SINGLE-TRANSFORMER LATENT DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent generative models typically rely on three separate networks: an encoder that tokenizes pixels, a generator that models the latent distribution, and a decoder that maps latents back to pixels. This contrasts with multimodal transformers, where heterogeneous modalities are processed by a single backbone in one unified framework. Recent work narrows the gap by casting tokenization and generation as one latent-inference problem solved by a single generative encoder, but the decoder remains a separate network. We investigate whether this last component can be folded in too, and what it takes for one diffusion transformer to encode, denoise, and decode. We propose a simple formulation that casts all three modes as position-aligned sequence maps over one fully shared backbone; the modes differ only in lightweight input and output heads and in their conditioning, namely a mode token, the timestep, and the class label. On ImageNet-1K 256×256 at matched parameters, our model, UNIFIED, improves gFID over both a separate tokenizer–generator pipeline and the two-network baseline, with the largest gains at the two smaller scales (2.79 vs. 3.04 at $\sim$60M, 2.44 vs. 2.77 at $\sim$115M) and a smaller gain at $\sim$215M (2.16 vs. 2.23). The results show that a single transformer can encode, denoise, and decode, and that full unification is not merely feasible but can be beneficial: the standalone decoder is a less parameter-efficient use of capacity than the shared backbone, and folding it in can improve generation quality at matched parameters.

1 INTRODUCTION

Foundation models increasingly absorb heterogeneous input–output shapes and training objectives into a single backbone, with modality-specific heads doing the work that separate networks used to do (Zhou et al., 2025; Mizrahi et al., 2023; Lu et al., 2023). Latent diffusion has not followed this trend. Even for a single modality, the standard pipeline has three trainable networks—an encoder, a decoder, and a generator—commonly optimized in two stages: encoder and decoder are trained first under a reconstruction objective, the latent space is then frozen, and a generator is fit to it (Rombach et al., 2022; Peebles & Xie, 2023; Ma et al., 2024). The split is a design commitment rather than a necessity: it fixes the representation before the model that must sample from it is trained.

A growing body of work aims to avoid two-stage training, motivated by the observation that a latent space optimized purely for reconstruction need not be one that enables easy generation (Yao et al., 2025). Most of these methods train the tokenizer and generator jointly but keep their networks separate (Vahdat et al., 2021; Leng et al., 2025; Zhao et al., 2026; Zheng et al., 2026; Heek et al., 2026). UNITE (Duggal et al., 2026) goes further and folds the encoder into the generator itself: tokenization and generation are the same latent-inference problem and can be solved by one *generative encoder* trained with reconstruction and flow-matching losses, with no adversarial term and no pretrained encoder, but pixel decoding still happens in a network of its own. DSD (Wang & Zhang, 2025) merges all three components into a single ViT, relying on alignment to DINOv3 (Siméoni et al., 2025) and on task-specific heads, including a dedicated diffusion head. Whether one backbone can encode, denoise, and decode with every block shared and without adversarial or representation-alignment losses remains open.

Encoding and decoding read clean inputs, whereas generation reads a latent whose noise level varies from pass to pass. Once every backbone weight is shared, nothing inside the backbone is specific to a task, so how the three tasks are told apart stops being an implementation detail and becomes the central design decision. We therefore ask not only whether one network can encode, denoise, and

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

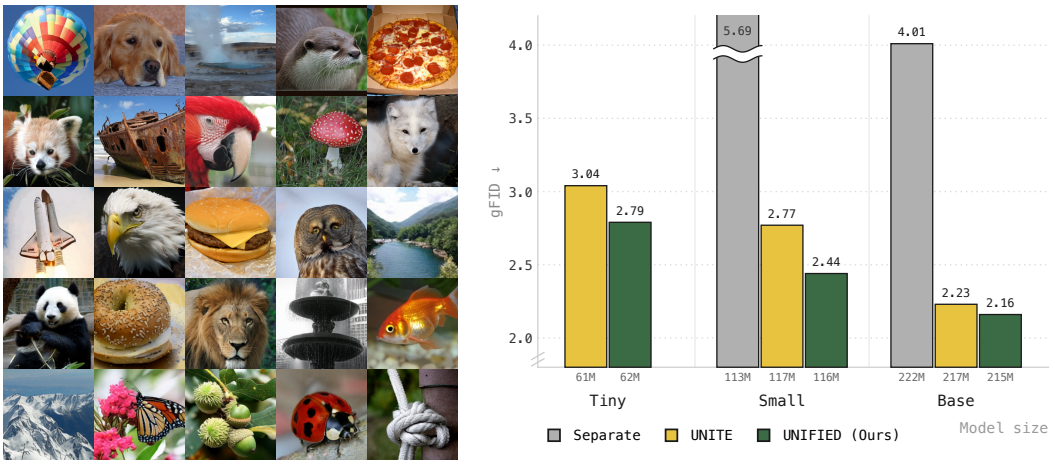


Figure 1: *(Left)* **Selected samples from UNIFIED** on ImageNet-1K 256×256 (CFG scale 4.0). *(Right)* **Full unification is not merely feasible but beneficial.** gFID (↓) at three sizes; bar labels give the total trainable parameters. UNIFIED improves on UNITE at every size, and Tiny UNIFIED nearly matches Small UNITE with half the parameters (2.79 at 62M vs. 2.77 at 117M).

decode, but (i) what it costs in parameters, (ii) what conditioning it needs to keep the three tasks distinct, and (iii) what it ends up computing in each mode.

We study these three questions through an architecture which we refer to as **UNIFIED** (Figure 2d). It is built on a single principle: every mode is a position-aligned sequence map over one backbone, i.e., a given sequence index refers to the same image location, irrespective of whether it holds a pixel patch, a latent, or a noisy latent. The mode is specified by a learned mode token and the timestep. Relative to UNITE (Duggal et al., 2026), UNIFIED changes only this layout and, with it, the encoding timestep, adds the mode tokens, and replaces UNITE's standalone decoder with a lightweight pixel output head on the shared backbone; the training recipe is otherwise unchanged. This makes the comparison attributable to these changes rather than to training details. As a control, we also build **UNITE++** (Figure 2c), which shares every weight and uses the same mode tokens but keeps UNITE's readout tokens, extra input tokens whose outputs are read as the result. Comparing the two, each at its matched encoding timestep (Section 3.3), isolates the effect of position alignment.

Three findings follow, one for each question. *(i) Parameters.* At matched parameter counts after 200 epochs on ImageNet-1K 256×256, UNIFIED improves gFID most where capacity is scarce: 2.79 vs. 3.04 at ∼60M, 2.44 vs. 2.77 at ∼115M, and 2.16 vs. 2.23 at ∼215M against UNITE, while a separate pipeline of three networks trained with the same recipe trails both at ∼115M and ∼215M (Figure 1). This pattern is what one should expect if the standalone decoder is a less parameter-efficient use of capacity than the shared backbone, and we test this premise directly: at a fixed ∼60M budget, shifting capacity from the decoder to the encoder of a two-network model improves gFID (3.04 vs. 3.76 at a 2:1 ratio), while shifting it the other way degrades it sharply (5.03 at 1:2). The gain requires position alignment: UNITE++, which shares every weight but keeps readout tokens, matches UNITE in gFID at ∼215M (2.24 vs. 2.23) and reconstructs worse (rFID 2.20 vs. 1.41; Tables 2 and 4). *(ii) Conditioning.* The timestep should match how much of the input is observed. Encoding and decoding both read clean inputs and use the clean endpoint; labeling the encoder's clean input as noise instead performs worse (2.20 vs. 2.16 gFID at ∼215M), and the learned mode token is what keeps decoding distinct from generation at the same timestep: without it, neither fully shared model trains. The two modes still compete at the shared endpoint, however: ending the sampling ODE at $t = 0.99$ improves gFID from 2.16 to 2.05. *(iii) Computation.* Layer-wise CKA (Appendix B) shows what position alignment changes. UNITE casts tokenization and generation as the same latent-inference problem (Duggal et al., 2026); in UNIFIED, this correspondence also holds at the level of computation. The encoding pass tracks the generation pass from the first block (CKA 0.48–0.80 in blocks 1–5), whereas the readout-based encoders of UNITE and UNITE++ align only in the last blocks.

Our contributions are:

- UNIFIED, in which encoding, generation, and decoding are position-aligned passes through the same DiT blocks, differing only in lightweight heads and conditioning, with no readout tokens, diffusion head, or representation alignment.
- A controlled study of how fully shared weights keep the three modes apart, showing that mode tokens are necessary, that each mode's timestep should match how much of its input is observed, and that decoding and generation compete at the shared clean endpoint.
- Evidence that the standalone decoder is a less parameter-efficient use of capacity than the shared backbone, and, from layer-wise CKA, that with position alignment the encoding pass tracks the generation pass from the first block, extending UNITE's view that tokenization and generation are one task.

2 Related Work

Autoencoders and latent diffusion. Latent generative models typically follow a two-stage recipe: an autoencoder is trained for reconstruction and frozen, and a separate generator is then fit to its latents. The tokenizer is usually a KL-regularized VAE (Kingma & Welling, 2014; Rezende et al., 2014) for latent diffusion (Rombach et al., 2022; Podell et al., 2024) or a vector-quantized autoencoder such as VQ-VAE and VQGAN (Van Den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021) for discrete generation, in both cases trained with the perceptual (Zhang et al., 2018) and adversarial losses popularized by VQGAN. Because the two stages are decoupled, progress has largely come one network at a time: generators have moved to transformer backbones (Peebles & Xie, 2023; Ma et al., 2024) and flow-based objectives (Lipman et al., 2023; Liu et al., 2023), while tokenizers have improved latent quantization (Mentzer et al., 2024; Yu et al., 2024a) and compactness (Yu et al., 2024b; Chen et al., 2025). This decoupling has a cost, however. The two stages optimize different objectives, and Hansen-Estruch et al. (2025) find that gains on one do not transfer to the other: reconstruction and generation quality need not improve together, and enlarging the decoder helps reconstruction while having mixed effects on generation. We revisit the recipe itself, a single-stage training of one backbone to serve as encoder, decoder, and generator.

End-to-end tokenization and generation. An alternative is to train the tokenizer and generator jointly (Yao et al., 2025; Leng et al., 2025; Duggal et al., 2026). Doing so raises two design questions: what prevents the latent from collapsing, and which networks, if any, share weights. Collapse is the central difficulty, since once the generation loss can shape the latent, the latent may become trivial rather than be modeled (Leng et al., 2025; Wang & Zhang, 2025), and existing methods differ mainly in their answer. Likelihood-based approaches keep a variational bound whose KL or entropy terms hold the latent in place and train a score-based (Vahdat et al., 2021), normalizing-flow (Zhao et al., 2026), or diffusion (Zheng et al., 2026; Heek et al., 2026) prior jointly with the autoencoder. Refinements fix the encoder variance (Zhao et al., 2026), shape the latent with the generative loss before strengthening reconstruction (Zheng et al., 2026), or tie the encoder noise to the prior to bound the latent bitrate (Heek et al., 2026). REPA-E (Leng et al., 2025) instead anchors the latent to an external representation, backpropagating a representation-alignment loss rather than the diffusion loss into the VAE. A third answer stops the gradient through the denoising target and predicts clean latents (Duggal et al., 2026; Wang & Zhang, 2025); it requires neither a variational bound nor an external encoder, and we adopt it. On the second question, two methods built on this recipe also share weights: UNITE (Duggal et al., 2026) merges the encoder and generator yet keeps a standalone decoder, and, in concurrent work, DSD (Wang & Zhang, 2025) merges all three into one ViT but attaches a lightweight diffusion head and reintroduces an external alignment loss. We share a single DiT backbone across all three roles with neither addition. Finally, a complementary route avoids the latent altogether and generates pixels directly (Li & He, 2026); we instead keep a learned latent that the shared backbone both produces and decodes.

One transformer, many tasks. Without task-specific weights or an external signal, a fully shared backbone must tell its tasks apart through its inputs alone, and multi-task transformers offer two ways to do so. The first specifies the task in the input format: a text prefix or prompt (Raffel et al., 2020; Chen et al., 2022; Lu et al., 2023; 2024), the choice of input and target modalities (Mizrahi et al., 2023), in-context example pairs (Wang et al., 2023), or a multimodal instruction (Xiao et al., 2025). Unified multimodal models (Chameleon Team, 2024; Zhou et al., 2025; Xie et al., 2025; Wu et al., 2025; Deng et al., 2025) push this furthest, combining understanding and generation in

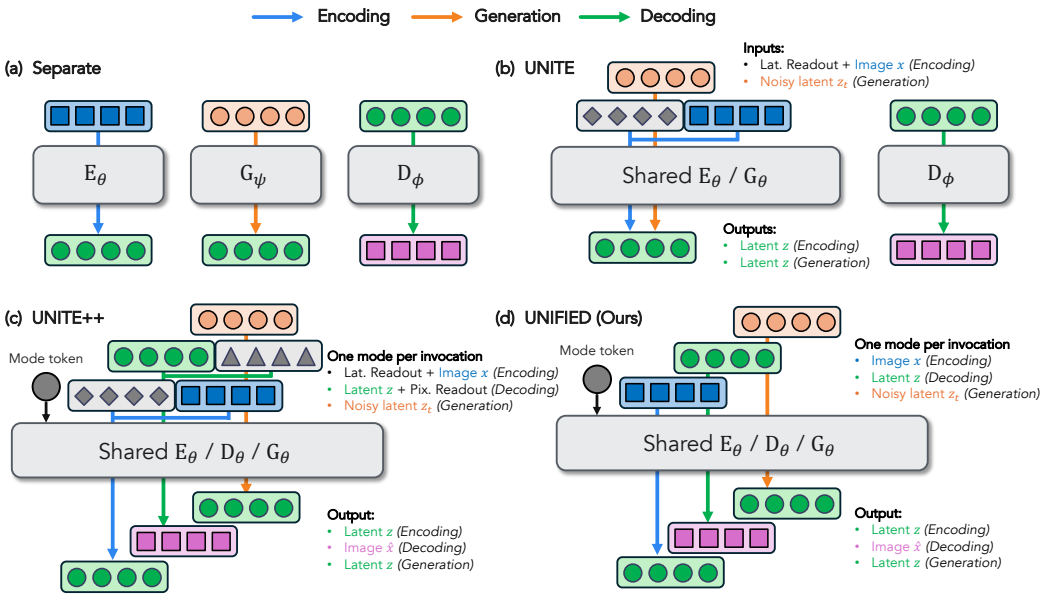


Figure 2: **From three networks to one shared transformer.** Arrows mark the encoding (blue), generation (orange), and decoding (green) passes; grey diamonds and triangles are latent and pixel readouts, and dark circles are learned mode tokens. **(a)** Separate: the standard latent-diffusion pipeline trains an encoder E_θ, a generator G_ψ, and a decoder D_ϕ. **(b)** UNITE shares one set of weights between the encoding and generation modes (E_θ/G_θ); its encoding pass reads image patches alongside $\mathcal{N}(0, I)$ latent readouts, whose outputs form the latent z, but a standalone decoder D_ϕ remains. **(c)** UNITE++ folds decoding into the same backbone: E_θ, G_θ, and D_θ run as three modes selected by a mode token, while keeping readout tokens—$\mathcal{N}(0, I)$ latent readouts for encoding and pixel readouts x_{readout} for decoding. **(d)** UNIFIED (ours) removes the readout tokens: every pass is a same-length, position-aligned sequence—image x, noised latent z_t, or latent z—after a mode token at position 0.

one backbone, yet still rely on a separately trained tokenizer. The second way specifies the task through the corruption level of each input. UniDiffuser (Bao et al., 2023) gives each modality its own timestep, so that a clean modality acts as a condition and a fully noised one is marginalized out; per-token noise levels let one network move between next-token prediction and full-sequence diffusion (Chen et al., 2024; Le et al., 2025); and corruption by masking (Li et al., 2023), or masking combined with noise (Hansen-Estruch et al., 2024), lets a single network serve both representation learning and generation. In diffusion transformers this signal reaches every block through AdaLN (Perez et al., 2018; Peebles & Xie, 2023). UNIFIED draws one signal from each family, a learned mode token as the minimal form of a task prefix and a timestep that reflects how much of the input is observed, and applies both to the tokenizer itself, so that encoding and decoding become tasks of the backbone rather than separate networks around it. A further choice is where the latent lives: UNITE and UNITE++ encode into appended query tokens whose outputs are read out as the result (Jaegle et al., 2022), which we call readout tokens, whereas UNIFIED keeps every mode position-aligned. Beyond the input and output heads and a class label that is null in encoding and decoding, the mode token and timestep are thus the only signals that distinguish the three modes, and how to set them is the central design question for a fully shared backbone, which we study in Section 4.3.

3 Method

UNIFIED replaces the encoder, generator, and decoder of latent diffusion with a single diffusion transformer T_θ that operates in three modes: encoding, generation, and decoding. Each training step runs one pass per mode, and every pass is a position-aligned sequence map, in which index i refers to the same image location whether it holds a pixel patch, a latent token, or a noised latent token.

The backbone weights are fully shared; the modes differ only in lightweight input and output heads, in a learned mode token prepended to the sequence, and in the AdaLN conditioning, namely the timestep and a class label that is null outside generation (Table 1). Section 3.1 fixes notation and reviews how UNITE uses one network for encoding and generation. Section 3.2 folds decoding into the same backbone, Section 3.3 explains how the backbone is informed about the three modes, and Section 3.4 gives the single-stage training objective.

3.1 Preliminaries: latent diffusion and UNITE

Let $x \in \mathbb{R}^{H\times W\times 3}$ denote an image, which a transformer reads as $N = HW/p^2$ patches of size $p\times p$, and let $z \in \mathbb{R}^{K\times d}$ denote its latent, a sequence of K tokens of width d. Following UNITE (Duggal et al., 2026), $t \in [0,1]$ is the noise level, with $t = 1$ corresponding to clean data and $t = 0$ to Gaussian noise, and c is the class label, with $\emptyset$ denoting the null label. A noised latent is formed by the rectified-flow interpolation (Liu et al., 2023; Lipman et al., 2023)

$$z_t = t\, z + (1-t)\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \quad t \sim \mathcal{U}[0,1], \tag{1}$$

so that $z_1 = z$ is the clean latent and z_0 is pure noise.

Latent diffusion. The standard pipeline (Figure 2a) trains three networks: an encoder E_θ that maps an image to its latent, a generator G_ψ that recovers the clean latent from a noised one, and a decoder D_ϕ that maps latents back to pixels (Rombach et al., 2022; Peebles & Xie, 2023). The encoder and decoder are trained first under a reconstruction objective and then frozen, so generative gradients cannot shape the latent space that the generator must later model.

UNITE. UNITE observes that tokenization and generation are the same latent-inference problem under different conditioning: tokenization infers the latent from a fully observed image, and generation infers it from a noised latent and a class label (Duggal et al., 2026). It therefore implements the encoding and generation modes with one network, the *generative encoder*, trained in a single stage (Figure 2b). Its encoding pass reads the image patches alongside K tokens drawn from $\mathcal{N}(0, I)$ and returns their outputs as the latent, and its generation pass places z_t at the same positions. UNITE calls these tokens registers; we call them *latent readouts*, since unlike the registers of Darcet et al. (2024), whose outputs are discarded, their outputs are the latent. The clean target of the generation loss is stop-gradient, which keeps the latent space from moving solely to make generation trivial, while reconstruction and generation still update the shared weights jointly. Writing each pipeline in terms of the three functions it learns, with a common subscript marking functions that run on the same backbone, gives

$$\text{LDM:} \qquad z = E_\theta(x), \quad \hat{z} = G_\psi(z_t, t, c), \quad \hat{x} = D_\phi(z), \tag{2}$$

$$\text{UNITE:} \qquad z = E_\theta(x), \quad \hat{z} = G_\theta(z_t, t, c), \quad \hat{x} = D_\phi(z), \tag{3}$$

$$\text{UNIFIED:} \qquad z = E_\theta(x), \quad \hat{z} = G_\theta(z_t, t, c), \quad \hat{x} = D_\theta(z). \tag{4}$$

Functions with a common subscript share every backbone weight; they differ only in lightweight input and output heads and, in UNIFIED, a learned mode token (Section 3.2). UNITE reduces three networks to two, but the decoder D_ϕ remains a standalone network. The rest of this section describes how UNIFIED (Equation 4) runs the decoding mode on the same parameters θ as well.

3.2 UNIFIED: one backbone, three modes

UNIFIED computes all three functions in Equation 4 with one backbone T_θ, a stack of DiT blocks (Peebles & Xie, 2023) with RoPE (Su et al., 2024), in which the timestep and class label reach every block through AdaLN. The backbone takes a token sequence together with (t, c) and returns a sequence of the same length. Four lightweight heads translate between values and the backbone width: P_x^{in} and P_x^{out} for pixel patches, and P_z^{in} and P_z^{out} for latent tokens. Only one of them is new: P_x^{in}, P_z^{in}, and P_z^{out} are the heads of UNITE's generative encoder, used unchanged, and P_x^{out} is added so that the backbone itself returns pixel patches, replacing UNITE's standalone decoder. A learned mode token, q_{enc}, q_{gen}, or q_{dec}, is prepended at position 0 to select the mode. The three passes are

$$E_\theta(x) = P_z^{\text{out}}\, T_\theta\big([q_{\text{enc}}; P_x^{\text{in}}(x)],\, t_{\text{enc}},\, \emptyset\big)\,, \tag{5}$$

$$G_\theta(z_t, t, c) = P_z^{\text{out}}\, T_\theta\big([q_{\text{gen}}; P_z^{\text{in}}(z_t)],\, t,\, c\big)\,, \tag{6}$$

$$D_\theta(z) = P_x^{\text{out}}\, T_\theta\big([q_{\text{dec}}; P_z^{\text{in}}(z)],\, t_{\text{dec}},\, \emptyset\big)\,, \tag{7}$$

Table 1: **What each pass receives.** Every pass runs through the same backbone T_θ after a mode token at position 0. In UNIFIED, all three passes are position-aligned sequences of length $N+1$, and the output is read at all N positions. UNITE++ keeps UNITE's readout layout: the encoding pass appends K latent readouts $r \sim \mathcal{N}(0, I)$ and reads the latent from them, and the decoding pass appends N pixel readouts x_{readout} and reads pixels from them.

Model	Mode	Input sequence	Output	t	c	Length
UNIFIED	Encoding	$[q_{\text{enc}}; P_x^{\text{in}}(x)]$	P_z^{out} at positions $1{:}N$	1	$\emptyset$	$N+1$
	Generation	$[q_{\text{gen}}; P_z^{\text{in}}(z_t)]$	P_z^{out} at positions $1{:}N$	t	c	$N+1$
	Decoding	$[q_{\text{dec}}; P_z^{\text{in}}(z)]$	P_x^{out} at positions $1{:}N$	1	$\emptyset$	$N+1$
UNITE++	Encoding	$[q_{\text{enc}}; r; P_x^{\text{in}}(x)]$	P_z^{out} at the K latent readouts	0	$\emptyset$	$1+K+N$
	Generation	$[q_{\text{gen}}; P_z^{\text{in}}(z_t)]$	P_z^{out} at positions $1{:}K$	t	c	$1+K$
	Decoding	$[q_{\text{dec}}; P_z^{\text{in}}(z); x_{\text{readout}}]$	P_x^{out} at the N pixel readouts	1	$\emptyset$	$1+K+N$

where $[\,\cdot\,;\,\cdot\,]$ denotes concatenation along the sequence dimension and the output at the mode-token position is discarded before the output head. The latent has one token per patch, $K = N$, so every pass has length $N+1$; the encoding and decoding timesteps t_{enc} and t_{dec} are discussed in Section 3.3. Every backbone parameter is shared across Equations 5–7; the only mode-specific parameters are the four heads and the three mode tokens, which together make up 1.63% of the parameters at Base ($\sim$215M). Because the generation and decoding passes read latents through the same head P_z^{in}, the two differ only in their mode token, their conditioning, and their output head. Table 1 lists what each pass receives.

3.3 Specifying modes in shared weights

With every backbone weight shared, the three passes differ only in what enters the backbone: the values supplied by the input heads, the layout of the sequence, the mode token, and the conditioning (t, c). The class label is null in both encoding and decoding, so the layout, the mode token, and the timestep must keep the modes apart. We set each by a single principle.

Position alignment. In UNITE and UNITE++, the latent occupies its own group of readout tokens (Table 1), so information crosses between pixels and latents only through attention: the content of every image location must be routed into readout tokens with no fixed spatial correspondence. This decouples the latent length K from the number of patches, as in Perceiver-style readouts (Jaegle et al., 2022), but the backbone must learn this routing on top of the computation itself. UNIFIED instead aligns the sequences of all three modes: the pixel patch, latent token, and noised latent token for a given image location share one index, so RoPE assigns them the same geometry and information about that location can pass through the residual stream at its own position. No readout tokens are needed, and each encoding or decoding pass is about half as long as in UNITE++; the cost is that $K = N$, so the compression rate is set by the latent width d alone. We therefore expect the encoding pass to track the generation pass from the early blocks rather than only near the output, which we test with layer-wise CKA in Section 4.4.

Timestep matches observation. The timestep is the backbone's only signal of how much of its input is observed, so we set it in every mode to match that amount. The generation pass uses the sampled t. The encoding pass reads fully observed image patches and the decoding pass reads a clean latent, so both use the clean endpoint, $t_{\text{enc}} = t_{\text{dec}} = 1$. This principle is coupled to the layout. In UNITE and UNITE++, the latent positions of the encoding pass hold $\mathcal{N}(0, I)$ latent readouts, the same input from which sampling starts, so the matched choice there is the noisy endpoint, $t_{\text{enc}} = 0$. Only under position alignment do those positions hold observed content, which makes the clean endpoint the matched choice for encoding; we therefore compare UNIFIED with UNITE++, each at its own matched timestep.

Mode token. At the clean endpoint, the generation pass reads $z_1 = z$, the same clean latent that the decoding pass reads, through the same head P_z^{in} at the same timestep, and apart from the class label, which is null in decoding, the two receive the same conditioning. What tells the backbone whether to return a latent or the features from which P_x^{out} produces pixels is therefore the mode token. It enters as a prefix token and influences the other positions only through attention, so every block

Table 2: ImageNet-1K 256×256 at matched sizes, with CFG at each model's best scale (without CFG: Table 7). Tiny UNITE uses its best allocation (2:1, Table 3).

Size	Method	Params	rFID ↓	gFID ↓	IS ↑
Tiny	UNITE	61M	**1.27**	3.04	271.9
	UNIFIED	62M	1.30	**2.79**	**281.1**
Small	Separate	113M	1.64	5.69	215.7
	UNITE	117M	1.13	2.77	284.4
	UNIFIED	116M	**0.95**	**2.44**	**287.4**
Base	Separate	222M	1.38	4.01	255.9
	UNITE	217M	1.41	2.23	281.2
	UNIFIED	215M	**0.97**	**2.16**	**314.2**

Table 3: Parameter allocation at the Tiny budget (∼60M). Enc.:Dec. is the ratio of encoder/generator to decoder parameters in UNITE; UNIFIED has no standalone decoder.

Model	Enc.:Dec.	Params	rFID ↓	gFID ↓	IS ↑
UNITE	1:2	60M	**1.24**	5.03	240.7
	1:1	60M	1.59	3.76	260.3
	2:1	61M	1.27	3.04	271.9
UNIFIED	–	62M	1.30	**2.79**	**281.1**

stays fully shared across modes. Section 4.3 evaluates encoding at the noisy endpoint and training without mode tokens.

3.4 Single-stage training and inference

Each training step runs the three passes of Section 3.2 on a batch of images. The encoding pass produces $z = E_\theta(x)$. Following UNITE (Duggal et al., 2026), the latent is perturbed before reconstruction, $\tilde{z} = z + \sigma_{\text{rec}}\epsilon_r$ with $\epsilon_r \sim \mathcal{N}(0, I)$, which makes the decoding mode robust to perturbations of the latent, and the decoding pass reconstructs $\hat{x} = D_\theta(\tilde{z})$ under

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{x,\epsilon_r}[\|\hat{x} - x\|_1 + \lambda_{\text{perc}}\,\text{LPIPS}(\hat{x}, x)]\,, \tag{8}$$

where LPIPS is the perceptual distance (Zhang et al., 2018). The generation pass noises the stop-gradient latent as in Equation 1 and predicts the clean latent directly:

$$\mathcal{L}_{\text{gen}} = \mathbb{E}_{x,t,\epsilon}\left[\|G_\theta(t\ \text{sg}(z) + (1-t)\epsilon,\ t,\ c) - \text{sg}(z)\|_2^2\right], \tag{9}$$

where sg denotes stop-gradient. The complete objective is

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{gen}}\mathcal{L}_{\text{gen}}. \tag{10}$$

Gradient flow. As in UNITE, the stop-gradient keeps the generation loss from reaching the encoding pass through its target, but all three passes run through T_θ, so both losses update the same weights. Sharing the decoder adds a coupling that UNITE lacks: every backbone weight now receives reconstruction gradients through the encoding and decoding passes and generation gradients through the generation pass, whereas UNITE's decoder receives only reconstruction gradients.

Inference. Encoding and decoding each take a single pass. To generate, we start from Gaussian noise at $t = 0$, integrate the learned flow toward $t = 1$ with repeated generation passes, and apply one decoding pass to the final latent.

4 Experiments

All ImageNet-1K 256×256 models, including the baselines, are trained for 200 epochs; each optimizer step combines one reconstruction step and 12 flow-matching steps on the same batch into a single gradient update. The separate pipeline uses a standalone encoder, generator, and decoder but is otherwise trained identically (Section 3.4). Appendix E extends the comparison to molecules (QM9), and Appendix G discusses limitations.

4.1 Full unification improves generation at matched size

UNIFIED obtains the best gFID and IS at every size (Table 2), improving gFID over UNITE by 0.25, 0.33, and 0.07 at Tiny, Small, and Base, while the separate pipeline trails both weight-shared models. Absorbing the decoder does not cost reconstruction: rFID improves at Small ($1.13 \to 0.95$) and Base ($1.41 \to 0.97$) and stays comparable at Tiny (1.30 vs. 1.27).

Table 4: **The encoding timestep should match the encoder input.** Base ($\sim$215M); all variants decode at $t = 1$.

Model	Encoder input	t_{enc}	rFID ↓	gFID ↓	IS ↑
UNITE++	noise latent readouts	0 (matched)	2.20	2.24	326.2
UNIFIED	image patches	0 (mismatched)	1.40	2.20	303.0
UNIFIED (default)	image patches	1 (matched)	**0.97**	**2.16**	**314.2**

Table 5: **Ending the sampling ODE at $t = 0.99$.** Base models, adaptive dopri5 solver, 50K samples; the $[0, 1]$ rows repeat Table 2. NFE counts generation passes per image, including rejected trial steps, which can overshoot the end of the interval.

Model	ODE range	gFID ↓	IS ↑	NFE	NFE, $t \in (0.99, 1]$	NFE, $t > 1$
UNITE	$[0, 1]$	2.23	281.2	124.8	22.3	19.2
	$[0, 0.99]$	2.22	281.9	120.2	17.4	14.9
UNIFIED	$[0, 1]$	2.16	314.2	199.9	76.5	32.6
	$[0, 0.99]$	**2.05**	**316.7**	164.6	19.4	22.5

4.2 Decoder parameters are better spent in the shared backbone

At a fixed Tiny budget (Table 3), shifting UNITE's capacity toward the encoder/generator improves gFID (1:1 $\rightarrow$ 2:1: 3.76 $\rightarrow$ 3.04), whereas shifting it to the decoder degrades gFID to 5.03 despite the best rFID. Removing the standalone decoder altogether (UNIFIED) improves further to 2.79. Appendix D follows these models through training.

4.3 How are three tasks kept apart in shared weights?

Section 3.3 sets each mode's timestep to match how much of its input is observed. UNITE++ encodes from $\mathcal{N}(0, I)$ readouts and thus at $t_{\text{enc}} = 0$; UNIFIED encodes from image patches, and labeling them as noise ($t_{\text{enc}} = 0$) hurts both reconstruction and generation (rFID 1.40 vs. 0.97, gFID 2.20 vs. 2.16; Table 4). We omit UNITE++ at $t_{\text{enc}} = 1$, which would label noise as clean.

Mode tokens are necessary: without them, neither UNITE++ nor UNIFIED trains; the loss does not converge, and generated images are only high-frequency noise patterns.

Sharing the clean endpoint with decoding also affects sampling. Decoding runs at $t = 1$ in every step, whereas generation draws $t \in [0.99, 1]$ for only 1% of samples, so the backbone near $t = 1$ is shaped mostly by decoding and the end of the generation trajectory is learned least. Table 5 tests this by ending the ODE at $t = 0.99$. Over $[0, 1]$, UNIFIED spends 38% of its evaluations in the last 1% of the interval (76.5 of 199.9), versus 18% for UNITE: because the velocity $(\hat{z} - z_t)/(1 - t)$ amplifies errors in $\hat{z}$ as $t \rightarrow 1$, the adaptive solver shrinks its steps wherever that prediction is inaccurate. Stopping at $t = 0.99$ cuts UNIFIED's NFE from 199.9 to 164.6 and improves gFID from 2.16 to 2.05; the residual noise is tolerated by the decoding pass, which is trained on latents perturbed by σ_{rec}.

UNITE, whose separate decoder leaves the endpoint to generation alone, is nearly unaffected (gFID 2.23 $\rightarrow$ 2.22, 4% fewer NFE), which attributes the effect to the shared endpoint rather than to the solver or the velocity parameterization. With both models ending at $t = 0.99$, UNIFIED's gFID lead grows from 0.07 to 0.17, although it still needs more evaluations (164.6 vs. 120.2).

4.4 What does the shared backbone compute in each mode?

Figure 3 tests the hypothesis of Section 3.3. With position alignment, the encoding pass tracks the generation pass from the first block: both UNIFIED variants reach a CKA of 0.48–0.80 in blocks 1–5, and through block 9 one of them is highest at every t. UNITE and UNITE++ stay at or below 0.15 in blocks 1–5, because their encoders read $\mathcal{N}(0, I)$ readouts that carry no image information until attention writes it in; they align only in late blocks: UNITE reaches 0.84–0.92 in its last two.

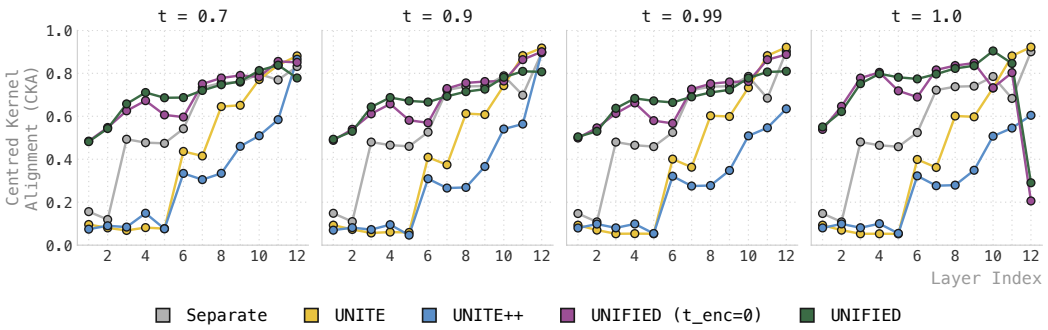


Figure 3: **Layer-wise CKA between the encoding and generation passes.** Following UNITE (Duggal et al., 2026), we record each block's activations while encoding an image and while running the generation pass on its encoded latent corrupted to level t ($t = 1$ clean); each panel reports CKA (Kornblith et al., 2019) at one t.

The exception is the last block at the clean endpoint. At $t = 1$ its CKA falls to 0.29 for UNIFIED and 0.21 for UNIFIED ($t_{\text{enc}} = 0$), and to 0.60 for UNITE++ (already 0.63 at $t = 0.99$), but stays at 0.92 and 0.90 for UNITE and Separate. The drop thus appears only when the shared backbone also decodes: at $t = 1$ its last block serves decoding, which outputs pixel features rather than a latent. This is consistent with Section 4.3, where sampling benefits from skipping this endpoint.

4.5 Generation quality vs. sampling budget

Figure 4 shrinks the sampling budget from 128 to 8 NFE. The separate pipeline degrades most (gFID 4.47 → 27.49), whereas UNITE, UNITE++, and UNIFIED all stay below 10 at 8 NFE. UNIFIED improves on UNITE on both metrics at every budget and has the lowest gFID at 32 and 64 NFE; at 64 it already surpasses UNITE and UNITE++ at 128 (2.20 vs. 2.35 and 2.32). UNITE++ has the lowest gFID at 8 and 16 NFE (7.16 and 3.13 vs. 8.56 and 3.32) and the highest IS throughout, suggesting that its separate readout pathway may help under coarse steps.

Encoding at the noisy endpoint shows the opposite sensitivity: UNIFIED with $t_{\text{enc}} = 0$ is worst at 8 and 16 NFE (gFID 39.05 and 9.20) but matches the default at 128 (2.09 vs. 2.11). We hypothesize that at $t = 0$ it must both encode clean patches and denoise Gaussian noise at the same positions, separated only by the mode token, which makes early velocity predictions unreliable; coarse steps carry this error into the sample, whereas finer steps correct it. UNITE++ avoids the conflict because its encoder also reads noise at $t = 0$.

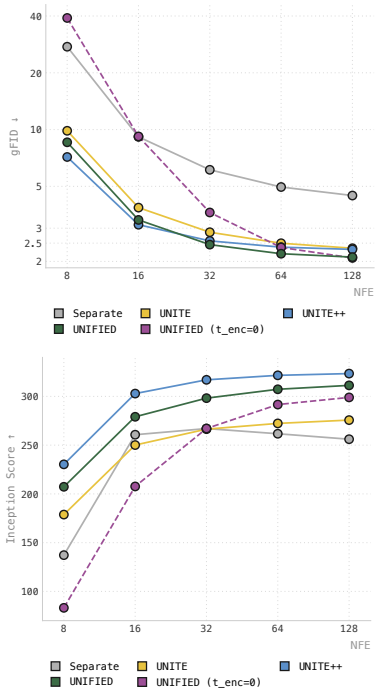


Figure 4: **Generation quality across sampling budgets.** Base models (∼215M), 8–128 NFE. *(Left)* gFID (↓, log scale). *(Right)* Inception Score (↑).

5 Conclusion

In this paper, we present UNIFIED, a single diffusion transformer that can encode, denoise, and decode with all backbone weights shared, provided that the three modes are cast as position-aligned sequence maps and distinguished by a learned mode token and a timestep that reflects how much of each input is observed. On ImageNet-1K 256 × 256, UNIFIED improves gFID over both parameter-matched baselines at all three model scales. Our analysis explains where the gains come from: capacity spent on a standalone decoder is better spent in the shared backbone, and, as the layer-wise CKA results indicate, position alignment, rather than weight sharing alone, is what makes full unification pay off. Overall, full unification of latent diffusion is not only simple but beneficial, and we hope it serves as a step toward generative models in which encoding, denoising, and decoding are modes of a single network rather than separate networks.

AI USE STATEMENT

We used LLMs (Claude and ChatGPT) to help implement parts of the training and evaluation code, write figure plotting scripts, and search for related literatures. They were also used to polish the manuscript for grammar and readability.

The LLMs played no role in forming hypotheses, developing the conceptual framework, designing the methodology or experiments, or interpreting results. The model formulation, choice of baselines and ablations, and all conclusions are the authors' own. Other uses such as synthetic data generation, qualitative data analysis, proof writing, and translation do not apply, since we use the standard ImageNet-1K and QM9 benchmarks, include no formal proofs, and wrote the paper in English.

The authors reviewed all AI-assisted work and take full responsibility for the final content. We inspected all AI-assisted code and confirmed every reported number against our own training logs, verified suggested papers and bibliography entries against their original sources, and checked that AI-edited text did not change any technical content or claims.

ETHICS STATEMENT

This work studies the architecture of latent generative models and does not involve human subjects, the collection of personal data, or new datasets. All experiments use the public ImageNet-1K and QM9 benchmarks under their respective terms of use.

REPRODUCIBILITY STATEMENT

Section 3 describes the model, the mode specification, and the training objective. The appendix gives the detailed training and sampling procedure. Our experiments follow the public protocols, and all datasets are publicly available. We will release code and checkpoints upon publication.

REFERENCES

Fan Bao, Shen Nie, Kaiwen Xue, Chongxuan Li, Shi Pu, Yaole Wang, Gang Yue, Yue Cao, Hang Su, and Jun Zhu. One transformer fits all distributions in multi-modal diffusion at scale. In *International Conference on Machine Learning*, pp. 1692–1717. PMLR, 2023.

A Patrícia Bento, Anne Hersey, Eloy Félix, Greg Landrum, Anna Gaulton, Francis Atkinson, Louisa J Bellis, Marleen De Veij, and Andrew R Leach. An open source chemical structure curation pipeline using rdkit. *Journal of cheminformatics*, 12(1):51, 2020.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, 2025.

Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David Fleet, and Geoffrey E Hinton. A unified sequence interface for vision tasks. *Advances in Neural Information Processing Systems*, 35: 31333–31346, 2022.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International conference on learning representations*, 2024.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Shivam Duggal, Xingjian Bai, Zongze Wu, Richard Zhang, Eli Shechtman, Antonio Torralba, Phillip Isola, and William T Freeman. End-to-end training for unified tokenization and latent denoising. *arXiv preprint arXiv:2603.22283*, 2026.

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*, pp. 12873–12883, 2021.

Philippe Hansen-Estruch, Sriram Vishwanath, Amy Zhang, and Manan Tomar. Unified autoencoding with masked diffusion. *arXiv preprint arXiv:2406.17688*, 2024.

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

Jonathan Heek, Emiel Hoogeboom, Thomas Mensink, and Tim Salimans. Unified latents (UL): How to train your latents. *arXiv preprint arXiv:2602.17270*, 2026.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

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.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Duong H Le, Tuan Pham, Sangho Lee, Christopher Clark, Aniruddha Kembhavi, Stephan Mandt, Ranjay Krishna, and Jiasen Lu. One diffusion to generate them all. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2671–2682. IEEE, 2025.

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. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18262–18272. IEEE, 2025.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: Masked generative encoder to unify representation learning and image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2142–2152, 2023.

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-IO: A unified model for vision, language, and multi-modal tasks. In *International Conference on Learning Representations*, 2023.

Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-IO 2: Scaling autoregressive multimodal models with vision, language, audio, and action. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26429–26445. IEEE, 2024.

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. In *European Conference on Computer Vision*, pp. 23–40, Cham, 2024. Springer Nature Switzerland.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations*, 2024.

David Mizrahi, Roman Bachmann, Oguzhan Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4M: Massively multimodal masked modeling. *Advances in Neural Information Processing Systems*, 36:58363–58408, 2023.

Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open Babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1):33, 2011.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024.

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

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

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in neural information processing systems*, 32, 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. IEEE, 2022.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021.

Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist Painter for in-context visual learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6830–6839. IEEE, 2023.

Xiyuan Wang and Muhan Zhang. Diffusion as self-distillation: End-to-end latent diffusion in one model. *arXiv preprint arXiv:2511.14716*, 2025.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12966–12977. IEEE, 2025.

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

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations*, 2025.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712. IEEE, 2025.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion – tokenizer is key to visual generation. In *International Conference on Learning Representations*, 2024a.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024b.

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

Qinyu Zhao, Guangting Zheng, Tao Yang, Rui Zhu, Xingjian Leng, Stephen Gould, and Liang Zheng. SimFlow: Simplified and end-to-end training of latent normalizing flows. In *European Conference on Computer Vision*, pp. 679–697, Cham, 2026. Springer Nature Switzerland.

Guangting Zheng, Yiyuan Zhang, Tao Yang, Yunpeng Chen, Rui Zhu, Jiajun Deng, and Yanyong Zhang. GenFirst: Generation before reconstruction for stable end-to-end latent generative modeling. *arXiv preprint arXiv:2608.29335*, 2026.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Learning Representations*, 2025.

APPENDIX

In this appendix, we first provide the ImageNet-1K training details and the evaluation protocols used across the paper. Then, we provide a compute analysis on model training and additional ablation results on ImageNet-1K. Next, we present results and implementation details for the QM9 molecule experiment. Finally, we discuss limitations.

A IMAGENET TRAINING DETAILS

Table 6 lists the training configuration shared by all ImageNet-1K 256×256 models, including the baselines. We follow UNITE (Duggal et al., 2026) except for two settings: we train for 200 epochs instead of 240, and each optimizer step combines one reconstruction step with 12 flow-matching steps instead of 14.

Table 6: Training configuration for ImageNet-1K 256×256, shared by all models. Settings follow UNITE (Duggal et al., 2026) except for the total epochs and the flow-matching steps per reconstruction step (UNITE: 240 and 14).

Hyperparameter	Value	Hyperparameter	Value
Base learning rate	1×10^{-4}	Warmup epochs	20
Global batch size	1024	Total epochs	200
Optimizer	Muon	LR schedule	Cosine
AdamW betas	$(0.9, 0.999)$	Min LR	1×10^{-6}
Weight decay	0	Gradient clip	3.0
Reconstruction noise (σ_{rec})	0.7	EMA decay	0.9978
Flow timesteps (training)	1000	Noise schedule shift (α)	0.5
Flow steps per recon. step	12	Precision	BF16 (mixed)

B EVALUATION PROTOCOLS

FID computation. We report Fréchet Inception Distance (FID) computed with the `torch-fidelity` library using InceptionV3 features. Reference statistics are computed over the full ImageNet-1K training set (1,281,167 images), and all FID scores are evaluated on 50K generated samples.

Sampling protocol. Following UNITE (Duggal et al., 2026), we use class-balanced sampling, generating exactly 50 images for each of the 1,000 ImageNet classes (50K images in total).

ODE solver. During inference, we follow UNITE (Duggal et al., 2026) to solve the probability flow ODE over the interval $[0.1, 1.0]$ with classifier-free guidance, and sweep the CFG scale ω from 1.0 to 4.0 in increments of 0.2 and report the best FID for each model. We use adaptive fifth-order Dormand–Prince solver (`dopri5`) as the main solver.

CKA analysis. To test whether the shared backbone processes clean images and noised latents in a similar way, we measure centered kernel alignment (CKA) (Kornblith et al., 2019) between its activations in the two modes. For each validation image, we run the backbone twice: once in *encode* mode on the image, and once in *denoise* mode on the noised latent $x_t = t\, z_n + (1 - t)\, \epsilon$, where z_n is the layer-normalized encoder output and $t = 1$ corresponds to the clean latent. The noise $\epsilon \sim \mathcal{N}(0, I)$ is drawn once per batch and reused across the entire t-grid, and class conditioning is set to the null label. We record activations at $D + 1$ taps (the input to the first block and the output of each of the D blocks) and retain only the N latent-token positions, so that token i corresponds to the same latent position in both modes. This yields, for each image and tap, matrices $X \in \mathbb{R}^{N \times H_{\text{enc}}}$ and $Y \in \mathbb{R}^{N \times H_{\text{den}}}$.

Unlike the token-level and pooled views, the per-image view treats the N tokens of a single image as the examples. We center each matrix over its tokens, $\tilde{X} = CX$ and $\tilde{Y} = CY$ with $C = I_N - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$,

and form the linear Gram matrices $K = \tilde{X}\tilde{X}^\top$ and $L = \tilde{Y}\tilde{Y}^\top$. The biased estimator is

$$\mathrm{CKA}(K, L) = \frac{\langle K, L \rangle_F}{\|K\|_F \, \|L\|_F}. \tag{11}$$

The per-image score thus asks whether encode and denoise modes induce the same token-to-token similarity structure within a single image.

C Compute analysis

Sharing every weight across encoding, denoising, and decoding is beneficial in our experiments: at matched parameters, UNIFIED improves gFID over UNITE at all three sizes (Table 2), and moving capacity from a standalone decoder into the shared backbone improves generation (Section 4.2). We attribute this to the fact that no parameters are reserved for any single mode. Training can therefore assign the capacity of the whole network to whichever task needs it, instead of fixing the split in advance, as UNITE's encoder-to-decoder ratio does (Table 3).

This flexibility comes with a caveat in the current, simple form of UNIFIED. Because every mode uses the full backbone, each denoising step runs the whole network, whereas UNITE runs only its generative encoder. At matched parameters (215.6M vs. 217.6M), the shared backbone is also wider (984 vs. 768), so each step costs more: one flow-matching forward pass takes 81.7 GFLOPs in UNIFIED and 50.1 in UNITE. This $1.63\times$ ratio follows the $(984/768)^2 \approx 1.64$ scaling of the matrix multiplies. The reconstruction branch, in contrast, costs about the same in both models (0.62 vs. 0.57 TFLOPs): UNIFIED encodes and decodes position-aligned 257-token sequences, while UNITE encodes a 512-token sequence of latent readouts and patches and decodes with a separate decoder. Since flow matching dominates the training step (83% of UNIFIED's compute and 76% of UNITE's), a full step costs 3.58 TFLOPs per image for UNIFIED and 2.39 for UNITE, or $1.50\times$ more. Over 200 epochs of ImageNet-1K (2.56×10^8 samples), this comes to 9.2×10^{20} and 6.1×10^{20} FLOPs (10.6 and 7.1 PF-days).

At sampling time, the higher cost per step does not have to mean a higher cost at equal quality. UNIFIED at 64 NFE already reaches a lower gFID than UNITE at 128 NFE (2.20 vs. 2.35, Section 4.5), which takes about 18% fewer denoising FLOPs per sample (5.2 vs. 6.4 TFLOPs). More generally, the overhead comes from running the entire backbone uniformly at every step and in every mode, not from weight sharing itself. Future versions could reduce it, for example by running only a subset of blocks for denoising or by adapting depth to the timestep.

Measurement details. We count FLOPs with PyTorch's `FlopCounterMode` (Paszke et al., 2019) on the exact training configurations (256×256 inputs, 256 latent tokens, and 12 flow-matching steps per reconstruction step), counting a multiply–add as two FLOPs. A training step includes the forward and backward passes of the reconstruction branch (encoder, decoder, and the LPIPS network) and of the 12 flow-matching steps. All figures above are model FLOPs. Gradient checkpointing of the generation passes recomputes every flow-matching forward pass, which raises the executed compute to 1.17×10^{21} (UNIFIED) and 7.7×10^{20} (UNITE) FLOPs. The Muon optimizer adds less than 0.2%.

D Additional ablation experiments

gFID on ImageNet-1K 256×256 **without classifier-free guidance.** In Table 7, we evaluate gFID of every method at each parameter level without classifier-free guidance. The ranking is unchanged from Table 2: UNIFIED obtains the best gFID and IS at every size, and the separate pipeline trails both weight-shared models.

rFID vs. gFID during training. In Figure 5, we evaluate the rFID and gFID every 40 epochs on Tiny models. Decoder-heavy UNITE (1:2) reconstructs best but generates worst at every checkpoint. UNIFIED generates the best over all UNITE models.

Table 7: ImageNet-1K generation at matched parameter levels without classifier-free guidance. Models and sizes match Table 2; at Tiny, UNITE uses its 2:1 encoder–decoder allocation.

Size	Method	Params	gFID ↓	IS ↑
Tiny	UNITE	61M	26.21	60.6
	UNIFIED	62M	**20.90**	**74.9**
Small	Separate	113M	42.54	36.9
	UNITE	117M	22.65	69.4
	UNIFIED	116M	**17.68**	**86.6**
Base	Separate	222M	29.74	52.4
	UNITE	217M	16.11	90.5
	UNIFIED	215M	**14.17**	**102.1**

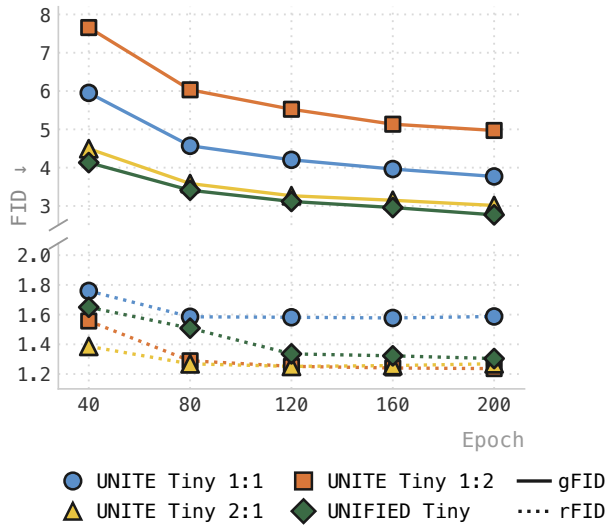


Figure 5: **Reconstruction and generation FID during training.** Tiny models of Table 3, evaluated every 40 epochs. Top: gFID (CFG tuned per checkpoint); bottom: rFID.

E Molecular experiments on QM9

Position alignment exploits the fact that every image shares one token grid. QM9 tests whether the benefit of full sharing carries over to data without such a grid, where each molecule is a variable-size set of atoms with no canonical ordering. Table 8 compares UNITE and UNIFIED on reconstruction (match rate, RMSD) and generation (validity, uniqueness).

Setup. We follow the QM9 protocol of UNITE (Duggal et al., 2026), built on ADiT (Joshi et al., 2025). QM9 (Ramakrishnan et al., 2014) contains ∼130K small molecules with up to 29 atoms including hydrogens, each represented by atom types and centered 3D coordinates, with one token and one 16-dimensional latent per atom. At matched capacity, UNITE uses an 8-block generative encoder and a separate 6-block decoder, and UNIFIED a single 12-block backbone, position-aligned as on ImageNet: atom i and latent i share an index, and encoding and decoding run at $t = 1$ (Table 9). Both are trained in a single stage with the same objective and one generation update per reconstruction update (12 on ImageNet); Appendix F gives the full details.

Results. Unlike on ImageNet, full sharing does not improve on UNITE on QM9 (Table 8). UNIFIED reconstructs nearly as well (match rate 97.69% vs. 98.38%) but generates far fewer valid molecules (48.48% vs. 87.26%), so the gap lies in modeling the latent distribution rather than in encoding and decoding. We hypothesize that position alignment relies on token indices that mean the same thing in every sample, as a fixed image grid provides. Atom indices follow an arbitrary ordering and carry no shared geometry, so aligning latent i with atom i gives the backbone no com-

Table 8: Molecular reconstruction and generation results on QM9. Both models have $\sim$32.7M parameters and are evaluated with EMA weights after 6,000 epochs. Reconstruction is measured on the full test split (12,831 molecules) and generation on 10,000 samples (500 Euler steps, no guidance).

	Reconstruction		Generation	
Method	Match (%) ↑	RMSD (Å) ↓	Valid (%) ↑	Unique (%) ↑
UNITE	**98.38**	**0.036**	**87.26**	98.72
UNIFIED	97.69	0.056	48.48	**99.69**

Table 9: **QM9 sequence layout per pass.** a_i is the token of atom i, $\ell(\cdot)$ the latent-token embedding, q a mode token, and N the number of atoms. UNITE's latent readouts are $r_i = \ell(\epsilon_i)$ with $\epsilon_i \sim \mathcal{N}(0, I)$.

Model	Pass	Input sequence	t	Label	Output	Length
UNIFIED	Encoding	$[q_{\text{enc}}; a_{1:N}]$	1	null	latent head at positions $1{:}N$	$N+1$
	Generation	$[q_{\text{gen}}; \ell(z_t)_{1:N}]$	t	QM9	latent head at positions $1{:}N$	$N+1$
	Decoding	$[q_{\text{dec}}; \ell(z)_{1:N}]$	1	null	molecular head at positions $1{:}N$	$N+1$
UNITE	Encoding (encoder)	$[a_{1:N}; r_{1:N}]$	0	null	latent head at the N readouts	$2N$
	Generation (encoder)	$[\ell(z_t)_{1:N}]$	t	QM9	latent head at all positions	N
	Decoding (decoder)	$[\ell(z)_{1:N}]$	–	–	molecular head at all positions	N

mon positional structure to exploit, whereas UNITE's readout tokens aggregate all atoms through attention. Because the two models differ in both weight sharing and layout, this comparison alone does not isolate position alignment (Appendix G).

F QM9 IMPLEMENTATION DETAILS

This section completes the setup summarized in Appendix E. We follow Duggal et al. (2026) for the dataset, optimizer, learning rate, batch size, latent width, and sample count. They do not specify the token layout, the sampler, the ratio of generation to reconstruction updates, or the evaluator; for these we follow ADiT (Joshi et al., 2025) where it applies and otherwise make our own choices, described below. The two models share the data pipeline, objective, optimizer, sampling plan, and evaluator, and differ only in the backbone and sequence layout. Each is trained once, with a single seed, for 6,000 epochs, whereas Duggal et al. (2026) train for 8,000. Our UNITE numbers are therefore not a reproduction of their QM9 results and should not be compared with them directly; the comparison in Table 8 is controlled instead.

Model. An atom token is the sum of an atom-type embedding over a 100-way atomic-number vocabulary, a two-layer MLP of its coordinates in Å, and a sinusoidal embedding of its index i. A latent token is a linear projection of its 16-dimensional latent plus the same index embedding, so atom i and latent i carry the same positional signal. The AdaLN condition is the sum of a timestep embedding and a dataset-label embedding; the label is QM9 in the generation pass and null in the encoding and decoding passes. The latent head is an AdaLN output layer followed by a LayerNorm with learnable scale and shift, shared by the encoding and generation passes as in UNITE. The molecular head predicts atom-type logits and coordinates with two linear layers after a LayerNorm and re-centers the coordinates on their mean over real atoms.

Both models use width 384, 6 attention heads, and AdaLN blocks with RMSNorm, gated self-attention, and a gated SwiGLU MLP of inner width 1,024. UNITE pairs an 8-block generative encoder, used for encoding and generation, with a separate 6-block decoder of pre-LN transformer blocks with GELU MLPs of width 1,536 and no timestep input (32.730M parameters). UNIFIED uses one 12-block backbone for all three passes (32.725M). The two models match in parameters, not in compute: UNITE encodes $2N$ tokens, whereas UNIFIED encodes $N+1$ tokens but decodes through the full 12-block backbone.

Training. We use the QM9 release of PyTorch Geometric (130,831 molecules) with hydrogens retained, split by a fixed random permutation into 100,000 training, 18,000 validation, and 12,831 test molecules. This matches ADiT's split sizes, though not necessarily its molecule IDs. Each molecule is translated so that its mass-weighted center of mass lies at the origin. Following ADiT, we augment with random rotations and corrupt only the encoder input by masking atom types and jittering positions (Table 10); reconstruction targets are always the clean molecules.

Each optimizer step runs one encoding, one decoding, and one generation pass on the same batch and takes a single gradient step on their summed loss. Before decoding, the latent is perturbed as $\tilde{z} = z + \sigma_{\text{rec}}\epsilon$. Following ADiT, the reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \Big[-\log p_\theta(a_i \mid \tilde{z}) + \lambda_{\text{pos}} \tfrac{1}{3} \big\| \hat{\mathbf{x}}_i - \mathbf{x}_i \big\|_2^2 \Big], \tag{12}$$

where $\mathcal{A}$ is the set of real atoms in the batch, and predicted and target coordinates are in nanometers and centered on their mean over real atoms. The generation pass draws $t \sim \mathcal{U}[0, 1)$ per molecule, forms $z_t = t\ \text{sg}(z) + (1 - t)\,\epsilon$ with i.i.d. Gaussian ϵ, and predicts the clean latent; its squared error to $\text{sg}(z)$ is added to $\mathcal{L}_{\text{rec}}$ with weight λ_{gen}. No perceptual, adversarial, KL, or chemistry-based loss is used. Table 10 lists the hyperparameters, which are identical for the two models.

Table 10: QM9 training hyperparameters, shared by UNITE and UNIFIED.

Setting	Value
Optimizer	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0
Learning rate	10^{-4}, constant, no warmup
Batch size	512
Epochs evaluated	6,000 (1.17M steps)
Gradient clipping	global norm 10
EMA decay	0.9999
Precision	bfloat16; latent normalization and losses in float32
Latent	16 channels per atom
σ_{rec}, λ_{pos}, λ_{gen}	0.7, 10, 1
Generation : reconstruction updates	1:1
Label dropout (generation pass)	0.1
Encoder-input corruption	10% types masked, 10% positions jittered ($\sigma = 0.1$ Å)
Augmentation	one random rotation per batch

Sampling and evaluation. Both models sample from the same plan: 10,000 atom counts drawn from the histogram of the training split and 10,000 initial noise tensors, both from a fixed seed. Starting from $z_0 = \epsilon$, we take 500 explicit Euler steps on the uniform grid $t_k = k/500$, using the velocity $(\hat{z} - z_{t_k})/(1 - t_k)$ from the predicted clean latent $\hat{z}$, and decode the final latent once, for 501 function evaluations per molecule, with the QM9 label and no classifier-free guidance. All metrics use EMA weights, with no corruption or latent noise at evaluation. Predicted atom types are the argmax of the logits, with the padding class mapped to hydrogen as in ADiT. For reconstruction, each test molecule is encoded and decoded once and counts as a match when pymatgen's (Ong et al., 2013) `MoleculeMatcher` (tolerance 0.01) aligns it with the ground truth, as in ADiT; the RMSD is averaged over matched molecules. For generation, each sample is converted to a PDB block with Open Babel (O'Boyle et al., 2011) and parsed by RDKit (Bento et al., 2020) with sanitization, explicit hydrogens, and proximity-based bonding. Following ADiT, a sample is valid if this conversion succeeds, which checks that RDKit can sanitize the structure, including its valences, but not bond lengths or steric clashes. Validity is computed over all 10,000 samples without rejection or resampling, and uniqueness is the fraction of distinct canonical isomeric SMILES of the largest fragment among the valid samples. We use RDKit 2024.3.5, Open Babel 3.1.1, and pymatgen 2024.11.13.

G LIMITATIONS

Our evidence that full unification helps comes from images, where every sample shares one token grid. Many modalities lack such a grid, and on QM9, where molecules are variable-size sets of

unordered atoms, UNIFIED falls well short of UNITE in generation (Appendix E). Adapting position alignment to such data, for example through a canonical atom ordering, a permutation-invariant form of alignment, or a hybrid layout that uses readout tokens where no shared grid exists, is left to future work.

A second limitation follows directly from full weight sharing. When the tokenizer is a network of its own, the latent space can be frozen at some point, either from the start as in the standard two-stage recipe or after a phase of joint training, and the generator can then be trained or fine-tuned alone, for example on precomputed latents. Full sharing rules this strategy out for UNIFIED: the encoder, decoder, and generator are the same backbone, so any update that improves generation also changes the encoding pass, which moves the latent space the generator is fitting, and the decoding pass, which changes how latents map back to pixels. The reconstruction objective must therefore stay active for the whole of training, including any stage devoted to generation, and training keeps paying for encoding and decoding passes alongside the generation passes.