LightningDiT-XL/2	VA-VAE	675M+49M	1.35	295.3
DDT-XL/2	SD-VAE	675M+49M	1.26	310.6
REPA-E	SD-VAE	675M+49M	1.15	304.0
RAE-DiT^{DH}-XL/2	DINOv2	839M+415M	1.13	262.6
Concurrent works				
LF-DiT-L	DINOv2	465M	2.48	–

Compared with concurrent works including Latent Forcing [2] (see LF-DiT-L in Tab. 1), Unified Latents [19] and Self-Flow [5], our method is fully trained from scratch while achieving a strong generation FID. Unified Latents reports results only on ImageNet-512, with best performance relying on second-stage fine-tuning. Self-Flow reports only unconditional gen results using DINOv2.

Comparison with Single-Stage Generation Pipelines. UNITE is most directly comparable to pixel-space generative models, which are also trained in a single stage and typically do not rely on DINOv2-based supervision. As shown in Tab. 2, UNITE-B achieves a gFID of 2.12, substantially outperforming JiT models of comparable or slightly larger size, including JiT-B (3.66) and JiT-L (2.36). While JiT does not use an LPIPS-based loss, PixelGen [36]-style variants add VGG- and DINO-based LPIPS losses to JiT. In our JiT-B ablation at epoch 120, adding gating to the LPIPS variant lowers gFID from 4.36 to 4.09. At epoch 240, the gated JiT-B variant reaches gFID 3.64, compared with 2.12 for UNITE-B. LPIPS is more critical for latent-space reconstruction, and in UNITE it is applied only to reconstructions from image-tokenized clean latents, rather than to decoded denoising trajectories at all timesteps. In contrast, PixelGen applies LPIPS across denoised reconstructions from low to high noise levels, which may introduce a harder optimization trade-off.

Reconstruction. Tab. 3 compares the reconstruction quality of UNITE against existing tokenizers. Most prior methods decouple reconstruction and generation, and low-rFID tokenizers such as VAEs and VQGANs typically strongly rely on adversarial objectives in addition to reconstruction losses.

[1] Our ImageNet training uses LPIPS loss, which requires a pretrained VGG. However, (a) training VGG is inexpensive; (b) our molecule gen. results do not use LPIPS. (c) LPIPS is important for latent-space models, see Tab. 2 for JIT+LPIPS results.

Table 2: **ImageNet-256 generation training dynamics.** We compare FID across training epochs for UNITE (flow 14 variants) against pixel-based JiT / PixelGen baselines. UNITE achieves SOTA performance at matched or with even fewer parameters (highlighted in green).

Model	Params	Ep 80	Ep 120	Ep 160	Ep 240	Ep 280	Ep 360	Ep 400	Ep 600
UNITE-B	217M	2.46	2.33	2.18	**2.12**	–	–	–	–
UNITE-L	589M	–	–	**1.75**	*stopped training due to compute*				
UNITE-XL	806M	–	**1.75**	*stopped training due to compute*					
JiT-B + LPIPS	131M	5.10	4.36	*stopped training, w/ gating is better (see PixelGen [36])*					
JiT-B + LPIPS (w/ gating)	131M	4.89	4.09	3.88	3.64	3.70	3.63	*converged*	
JiT-B	131M	10.10	6.75	5.50	4.50	4.25	3.95	3.80	3.66
JiT-L	459M	–	–	–	–	–	–	–	2.36
PixelGen-XL (DINOv2 for LPIPS)	677M	–	–	–	–	–	–	–	1.83
JiT-H	953M	–	–	–	–	–	–	–	1.86
JiT-G	2B	–	–	–	–	–	–	–	1.82

More recent approaches further improve reconstruction fidelity using externally pretrained self-supervised encoders such as DINOv2. As reference, a ViTok-B/16 Stage 1 [17], trained only with L2+LPIPS+KL losses, no adversarial training or DINOv2, obtains an rFID = 1.63.

Despite being trained jointly with a generative objective, **UNITE-B achieves an rFID of 1.01** after 120 epochs, outperforming the vanilla autoencoder baseline (rFID=1.63). A lightweight adversarial fine-tuning stage, which freezes the Generative Encoder and updates only the decoder for 16 epochs, further reduces **rFID to 0.51**, surpassing RAE (0.58) and SD-VAE (0.62) without DINOv2-based pretraining. Increasing the generative encoder size from base to XL improves both gFID $(2.12 \rightarrow 1.75)$, rFID $(1.01 \rightarrow 0.89)$, supporting scaling. We refer to detailed ablation on more reconstruction metrics with generation tradeoff in App. Sec. D, Tab. 6.

Table 3: **ImageNet reconstruction.** Our tokenizer achieves competitive rFID w/o adversarial loss (Adv.) or pretrained encoders. UNITE trained for 120 eps.

Tokenizer	Adv.	Pretrained Encoder	rFID↓
With adversarial / external supervision			
SD-VAE	✓	-	0.62
DC-AE-f32	✓	-	0.69
RAE	✓	DINOv2	0.58
VA-VAE	✓	DINOv2	**0.28**
Without adversarial or external supervision			
ViTok-B/16*	-	-	1.63
UNITE-XL	-	-	0.89
UNITE-B	-	-	1.01
+ GAN decoder ft†	✓	-	**0.51**

* Stage-1 only. † Decoder-only ft, 16 epochs.

Unifying Tokenization and Generation Boosts Generation. One key result of this work is that jointly training tokenization and generation can enhance generative capability. First, UNITE achieves state-of-the-art generation fidelity among methods trained from scratch, without relying on externally pretrained DINOv2 encoders trained on $140\times$ more data than ImageNet (Tab. 1, Tab. 2). Second, compared with our separate-weight ablation and standard ImageNet-trained RAE-MAE / VAE-based DiT baselines, UNITE achieves a stronger reconstruction–generation trade-off, suggesting that shared tokenization and denoising improves the *generative usability of the learned latent space* (Fig. 5, Tab. 2).

Unifying Tokenization and Generation Improves Generation-Aware Tokenization. We next ask whether unified training improves the tokenizer itself. First comparing with prior works, with lightweight GAN-based decoder fine-tuning, UNITE achieves a strong rFID of 0.51 from scratch, without pretrained DINOv2/SigLIP encoders. Second, compared with approaches such as RAE and VAVAE that rely on DINOv2, which is trained on roughly $140\times$ more data than ImageNet, UNITE remains comparable to VAVAE in PSNR/SSIM, rFID, and gFID, while RAE has substantially worse single-image reconstruction metrics (see Tab. 6 for detailed comparison). Moreover, replacing DINOv2 with ImageNet-trained MAE further hurts these approaches (see RAE Appendix G.1).

Next, compared with separate-weight ablation, UNITE achieves comparable PSNR/SSIM, better rFID, and substantially better gFID, suggesting an improved generation–reconstruction trade-off. We further test *generation-aware tokenization* by freezing either the co-trained UNITE tokenizer or a reconstruction-only tokenizer, then training a new denoiser from scratch on top of each, as in a standard two-stage LDM setup. This improves gFID from 2.99 with the reconstruction-only tokenizer to 2.60 with the UNITE tokenizer, providing evidence that generative co-training improves tokenization (for two-stage generation). Table 4 compares these frozen-tokenizer pipelines with joint training using separate or shared weights, including the separate-weight variant without stop-gradient. These results highlight the benefits of joint training. Finally, Linear probing (LP) remains comparable

Table 4: **Unifying Tokenization and Generation Improves Generation-Aware Tokenization.** Training schemes on ImageNet 256×256, all with the same geometry with tokenizer configuration: f16d32, denoising on 1×1 latent patches, 256 tokens).

Tokenizer	Denoiser training	gFID ↓
Reconstruction-only, frozen	Fresh second-stage denoiser	2.99
UNITE, frozen	Fresh second-stage denoiser	2.60
Joint, separate weights, with stop-gradient	Trained with tokenizer	2.93
Joint, separate weights, no stop-gradient	Trained with tokenizer	2.24
Joint, shared weights (UNITE)	**Trained with tokenizer**	**2.12**

to standard autoencoder tokenizers such as VAEs/VQGANs ($\approx 30\%$), as expected. See Appendix Sec. B on why LP might not be the best metric to test the utility of compressed generative tokenizers.

5.2 Beyond Vision: Application to Domains Without Pretrained Encoders

Recent approaches such as REPA [51] and RAE [54] rely on pretrained representation models, e.g., DINOv2 [38], to strengthen latent diffusion. This makes transfer less straightforward in domains where such encoders are unavailable or expensive to train. In contrast, UNITE trains tokenizer & generator end-to-end from scratch, making it applicable beyond vision.

Table 5: **QM9 molecule generation.** UNITE-S achieves best reconstruction accuracy & uniqueness under single-stage training. Crystal gen. results on MP20 in Appendix G.2.

Method	Reconstruction		Generation	
	Match (%)	RMSD (Å)	Valid (%)	Unique (%)
EDM [23]	–	–	91.9	90.7
GeoLDM [48]	–	–	93.8	92.9
ADiT Tokenizer [25]	97.20	0.075	–	–
ADiT-S QM9-only [25]	–	–	**96.02**	97.76
UNITE-S (Ours)	**99.37**	**0.039**	94.90	**99.71**

We demonstrate this on QM9 molecule generation, a setting without a DINO-like pretrained encoder. As shown in Tab. 5, UNITE matches or surpasses the All-atom Diffusion Transformer (ADiT) [25], which trains separate tokenizer and denoiser in stages. UNITE achieves a 99.37% reconstruction match rate (vs. 97.20% for ADiT) and 99.71% uniqueness among generated molecules (vs. 97.76%), training fully end-to-end. These results motivate studying end-to-end tokenization and generation, where both objectives jointly shape the same representation, as an alternative to relying on pretrained encoders for training LDMs.

6 Conclusion

We present UNITE, a unified framework for joint tokenization and generation. Its Generative Encoder serves as both tokenizer and latent denoiser, sharing weights across reconstruction and denoising objectives so both gradients shape a common latent space. UNITE trains end-to-end in a single stage, using one forward pass for tokenization/reconstruction and another for latent denoising through the same module. Across ImageNet 256×256 and molecule generation, UNITE achieves strong near-state-of-the-art fidelity, with base model reaching 2.12 gFID and Large model improving to 1.75 gFID. Our experiments suggest that unification provides mutual benefits for tokenization and generation. We also demonstrate representation-alignment and compression analyses of the Generative Encoder.

Furthermore, our results suggest two practical implications. First, UNITE reduces reliance on pretrained encoders such as DINOv2, enabling latent generative modeling in domains where such encoders are unavailable. Second, its unified architecture simplifies conventional two-stage pipelines, reducing implementation complexity and avoiding additional pretrained-encoder overhead.

Acknowledgements

We are grateful to Jyo Pari, Shamit Lal, Tianyuan Zhang, Suwan Kim, Peter Holderrieth & Qianwei Jia for fruitful discussions and constructive suggestions. We also thank Prof. Kaiming He for inspiring discussions on earlier iterations of this project. This work is in part supported by MIT-IBM Watson AI Lab; ONR MURI grant #033697-00007; the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/). Shivam Duggal is further supported by Amazon AI Research Innovation Fellowship; Xingjian Bai is supported by MongoDB PhD fellowship.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation, 2026.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[5] Hila Chefer, Patrick Esser, Dominik Lorenz, Dustin Podell, Vikash Raja, Vinh Tong, Antonio Torralba, and Robin Rombach. Self-supervised flow matching for scalable multi-modal synthesis. *arXiv preprint arXiv:2603.06507*, 2026.

[6] Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[7] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.

[8] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[9] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

[10] Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

[11] Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. *arXiv preprint arXiv:2511.18822*, 2025.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] Shivam Duggal, Phillip Isola, Antonio Torralba, and William T. Freeman. Adaptive length image tokenization via recurrent allocation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[14] 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, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12606–12633. PMLR, 2024.

[15] Patrick Esser, Robin Rombach, and Björn 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.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[17] Philippe Hansen-Estruch, David Yan, Ching-Yao Chung, Orr Zohar, Jialiang Wang, Sriram Vishwanath, Peter Vajda, and Xinlei Chen. Learnings from scaling visual tokenizers for reconstruction and generation. *arXiv preprint arXiv:2501.09755*, 2025.

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[19] Jonathan Heek, Emiel Hoogeboom, Thomas Mensink, and Tim Salimans. Unified latents (ul): How to train your latents, 2026.

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

[21] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.

[22] Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. *arXiv preprint arXiv:2410.19324*, 2024.

[23] Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[24] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[25] Chaitanya K. Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary W. Ulissi. All-atom diffusion transformers: Unified generative modelling of molecules and materials. *arXiv preprint arXiv:2503.03965*, 2025.

[26] 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.

[27] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[28] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 2019.

[29] Mikhail V Koroteev. Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*, 2021.

[30] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025.

[31] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[32] Tianhong Li, He Li, and Mingyang Deng. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37, 2024.

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

[34] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2023.

[35] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024. ECCV 2024.

[36] Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Pixel diffusion beats latent diffusion with perceptual loss, 2026.

[37] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L. Chevrier, Kristin A. Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[38] 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. *arXiv preprint arXiv:2304.07193*, 2023.

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

[40] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

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

[42] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, 2014.

[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] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[45] Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[46] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[47] Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. *arXiv preprint arXiv:2507.23268*, 2025.

[48] Minkai Xu, Alexander S. Powers, Ron O. Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

[49] Jiawei Yang, Tianhong Li, Lijie Fan, Yonglong Tian, and Yue Wang. Latent denoising makes good tokenizers. In *The Fourteenth International Conference on Learning Representations*, 2026.

[50] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models, 2025.

[51] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

[52] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. *arXiv preprint arXiv:2511.20645*, 2025.

[53] Shuangfei Zhai et al. Normalizing flows are capable generative models. *arXiv preprint arXiv:2412.06329*, 2024.

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

Appendix

In this appendix, we first summarize the key points and contributions of UNITE in Sec. A, and discuss broader implications, future work, and limitations in Sec. B. We then cover relevant concurrent works in Sec. C, followed by additional reconstruction and generation results in Sec. D. Sec. E presents additional end-to-end training alternatives, representation alignment analysis, and compression analysis. We provide training-efficiency analysis in Sec. F, evaluation details in Sec. H, and additional samples from UNITE-XL in Fig. 9 and uncurated samples in Fig. 10. Sec. G presents additional molecule-generation results and ImageNet ablations. Finally, architectural and training details are provided in Tabs. 13 and 14.

A Summary of Key Contributions

We summarize the main points and contributions of UNITE.

- **Joint tokenization and latent diffusion.** UNITE jointly trains tokenization and latent denoising at ImageNet scale, achieving near-state-of-the-art generation fidelity among single-stage methods and strong performance against two-stage methods, without using DINOv2 encoders trained on substantially more data.
- **Weight-sharing between tokenization and generation.** UNITE studies a novel weight-sharing perspective, using the same Generative Encoder for both image-to-latent tokenization and latent denoising.
- **Simplified training without adversarial losses.** Unless otherwise stated, UNITE does not use GAN-based adversarial training, unlike many VAE-, VQGAN-, and RAE-style tokenizers, while still achieving strong generation FID.
- **Controlled end-to-end ablations.** We compare weight sharing against separate encoder–denoiser weights and full denoising-gradient backpropagation through the tokenization pathway.
- **Analysis beyond headline metrics.** Beyond reconstruction and generation metrics, we analyze training dynamics, tokenization–generation representation alignment, and proxy description-length measures to better understand why weight sharing works.
- **Applications to domains without DINO-like encoders.** UNITE extends beyond images to molecular generation from scratch, suggesting a path toward generative tokenization where large pretrained encoders are unavailable.

B Broader Implications, Future Works & Limitations

The core contribution of UNITE is to align tokenization and generation by training both over a shared latent space. The two objectives we consider are reconstruction and denoising. While reconstruction is a natural objective for learning compressed representations that preserve input information, *exploring alternative objectives for tokenization beyond reconstruction* is an interesting direction for research—for example, jointly training the Generative Encoder with DINO- or JEPA-style objectives. This is especially appealing for robotics, where generative modeling can provide a useful world model of the environment. However, naively training such a world model on standard VAE latents may not yield *actionable* latents that matter most for decision-making.

Another point worth discussing is the *vision-language modeling capability of the Generative Encoder*. The Generative Encoder idea is loosely reminiscent of the classical wake-sleep algorithm, whose broader goal was to bridge discriminative and generative modeling. While UNITE achieves strong reconstruction and generation fidelity, the linear probing accuracy of the Generative Encoder remains comparable to that of other generative tokenizers, such as VAEs and VQGANs, at around 30%. We believe that linear probing (LP) alone may not be fully predictive of the discriminative strengths of highly compressed latent representations. In particular, stronger compression may require greater downstream decoding capacity before the representation becomes predictive for a given task. For this reason, evaluating the tokenizer in a VLM setting may provide a more informative picture of its discriminative capabilities than LP alone. In fact, modern unified vision-language-action models use VAE/VQGAN as visual tokenizers (despite low LP).

Table 6: **ImageNet-256 reconstruction and generation comparison.** We compare reconstruction quality using rFID and single-image metrics, and generation quality using gFID. Empty entries indicate metrics not reported here. UNITE results from max-trained epochs.

Approach	Latent Dim	Pretrained Encoder	rFID↓	PSNR↑	SSIM↑	gFID↓
External / pretrained-tokenizer baselines						
VAVAE-XL	32	DINOv2	**0.28**	26.30	0.762	1.35
RAE-DINO-XL	768	DINOv2	0.58	18.41	0.453	**1.13**
RAE-SigLIP-XL (XL decoder)	768	SigLIP	0.53	19.10	0.496	–
VAE baseline						
SD-VAE (w/ DiT-XL)	–	×	0.62	24.98	0.708	–
UNITE shared encoder–denoiser						
UNITE-B	32	×	1.12	26.44	0.763	**2.12**
+ GAN decoder ft	32	×	0.51	26.44	0.760	2.12
UNITE-L	32	×	0.94	27.36	0.792	1.75
UNITE-B separate encoder–denoiser						
UNITE-B-Sep	32	×	0.93	28.03	0.831	**2.93**
UNITE-B-Sep	16	×	1.34	25.82	0.753	2.40

Furthermore, the results in Fig. 8 suggest that weight sharing and end-to-end joint training of tokenization and generation may support further *progress toward faster generative models*, potentially enabling high-quality one- to few-step generation. This also raises another interesting related question: can the process of mapping images to latents itself benefit from multiple iterative refinement loops? Prior work, such as ALIT [13], has explored this direction and reported improvements in linear probing and token-level object binding with additional iterations. Overall, UNITE opens up several interesting future directions on both the discriminative and generative fronts.

Limitations: While UNITE demonstrates the benefits of joint tokenization and generation in controlled single-stage settings, our study does not exhaust the broader space of joint discriminative and generative modeling. In particular, we do not evaluate large-scale VLM training, higher resolution or text-to-image generation, or one-/few-step diffusion distillation, all of which could further test whether the shared Generative Encoder yields benefits beyond the settings studied here. These experiments are substantially more compute-intensive. Accordingly, this work focuses on introducing a novel shared-weight framework for unified tokenization & generation, and evaluates it through dense experiments and ablations on Imagenet 256×256 and molecule datasets. We leave larger-scale evaluations as important directions for future.

C Concurrent Related Work

Several concurrent papers have explored closely related directions toward unifying tokenization and latent generative modeling. The closest to our setting is Google's Unified Latents [19], which studies end-to-end training of a tokenizer together with a latent generator and is closely aligned with our separate-weights ablation (i.e., an encoder and denoiser trained jointly without parameter sharing). In contrast to our single-stage results, their strongest numbers rely on an additional second-stage diffusion fine-tuning step (see Appendix B of [19]). Latent Forcing [2] extends pixel-space diffusion (JiT) by denoising pretrained DINO latents alongside image patches through a shared bottleneck, but does not learn the latent space from scratch. We still compare against them in Tab. 1. Another concurrent effort [5] adds an auxiliary self-supervised objective alongside the diffusion/flow objective, but similarly operates in a latent space defined by a pretrained encoder rather than jointly learning the tokenizer and generator end-to-end. In contrast to these works, our primary emphasis is on understanding the capabilities of a single-stage, end-to-end trained latent diffusion model. To this end, we study a perspective where encoding & denoising are performed by the same network parameters.

D Additional Reconstruction Fidelity Results

In Table 7, both models at number of latent features or channels, $d = 16$, use $16 \times 16 \times 16 = 4096$ latent scalars, the same count as SD-VAE f8d4. $d = 32$ means 8192 latent scalars. At 16 latent feature dimension, equivalent to 4096 latent scalars, sharing weights improves gFID from 2.58 to

Table 7: Generation FID on ImageNet-256 for Base models with separate or shared (UNITE) weights across denoising update counts. A dash marks an unreported result.

			gFID ↓ by denoising updates per reconstruction		
Model	Channels	Latent scalars	3	7	14
Separate-B	32	8192	3.46	3.16	2.93
UNITE-B	32	8192	2.65	2.15	2.12
Separate-B	16	4096	2.73	2.58	2.40
UNITE-B	16	4096	2.36	2.24	—

2.24 (using 7 denoising updates per reconstruction). For reference, DiT-XL/2, a larger denoiser trained on SD-VAE latents, achieves a gFID of 2.27 with the same number of latent scalars (4096).

Table 3 in the main paper summarizes our reconstruction fidelity results. Tab. 6 provides more elaborate analysis on reconstruction performance using single-image metrics and tradeoff with generation. We also provide additional details on the adversarial fine-tuning procedure that reduces rFID from 1.01 to 0.51, without changing gFID.

Detailed reconstruction comparison. Tab. 6 compares UNITE with prior tokenizers and separate encoder–denoiser ablations using PSNR, SSIM, rFID, and gFID. Compared to RAE-based approaches, UNITE achieves substantially better single-image reconstruction metrics (PSNR and SSIM), despite not using pretrained DINOv2/SigLIP encoders. With lightweight GAN-based decoder fine-tuning, UNITE further improves rFID, even outperforming RAE-DINO. Compared to separate encoder–denoiser ablations, the shared-weight model achieves stronger generation fidelity, even at the same or lower latent dimensionality, indicating that unifying tokenization and generation improves the generation–reconstruction trade-off. While separate weights can marginally improve pixel-level reconstruction metrics, they lead to much worse gFID, suggesting that weight sharing encourages latents that are more suitable for downstream generation. Overall, these results suggest that unification improves not only generation, but also generation-aware tokenization: reconstruction quality remains competitive while the learned latents become easier to model generatively.

GAN Decoder Fine-Tuning. After UNITE joint training converges, we optionally apply a lightweight adversarial fine-tuning stage that targets *only* the decoder. Concretely, we freeze the Generative Encoder entirely and train the decoder with an additional GAN loss for 16 epochs. The discriminator is initialized from our Generative Encoder, which already encodes rich semantic features from joint training; this eliminates the need for an external pretrained network (e.g., DINOv2) as the discriminator backbone. Due to limited compute budget, we did not explore fine-tuning both the encoder and decoder jointly with adversarial training.

E Analyzing UNITE's Generative Encoder

This section continues the analysis from Sec. 4. We first study an additional alternative for end-to-end training of tokenization and generation: removing the stop-gradient through clean latents, allowing denoising gradients to backpropagate into the tokenization pathway. We then examine how this gradient flow affects tokenization–generation representation alignment. Finally, we compare the weight-shared model against the separate encoder–denoiser ablation through an additional compression lens.

Backpropagating Denoising Gradients through the Encoder: Throughout this work, we stop denoising gradients from flowing through the clean latent into the tokenization pathway. Concretely, after the tokenization pass produces $z_0 = \mathrm{GE}_\theta(x)$, we apply $\mathrm{sg}(\cdot)$ before constructing the noised latent z_t used in the denoising pass. As a result, the flow-matching objective updates GE_θ only through the second (denoising) forward pass, rather than also directly shaping tokenization through gradients flowing into z_0.

Importantly, this does *not* decouple tokenization and generation: in the weight-shared Generative Encoder, reconstruction and denoising still act on the same set of network parameters, so both objectives jointly shape the learned representation. The stop-gradient only removes the more direct route in which denoising gradients also flow through the clean latent itself. In the separate encoder–denoiser setting, removing this stop-gradient yields a two-network end-to-end regime closely analogous to

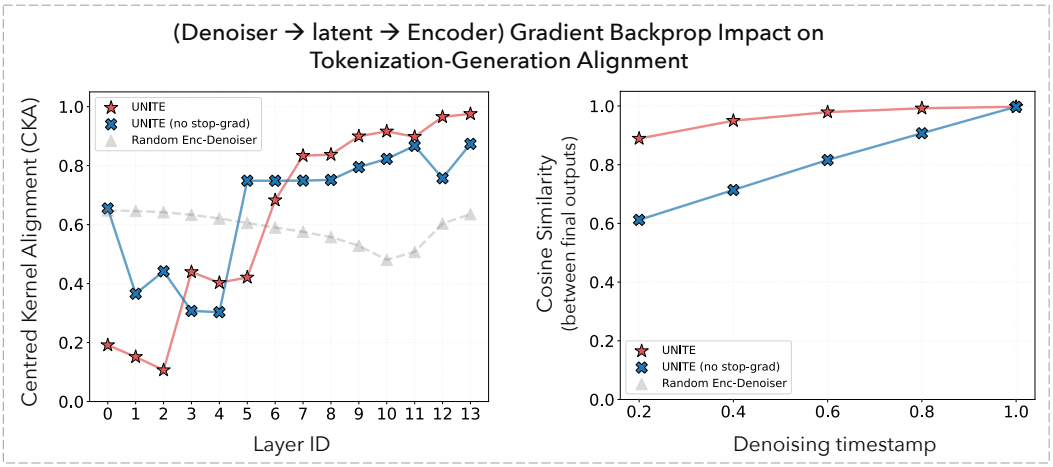

Figure 7: **Representation alignment between tokenization and generation pathways.** We measure alignment between tokenization and denoising activations using CKA and cosine similarity. Given an input image, we first record intermediate activations along the tokenization pathway, then corrupt the encoded latent and record the corresponding denoising-pathway activations. **Left:** removing the stop-gradient and backpropagating denoising gradients through the latent *weakens late-layer alignment*, even though the denoising objective still matches the final latent target. **Right:** cosine similarity on the final latents decreases at lower denoising timesteps in the no-stop-gradient setting, suggesting that direct gradient backpropagation from denoising into tokenization leads to a less cleanly shared representation (see Fig. 8 for visual interpretation).

concurrent work on Unified Latents (UL) [19], which jointly trains separate encoder and denoiser modules without parameter sharing. We therefore study what happens when denoising gradients are allowed to backpropagate through the clean latent (termed the no-stop-grad setting in the following paragraphs), both in our weight-shared GE setting and in the separate encoder–denoiser ablation.

For separate encoder and denoiser weights at latent dimension 32, the best gFID and corresponding rFID improve from 2.93/1.10 with stop-gradient to 2.22/0.82 without stop-gradient, indicating that end-to-end joint training of tokenization and generation is promising. As noted in concurrent work [19] (Appendix B), the no-stop-gradient setting requires tuning the denoising-to-reconstruction loss ratio. In contrast, with shared weights, UNITE-B achieved its best gFID of 2.12 with a corresponding rFID of 1.12 using stop-gradient. One possible hypothesis is that, under weight sharing, the two objectives already interact through a common parameter set, so allowing denoising gradients to additionally flow through the clean latent introduces extra (asymmetric) gradient interference. In this sense, weight sharing itself acts as a natural coupling mechanism between the two tasks: simply increasing the number of flow iterations improves performance, without much loss-weight tuning.

Tokenization-Generation Representation Alignment Analysis: When analyzing the no-stop-gradient alternative (Fig. 6, middle and right), we observe that, for both the weight-shared and separate encoder–denoiser settings, CKA and cosine-similarity alignment between the outputs of the tokenization and denoising pathways is reduced, relative to the stop-gradient variants, despite the denoising objective encouraging agreement at the final latent target. Further, Fig. 8 shows that the no-stop-gradient models produce noticeably noisier intermediate denoised reconstructions. Taken together, these observations suggest that stopping denoising gradients through clean latent *may* help preserve a more cleanly shared representation between tok. & generation.

Entropy / Compression Analysis. We next study encoder–denoiser relationship through the lens of *compressibility*, motivated by a Minimum Description Length (MDL) perspective: if tokenization and denoising implement closely related computations, then a unified latent-generation program should admit a shorter description than two independently parameterized modules. Concretely, we estimate an empirical description-length proxy for model weights using per-tensor histogram entropy, in Tab. 8.

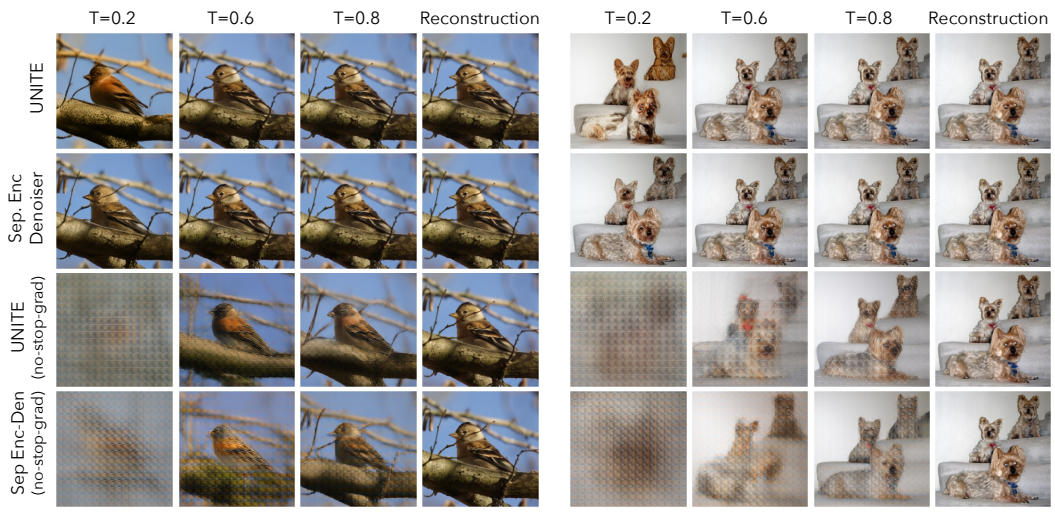

Figure 8: **Analyzing the denoising trajectory.** Given an input image, we first encode it into latents, corrupt the latent with noise, and then decode the denoised prediction at different noise levels (first three columns). The final column shows direct decoding of the clean latent. Although all four models achieve competitive aggregate rFID/gFID, the stop-gradient variants (first two rows)–UNITE and the separate encoder-denoiser ablation–exhibit markedly cleaner intermediate denoising trajectories, with higher PSNR to the input image across all noise levels. This result is consistent with the representation-alignment in Fig. 6, which shows drop in alignment at final layers.

We begin with the separate encoder–denoiser setting. Compared to random weights, the total entropy of the encoder drops from 179.2MB at random initialization to 121.9MB after training, with both normalization parameters (60.0 $\rightarrow$ 30.7MB) and functional attention/MLP parameters (119.2 $\rightarrow$ 91.2MB) becoming substantially more structured as a result of training.

In the weight-shared Generative Encoder setting, the entropy of the functional attention/MLP parameters remains nearly *unchanged* relative to the separate encoder (91.2MB $\rightarrow$ 90.8MB), while the main increase is concentrated in normalization-related parameters, whose entropy rises modestly from 30.7MB to 42.0MB and closely matches that of the separate denoiser (42.0MB). Thus, *unifying tokenization and denoising does not require a more complex functional backbone*; instead, the shared model reuses essentially the same attention/MLP computation and expresses the residual mode-specific adaptation primarily through normalization and scale parameters. This provides a complementary MDL-style interpretation of why sharing works in our setting: parameter tying may yield a shorter description of the joint latent-generation program, not by substantially altering the main reusable computation, but by preserving a common functional backbone while allocating only a small additional entropy budget to normalization. This interpretation aligns with our CKA analysis, since pathway alignment remains high while CKA is largely insensitive to norm/scale changes, suggesting that tokenization & denoising differ more in feature calibration than in core representational geometry.

F Training Efficiency

We report total training FLOPs measured with gradient checkpointing enabled in Tab. 9. For UNITE-B, each training sample costs approximately 3.5 TFLOPs (forward + backward), including one tokenization pass through GE_θ (512 tokens), fourteen denoising mini-batch passes (256 tokens each), one decoder pass, and one forward pass through a frozen VGG network for the LPIPS loss. Over 120 ImageNet epochs, UNITE-B requires approximately 6.7×10^{20} FLOPs and reaches an FID of 2.18. Reducing the number of denoising iterations per reconstruction iteration can further lower training cost, at the expense of a modest increase in gFID. This is approx. $\mathbf{15}\times$ cheaper than the end-to-end cost of methods that rely on pretrained DINOv2 encoders. RAE [54] and LF-DiT [2] both depend on DINOv2 features, whose ViT-g/14 pretraining and distillation together require approx.

Table 8: **Entropy / compression analysis of encoder–denoiser weights.** We report per-tensor histogram entropy as an empirical description-length proxy. Training substantially reduces entropy relative to random initialization. In the shared Generative Encoder, the functional attention/MLP entropy remains nearly unchanged relative to the separate encoder and denoiser (comparing row 2/3 with row 5), while the main increase appears in normalization parameters. This suggests that the shared model reuses a common functional backbone and differentiates the two tasks mainly through scale parameters, consistent with similar CKA scores since CKA is invariant to isotropic scaling.

Model / Weights	Norm. Params	Attn./MLP Params	Total
Random encoder	60.0	119.2	179.2
Separate encoder	30.7	91.2	121.9
Separate denoiser	42.0	86.5	128.4
Total encoder + denoiser	72.7	177.7	250.3
Shared GE	42.0	90.8	132.8

Values are reported in MB using per-tensor histogram entropy as a description-length proxy.

27,000 A100-GPU-hours, corresponding to $\sim 1.0 \times 10^{22}$ model FLOPs [2]. This constitutes a fixed upfront cost inherited by any downstream method built on top of these features. In contrast, UNITE eliminates this overhead entirely by training from scratch.

Table 9: **Training efficiency on ImageNet-256.** We decompose training cost into tokenizer/encoder, decoder, and generator components. UNITE trains tokenization and generation jointly from scratch, avoiding pretrained SSL encoder overhead.

Method	Params	Tokenizer / Encoder	Explicit Decoder / GAN	Generator	Total Cost	FID↓
DiT-XL/2	724M	$C_{\text{SD-VAE}}$	None	6.4×10^{20}	6.4×10^{20}	2.27
JiT-G/16	2B	None	None	$\sim 8.8\times10^{20}$	$\sim 8.8\times10^{20}$	1.82
UNITE-B	217M	Joint	None	Joint	6.7×10^{20}	2.18
UNITE-L	589M	Joint	None	Joint	$\sim 2.5\times10^{21}$	1.75
RAE-DiT^{DH}	839M+415M	$\sim 1.0\times10^{22}$	C_{dec}	$\sim 4\times10^{20}$	$\sim 1.0\times10^{22} + C_{\text{dec}} + 4\times10^{20}$	1.13

$(C_{\text{SD-VAE}}), (C_{\text{dec}})$ denotes SD-VAE and RAE decoder training costs.

Compared with standard two-stage latent diffusion models, our total compute is comparable: UNITE-B surpasses DiT-XL/2 [39] (FID 2.27) at nearly matched total FLOPs (6.7×10^{20} vs. 6.4×10^{20}), while using $3\times$ fewer parameters (217M vs. 724M). In addition, UNITE jointly learns a tokenizer whose latent space is shaped by both reconstruction and generation objectives.

Among single-stage methods, UNITE-B (6.7×10^{20} FLOPs, 217M parameters) achieves an FID of 2.18 (160 epochs) at total compute comparable to JiT-G/16 [31] ($\sim 8.8 \times 10^{20}$ FLOPs, 2B parameters, FID 1.82), while using approximately $10\times$ fewer parameters. UNITE-L achieves a lower gFID of 1.75 with slightly higher flops and fewer parameters. Moreover, UNITE produces a reusable latent tokenizer alongside the generator, a capability that pixel-space methods such as JiT do not offer.

G Additional Results

G.1 QM9 Molecular Generation

The QM9 dataset [42] contains approximately 130K stable small organic molecules with up to 9 heavy atoms from the set {C, N, O, F}. Following [25], we represent molecules with explicit hydrogen atoms and use 3D Cartesian coordinates for both training and generation. Each molecule is preprocessed to ensure correct bond valencies and stable conformations, with coordinates normalized to have zero center of mass.

Our training configuration employs the UNITE-S architecture with a DiT-S backbone containing approximately 33M parameters. The model is trained end-to-end for 8000 epochs with a batch size of 512 using the AdamW optimizer with a learning rate of 1×10^{-4}. This single-stage approach contrasts with ADiT's two-stage training, which requires 5000 epochs for the tokenizer followed by

[2] 27,316 A100-GPU-hours $\times$ 312 TFLOP/s (A100 BF16 peak) $\times$ 0.4 model-FLOP utilization $\approx 1.0 \times 10^{22}$ [38].

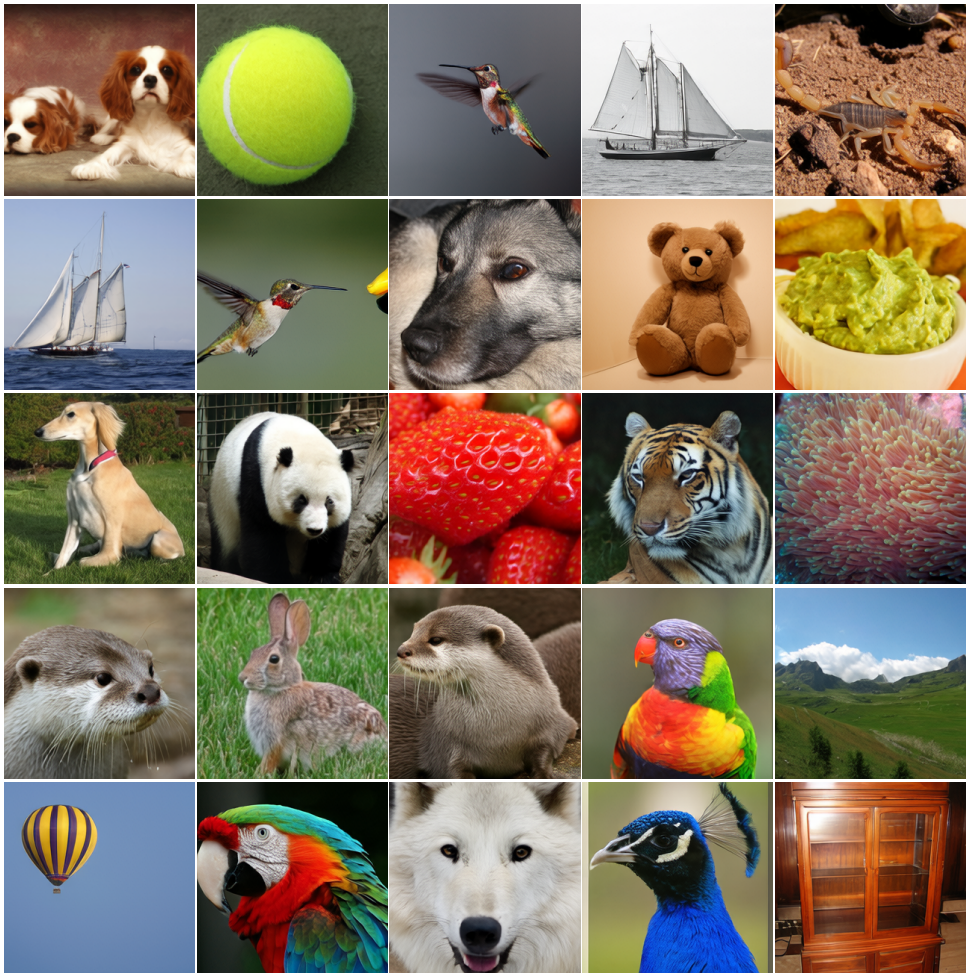

Figure 9: **Selected samples from UNITE-XL.** Generated using 50 steps with CFG. This model achieves FID 1.75. More class-conditioned results in Fig. 10.

another 5000 epochs for the diffusion model, totaling 10000 epochs of training across two separate optimization phases.

For evaluation, we compute four primary metrics on 10000 generated samples. The match rate measures the percentage of reconstructed molecules that exactly match the input structure after discretization. The RMSD (Root Mean Square Deviation) in Angstroms quantifies reconstruction error in atomic positions. Validity percentage indicates the proportion of generated molecules satisfying chemical constraints including proper valencies, reasonable bond lengths, and absence of steric clashes. Uniqueness measures the percentage of distinct molecules among valid generations, computed using canonical SMILES representations to identify duplicates.

The UNITE-S architecture employs a weight-shared encoder-denoiser operating in a 16-dimensional latent space, significantly compressed from the original 3D coordinate space. This compression factor of approximately 20:1 (from 29 atoms × 3 coordinates to 16 dimensions) requires the model to learn highly efficient representations while maintaining recon. fidelity.

G.2 MP20 Crystal Generation

The MP20 dataset from the Materials Project [24] contains 45,231 inorganic crystal structures with up to 20 atoms per unit cell. We train UNITE-S for 10000 epochs with batch size 512, following the same single-stage approach as QM9. Evaluation follows [25], computing structural validity (pairwise distances > 0.5 Å, unit cell volume > 0.1 Å^3), compositional validity (charge neutrality and electronegativity balance), and match rate using pymatgen's [37] Structure Matcher.



class 12: house finch, linnet, Carpodacus mexicanus



class 39: common iguana, iguana, Iguana iguana



class 99: goose



class 108: sea anemone, anemone



class 144: pelican



class 207: golden retriever



class 309: bee



class 470: candle, taper, wax light



class 725: pitcher, ewer



class 930: French loaf

Figure 10: **Uncurated, class-conditional samples on ImageNet 256×256 using UNITE-XL.** We show images using CFG $\omega = 4.0$. Each grid contains 21 randomly sampled images, demonstrating consistent quality across diverse categories including animals, objects, and scenes.

Tab. 10 shows that UNITE-S achieves 87.9% overall validity on MP20, approaching ADiT's 90.1% despite using single-stage training. Our structural validity of 99.0% nearly matches ADiT's 99.6%, demonstrating effective learning of crystal geometry constraints. The match rate of 75.7% is reasonable considering ADiT's dedicated tokenizer achieves 84.50% after separate optimization. These results validate that our unified approach generalizes well from molecules to crystals—the same architecture that achieves a 99.37% match rate on QM9 also performs competitively on the more complex MP20 dataset without modification.

Table 10: **MP20 crystal generation results.** Evaluation on 10K generated samples.

Method	Size	Training	Struct.	Comp.	Overall	Match
ADiT Tokenizer	-	-	-	-	-	84.50
ADiT MP20-only	DiT-B	Two-stage	99.6	90.5	90.1	-
ADiT Joint	DiT-B	Two-stage	**99.7**	**92.1**	**91.9**	-
UNITE-S (Ours)	DiT-S	Single-stage	99.0	89.9	87.9	75.7

G.3 Ablation Studies on ImageNet 256×256

In addition to the weight-sharing and stop-gradient ablations shown in the main paper, we provide a few additional ablations that further improve UNITE's reconstruction and generation fidelity.

Reconstruction Noise Level. Table 12 (top half) investigates the impact of noise augmentation during reconstruction training, where Gaussian noise is injected into latent representations prior to decoding. Consistent with recent findings in RAE [54] and TARflow [53], this acts as a useful regularizer by preventing the decoder from overfitting to noise-free latent codes, thereby improving generative capability. Notably, due to our model's learnable affine normalization, the system can autonomously calibrate its internal signal-to-noise ratio (SNR) to accommodate varying noise scales. As a result, the model exhibits strong robustness to the exact noise level, maintaining a nearly constant FID of around 2.7 across a range of noise levels.

Noise Schedule Shifting. Following RAE [54], we find that noise-schedule shifting is important for our 32-dimensional latent space. Table 12 (bottom half) shows that, without shifting, FID degrades to 3.14. Our best setting, a shift of 0.5, adapts the noise schedule to the compressed latent dimensionality and improves both FID and IS. This shift is equivalent to using an anchor dimension of $d_{\text{anchor}} = 4096$, matching the uncompressed token dimension before projection into the 32-dimensional latent space. Overall, this adaptation is important when working with highly compressed latents.

H Evaluation Protocol

For reproducibility, we detail the full evaluation protocol used for all ImageNet-256 generation results. See also Fig. 10 for additional uncurated samples generated by UNITE-XL.

FID Computation. We compute Fréchet Inception Distance (FID) using the `torch-fidelity` library with InceptionV3 features. Reference statistics are computed on the full ImageNet-1K training set (1281167 images). All reported FID scores use 50K generated samples.

Sampling Protocol. We adopt class-balanced sampling: exactly 50 images are generated per class for the 1K ImageNet classes, totaling 50K images. This follows the protocol used by VAR [44], MAR [32], and RAE [54], among others. As shown in RAE paper Table 14, class-balanced sampling yields about 0.1 lower FID than the uniform random class sampling used in some prior work (e.g., DiT [39], SiT [35]). We note this systematic difference when comparing absolute FID values across methods.

ODE Solver and Inference Details. At inference time, we solve the probability flow ODE over the interval $[0.1, 1.0]$ with classifier-free guidance. We sweep the CFG scale ω from 1.0 to 4.0 in increments of 0.2 and report best FID for each model.

For all reported FID numbers in the main paper (Tab. 1), we use the **adaptive fifth-order Dormand–Prince solver** (`dopri5`, from `torchdiffeq` [8]), following the default configuration of the SiT codebase [35]. For our model, we observe that `dopri5` uses $\sim$108 NFEs on average per sample (estimated from wall-clock timing), compared to exactly 100 NFEs for the fixed-step Heun solver with 50 steps.

Tab. 11 compares FID under different evaluation protocols for the same UNITE-B checkpoint. Switching to a fixed-step second-order Heun solver with 50 steps (100 NFEs) yields FID within $\sim$0.05 of `dopri5`, consistent with prior observations that flow-matching models produce near-linear

Table 11: **Effect of evaluation protocol on reported FID.** All rows use the same UNITE-B checkpoint (120 epochs, 3 flow steps per reconstruction). "Balanced" denotes 50 images per class; "Random" denotes uniformly sampled class labels. NFE = number of function evaluations.

ODE Solver	Class Sampling	NFE	FID ↓	IS ↑
Heun (50 steps)	Balanced	100	2.789	287.6
dopri5 (adaptive)	Balanced	∼108	2.735	268.1
dopri5 (adaptive)	Random	∼108	2.885	274.7

Table 12: **Ablation study on ImageNet-256 generation.** We systematically evaluate key design choices across architecture, normalization, training dynamics, and augmentation strategies. All ablations are done using the base backbone and trained for 120 epochs.

Steps	FID↓	r-FID↓	IS↑
Reconstruction noise σ (c)			
1.0 (full noise)	2.87	1.09	281.1
0.4	2.80	1.21	267.1
0.3	**2.71**	**1.01**	282.2
0.2	2.87	1.42	**292.1**
0.0 (no augmentation)	6.58	1.60	275.7
***Noise schedule or timestamp shifting α* (e)**			
0.0 (No Shift)	3.14	1.41	267.1
0.5	**2.71**	**1.01**	**282.2**
0.75	2.93	1.26	278.0

trajectories that are well approximated by low-order fixed-step integrators [33, 34]. This small gap is also consistent with the SiT authors' report that the FID difference between `dopri5` and fixed-step solvers is <0.1.[3]

I Architectural & Training Details

Tab. 13 and Tab. 14 provide additional architectural and training details. For more details, refer to the codebase.

Time and class conditioning details during training. During tokenization, we set t to the minimum-noise timestep and disable class conditioning, as in the no-CFG setting, so the block conditioning is given only by the corresponding timestep embedding. This conditioning is applied through standard adaptive normalization. During denoising, when class conditioning is enabled, we follow JiT by prepending 32 in-context class tokens to the latent-token sequence.

[3] See `https://github.com/willisma/SiT/issues/21`.

Table 13: **Detailed architecture configurations for UNITE models.** Throughout, we only use base decoder, with either base, large or XL encoder/denoiser.

Component	Encoder/Denoiser			Decoder	
	DiT-B	DiT-L	DiT-XL	ViT-B	ViT-L
Hidden Dimension	768	1024	1152	768	1024
Layers	12	24	28	12	24
Attention Heads	12	16	16	12	16
MLP Ratio	4	4	4	4	4
Patch Size	16	16	16	-	-
Latent Dimension	32	32	32	32	32
Latent Resolution	16×16	16×16	16×16	16×16	16×16
Parameters (M)	86.2	458.2	675.3	130.6	303.9

Table 14: **Training configuration for ImageNet-256 experiments**. All models are trained with mixed precision BF16 and gradient clipping at 3.0.

Hyperparameter	**Value**
Base Learning Rate	1×10^{-4}
Global Batch Size	1024
Optimizer	Muon
AdamW Betas	(0.9, 0.999)
Weight Decay	0
Reconstruction Noise (σ)	0.3
Default Denoising/Flow steps per Recon	14
CFG Scale (ω)	Sweep $[1.0,\ 4.0]$, step 0.2
Warmup Epochs	20
Total Epochs	240
LR Schedule	Cosine
Min LR	1×10^{-6}
Gradient Clip	3.0
EMA Decay	0.9978
ODE Solver (Inference)	dopri5 (adaptive, $\sim$108 NFE)
Noise Schedule Shift (α)	0.5
Integration Interval	$[0.1,\ 1.0]$

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 summarize the paper's contributions—a single-stage end-to-end training framework that unifies tokenization and latent diffusion via a shared Generative Encoder—which are supported by the methods in Section 3 and the empirical results in Section 5. The scope of these claims is bounded to the experimental settings reported, with limitations stated in the appendix.

   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: The appendix includes an explicit Limitations subsection that discusses the settings we do not evaluate (large-scale VLM training, higher-resolution and text-to-image generation, few-step distillation) and the compute scope of our study.

   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 is empirical and methodological; the equations describe the model and training objective rather than stating formal theorems requiring proofs.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The appendix documents the architecture, optimizer and training schedule, training efficiency in FLOPs, and the full evaluation protocol used for ImageNet and molecule experiments.

   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 is available at `https://github.com/ShivamDuggal4/UNITE-tokenization-generation`. The repository documents ImageNet training and evaluation; it does not provide the full molecule pipeline. The datasets and pretrained components we build on (ImageNet, QM9, MP20, VGG for LPIPS) are already 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: Section 5 reports the experimental setup and main results, with full architectural and training configurations, evaluation protocol, and ablation details provided in the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: Following standard practice in image generative modeling, we report FID and reconstruction metrics as single-point estimates over a fixed-size sample set (50K for ImageNet) without error bars, due to the cost of repeating full training runs.

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: The appendix reports total training FLOPs for our models alongside relevant baselines, together with the optimizer, batch size, and precision used in the main runs.

   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: The research conforms with the NeurIPS Code of Ethics. The work uses publicly available pretrained models and datasets, and the experiments do not involve human subjects.

   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: [N/A]

    Justification: This work is foundational research on tokenization and generation. It does not have any direct impact to any specific societal applications.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release new high-risk models or scraped datasets.

    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 existing assets (baselines, datasets used and pretrained models)—are cited in the paper, and we use them in accordance with their public 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: [Yes]

Justification: The project uses public datasets, so do not release any data assets. We only release pretrained models trained on Imagenet dataset.

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: This work does not involve crowdsourcing or 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: This work does not involve 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 are not an important, original, or non-standard component of the methods in this work.

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.