UNIFYING LATENT DENOISER WITH DECODER MAKES ROBUST IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the substantial progress of latent diffusion models, the gap between generated latents and those encoded from real data remains, posing a fundamental tension between reconstruction fidelity and generative modeling. In this study, we explore a simple-yet-effective approach that jointly learns denoising and image reconstruction from encoded image latents, termed the **D**enoise-decode **J**oint **T**ransformer (**DJT**). During training, the denoising and reconstruction objectives shape the shared backbone together, while each task-specific head specializes in its corresponding task. At inference time, the model acts like a recursive transformer: it first operates as a denoiser to iteratively produce a final latent sample and then as a decoder to map the latent into an image. Experiments on ImageNet show that DJT-XL achieves gFID=**0.97**, with **30%** less parameters than the counterpart pipeline. Representation analysis further shows that the joint training strategy strengthens the alignment between denoising and decoding representations and improves their functional compatibility. The code and models will be publicly available.

1 INTRODUCTION

Latent diffusion has become the dominant paradigm for image generation by moving the generative process from pixels to a compact latent space (Ho et al., 2020; Rombach et al., 2022; Peebles & Xie, 2023). However, aggressive compression commonly used in low-dimensional variational autoencoders (VAEs) (Kingma & Welling, 2014) limits its information capacity and semantic quality (Tong et al., 2026). Representational autoencoders (RAEs) (Zheng et al., 2026) propose to pair pretrained vision foundation models, such as DINOv2 (Oquab et al., 2024), with learned decoders and demonstrate strong generation performance in these high-capacity semantic spaces.

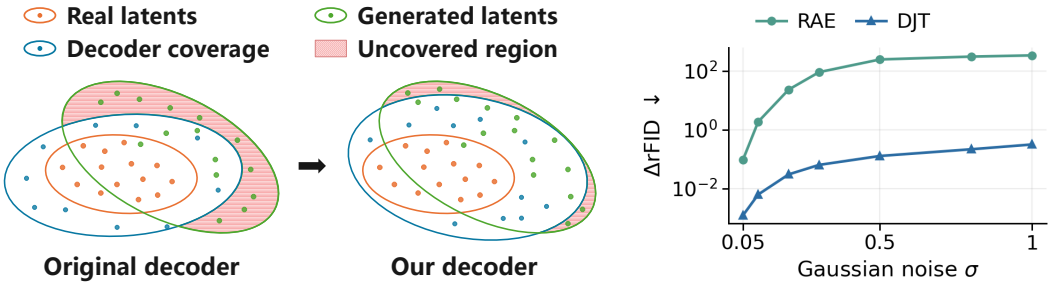
Nevertheless, RAE's deterministic encoder comes with a price: unlike VAEs, it does not expose the decoder to stochastic variations during training (Razavi et al., 2019; Vahdat & Kautz, 2020; Zheng et al., 2026), reducing its robustness to latent perturbations. RAE mitigates this with an additive Gaussian noise augmentation. However, its much higher-dimensional latent space is harder to model (Li & He, 2025), leading to a substantial distribution gap than in VAEs, as shown in Table 1. This larger gap places greater demands on the decoder (Tschannen et al., 2023), contributing to the much heavier decoder than VAE's (415.4M vs. 41.4M). [1]

Table 1: **Distribution gap between generated and real (encoded) latents.** Latent-FD is the Fréchet distance between the two sets of latents. C2ST (Lopez-Paz & Oquab, 2017) is the test accuracy of a discriminator trained to classify a latent as generated or real, for which 50% means the two sets are indistinguishable. The representational latents show significantly larger shift than VAE latents during the generative process.

Latent space	Latent-FD ↓	C2ST acc. ↓
VA-VAE $_{(16\times16\times32)}$	0.00299	60.30%
DINOv2 $_{(16\times16\times768)}$	2.033	85.22%

[1] Parameter computed: 415.4M for RAE Dec-XL and 41.4M for the VA-VAE decoder.

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.



(a) Train–test mismatch.

(b) Robustness to distribution shift.

Figure 1: **Train-test mismatch in decoding, and robustness comparison to latent perturbation.** (a) Illustration of the mismatch between encoded and generated latents. Our decoder (DJT) has broader decoding coverage than the original decoder (RAE), as evidenced in (b): across all noise perturbation levels, DJT exhibits substantially smaller degradation than RAE. σ controls the noise stength, and $\Delta\text{rFID} = \text{rFID}(\sigma) - \text{rFID}(0)$.

This motivates us to reconsider the functionality of denoiser and decoder when the image latent is *pre-acquired*. The decoder translates noise-augmented latents into images, while the denoiser predicts the denoising velocity from the noised latent. Training them requires similar stochastic noise injection, and most of the layers might be extracting information of high similarity. A natural question arises: *Can latent denoising and image reconstruction be learned jointly?*

The answer is promisingly positive. By comparing the layer-wise representational similarity between the RAE's independently trained decoder and denoiser, we observe that the reconstruction and generative tasks share considerable intermediate structure, showing the potential of unification with minimal training and inference overhead. Nevertheless, a simple stitching and recursive calculation of reconstructive and generative objectives leads to deteriorated performance, indicating that the architecture needs to be crafted wisely.

In pursuit of an elegantly unified generative model for representational latents, we propose **DJT** (**D**enoise-decode **J**oint **T**ransformer), where the denoiser and decoder share the backbone, and branch into two shallow task-specific heads for velocity prediction and image reconstruction, respectively. During training, the denoising and reconstruction objectives are assigned by the Bernoulli router (Hu et al., 2026), and jointly optimize the shared backbone. At inference time, the same backbone is reused for both stages: it first operates with the denoising head to iteratively produce a final latent sample, and then with the reconstruction head to map that latent to an image.

Experiments on ImageNet demonstrate that the proposed unified model achieves state-of-the-art **0.97** gFID. It can also save up to **30%** parameters. Through detailed analyzes, we validate that jointly learning the two stages does more than provide a parameter-sharing architecture: it reshapes the representations used by the generative pipeline and yields significantly better decoder robustness to the imperfect latents encountered at inference time. Extended results on VAE space also show that the unified design can be effectively utilized beyond representational latents, improving generation quality while substantially reducing decoding computation.

2 Related Work

Representations for latent diffusion. Latent diffusion models reduce generation costs by operating in compact spaces defined by pretrained autoencoders (Rombach et al., 2022). Subsequent work improves the trade-off between compression efficiency and reconstruction fidelity (Chen et al., 2025b;a). However, reconstruction quality alone does not ensure that latents are easy to model (Yao et al., 2025). DC-AE 1.5 (Chen et al., 2025c) further addresses this tension with a structured latent space and augmented diffusion training. Equivariance-based methods promote consistency between corresponding image and latent transformations, improving transformation robustness and latent-space learnability (Zhou et al., 2025; Kouzelis et al., 2025; Skorokhodov et al., 2025). Another direction incorporates pretrained visual representations. VA-VAE (Yao et al., 2025) aligns VAE la-

tents with features from vision foundation models, whereas RAE (Zheng et al., 2026) uses frozen pretrained vision encoders to map images into high-dimensional semantic representations that serve directly as the diffusion latent space. LV-RAE (Liu et al., 2026) supplements these representations with low-level information, and PAE (Yue et al., 2026) explicitly regularizes latent spatial structure, continuity, and semantics.

Diffusion transformers and joint modeling. DiT (Peebles & Xie, 2023) replaces the convolutional U-Net architecture (Ho et al., 2020; Song et al., 2021) with a transformer backbone. LightningDiT (Yao et al., 2025) improves its architecture and training recipe, while DDT (Wang et al., 2026) separates semantic encoding from velocity decoding within the denoiser. Beyond denoiser design, REPA-E (Leng et al., 2025) utilizes the representation-alignment objective of REPA (Yu et al., 2025) to jointly tune a pretrained VAE and diffusion model. Unified Latents (Heek et al., 2026) jointly learns an encoder, a diffusion prior, and a diffusion decoder, followed by training a separate generative model on the frozen latents, and UNITE (Duggal et al., 2026) uses a shared backbone for image encoding and latent denoising. In language generation, ELF (Hu et al., 2026) projects discrete indices into embeddings and uses a shared network between continuous denoising and discrete decoding.

3 Motivating Joint Denoising and Decoding

In this section, we examine whether independently trained denoisers and decoders learn similar representations, and discuss the possible design options for unification.

Separately trained denoisers and decoders learn similar representations. In RAE (Zheng et al., 2026), the denoiser and decoder are usually independently trained for velocity prediction and image reconstruction. Despite being optimized for different objectives with or without conditioning, they operate on the same latent space. Inspired by prior work on comparing neural representations (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019; Huh et al., 2024), we compute layer-wise unbiased centered kernel alignment (unbiased CKA) (Song et al., 2012) between the denoiser and decoder on $10{,}000$ ImageNet validation images. Specifically, for each image, we collect the outputs of the denoiser's 12 backbone layers when denoising a corrupted latent and the decoder's 12 layers when decoding its clean counterpart, excluding the denoiser's output head. We then compute the unbiased CKA between the outputs of corresponding layers using all sampled images and average the resulting scores across noise timesteps. To control for trivial similarity arising from shared input or chance, we include random-initialization and sample-permutation baselines.

Figure 2 shows a clear depth-dependent pattern. The denoiser and decoder exhibit substantial representational alignment in early layers, while the similarity progressively decreases with depth. Furthermore, the observed alignment remains above both random-initialization and sample-permutation baselines throughout the network. Mutual kNN, CKNNA (Huh et al., 2024), and biased CKA exhibit qualitatively similar depth-dependent trends (Appendix B).

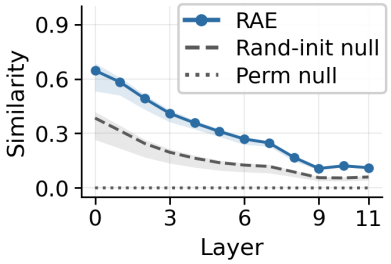


Figure 2: Unbiased CKA between the independently trained RAE denoiser and decoder.

The substantial alignment between independently trained denoisers and decoders suggests that the two tasks may not require entirely separate feature-processing pipelines, motivating a shared backbone with task-specific branches. Representational similarity alone, however, does not establish how much of the network can be shared or where the branches should begin; we examine the choice of branch point empirically in Section 5.

Optimizing the shared representation. Representational similarity does not imply direct reusability. We freeze the denoiser and only train an attached lightweight head consisting of two transformer blocks followed by a pixel output layer. As shown in Table 2, this variant (denoted as Stitched Decoder) performs poorly in reconstruction, achieving an rFID of 7.17 compared with 0.58 for the independently trained RAE decoder (denoted as RAE Dec-B). Thus, although the two

networks learn similar representations, the frozen denoiser features are not directly suitable for high-quality reconstruction with only a lightweight decoding branch, which suggests that the backbone itself needs to adapt to the reconstruction objective.

We explore another variant by unfreezing the denoiser and optimize the entire model, denoted as Optimized Jointly. Joint training reduces rFID to 0.77, approaching the reconstruction quality of the independently trained RAE decoder. It also substantially improves gFID from 15.36 to 4.75, surpassing the RAE Dec-B baseline. Note that in this setting, although attached to the denoiser backbone, the entire parameter and Gflops used for reconstruction is $\sim$ **50%** smaller than RAE's standalone decoder (48M 9.4 Gflops vs. 86M 22.2 Gflops). The gFID reported in Tab. 2 is computed by decoding latents generated by **the same frozen RAE denoiser** for all variants, and therefore reflects the ability of decoders to handle generated latents.

Table 2: Reconstruction and decoding quality under different optimization strategies. rFID denotes FID (Heusel et al., 2017) computed between original and reconstructed images.

Model	rFID ↓	gFID ↓
RAE Dec-B	0.58	7.51
Stitched Decoder	7.17	15.36
Optimized Jointly	0.77	4.75

Together, these results suggest that representational similarity makes backbone sharing plausible, but effective sharing requires the backbone to adapt to both tasks. This motivates our shared-backbone design, as introduced in the next section.

4 METHOD

Notation. Let $\boldsymbol{x} \sim p_{\text{data}}$ denote the image data, and $\boldsymbol{z}$ as its corresponding latent. $\boldsymbol{\epsilon} \sim p_{\text{noise}}(\boldsymbol{\epsilon})$ as a noise vector, and $t \sim \mathcal{T}$ as a timestep distributed in range $[0, 1]$. We construct the noised latent as

$$\boldsymbol{z}_t = (1-t)\boldsymbol{z} + t\boldsymbol{\epsilon}, \tag{1}$$

which denotes $\boldsymbol{z}_0$ as the clean latent, and $\boldsymbol{z}_1$ as the fully noised one. Following the practice of flow matching (Lipman et al., 2023; Ma et al., 2024), we define the velocity $\boldsymbol{v}$ as as the time-derivative of $\boldsymbol{z}_t$:

$$\boldsymbol{v} = \frac{d\boldsymbol{z}_t}{dt} = \boldsymbol{\epsilon} - \boldsymbol{z}. \tag{2}$$

Let $y \in \mathcal{C}$ denote the category, and $\varnothing$ as the null label. To allow a shared backbone to support both tasks within the same batch, we introduce a mode token $m \in \{m_{\text{diff}}, m_{\text{dec}}\}$, which explicitly specifies the task being performed. We respectively refer to the tuple of variables $\{t, y, m\}$ as condition $\boldsymbol{c}_{\text{diff}}$ or $\boldsymbol{c}_{\text{dec}}$ for brevity. As illustrated in Figure 3, the time and category condition in $\boldsymbol{c}_{\text{dec}}$ is fixed at $t = 0$ and $y = \varnothing$, indicating that the model must reconstruct the clean image without category information.

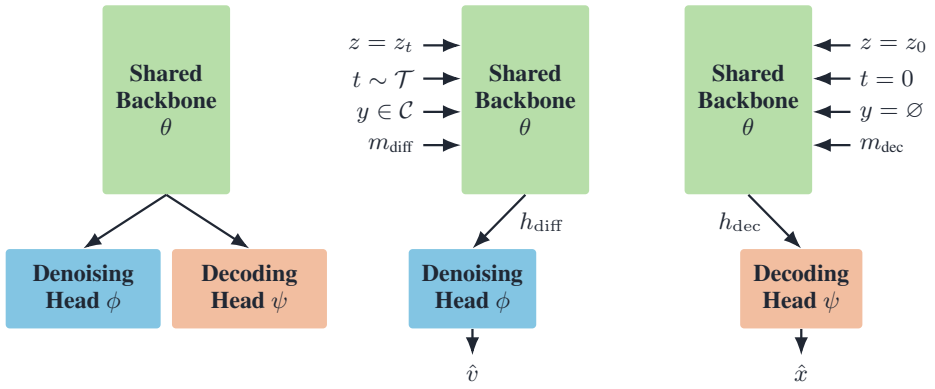
Architecture. We use E_θ, D_ϕ, and D_ψ to describe the function parameterized by θ, ϕ, and ψ. The backbone θ of DJT is the front portion of a DiT—a patch embedder followed by a stack of transformer blocks, which produces $h_{\text{diff}} = E_\theta(\boldsymbol{z}_t, \boldsymbol{c}_{\text{diff}})$. We use AdaLN-Zero (Peebles & Xie, 2023) to modulate the category, time and mode conditions. The denoising head ϕ is of two transformer blocks, together with a linear layer to predict the velocity $\hat{v} = D_\phi(h_{\text{diff}}, \boldsymbol{c}_{\text{diff}})$, which we also write as $v_{\theta,\phi}(\boldsymbol{z}_t, \boldsymbol{c}_{\text{diff}})$. Formally, we optimize the denoising path with

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\boldsymbol{c}_{\text{diff}}, \boldsymbol{z}} \| v_{\theta,\phi}(\boldsymbol{z}_t, \boldsymbol{c}_{\text{diff}}) - \boldsymbol{v} \|_2^2. \tag{3}$$

We add a decoding head in parallel while keeping the denoising path unchanged. The decoding head ψ is also of two transformer blocks, together with a linear layer and pixel unshuffle operation. It takes the backbone's output hidden state $h_{\text{dec}} = E_\theta(\boldsymbol{z}_0, \boldsymbol{c}_{\text{dec}})$ as input and decodes it into a reconstructed image $\hat{x} = D_\psi(h_{\text{dec}})$. Similarly, we can write the objective as

$$\mathcal{L}_{\text{dec}} = \mathbb{E}_{\boldsymbol{c}_{\text{dec}}, \boldsymbol{z}} \big[\ell_{\text{dec}} \big(x_{\theta,\psi}(\boldsymbol{z}_0, \boldsymbol{c}_{\text{dec}}), \boldsymbol{x} \big) \big]. \tag{4}$$

Following the common practice of RAE and VAEs, we instantiate ℓ_{dec} as a combination of the ℓ_1, the perceptual (Zhang et al., 2018; Weber et al., 2024), and adversarial losses (Goodfellow et al., 2014). See Appendix A for full details.



(a) Overall architecture (b) Denoising branch (c) Decoding branch

Figure 3: **Overview of DJT.** (a) Our unified architecture contains a shared backbone θ and two task-specific heads, ϕ and ψ, to support latent denoising and image reconstruction simultaneously. (b) In denoising mode, the backbone processes a noised latent z_t conditioned on timestep t, class label y, and mode token m_{diff}, producing h_{diff} for velocity prediction. (c) In decoding mode, it processes a clean latent z_0 with $t = 0$, $y = \varnothing$, and m_{dec}, producing h_{dec} for image reconstruction. Both objectives jointly optimize the shared backbone. At inference, DJT first iteratively denoises a latent and then switches to decoding mode to decode the latent into an image.

During training, we employ a Bernoulli router (Hu et al., 2026), to assign each sample in a batch to one of the two tasks. Specifically, each sample is independently assigned to the reconstruction task (condition $\boldsymbol{c}_{\text{dec}}$) with probability p and to the denoising task (condition $\boldsymbol{c}_{\text{diff}}$) with probability $1 - p$. Every processed sample first passes through the shared backbone to produce a hidden state, and is then routed to the corresponding branch based on its mode. As a result, gradients from both objectives jointly update the shared backbone during training, with the only additional hyperparameter p that determines the relative contribution of each task.

At inference time, the denoising branch is activated to function as a denoiser, iteratively refining sampled Gaussian noise into a clean latent $\hat{\boldsymbol{z}}_0$. Formally, for the given category label y, we start from a noise sample $\boldsymbol{z}_{t_0}$ at $t_0 = 1$, discretize $[1, 0]$ into N timesteps and apply the update as

$$\boldsymbol{z}_{t_{i+1}} = \boldsymbol{z}_{t_i} + (t_{i+1} - t_i) v_{\theta,\phi}(\boldsymbol{z}_{t_i}, t_i, y, m_{\text{diff}}). \tag{5}$$

The final state $\boldsymbol{z}_{t_N}$ at $t_N = 0$ is then used as the generated clean latent $\hat{\boldsymbol{z}}_0$. The decoding branch then function as a decoder, mapping the generated latent into an image $\hat{\boldsymbol{x}} = x_{\theta,\psi}(\hat{\boldsymbol{z}}_0, 0, \varnothing, m_{\text{dec}})$.

5 Experiment

5.1 Setup

Dataset and Evaluation. Experiments are conducted on ImageNet (Deng et al., 2009) with 256×256 image resolution, using the ADM protocol (Dhariwal & Nichol, 2021) for training and evaluation. We use the default Euler sampler and set the number of function evaluations (NFE) as 50. We evaluate generation quality on 50,000 generated images, and report the generation Fréchet Inception Distance (gFID) (Heusel et al., 2017), Inception Score (IS) (Salimans et al., 2016), precision, and recall (Kynkäänniemi et al., 2019).

Implementation. For the shared backbone, we use the DiT^{DH} transformer, following the architecture of RAE. For the DJT-S variant, the shared backbone is of 12 blocks with hidden size 384, while the DJT-XL has 28 blocks with hidden size 1152. Then the model branches into the 2-layer denoising and decoding heads, as described in Section 4. We train the model under `fp32` precision, set the global batch size to 1024, and adopt a learning rate scheduler that keeps the learning rate constant for the first 40 epochs and then decays it linearly until epoch 800. The image encoder is frozen throughout the training process for fair comparison. More details are included in Appendix A.

Table 3: **Comparison of class-conditional image generation** on ImageNet 256 × 256. Params count the unique parameters (denoiser+decoder) required by the complete generation pipeline used at inference time. Dim refers to the corresponding autoencoder's bottleneck dimension. * denotes that the encoder (DINOv2) is finetuned for reconstruction.

Method	Params	Dim	Epochs	Generation w/o guidance				Generation w/ guidance			
				gFID ↓	IS ↑	Prec. ↑	Rec. ↑	gFID ↓	IS ↑	Prec. ↑	Rec. ↑
DiT (Peebles & Xie, 2023)	675+49M	4	1400	9.62	121.5	0.67	0.67	2.27	278.2	0.83	0.57
SiT (Ma et al., 2024)	675+49M	4	1400	8.61	131.7	0.68	0.67	2.06	270.3	0.82	0.59
REPA (Yu et al., 2025)	675+49M	4	800	5.78	158.3	0.70	0.68	1.29	306.3	0.79	0.64
DDT (Wang et al., 2026)	675+49M	4	400	6.27	154.7	0.68	0.69	1.26	310.6	0.79	0.65
LightningDiT (Yao et al., 2025)	675+41M	32	800	2.17	205.6	0.77	0.65	1.35	295.3	0.79	0.65
REPA-E (Leng et al., 2025)	675+41M	32	800	1.70	217.3	0.77	0.66	1.15	304.0	0.79	0.66
PAE* (Yue et al., 2026)	675+309M	32	800	1.43	244.8	0.78	0.66	1.03	296.9	0.79	0.67
RiT (Zhang et al., 2026)	676+415M	384	800	1.45	231.6	0.81	0.62	1.14	299.7	0.80	0.63
RAE (Zheng et al., 2026)	839+415M	768	800	1.51	242.9	0.79	0.63	1.13	262.6	0.78	0.67
LV-RAE* (Liu et al., 2026)	839+415M	768	800	2.42	223.8	0.77	0.64	1.82	249.7	0.75	0.67
DJT	**870M**	768	800	**1.35**	234.6	0.79	0.65	**0.97**	264.7	0.77	0.66

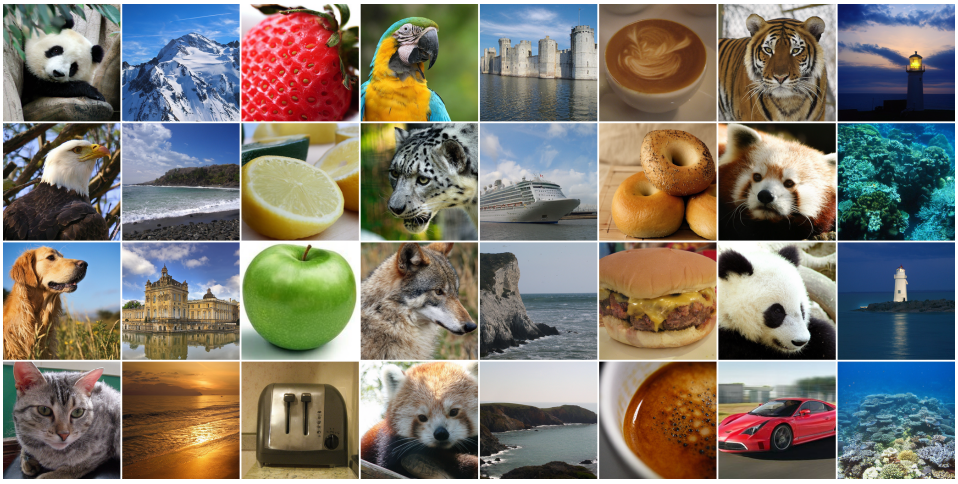


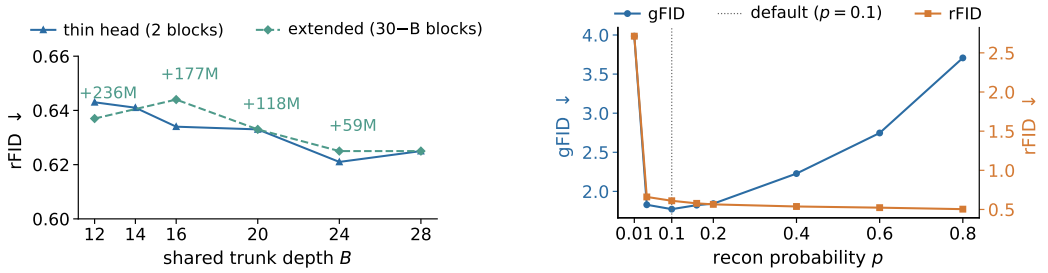
Figure 4: **Qualitative results.** Generated samples on ImageNet 256 × 256 using DJT-XL.

5.2 Main Result

Table 3 compares DJT-XL with existing methods after 800 epochs of joint training. DJT-XL achieves a gFID of **0.97** with guidance, the best result among the compared methods operating in representation latent spaces, surpassing RAE (1.13) and RiT (1.14). It also compares favorably with leading VAE-style methods: DJT-XL outperforms the end-to-end trained REPA-E (1.15) and achieves performance better than PAE (1.03), which explicitly optimizes the latent space. At the same time, it achieves the state-of-the art performance of gFID=**1.35** without guidance, surpassing all the compared methods. These results show that joint denoising and decoding can achieve competitive generation quality in a fixed, high-dimensional representation space despite the larger gap between encoded and generated latents documented in Table 1. We provide qualitative results in Figure 4, and more uncurated samples in Appendix E.

DJT-XL also offers a favorable balance between generation quality and model cost. The complete generation pipeline uses 870M unique parameters, approximately **31%** fewer than RAE's 1,254M and **11%** fewer than PAE's 984M. Meanwhile, DJT-XL achieves a relatively low decoding cost of 122.8 Gflops per forward pass, requiring less computation than most VAE-based methods. Higher efficiency can also be achieved by switching the decoder to smaller models such as DJT-S, which reduces the decoding calculation by more than 90% while maintaining comparable performance. See Section 6 for detailed comparison.

5.3 Design Choices



(a) Branch point ablation.

(b) Routing probability ablation.

Figure 5: **Ablations on DJT design.** (a) Effect of the branch point. The thin-head variant fixes the reconstruction head to two blocks while varying the branch point at B-th layer. For the extended-head variant, the total decoding depth is fixed to 30 blocks: decreasing the shared backbone depth B correspondingly increases the reconstruction head. Annotations indicate the additional parameters relative to B=28. (b) Effect of the routing probability p on generation and reconstruction quality.

Where should the two tasks branch? We first examine how the branch point between shared backbone and task-specific heads affects reconstruction. To obtain a general guidance, we conduct this ablation on DJT-XL, whose denoising and decoding paths each contain 30 transformer blocks. We vary the branch point B, while extending the reconstruction-specific head so that the complete decoding path always remains 30 blocks deep. Thus, decreasing B redistributes blocks from the shared backbone to the reconstruction-specific head without changing the total decoding depth, but increases the number of unique parameters. All the variants are trained for 100 epochs.

Interestingly, increasing reconstruction-specific capacity does not improve reconstruction. As shown by the extended-head curve in Figure 5a, moving the branch point from B=28 to B=12 adds 236M unique parameters, yet results in a higher (worse) rFID. In other words, allocating blocks to the shared backbone is more effective than allocating the same blocks exclusively to reconstruction-specific head.

We further fix the reconstruction head to only two blocks and vary the depth of the shared backbone. Reconstruction generally improves as the shared backbone becomes deeper, with the gain largely saturating around B=24. Together, these results show that, *under joint training, allocating capacity to the shared backbone benefits more than increasing reconstruction-specific capacity.* We therefore adopt B=28 in DJT-XL, leaving only a lightweight two-block reconstruction head while maximizing parameter sharing. Note that the gFID remains approximately the same, as the denoising branch remains unchanged during this sweep.

Routing probability. We next study the effect of reconstruction probability p, which controls how training budgets are allocated between the denoising and reconstruction objectives. Figure 5b reports the results after 100 epochs training, and shows the trade-off clearly. Increasing p consistently improves reconstruction, but eventually degrades generation as fewer samples are assigned to denoising. In contrast, an extremely small reconstruction probability (p=0.01) is insufficient to train the decoding path effectively. We use p=0.1 by default, which preserves the best generation performance while already capturing most of the reconstruction gain.

6 Discussion

Our unified design, which contains a shared backbone with lightweight task-specific heads and joint training, brings two complementary benefits to the generation pipeline. In this section, we first detail these two benefits, then show that the same design can be extended beyond RAE, and finally provide one possible explanation for our better performance. Unless otherwise stated, all experiments in this section use DJT-S trained for 100 epochs.

Table 4: **Comparison of generation quality and efficiency.** All pipelines use the same DiT^{DH}-S (Zheng et al., 2026) denoising architecture and are evaluated on 50K samples without guidance.

Pipeline	gFID ↓	Params	Dec. Gflops
RAE + Dec-B	7.41	283M	22.2
RAE + Dec-XL	5.77	612M	106.7
DJT	**5.02**	**212M**	**9.4**

A more efficient generation pipeline. By reusing the same backbone for denoising and decoding, DJT substantially reduces the number of unique parameters required by the complete generation pipeline and enables a lightweight, computationally efficient decoding path. While Table 3 demonstrates this advantage at the XL scale, we further examine whether it persists at a smaller scale. We compare DJT-S with RAE pipelines using the same denoising architecture but different independently trained RAE decoders. As shown in Table 4, DJT-S achieves a 50K unguided gFID of 5.02 with only 212M unique parameters and 9.4 decoding Gflops. In comparison, the RAE pipeline with Dec-B uses more parameters and more than twice the decoding computation, yet obtains a substantially worse gFID of 7.41. Scaling the decoder to Dec-XL improves the gFID to 5.77, but requires nearly $3\times$ the total parameters and over $10\times$ the decoding computation of DJT-S. These results show that the efficiency gains of the unified design do not come at the cost of generation quality, yielding a better balance between generation quality, parameter count, and decoding cost.

Joint training strengthens representation alignment. We compute the layer-wise representational similarity of DJT's shared backbone under denoising and reconstruction modes following the same protocol as in Section 3. As shown in Figure 6, DJT maintains substantially higher similarity than RAE across all layers, indicating that joint optimization reinforces cross-task alignment. It also surpasses the randomly initialized model with identical design, which validates its co-optimization beyond simple parameter sharing. To verify whether this similarity also reflects functional compatibility rather than simple statistical alignment, we take one step further and conduct activation patching between the two modes following Zhang & Nanda (2024). It shows that early-layer representations can be exchanged with little effect on the output, whereas deeper-layer exchanges lead to rapid degradation, consistent with the depth-dependent similarity trend. See Appendix C for details.

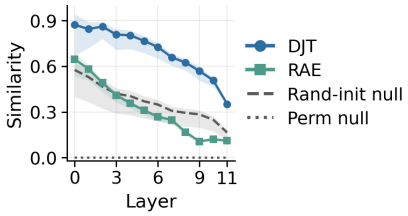


Figure 6: Layer-wise Unbiased CKA between denoising and reconstruction modes in DJT.

Joint training goes beyond parameter sharing. Beyond the parameter and computation efficiency enabled by sharing, DJT also brings denoising and reconstruction into the same optimization framework. Through the shared backbone, the denoising objective can directly shape the representations that are subsequently used for decoding. This captures a key motivation behind end-to-end optimization of the generation pipeline (Leng et al., 2025; Heek et al., 2026): the generative objective can influence components beyond the denoiser itself. To examine whether this influence translates into better generation performance, we remove the denoising task and train the same backbone and decoding head solely for reconstruction, as shown in Figure 7. Using DJT-S as the denoiser, this variant (DJT w/o Gen) obtains a 50K gFID of 6.58, compared with 5.02 for DJT. The 1.56 gFID improvement shows that unification provides more than parameter and computation efficiency, it also allows the denoising objective to shape the shared backbone that eventually improve the final generation quality.

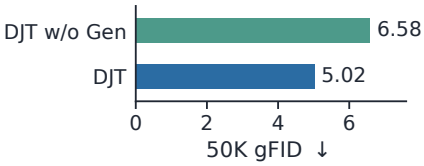


Figure 7: Effect of joint training on decoding generated latents. DJT w/o Gen shares the decoding architecture of DJT-S but is trained only for reconstruction. Both decode the same 50K unguided latents generated by DJT-S.

Table 5: **Generation results in the VAE space.** gFID evaluates the complete generation pipeline. Since both settings use the identical denoiser architecture, Params and Gflops are reported only for the latent-to-image decoding path.

Pipeline	gFID ↓	Params	Gflops
DiT + VA-VAE	16.00	41M	126.1
DJT	11.85	48M	9.4

Extension to VAE Space. The proposed architecture is not specific to the high-dimensional RAE space, as it is latent-agnostic and designed to forge a shared optimization dynamic. To examine its broader applicability, we directly apply the same joint denoising-and-decoding formulation on latents produced by VA-VAE (Yao et al., 2025). As shown in Table 5, the resulting DJT-S decoding path, with a parameter count comparable to that of the original VA-VAE decoder (48M vs. 41M), yet substantially reduces the decoding cost from 126.1 to 9.4 Gflops. Furthermore, our unified design improves gFID from 16.00 to 11.85. These results show that the unified design extends beyond RAE, improving generation quality while substantially reducing decoding computation.

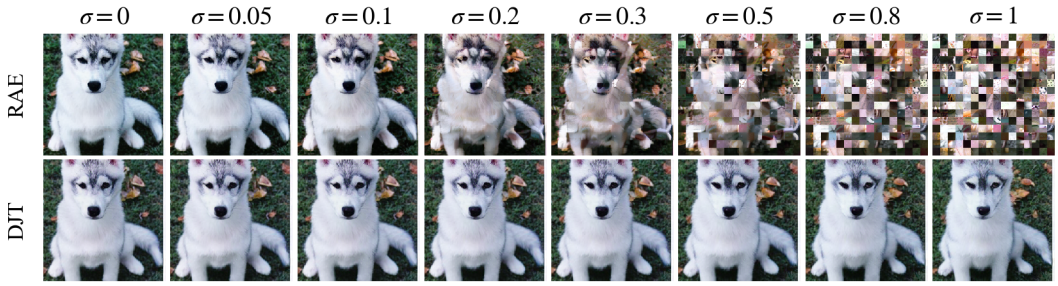


Figure 8: **Qualitative comparison of decoder robustness.** In each column, we show RAE and DJT's decoded results from latents perturbed with strength σ.

Robustness to distribution shift. As the train-test mismatch affects the generation quality (Razavi et al., 2019; Tschannen et al., 2023), a decoder that is more robust to this deviation should help improve the final performance. To verify this argument, we perturb the latents with Gaussian noise ϵ at strength σ, as $z' = z + \sigma\epsilon$, and visualize the decoded results. As shown in Figure 8, the RAE decoder degrades rapidly as the perturbation strength increases, whereas DJT exhibits substantially smaller quality degradation and remains robust over the full range of noise levels. These results suggest that joint training enables the decoder to better handle deviations from the encoded-latent distribution. We hypothesize that this robustness arises because the shared backbone is also optimized by the denoising objective, which requires processing latents across a broad range of signal-noise ratios. For decoding, this improves its ability to extract useful information from imperfect latents, thereby enhancing robustness. Although Gaussian perturbations do not exactly reproduce the distribution shift between encoded and generated latents, the improved robustness provides a plausible explanation for DJT's better generation quality.

7 Conclusion

We proposed DJT, a Denoise-decode Joint Transformer which simultaneously learns latent denoising and image reconstruction through a shared backbone and two task-specific heads. The analysis shows that joint training reshaped the shared representations and improved robustness of the decoder to distribution shift between encoded and generated latents. Experiments within the VA-VAE latent space validated that joint training extends beyond high-dimensional representation spaces. On ImageNet 256×256, DJT-XL achieved a gFID of 0.97 using fewer parameters than the RAE baseline. These findings demonstrated that joint denoising and decoding offers an effective approach to improving image generation by making the decoder more robust to latent distribution shift.

AI USE STATEMENT

In this work, we use generative AI tools to improve readability, and provide feedback on research methodology.

We **do not** use generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, assist with translation, implement methods, or interpret results. Cleaning and reformatting datasets and supporting qualitative and thematic data analysis are not applicable to this work.

We have reviewed all AI-assisted work and manually verified the technical accuracy, claims, and references in the manuscript, as well as the correctness of any AI-assisted code. As human authors, we assume full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 4 specifies the DJT architecture, task routing, training objectives, and sampling procedure. Section 5 describes the experimental setup, baselines, and evaluation metrics. Appendix A provides details on ImageNet preprocessing, training details, model configurations, and evaluation settings. The protocols for the representation comparisons and activation-patching experiments are given in Appendices B and C, respectively. We also provide the core code implementation in the supplementary material.

REFERENCES

Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. Softvq-vae: Efficient 1-dimensional continuous tokenizer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 28358–28370, 2025a.

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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28*, 2025b.

Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. DC-AE 1.5: Accelerating diffusion model convergence with structured latent space. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25*, pp. 1–10, 2025c.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 20-25 June 2009, Miami, Florida, USA*, pp. 248–255, 2009.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, December 6-14, 2021, virtual*, pp. 8780–8794, 2021.

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.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2672–2680, 2014.

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

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, pp. 20617–20642, 2024.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10-15*, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, 9-15 June 2019, Long Beach, California, USA*, pp. 3519–3529, 2019.

Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. EQ-VAE: equivariance regularized latent space for improved generative image modeling. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19*, 2025.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3929–3938, 2019.

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 *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 18262–18272, 2025.

Tianhong Li and Kaiming He. Back to Basics: Let Denoising Generative Models Denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5*, 2023.

Siyu Liu, Chujie Qin, Hubery Yin, Qixin Yan, Zheng-Peng Duan, Chen Li, Jing Lyu, Chun-Le Guo, and Chongyi Li. Improving reconstruction of representation autoencoder. *arXiv preprint arXiv:2602.08620*, 2026.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

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 *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXVII*, pp. 23–40, 2024.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 5732–5741, 2018.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido 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. *Trans. Mach. Learn. Res.*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4172–4182, 2023.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6076–6085, 2017.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9*, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 10674–10685, 2022.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2226–2234, 2016.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved Baselines with Representation Autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19*, 2025.

Le Song, Alexander J. Smola, Arthur Gretton, Justin Bedo, and Karsten M. Borgwardt. Feature selection via dependence maximization. *J. Mach. Learn. Res.*, pp. 1393–1434, 2012.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7*, 2021.

Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, Rob Fergus, Naila Murray, Marjan Ghazvininejad, Mike Lewis, Nicolas Ballas, Amir Bar, Michael Rabbat, Jakob Verbeek, Luke Zettlemoyer, Koustuv

Sinha, Yann LeCun, and Saining Xie. Beyond Language Modeling: An Exploration of Multimodal Pretraining. *arXiv preprint arXiv:2603.03276*, 2026.

Michael Tschannen, Cian Eastwood, and Fabian Mentzer. GIVT: Generative Infinite-Vocabulary Transformers. *arXiv preprint arXiv:2312.02116*, 2023.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: decoupled diffusion transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pp. 40633–40642, 2026.

Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. MaskBit: Embedding-free Image Generation via Bit Tokens. *arXiv preprint arXiv:2409.16211*, 2024.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 15703–15712, 2025.

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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28*, 2025.

Zhengrong Yue, Taihang Hu, Mengting Chen, Haiyu Zhang, Zihao Pan, Tao Liu, Zikang Wang, Jinsong Lan, Xiaoyong Zhu, Bo Zheng, et al. What matters for diffusion-friendly latent manifold? prior-aligned autoencoders for latent diffusion. *arXiv preprint arXiv:2605.07915*, 2026.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11*, 2024.

Le Zhang, Ning Mang, and Aishwarya Agrawal. Rit: Vanilla diffusion transformers suffice in representation space. *arXiv preprint arXiv:2605.21981*, 2026.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *arXiv preprint arXiv:1801.03924*, 2018.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pp. 35791–35820, 2026.

Yifan Zhou, Zeqi Xiao, Shuai Yang, and Xingang Pan. Alias-free latent diffusion models: Improving fractional shift equivariance of diffusion latent space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 34–44, 2025.

A Experiment Details

Data and latent space. All experiments use ImageNet-1K (Deng et al., 2009) at 256×256 resolution with the ADM preprocessing pipeline (Dhariwal & Nichol, 2021): every training image is center-cropped to 256×256 and randomly flipped horizontally. Following RAE (Zheng et al., 2026), the latent space is produced by a frozen DINOv2-B encoder with registers (Oquab et al., 2024; Darcet et al., 2024), which yields $16 \times 16 = 256$ patch tokens of 768 channels.

Model architecture. The shared backbone follows DiT^{DH} (Zheng et al., 2026; Wang et al., 2026), with an additional mode embedding in the AdaLN conditioning:

$$\boldsymbol{c} = \mathrm{SiLU}\big(\mathrm{Emb}(t) + \mathrm{Emb}(y)\big) + \mathrm{Emb}(m), \tag{6}$$

where m denotes the mode token introduced in Section 4. The mode embedding is initialized to zero. In reconstruction mode, the null label is used and t is fixed to 0. The time embedding is re-injected into the backbone output only in denoising mode, so the decoding head receives a time-independent representation.

The denoising head follows the shallow-but-wide head of DiT^{DH}. The decoding head is a lightweight ViT with pixel-unshuffle operation. Its width and feed-forward dimension follow the RAE decoder convention at the corresponding scale ($768/3072$ for DJT-S and $1152/4096$ for DJT-XL), while its depth stays at two blocks.

Training configuration. We follow RAE for timestep sampling and latent noise augmentation. Before latent normalization, each reconstruction latent is perturbed as $\boldsymbol{z} + \sigma\boldsymbol{\epsilon}$, where $\sigma \sim \mathcal{U}(0, \tau)$ is drawn independently per sample and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$.

The reconstruction loss is

$$\ell_{\text{dec}} = \ell_1 + \ell_{\text{perc}} + \lambda_{\text{adv}}\, a\, \ell_{\text{adv}}, \tag{7}$$

where ℓ_1 is the pixelwise ℓ_1 reconstruction loss and ℓ_{perc} is the weighted average of VGG16 LPIPS (Zhang et al., 2018) and a perceptual loss computed with a frozen ConvNeXt-S, with weights 1.0 and 0.1, respectively, following MaskBit (Weber et al., 2024). The adversarial loss ℓ_{adv} uses a ViT-S/8 discriminator. Following RAE, we set the base weight $\lambda_{\text{adv}} = 0.75$ and use an adaptive weight a capped at 10^4. More training hyperparameters are listed in Table 6.

Evaluation protocol. For generation, we use EMA weights and generate $50{,}000$ images with 50 images per class. Reconstruction quality (rFID) is evaluated on $50{,}000$ ImageNet validation images using the same InceptionV3 feature extractor.

For guided generation, we follow RAE's implementation of autoguidance (Karras et al., 2024) over the full interval $t \in [0, 1]$. The auxiliary models for DJT-S and DJT-XL are DJT-S checkpoints from epochs 20 and 14, respectively. Guidance scales are listed in Table 6.

B Representation Analysis Protocol and Supplementary Results

B.1 Evaluation Protocol

Models. For RAE, we use DiT^{DH}-S, trained for 100 epochs with the official codebase, as the denoiser and the released ViT-B as the decoder. As shown in Figure 9, we compare the outputs of their 12 corresponding layers (0–11), excluding the denoiser's shallow wide head to match the decoder's 12-layer architecture. For our unified model, we use DJT-S trained for 100 epochs and compare the outputs at each of the 12 shared backbone layers under denoising and reconstruction modes, excluding both task-specific heads.

Implementation. In both RAE and DJT, the denoiser and the decoder are compared over the same number of layers, $N = 12$; for DJT, they correspond to the shared backbone in denoising and reconstruction modes, respectively. Given an image x_i, the frozen DINOv2 encoder produces its clean latent $\mathbf{z}_i$. The decoder takes $\mathbf{z}_i$ directly as input, and we record the outputs of its N Transformer blocks as $\{\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,N-1}\}$. The input to the denoiser instead depends on the timestep t: at timestep t, it receives the noisy latent

$$\mathbf{z}_{i,t} = (1-t)\mathbf{z}_i + t\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, I), \tag{8}$$

and we record the outputs of its N Transformer blocks as $\{\mathbf{h}'_{i,0}(t), \ldots, \mathbf{h}'_{i,N-1}(t)\}$. For DJT, both inputs are fed to the same backbone together with the corresponding mode token. Each block output is mean-pooled over tokens, giving one representation vector per image and layer.

We repeat this procedure on 10,000 ImageNet validation images. Given a similarity metric M, the similarity at layer j and timestep t is computed between the representations of all images from the

Table 6: Experiment details for DJT on ImageNet-1K at 256×256. The two variants share the same latent space, objective, optimizer, and sampler.

Setting	DJT-S	DJT-XL
Latent space		
Encoder (frozen)	DINOv2-B	DINOv2-B
Encoder input resolution	224×224	224×224
Latent size	$16 \times 16 \times 768$	$16 \times 16 \times 768$
Sequence length	256	256
Architecture		
Shared backbone depth B	12	28
Shared backbone width	384	1152
Backbone attention heads	6	16
Denoising head depth / width	2 / 2048	2 / 2048
Denoising head attention heads	16	16
Decoding head depth / width	2 / 768	2 / 1152
Decoding head patch size	16	16
Params (shared backbone)	33.1M	673.7M
Params (denoising head)	163.4M	165.0M
Params (decoding head)	15.3M	32.0M
Params (total)	211.7M	870.7M
Training		
Epochs	100	800
Global batch size	1024	1024
Steps per epoch	1,251	1,251
Precision	`fp32`	`fp32`
Optimizer	AdamW	AdamW
(β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)
Weight decay	0	0
Peak learning rate	2×10^{-4}	2×10^{-4}
LR schedule target at epoch 800	2×10^{-5}	2×10^{-5}
LR schedule	constant for 40 epochs, then linear decay until epoch 800	
Gradient clipping	1.0	1.0
EMA decay	0.9995	0.9995
Reconstruction routing probability p	0.1	0.1
Class-dropout probability	0.1	0.1
Latent noise augmentation τ	0.8	0.8
Timestep distribution	logit-normal$(0, 1)$, shift $s \approx 6.93$	
Perceptual weights (LPIPS / ConvNeXt-S)	1.0 / 0.1	1.0 / 0.1
Adversarial base weight λ_{adv}	0.75	0.75
Adaptive adversarial weight cap	10^4	10^4
Discriminator	frozen DINO ViT-S/8, spectral-norm heads, hinge loss	
Sampling and evaluation		
Sampler	Euler ODE	Euler ODE
NFE	50	50
Samples for gFID	50,000	50,000
Guidance (guided evaluation)	autoguidance, $w = 1.81$	autoguidance, $w = 1.44$

two models:

$$S_j(t) = M\big(\{\mathbf{h}_{i,j}\}_{i=1}^{10,000},\ \{\mathbf{h}'_{i,j}(t)\}_{i=1}^{10,000}\big). \tag{9}$$

Each timestep therefore yields one layer-wise similarity curve. We use $t \in \mathcal{T} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, which gives five curves; the reported curve is their mean, and the shaded region spans their minimum and maximum.

Baselines. we introduce two null baselines: (1) a *random-initialization baseline* that replaces the trained denoiser with a randomly initialized network of the same architecture, measuring the similarity attributable solely to architectural inductive biases and similar inputs; and (2) a *permutation baseline* that pairs representations derived from different underlying images and under different tasks, establishing the chance-level lower bound.

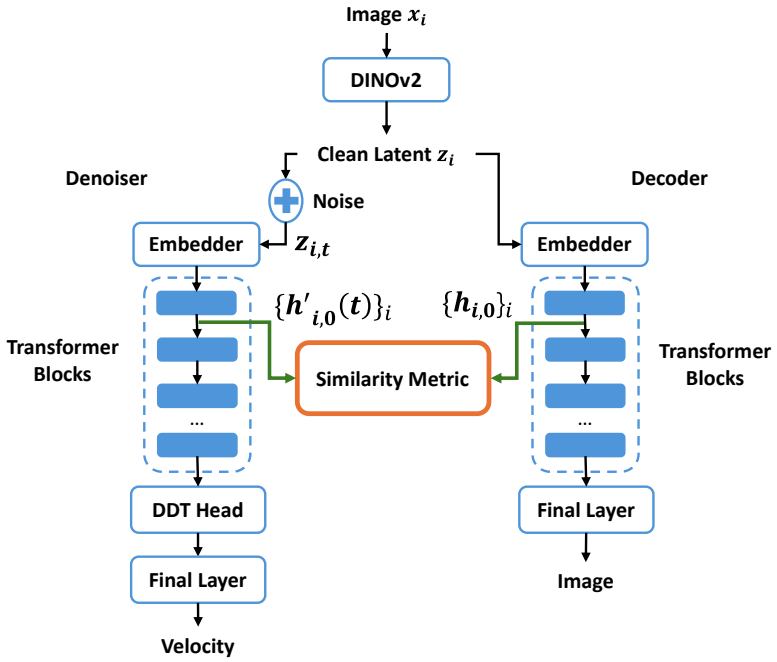


Figure 9: Illustration of the layer-wise similarity computation for RAE, using the first layer as an example. The denoiser receives the noisy latent and the decoder receives the clean latent of the same image. Green lines mark the outputs of the two layers being compared; the same computation is applied to all 12 pairs of corresponding layers. For DJT, the denoiser and decoder are replaced by the shared backbone in denoising and reconstruction modes, respectively.

B.2 Supplementary Results

The main text measures layer-wise similarity with Unbiased CKA. To check that the conclusions do not depend on this choice of metric, we repeat both analyses with three additional metrics: Mutual kNN, CKNNA, and biased CKA, where Mutual kNN and CKNNA use $k = 10$ nearest neighbors.

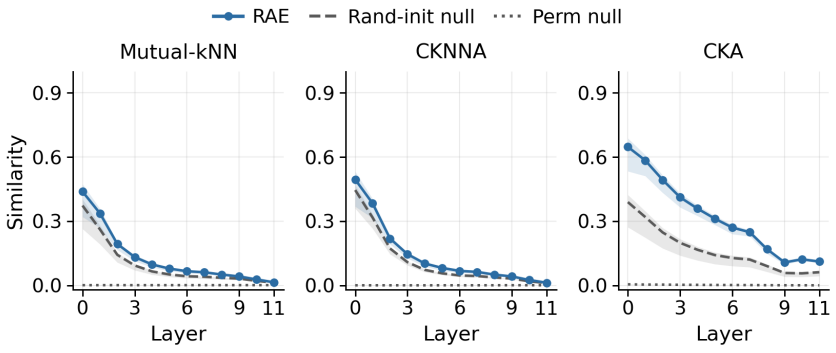


Figure 10: Layer-wise representational similarity between the independently trained RAE denoiser backbone and decoder, measured by Mutual kNN, CKNNA, and biased CKA.

Independently trained models. As shown in Figure 10, all three metrics reproduce the pattern that Unbiased CKA shows in Section 3: the similarity between the RAE denoiser and decoder is high in the early layers, decreases with depth, and stays above both null baselines at every layer.

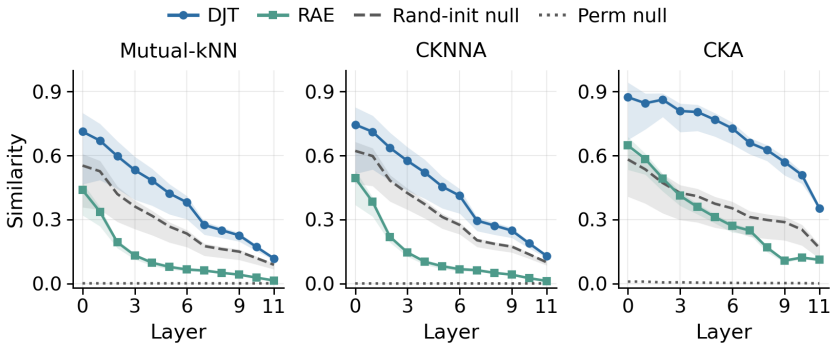
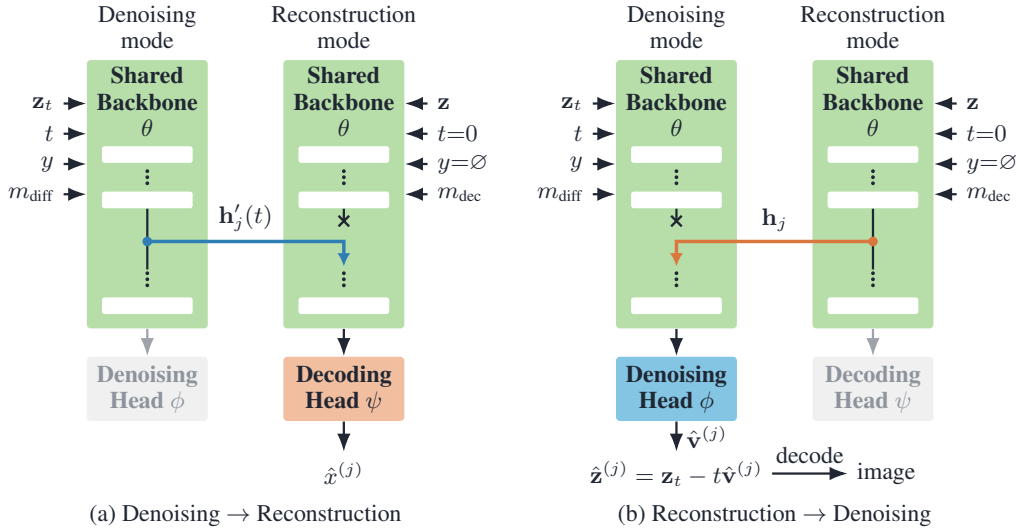


Figure 11: Layer-wise representational similarity between the denoising and reconstruction modes of DJT, measured by Mutual kNN, CKNNA, and biased CKA. The RAE curve from Figure 10 is included for reference; both null baselines are computed for DJT.

Jointly trained backbone. The DJT result in Section 6 likewise holds under all three metrics (Figure 11): the DJT curve lies above the RAE curve and both null baselines at every layer, and still decreases with depth.

C Functional Compatibility via Activation Patching

Section 6 shows that the shared backbone of DJT produces similar representations in its two modes. A high similarity score, however, does not guarantee functional compatibility: replacing the activations of one mode with those of the other may still change the output. We test this directly by patching activations between the two modes, one layer at a time.



(a) Denoising → Reconstruction (b) Reconstruction → Denoising

Figure 12: Activation patching between the two modes of DJT, illustrated for layer j. In each panel, the two columns are forward passes of the same model in denoising mode and reconstruction mode. The cross marks the replaced output of layer j, and the heads shown in gray are not used.

Implementation. We use the same DJT-S model as in Appendix B.1. Given an image x, the frozen DINOv2 encoder produces its clean latent $\mathbf{z}$. In reconstruction mode, the backbone receives $\mathbf{z}$ directly; in denoising mode, it receives the noisy latent $\mathbf{z}_t = (1-t)\mathbf{z}+t\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. We record the outputs of the 12 Transformer blocks of the shared backbone as $\{\mathbf{h}_0, \ldots, \mathbf{h}_{11}\}$ in reconstruction mode and $\{\mathbf{h}'_0(t), \ldots, \mathbf{h}'_{11}(t)\}$ in denoising mode. Without patching, the reconstruction-mode

forward pass outputs the reconstructed image $\hat{x}$, and the denoising-mode forward pass outputs the velocity prediction $\hat{\mathbf{v}}$, from which the clean latent is estimated as $\hat{\mathbf{z}} = \mathbf{z}_t - t\hat{\mathbf{v}}$.

To patch layer j, we replace its output in one mode with the recorded output of the same layer in the other mode and let the forward pass continue from there. Each of the 12 layers is patched separately, in two directions (Figure 12):

Denoising → Reconstruction. In the reconstruction-mode forward pass, we replace $\mathbf{h}_j$ with $\mathbf{h}'_j(t)$, which gives the patched reconstruction $\hat{x}^{(j)}$. Figure 13 shows $\hat{x}$ and $\hat{x}^{(0)}, \ldots, \hat{x}^{(11)}$.

Reconstruction → Denoising. In the denoising-mode forward pass, we replace $\mathbf{h}'_j(t)$ with $\mathbf{h}_j$, which gives the patched velocity prediction $\hat{\mathbf{v}}^{(j)}$ and the estimate $\hat{\mathbf{z}}^{(j)} = \mathbf{z}_t - t\hat{\mathbf{v}}^{(j)}$. For visualization, each estimate is decoded into an image by DJT in reconstruction mode, without patching. Figure 14 shows the images decoded from $\hat{\mathbf{z}}$ and $\hat{\mathbf{z}}^{(0)}, \ldots, \hat{\mathbf{z}}^{(11)}$.



Figure 13: Denoising → Reconstruction activation patching on an ImageNet validation image. The first image ("clean") is the unpatched reconstruction $\hat{x}$, and the image labeled Layer j is the patched reconstruction $\hat{x}^{(j)}$.

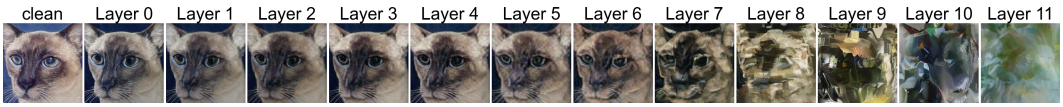


Figure 14: Reconstruction → Denoising activation patching on the same image as Figure 13. The first image ("clean") is decoded from the unpatched estimate $\hat{\mathbf{z}}$, and the image labeled Layer j is decoded from the patched estimate $\hat{\mathbf{z}}^{(j)}$.

The effect of patching grows with depth. In Denoising → Reconstruction (Figure 13), patching at early layers preserves the structure of the reconstruction and changes only its colors and textures. As the depth of the patched layer increases, colors desaturate, the image collapses to near-grayscale output with dense regular artifacts, and finally the original content is almost entirely lost. Reconstruction → Denoising (Figure 14) exhibits the same trend but with a different failure mode: rather than collapsing to grayscale, the decoded image degenerates into colorful abstract patterns bearing no resemblance to the original image.

Functional compatibility follows representational similarity. In layers where the two modes are highly similar (Section 6), patching leaves the overall structure of the output unchanged, so the image remains recognizable; as the similarity decreases, the effect of patching grows and the image becomes increasingly unrecognizable. The shallow layers of the shared backbone thus hold representations that both tasks can use, whereas the deep layers are already task-specific before the heads and specialize differently for each task, as their distinct failure modes suggest.

D LIMITATION

As shown in Table 7, the reconstruction metrics of our model, while outperform RAE, are modest compared to those of modern VAEs, but the original RAE decoder exhibits the same limitation. This is due to the use of DINOv2 features as the latent representation: they are highly structured and semantic, but of larger patch size and lack low-level details needed for faithful reconstruction. Specifically, a modern VAE (Rombach et al., 2021) usually uses patch size of 4×4 or 8×8, tuned with reconstruction objectives, while representational encoders use 14×14 or 16×16, optimized with semantic-aware objectives. A promising direction is to enrich the latent space with more low-level information. For example, RAEv2 (Singh et al., 2026) combines features from multiple DINOv2

Table 7: Reconstruction performance of RAE and DJT.

Decoder	rFID↓	PSNR↑	SSIM↑
RAE Dec-B	0.58	19.11	0.49
RAE Dec-XL	0.49	18.86	0.48
DJT-S	0.77	18.09	0.45
DJT-XL	0.49	19.47	0.52

layers rather than using the output of a single layer. We leave the exploration of such representations to future work.

E More Samples

We provide more qualitative samples in Figure 15.

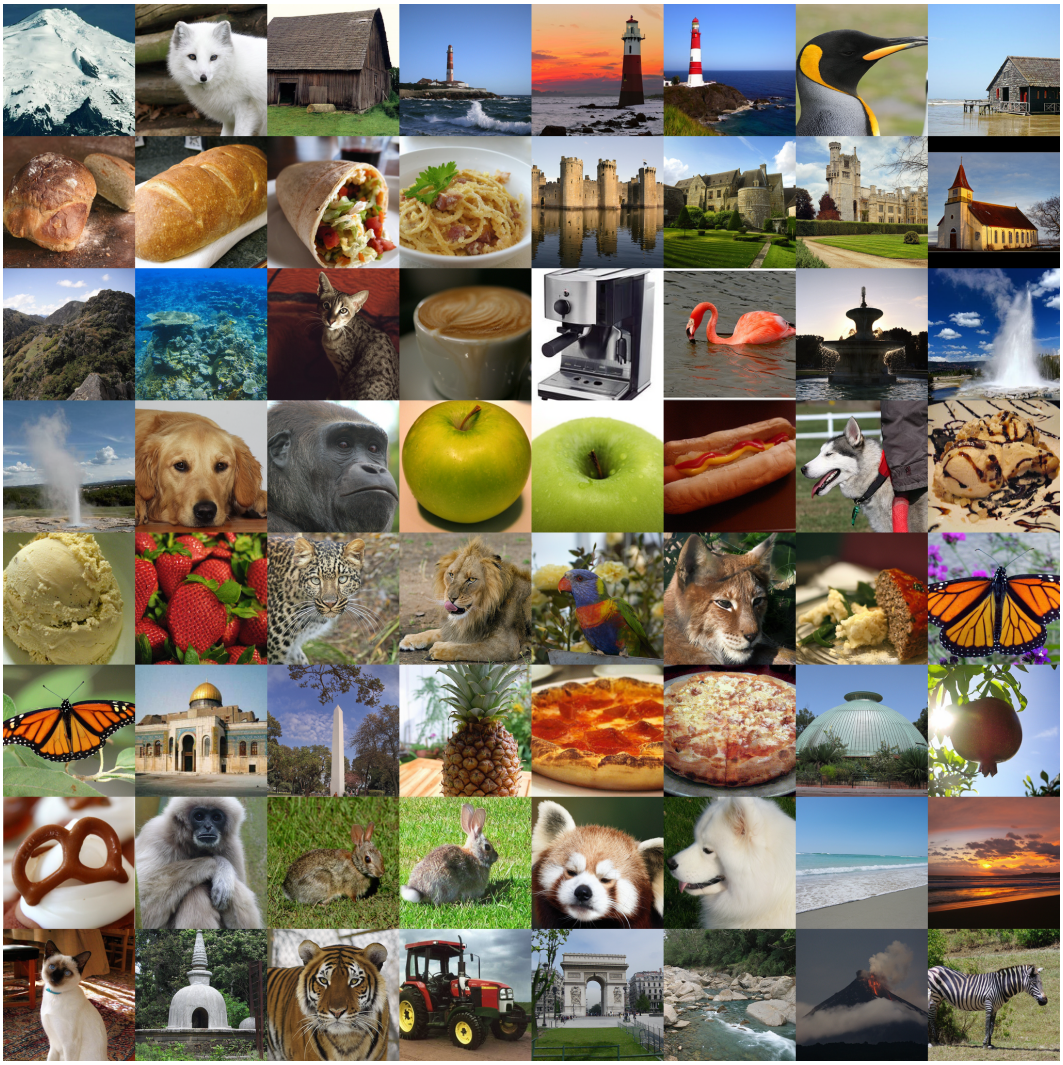


Figure 15: Uncurated generation results of DJT-XL.