

Representation-Augmented Diffusions

Farrin Marouf Sofian^{1,*}, Kushagra Pandey^{1,*}, Stephan Mandt¹

¹University of California, Irvine

*Equal contribution

Abstract

Diffusion models achieve state-of-the-art sample quality but lack an inherent mechanism to learn explicit semantic representations of the data. We propose *Representation-Augmented Diffusions* (RAD), a new class of diffusion models that augments the denoising process with semantic representations as latent variables. We derive a variational lower bound for this model that recovers representation alignment methods such as REPA as special cases and motivates a practical training objective. Interestingly, our framework reveals a simple design choice that existing representation alignment methods leave unused: *feeding the aligned representation back into the denoiser*. This idea leads to order-of-magnitude reductions in the training iterations and sampling steps needed to reach a given sample quality, along with improved data efficiency, over representation alignment. For instance, on class-conditional ImageNet 256×256 without classifier-free guidance, RAD matches the FID of REPA (trained for 4M iterations) with $27\times$ fewer training iterations while preserving discriminative properties, demonstrating that strong representations can be learned without compromising sample quality. RAD is also particularly effective with few sampling steps, and transfers to text-conditioned image generation. Overall, RAD emerges as a strong alternative for efficiently training diffusion models.

(a) ImageNet



(b) Text-to-image



(c) Training convergence

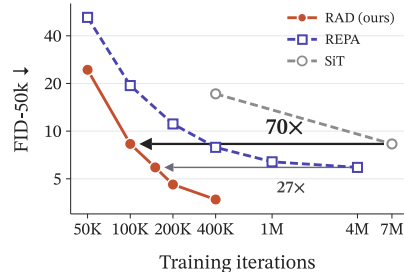


Figure 1: Representation-Augmented Diffusions (RAD). (a) ImageNet 256×256 samples. (b) Text-to-image samples. (c) Training convergence of XL/2 models on ImageNet: RAD matches the FID of SiT with $70\times$ and of REPA with $27\times$ fewer iterations.

Correspondence: Farrin Marouf Sofian at fmroufs@uci.edu

1 Introduction

Diffusion models [15, 46] and related families [1, 28] achieve state-of-the-art sample quality by iteratively converting noise into data. Beyond sample quality, however, many downstream tasks need compact representations of the data; world models, for instance, plan and act in a learned latent state rather than in the pixel space [2, 12, 32, 65]. This raises a natural question: *Does denoising also yield useful representations of the data?* The answer is unclear for two reasons. First, diffusion models are trained only to denoise, and although their features carry semantic information [54], they lag well behind those of self-supervised encoders [59]. Second, unlike GANs and VAEs [11, 20], which generate from a semantic latent space by construction, diffusion models lack such a compact latent space as every intermediate latent is a noisier version of the data. Moreover, semantic features stay implicit within the denoiser.

A recent line of work based on representation alignment addresses the first problem by proposing to align the representations of an intermediate layer in the denoiser with those of a frozen pretrained encoder [59] or toward the model’s own deeper features [17]. These methods show that good representations substantially accelerate the training of state-of-the-art diffusion transformers [31, 37]. However, the representation serves only as a training-time regularizer and is discarded at generation, which leaves the second problem unresolved. REG [53] denoises a semantic token jointly with the image latents so that it guides generation, and REED [48] formulates representation guidance variationally but drops the explicit conditioning in practice (App. A.3). Conversely, another line of work adds a latent variable that steers generation by combining diffusion models with auxiliary generative models such as a VAE [36, 38]. These methods address the second problem but are often learned from a pixel reconstruction objective that favors low-level detail over semantics. We discuss additional related work in App. A.3.

To address both problems, we make the representation a latent variable of the diffusion model. Intuitively, at every denoising step, the model first predicts a representation from the current noisy state and then takes a denoising step conditioned on it (Fig. 2). We derive a variational lower bound on the data likelihood, in which the alignment loss appears as a KL regularizer rather than a heuristic add-on and existing methods in representation alignment are recovered as a special case (Sec. 2.3 and App. A.3.2) and use it to motivate a practical training objective. Our framework reveals a simple design choice that representation alignment misses: *feeding the learned representation back into the denoiser* (Fig. 1). We call the resulting framework *Representation-Augmented Diffusions* (RAD). To summarize, our core contributions are: **KP: TODO: Double check the claims in the introduction**

- **A principled framework.** We formulate representation learning in diffusion models as a latent-variable model in which the denoiser conditions on a representation at every step. Our framework reveals a simple trick: feed the representation back into the denoiser, which leads to large improvements in sample quality (Sec. 3.5).
- **Faster, parameter- and data-efficient training.** On ImageNet, RAD-XL/2 surpasses REPA-XL/2 trained $27\times$ longer, while RAD-B/2 matches REPA-XL/2 with $4.8\times$ fewer parameters and outperforms REPA-B/2 trained on $10\times$ more data (Sec. 3.1).
- **Fewer sampling steps.** RAD induces more expressive denoising policies. As a result it has a better compute-vs-quality tradeoff than standard diffusion models. For instance, with 20 sampling steps RAD-XL/2 outperforms a REPA-XL/2 model of similar quality using 200 steps, a $10\times$ reduction (Sec. 3.2).
- **More discriminative and controllable representations.** We show that RAD learns smooth (see Fig. 5) and semantic representations while preserving discriminative quality and preserves linearly decodable semantics deeper into the denoiser (Sec. 3.3).
- **Generalization Capabilities.** RAD can be readily applied on top of existing baselines and transfers to text-conditioned image synthesis (Sec. 3.4).

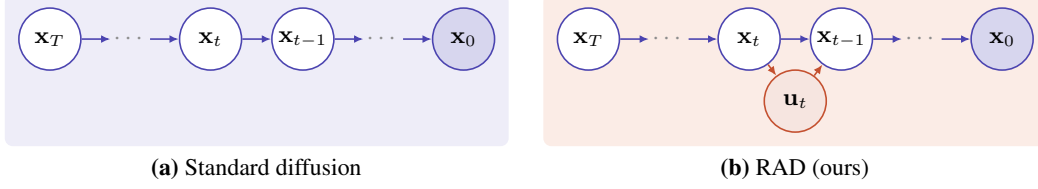


Figure 2: RAD inserts a latent representation into every denoising step. Graphical models of the generative process of (a) Standard Diffusion and (b) RAD.

2 Representation-Augmented Diffusions

Intuition We introduce Representation-Augmented Diffusions (RAD), a class of diffusion models in which a semantic representation guides the generative process itself. The core idea is simple: at every denoising step, the model first predicts a representation from an intermediate noisy state, and then conditions the denoiser on it. This is in contrast to recent methods like REPA which predict representations but never condition the denoiser on them. During training, the target for learning such semantic representations can be provided by additional supervision; At inference, the model generates these representations from its own features, and no external supervision is required. Next, we discuss the core RAD modeling paradigm.

2.1 Denoising meets representations

Our key modeling choice is to augment the denoising process in standard diffusion models with a sequence of latent representations $\mathbf{u}_{1:T}$, one for every denoising step (see Fig. 2).

Generative process. Starting from noise $\mathbf{x}_T \sim p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, each denoising step first samples a latent representation $\mathbf{u}_t$ from a *representation policy* $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$. Consequently, the lesser noisy state $\mathbf{x}_{t-1}$ is then sampled from a *denoising policy* $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)$ conditioned on the latent representation $\mathbf{u}_t$. More formally,

$$p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\phi(\mathbf{u}_t | \mathbf{x}_t) p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t). \quad (1)$$

The representation $\mathbf{u}_t$ therefore acts as a hierarchical latent variable of the transition [40]: the model first samples a semantic representation of the data and then takes a denoising step that is consistent with it. Consequently, we denote this class of diffusion models as *Representation-Augmented Diffusions* (RAD). It is worth noting that marginalizing over $\mathbf{u}_t$ yields the transition,

$$p_{\theta, \phi}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \int p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t) p_\phi(\mathbf{u}_t | \mathbf{x}_t) d\mathbf{u}_t = \mathbb{E}_{\mathbf{u}_t \sim p_\phi} [p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)], \quad (2)$$

which intuitively, is a continuous mixture that is more expressive than the Gaussian modeling choice in standard diffusion models. Since denoising is iterative and can be compute intensive, this property is specifically favorable for few-step sampling in diffusion models where the Gaussianity assumption over denoising breaks. We empirically demonstrate this property later where RAD strikes much better balance between sample quality vs inference compute (see Sec. 3.2). Moreover, when the denoising policy $p_\theta(\cdot | \mathbf{x}_t, \mathbf{u}_t)$ ignores $\mathbf{u}_t$, the transition in Eq. 2 reduces to a Gaussian distribution reducing the setup to standard diffusion models.

Noising process. Next, we define the forward noising (a.k.a inference) process as,

$$q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}). \quad (3)$$

Intuitively, at each timestep t , we infer the next noisy state $\mathbf{x}_t$ from the cleaner state $\mathbf{x}_{t-1}$ via the noising kernel $q(\cdot | \mathbf{x}_t)$, and then infer a representation $\mathbf{u}_t$ from the entire trajectory, $\mathbf{x}_{\leq t}$ up to time t using the *reference policy* $q_{\theta'}(\cdot | \mathbf{x}_{\leq t})$. The representation $\mathbf{u}_t$ is therefore inferred from information that is unavailable to the generative model at step t , i.e. the cleaner trajectory and thus can serve as an external supervision for the representation policy $p_\phi(\cdot | \mathbf{x}_t)$ (more on this in Sec. 2.3). With the modeling choices in place, we formulate a tractable variational bound for RAD next.

2.2 Tractable optimization objective

We derive a variational lower bound on the log-likelihood of the data, $\log p(\mathbf{x}_0)$, which motivates our training objective. With Eq. 3 as the variational distribution, we can show that (see proof in App. A.2),

$$\log p(\mathbf{x}_0) \geq \sum_{t=1}^T \mathbb{E}_{\mathbf{x}_{\leq t}} \left[\underbrace{\mathbb{E}_{\mathbf{u}_t} [\log p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)]}_{\text{Denoising}} - \underbrace{D_{\text{KL}}(q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) \| p_\phi(\mathbf{u}_t | \mathbf{x}_t))}_{\text{Alignment}} \right] + C, \quad (4)$$

where the trajectory $\mathbf{x}_{\leq t}$ is sampled from the noising process, i.e., $\mathbf{x}_{1:t} \sim q(\mathbf{x}_{1:t} | \mathbf{x}_0)$, and $\mathbf{u}_t$ is sampled from the reference policy $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})$. Here, C is a constant that depends only on the noising process and not on (θ, ϕ) and can be ignored in practice (see App. A.2.1). Intuitively, the objective in Eq. 4 is equivalent to learning a conditional VAE at any time t . The first term in Eq. 4 is analogous to the denoising objective of a standard diffusion model with an additional control input. During training, the denoiser learns to reconstruct $\mathbf{x}_{t-1}$ from the current noisy state $\mathbf{x}_t$ and a representation $\mathbf{u}_t$ drawn from the reference policy $q_{\theta'}(\cdot | \mathbf{x}_{\leq t})$. Similarly, the second term penalizes the deviation of the learned generative policy $p_\phi(\cdot | \mathbf{x}_t)$ from the reference policy $q_{\theta'}(\cdot | \mathbf{x}_{\leq t})$ therefore creating an information bottleneck. Due to the nature of this objective, one would expect the latents $\mathbf{u}_t$ to capture meaningful semantic information. Indeed, the latent $\mathbf{u}_t$ provides an interface for steering generation, e.g., for interpolating between images (Fig. 5) which is often hard to achieve with standard diffusion models [38]. Next, we discuss parameterization of different distributions and draw connections to existing work in representation alignment which leads to a surprisingly simple recipe for training RAD.

2.3 Design Choices and Training recipe

We start by proposing different choices for the forward and the generative distributions from Sec. 2.1 and then simplify the RAD training objective bound in Eq. 4.

Noising Process. We choose the forward noising process the same as standard diffusion/stochastic interpolant formulations i.e. $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$. Following recent work on stochastic interpolants, we set $\alpha_t = 1 - t$ and $\sigma_t = t$ for $t \in [0, 1]$. Next, we parameterize the representation policy at any time t as,

$$q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = \mathcal{N}(\alpha_t f_{\text{pre}}(\mathbf{x}_0), \beta^2 \sigma_t^2 \mathbf{I}), \quad \beta > 0 \quad (5)$$

Throughout this work we fix f_{pre} to a frozen pretrained DINOv2 [34] encoder which extracts clean representations from data $\mathbf{x}_0$; in the absence of an external supervisor, alternative choices of the reference policy, such as the model’s own EMA features, recover other methods (App. A.3.2). Our choice is motivated by empirical results (Table 6). Moreover, we adopt the same noising schedule for the reference policy as the forward diffusion process. Our intuition behind this choice is that during denoising at higher noise levels, only very coarse structures emerge and therefore the representations at these noise levels must encode more uncertainty. However, we emphasize that different choices of the noising schedule are possible and remains an interesting direction for further work. Moreover, since the supervision comes from $\mathbf{x}_0$ alone, the reference policy drops its dependence on the trajectory and carries no learnable parameters.

Choice of Representation Policy. We first consider a simple class of policies modeled by a unimodal Gaussian distribution,

$$p_\phi(\mathbf{u}_t | \mathbf{x}_t) = \mathcal{N}(\alpha_t \mathbf{h}_\phi(\mathbf{x}_t, t), \beta^2 \sigma_t^2 \mathbf{I}) \quad (6)$$

where β is a scalar which controls the width of the policy. We will later show that controlling the width of the policy can significantly improve sample quality (Fig. 9). Similar to REPA, we write $\mathbf{h}_\phi(\mathbf{x}_t, t) = h_\phi(\mathbf{h}_t)$, where $\mathbf{h}_t = \mathbf{v}_\theta^{(\ell)}(\mathbf{x}_t, t)$ is the hidden state of the denoiser $\mathbf{v}_\theta$ after block ℓ and h_ϕ is an MLP projector. The semantic representation can then be sampled via $\mathbf{u}_t = \alpha_t \mathbf{h}_\phi(\mathbf{x}_t, t) + \beta \sigma_t \bar{\epsilon}$ where $\bar{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. From a VAE perspective (Sec. 2.2), the initial ℓ blocks of the denoiser combined with the MLP head act as a noisy encoder for extracting representations from $\mathbf{x}_t$.

Algorithm 1: Training	Algorithm 2: Sampling (Euler, N steps)
repeat $(\mathbf{x}_0, \mathbf{c}) \sim \text{data}, t \sim \mathcal{U}(0, 1), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\mathbf{x}_t \leftarrow \alpha_t \mathbf{x}_0 + \sigma_t \epsilon, \mathbf{y} \leftarrow f_{\text{pre}}(\mathbf{x}_0)$ $\mathbf{h}_t \leftarrow \mathbf{v}_\theta^{(\ell)}(\mathbf{x}_t, t, \mathbf{c})$ ▷ first ℓ blocks $\mathbf{u}_t \sim q(\mathbf{u}_t \mathbf{x}_0)$ $\mathbf{v} \leftarrow \mathbf{v}_\theta^{(>\ell)}(\mathbf{h}_t + \text{Inject}(\mathbf{u}_t), t, \mathbf{c})$ $\mathcal{L} \leftarrow \ \mathbf{v} - \mathbf{v}_{\text{ref}}\ _2^2$ $\quad + \frac{\lambda_t}{2} \ \mathbf{y} - h_\phi(\mathbf{h}_t)\ _2^2 + (1 - \lambda_t) \ell_{\text{flow}}$ Gradient step on (θ, ϕ, ζ) until converged	$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \Delta \leftarrow 1/N$ for $i = N, \dots, 1$ do $t \leftarrow i\Delta$ $\mathbf{h}_t \leftarrow \mathbf{v}_\theta^{(\ell)}(\mathbf{x}, t, \mathbf{c})$ ▷ first ℓ blocks $\mathbf{u}_t \sim p_\phi(\mathbf{u}_t \mathbf{x}_t)$ $\mathbf{v} \leftarrow \mathbf{v}_\theta^{(>\ell)}(\mathbf{h}_t + \text{Inject}(\mathbf{u}_t), t, \mathbf{c})$ $\mathbf{x} \leftarrow \mathbf{x} - \Delta \mathbf{v}$ return $\mathbf{x}$

Highlighted parts are added or changed by RAD relative to REPA; removing them recovers REPA, whose alignment term is $-\lambda \cos(\mathbf{y}, h_\phi(\mathbf{h}_t))$ and which discards h_ϕ at sampling. Inject is the control pathway of Eq. 8 and ℓ_{flow} is the flow likelihood of Eq. 39.

While the choice of unimodal Gaussian for the representation policy can induce multimodal denoising transitions (Eq. 2), we also consider conditional normalizing flow [8] based policies with the following form (see App. A.2.4 for more details),

$$\mathbf{u}_t = \alpha_t \mathbf{h}_\phi(\mathbf{x}_t, t) + \beta \sigma_t \mathbf{g}_\zeta(\mathbf{h}_t, \bar{\epsilon}), \quad \bar{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \quad (7)$$

We find that using more expressive policies improves sample quality (Table 5).

Choice of Denoising Policy. We parameterize the denoising policy as $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, \mathbf{u}_t, t), \bar{\sigma}_{t-1|t}^2 \mathbf{I})$ and model $\boldsymbol{\mu}_\theta$ through the velocity network $\mathbf{v}_\theta$, which uses the SiT architecture throughout this work. The sampled representation $\mathbf{u}_t \in \mathbb{R}^{N \times D}$ enters the backbone $\mathbf{v}_\theta$ at the same layer ℓ where it is predicted, after three operations (Fig. 7):

$$\mathbf{u}_t^n = \frac{\mathbf{u}_t - \gamma \bar{\mathbf{u}}_t}{\bar{\sigma}_t + \delta} \implies \mathbf{u}_t^c = \text{MLP}(\mathbf{u}_t^n) \implies \mathbf{u}_t^f = \text{Scale-Shift}(\mathbf{u}_t^c, \mathbf{c}), \quad (8)$$

where $\bar{\mathbf{u}}_t, \bar{\sigma}_t \in \mathbb{R}^D$ are the per-channel mean and standard deviation of $\mathbf{u}_t$ across its N tokens, $\gamma \in [0, 1]$ controls how much of the mean is removed (we use $\gamma = 0.6$), and δ is a small constant. Intuitively, Eq. 8 first removes global information, shared across all tokens, from $\mathbf{u}_t$ (as it could be harmful for sample quality [43]), then compresses the information into a single token $\mathbf{u}_t^c$ with an MLP, and finally applies a learnable AdaLN-style scale-shift transform conditioned on the conditioning embedding $\mathbf{c}$. We then add $\mathbf{u}_t^f$ back into the hidden state $\mathbf{h}_t$ which is then passed into downstream SiT blocks to predict the velocity field. We present an ablation over these design choices in Sec. 3.5 and defer full details to App. A.2.4.

Simplified Training Objective Under these parameterizations, reweighting the terms of the bound in Eq. 4 reduces to the following loss (see App. A.2.3):

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{t, \mathbf{x}_0, \epsilon, \mathbf{u}_t \sim q} \left[\|\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) - \mathbf{v}_{\text{ref}}\|_2^2 + \frac{\text{SNR}(t)}{2\beta^2} \|\mathbf{f}_{\text{pre}}(\mathbf{x}_0) - \mathbf{h}_\phi(\mathbf{x}_t, t)\|_2^2 \right], \quad (9)$$

where $\text{SNR}(t) = \alpha_t^2 / \sigma_t^2$ and $\mathbf{v}_{\text{ref}} = \dot{\alpha}_t \mathbf{x}_0 + \dot{\sigma}_t \epsilon$. The simplified objective in Eq. 9 is similar to the heuristic alignment objective proposed in REPA with the exception of the latent representations $\mathbf{u}_t$ now also part of the denoiser. Indeed, we show that existing external alignment methods like REPA are special cases of our framework (see App. A.3.2). During training, we always fix $\beta = 1.0$. However, we find that tuning β during inference improves sample quality (Fig. 4c).

Sampling. Existing off-the-shelf samplers for flow matching models can be directly applied to RAD by estimating the marginal vector field and plugging it into the ODE/SDE sampler. For instance, the probability flow ODE for RAD reads,

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}_{\theta, \phi}(\mathbf{x}, t) = \mathbb{E}_{\mathbf{u}_t \sim p_\phi(\mathbf{u}_t | \mathbf{x}_t)} [\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t)] \approx \frac{1}{M} \sum_{i=1}^M \mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t^{(i)}, t). \quad (10)$$

where $\mathbf{u}_t^{(i)} \sim p_\phi(\mathbf{u}_t | \mathbf{x}_t)$ are samples from the representation policy. In early experiments, we find that a single sample ($M = 1$) is sufficient for high-quality generation. We provide a pseudo-code for sampling in Alg. 2.

However, estimating the expectation via Monte Carlo sampling in Eq. 10 makes each velocity evaluation stochastic. Therefore, when sampling $\mathbf{u}_t \sim p_\phi(\mathbf{u}_t | \mathbf{x}_t)$, we consider two choices for the policy noise. With *fresh* policy noise, a new $\bar{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is drawn at every denoising step, so each step uses an independent sample from the representation policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$, while the update of $\mathbf{x}_t$ itself injects no noise (Alg. 2). With *fixed* policy noise, $\bar{\epsilon}$ is drawn once together with $\mathbf{x}_1$ and kept fixed along the trajectory. This ensures that the sample is a deterministic function of the initial noise. Both variants require one network evaluation per step. In practice, we find that both variants yield similar sample quality (see Table 15) and therefore, use the fresh policy noise variant for all experiments in the main paper.

3 Experiments

We evaluate RAD primarily on class-conditional ImageNet 256×256 and study the effect of feeding the aligned representation back into the denoiser. We make the following observations. **(i) Training and data efficiency.** RAD trains faster and is more parameter- and data-efficient (Sec. 3.1). At XL/2 scale, RAD matches the sample quality of REPA in $27\times$ fewer training iterations (Fig. 1c), and RAD-B/2 outperforms REPA-B/2 trained on $10\times$ more data (Fig. 3b,c). **(ii) Sampling efficiency.** RAD improves the quality-vs-NFE tradeoff (Sec. 3.2). With 20 Euler steps, RAD-XL/2 outperforms a REPA-XL/2 model of similar quality sampled with 200 steps. **(iii) Representation quality.** RAD preserves linearly decodable semantics deeper into the network and supports smooth interpolation in representation space (Sec. 3.3). **(iv) Generalization.** RAD can be combined with other alignment methods such as REG and transfers to text-to-image generation (Sec. 3.4). We summarize the experimental setup below and defer full details to App. A.4.

Tasks and Datasets. We evaluate RAD under two settings. For **class-conditional generation**, we exactly follow the experimental setup of REPA and train on ImageNet 256×256 in the latent space of the Stable Diffusion VAE [41]. For **text-to-image synthesis**, we train on the BLIP3o pre-training captions [4] at 256×256 resolution, conditioning on caption features from a frozen Qwen3-0.6B text encoder [39], and, for qualitative samples only, fine-tune on the curated BLIP3o-60k instruction set.

Models and Training. We use SiT- $\{\text{B/2, L/2, XL/2}\}$ backbones for ImageNet and SiT- $\{\text{B/2, XL/2}\}$ for text-to-image synthesis (/2 indicates 2×2 patch size). For all experiments, we use the DINOv2-B model as the default external supervisor f_{pre} . For notational convenience, we denote a SiT backbone trained with RAD by its size, e.g., RAD-XL/2. Following REPA, all models are trained with a global batch size of 256 using the AdamW optimizer [30] with a learning rate of $1e-4$, and an Exponential Moving Average (EMA) of the weights, for up to 400K iterations (primarily due to compute limitations) unless stated otherwise.

Baselines. Our primary baseline is REPA, trained with an identical backbone, encoder, data, and optimization schedule. The difference is that RAD conditions the denoiser on a sampled representation, drawn from the reference policy during training and from the learned policy at inference (Sec. 2.3). We additionally compare against the unaligned SiT baseline and existing representation alignment methods which build directly on top of the REPA framework [26], though we hypothesize that such methods would directly benefit from our framework (see Sec. 3.4).

Evaluation Metrics. To measure sample quality, we report FID [13], and sFID [33] on 50k samples. For class conditional ImageNet, we additionally report Inception Score [42], and precision/recall [24] to measure the fidelity-diversity tradeoffs. We use the Euler-Maruyama SDE sampler with 250 steps no classifier-free guidance [14] unless noted. For text-to-image synthesis, we report compositional accuracy on GenEval [10].

Table 1: Comparison with existing baselines on ImageNet 256×256 . FID-50k without CFG (Euler SDE, 250 steps).

Method	Epochs	B/2		L/2		XL/2	
		#Params	FID↓	#Params	FID↓	#Params	FID↓
<i>Diffusion transformers</i>							
DiT [37]	80	130M	43.5	458M	23.3	675M	19.5
DiT [37]	1400	—	—	—	—	675M	9.6
SiT [31]	80	130M	33.0	458M	18.8	675M	17.2
SiT [31]	1400	—	—	—	—	675M	8.3
<i>Representation alignment</i>							
DiT + REPA [59]	80	—	—	458M	15.6	675M	12.3
REPA [59]	80	130M	24.4	458M	10.0	675M	7.9
iREPA [43]	80	130M	21.4	458M	9.4	675M	7.5
sREPA [56]	80	130M	20.5	458M	8.7	675M	7.2
HASTE [52]	80	130M	19.6	458M	8.9	675M	7.3
REED [48]	80	130M	19.5	458M	8.9	675M	7.3
TREAD [23]	80	130M	16.0	458M	5.9	675M	4.9
REG [53]	80	132M	15.2	460M	4.6	677M	3.4
REPA-E [†] [26]	80	—	—	—	—	675M	4.1
<i>Representation augmentation</i>							
RAD (ours)	80	142M	7.7	472M	3.9	690M	3.7

[†]REPA-E also fine-tunes the VAE end-to-end.

Table 2: Comparison with CFG on ImageNet 256×256 . NFE: network evaluations per sample, counting a CFG pair once (RAD: Euler SDE, 50 steps). FID uses class-balanced sampling, except rows marked [‡], which use random sampling.

Method	Ep.	#Params	NFE	FID↓	IS↑	Prec.↑	Rec.↑
<i>Pixel diffusion</i>							
ADM [‡] [7]	400	554M	500	3.94	215.8	0.83	0.53
PixelFlow [5]	320	677M	–	1.98	282.1	0.81	0.60
SiD2 [16]	1280	–	512	1.38	–	–	–
<i>Latent diffusion</i>							
DiT [‡] [37]	1400	675M	250	2.27	278.2	0.83	0.57
MaskDiT [‡] [64]	1600	675M	79	2.28	276.6	0.80	0.61
SiT [‡] [31]	1400	675M	250	2.06	270.3	0.82	0.59
<i>Representation alignment</i>							
REPA [59]	800	675M	250	1.29	306.3	0.79	0.64
DDT [50]	80	675M	250	1.52	263.7	0.78	0.63
REPA-E ^{†‡} [26]	80	675M	250	1.67	266.3	0.80	0.63
REG [‡] [53]	80	677M	250	1.86	321.4	0.76	0.63
iREPA [‡] [43]	80	675M	250	1.93	268.8	0.80	0.60
sREPA [56]	80	675M	250	2.25	257.5	0.83	0.58
<i>Representation augmentation</i>							
RAD (ours)	80	690M	50	2.02	252.6	0.77	0.62
RAD (ours)	200	690M	50	1.85	269.7	0.79	0.62

3.1 Generation Quality

Faster training convergence at every model scale. We compare the training efficiency of RAD in Fig. 3 (full results in Table 8 in App. A.5). Interestingly, after only 100K training iterations, RAD matches or surpasses REPA at 400K iterations in sample quality for all three backbones, a $4\times$ minimum reduction in training steps. Moreover, the gains grow further against long training runs (Fig. 1c): RAD-XL/2 surpasses SiT-XL/2 trained for 7M iterations after 100K iterations, and REPA-XL/2 trained for 4M iterations after 150K iterations, reducing the required training steps by $70\times$ and $27\times$, respectively. Since RAD and REPA share the backbone, encoder, and training schedule, we attribute these gains to conditioning the denoiser on sampled representations (we provide more detailed ablations in Sec. 3.5).

Better parameter and data efficiency. We find that RAD exhibits superior parameter efficiency compared to REPA. For instance, RAD-B/2 outperforms REPA-L/2 (FID 7.7 vs. 10.0) with $3.2\times$ fewer parameters (Fig. 3b). Similarly, RAD-B/2 outperforms SiT-XL/2 (FID: 8.3) trained for 7M iterations and also closely tracks REPA-XL/2 throughout training (Fig. 3a) and performs on par with it at 400K iterations (FID 7.7 vs. 7.9), while using $4.8\times$ fewer parameters.

RAD is also more data efficient. We train B/2 models for the same 400K iterations on subsets of ImageNet and report the best FID of each run over its checkpoints. Trained on 10% of ImageNet, RAD-B/2 outperforms REPA-B/2 (FID: 13.0 vs 22.9) trained on the full dataset i.e., RAD needs $10\times$ fewer distinct training images under the same iteration budget. Moreover, REPA-B/2 does not improve beyond 10% of the data suggesting underfitting while RAD keeps improving with more data.

Comparison with other baselines. Next, we compare RAD with additional alignment baselines (Table 1) reporting FID-50k after 80 epochs without classifier-free guidance. Across different model scales, RAD registers strong sample quality achieving best FID among competitors on SiT-B/2 and L/2 while being competitive to REG at the XL/2 scale. However, we show that our framework can be readily combined with other baselines (Sec. 3.4) and thus hypothesize that a lot of the techniques used by existing representation alignment would be complementary to RAD. We show class-conditional samples of RAD-XL/2 in App. A.9.

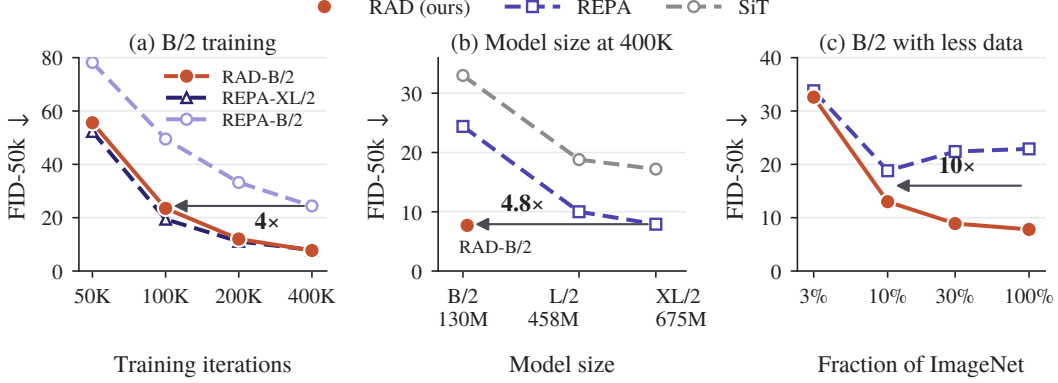


Figure 3: RAD converges faster and is more parameter- and data-efficient than REPA and SiT. (a) RAD-B/2 converges $4\times$ faster than REPA-B/2 and is comparable to REPA-XL/2 during training. (b) RAD-B/2 is $4.8\times$ more parameter efficient than REPA. (c) RAD-B/2 is $10\times$ more data efficient than REPA-B/2.

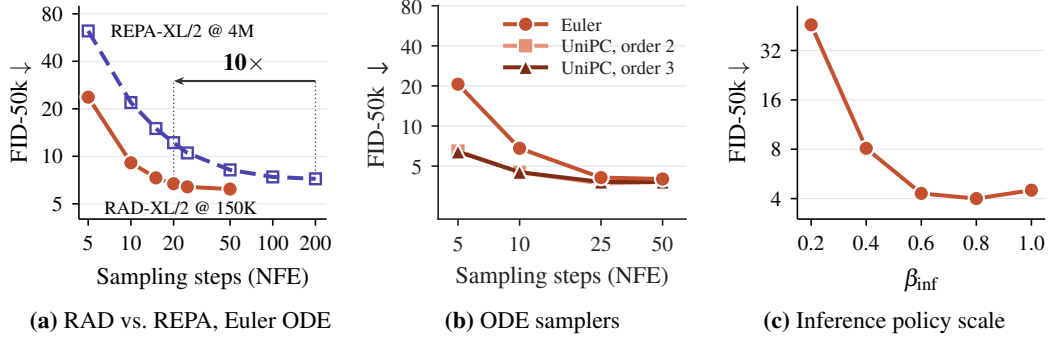


Figure 4: Inference with RAD. (a) FID-50k vs. sampling steps for RAD-XL/2 (150K) and REPA-XL/2 (4M) with the Euler ODE sampler. (b) Euler vs. UniPC for RAD-XL/2 (400K). (c) FID-50k of RAD-XL/2 (400K) as β_{inf} varies; training uses $\beta = 1$, Euler ODE sampler with 50 steps. No guidance.

Token noise coupling makes classifier-free guidance effective. Classifier-free guidance (CFG) [14] is the standard way to improve the sample quality of diffusion models. However, we find that vanilla CFG yields less significant improvements in sample quality in RAD (Table 10). We hypothesize this to a lack of sample-level diversity from the representation policy: the policy noise is independent across the N tokens of the representation $\mathbf{u}_t$, so it largely averages out once $\mathbf{u}_t$ is compressed into a single token (Eq. 8). We therefore correlate noise between different tokens in the representation $\mathbf{u}_t$ by introducing *noise coupling* (NC) which shares part of the policy noise across tokens (to prevent averaging),

$$\bar{\epsilon}_n = \rho \boldsymbol{\xi} + \sqrt{1 - \rho^2} \boldsymbol{\eta}_n, \quad \boldsymbol{\xi}, \boldsymbol{\eta}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad n = 1, \dots, N, \quad (11)$$

so that the policy of each individual token is unchanged. The representation $\bar{\mathbf{u}}_t$ is then sampled using the policy noise $\bar{\epsilon}_t$. On its own, noise coupling worsens sample quality, but combined with CFG it yields the largest improvement (Table 10). We find that similar observations also hold for autoguidance (AG) [18]. Our best model combines such noise coupling with CFG and AG and achieves an FID-50k of 2.02 after only 80 epochs and with 50 sampling steps (Table 2) which is comparable to SOTA methods while requiring $5\times$ less NFE evaluations (see App. A.6 for more details).

3.2 Sampling Efficiency

Better quality-vs-compute tradeoff at inference. Marginalizing out the representations can lead to more expressive denoising policies than standard diffusion models (Sec. 2). This could be particularly useful in a few step sampling regime where a more expressive transition should tolerate larger steps. To test this, we compare



Figure 5: RAD allows smooth interpolation in the representation space compared to REPA. Top: RAD-XL/2 (400K iters) keeps attributes like pose and viewpoint consistent and allows a smooth interpolation between two images. Bottom: REPA-XL/2 (4M iters) yields undesirable artifacts often changing the semantic image content.

two models of similar quality under the 250-step SDE sampler: RAD-XL/2 trained for 150K iterations (FID 5.7) and the official REPA-XL/2 checkpoint trained for 4M iterations (FID 5.9). Fig. 4a and Table 13 show that RAD outperforms REPA at every step budget evaluated for both. We find that RAD is roughly $10\times$ more NFE efficient compared to REPA in the deterministic sampling regime. Similarly for stochastic sampling: with the Euler–Maruyama SDE sampler (Fig. 8 and Table 14), RAD is around $5\times$ more efficient than REPA.

Compatibility with SOTA samplers. RAD is readily compatible with alternative state-of-the-art samplers for diffusion models (Fig. 4b). With UniPC [62], RAD-XL/2 (at 400k steps) reaches FID 6.5 at 5 steps. Moreover, sampler performance is robust to the choice of fresh or fixed policy noise (Tab. 15).

3.3 Representation Quality

Interpolation in Representation Space As the denoiser conditions on $\mathbf{u}_t$ at every step, the learned representation can be used to manipulate generations. We test this by interpolating between two real ImageNet images, $\mathbf{x}_0^A$ and $\mathbf{x}_0^B$. From each image we extract a representation $\bar{\mathbf{u}} = \mathbf{h}_\phi(\mathbf{x}_{t_c}, t_c)$ at a small noise level t_c . For a weight $m \in [0, 1]$, we linearly interpolate the representations, $\bar{\mathbf{u}}_m = m\bar{\mathbf{u}}^A + (1 - m)\bar{\mathbf{u}}^B$, and likewise the class embeddings, and obtain the starting noise $\mathbf{x}_{1,m}$ by spherical linear interpolation (slerp) between the inverted noises $\mathbf{x}_1^A$ and $\mathbf{x}_1^B$. We then sample from $\mathbf{x}_{1,m}$. Fig. 5 interpolates from a baboon to a macaque with RAD-XL/2 at 400K iterations and the released REPA-XL/2 checkpoint, which shares our backbone. RAD changes the subject gradually while the pose, viewpoint, and framing stay fixed, indicating a smooth representation space. REPA, which offers no representation to steer, interpolates only the noise and the class embedding, and its intermediate samples switch between poses and scenes, even though it is trained for 4M iterations, i.e., $10\times$ longer. Additional samples and implementation details are given in App. A.8.

Classification Accuracy Following REPA, we train a linear classifier on the hidden state at block k of a noised input $\mathbf{x}_t$ and report ImageNet top-1 accuracy for both RAD and REPA trained for 400K iterations, both aligned at block 4 (see App. A.4 for details). We make two key observations from Fig. 6. First, RAD preserves discriminative features deeper into the network than REPA (Fig. 6, top). At the aligned block, where both models regress the reference DINOv2 features, the two have similar linear probing accuracy. For the remaining blocks, RAD is more accurate than REPA by up to 4.4 points at $t = 0.1$. More interestingly, the linear probing accuracy of RAD decreases more slowly than that of REPA. Intuitively, this is likely due to feeding the representations back into the denoiser which keeps the later blocks linearly separable. Second, the features improve with scale (Fig. 6, bottom). RAD-XL/2 improves over RAD-B/2 by 8 to 9 points consistently for the most part of the diffusion trajectory $t \leq 0.5$ (71.7% vs. 63.4% at $t = 0.1$).

3.4 Generalization Capabilities

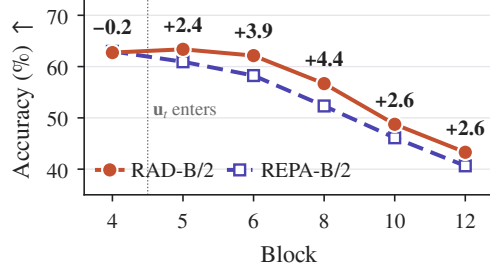
RAD benefits REG convergence. REG denoises the class token of the encoder jointly with the image tokens. We extend REG within the RAD framework by treating the projection of the intermediate noisy patch tokens

Table 3: RAD improves REG training convergence. FID-50k, Euler SDE, w/o CFG

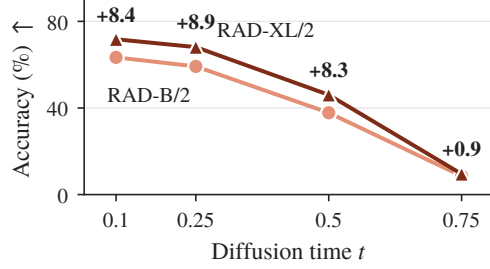
Model	#Params	Iter.	FID↓
<i>SiT-B/2</i> [31]			
REG [53]	132M	100K	36.1
REG + RAD (ours)	143M	100K	22.4
REG [53]	132M	400K	15.2
REG + RAD (ours)	143M	400K	8.9
<i>SiT-L/2</i> [31]			
REG [53]	460M	100K	11.4
REG + RAD (ours)	474M	100K	7.8
REG [53]	460M	400K	4.6
REG + RAD (ours)	474M	400K	3.9

Table 4: RAD improves text-to-image generation over REPA at both model scales. FID-50k and GenEval overall score on the BLIP3o pre-training captions at 256×256 with a frozen Qwen3-0.6B text encoder. Best per backbone in bold. No guidance.

Method	B/2		XL/2	
	FID↓	GenEval↑	FID↓	GenEval↑
REPA [59]	10.77	0.138	4.31	0.258
RAD (ours)	8.42	0.166	3.94	0.266



(a) Per-block accuracy for RAD-B/2 vs REPA (aligned at block 4)



(b) Best block against t for RAD-{B/2, XL/2}

Figure 6: Better linear probing with RAD.

in REG as the mean of the representation policy (see App. A.4 for details). Table 3 shows that combining REG with our approach improves the training convergence of REG at both model scales. Notably, REG combined with RAD converges twice as fast: on SiT-B/2 it matches the 200K-iteration FID of REG after 100K iterations, and on SiT-L/2 it matches the 400K-iteration FID of REG after 200K iterations (Table 9).

RAD generalizes to Text-to-Image Generation Table 4 shows that RAD outperforms REPA on text-to-image synthesis in both sample quality and compositional accuracy at both scales.

3.5 Ablation Experiments

Impact of β . The policy scale β sets the noise level of the representation, both during training and, as β_{inf} , at inference. During training, removing the noise entirely ($\beta = 0$, i.e., noise-free teacher features) heavily degrades sample quality (See Fig. 10, Table 17) which implies the importance of using stochastic policies for training RAD. We find that $\beta = 1.0$ works reasonably well at all model scales and therefore, propose to set it to 1.0 in practice. At inference, we find that reducing β usually helps with improving sample quality without affecting recall adversely (Figs. 4c and 9, and Table 16).

Reference vs. on-policy supervision. During training, the controls fed to the denoiser can be sampled either from the reference policy $q(\mathbf{u}_t | \mathbf{x}_0)$, as in our objective (Eq. 9), or from the learned representation policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$, which matches how the controls are obtained at inference. We find that training with the reference policy yields better sample quality: with on-policy supervision, RAD-B/2 reaches an FID of 22.4, compared with 12.6 with reference supervision. Therefore, we sample teacher representations for all experiments considered in this work.

Design Choices. Table 6 highlights the impact of the choice of external teacher, the representation injection depth and the alignment loss weighting in RAD-B/2. RAD is relatively robust to the choice of the encoder. DINOv2-B and DINOv2-L perform comparably, and, similar to REPA and REG, we use DINOv2-B for all our analysis. Moreover, we find that an injection depth of 2 and an alignment weighting coefficient of $p = 1$ works the best at RAD-B/2 scale.

Table 5: Elucidating the role of different design components in RAD. FLOPs is the increase in inference FLOPs over REPA (23.0 GFLOPs). ✓ indicates tuned β_{inf} , ✗ indicates $\beta_{\text{inf}} = 1$.

Training configuration	β_{inf}	FID↓	Δ	FLOPs
A REPA [59]	–	24.4	–	–
B + $\mathbf{u}_t \sim \mathcal{N}(\cdot) \rightarrow$ denoiser	✗	12.6	−11.8	+8.4%
C + Tuned β_{inf}	✓	10.3	−2.3	+8.4%
D + Flow policy	✓	8.7	−1.6	+11.8%
E + Control Norm + Mod	✓	8.2	−0.6	+11.8%
F + Early inject + $p=1.0$	✓	7.7	−0.5	+11.8%

Table 6: Ablation: Impact of the choice of external encoder, injection depth and alignment loss weighting. Sample quality on ImageNet 256×256 of SiT-B/2 models trained for 400K iterations

Method	Encoder	Depth	p	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
SiT-B/2 [31]	–	–	–	33.0	6.5	43.7	0.53	0.63
+ REPA [59]	DINOv2-B	4	–	24.4	6.4	59.9	0.59	0.65
+ REG [53]	DINOv2-B	4	–	15.2	6.9	94.6	0.64	0.63
<i>+ RAD (ours): encoder</i>								
	DINOv2-B	4	0	8.5	6.1	129.5	0.76	0.53
	DINOv2-L	4	0	8.4	7.0	129.8	0.76	0.54
	SigLIP-L	4	0	8.9	7.1	133.1	0.77	0.51
<i>+ RAD (ours): injection depth</i>								
	DINOv2-B	2	0	8.2	7.1	141.9	0.78	0.51
	DINOv2-B	4	0	8.5	6.1	129.5	0.76	0.53
	DINOv2-B	6	0	10.5	7.4	108.8	0.71	0.55
	DINOv2-B	8	0	15.1	7.7	85.6	0.66	0.56
<i>+ RAD (ours): alignment loss weighting</i>								
	DINOv2-B	4	0	8.5	6.1	129.5	0.76	0.53
	DINOv2-B	4	0.5	8.7	6.5	125.7	0.75	0.53
	DINOv2-B	4	1.0	8.3	6.7	129.0	0.75	0.53
	DINOv2-B	4	1.5	8.6	6.5	129.6	0.75	0.52

System Analysis. Table 5 isolates the contribution of each component of RAD-B/2 towards sample quality. We start from REPA (config A) and add components sequentially, ending at the configuration used in Table 1 (config F). We emphasize that conditioning the denoiser on representations $\mathbf{u}_t$ drawn from a Gaussian policy of Eq. 6 (configs B–C) already lowers the FID from 24.4 to 10.3. Using a more expressive representation policy, the normalizing flow of Eq. 34 (config D), lowers FID further from 10.3 to 8.7, indicating that more expressive policies can be readily adopted with RAD. Control normalization and modulation (config E) and earlier injection with the weighted alignment loss (config F) add smaller gains resulting in FID of 7.7.

Additional compute costs. Our control pathway is relatively lightweight. At B/2 model scale, the whole pathway costs 12% additional inference FLOPs over REPA, most of it from the projector h_ϕ , which REPA also trains but discards at inference. However, the control pathway has the same architecture for L/2 and XL/2: therefore the overhead falls to 3.5% at L/2 and 2.5% at XL/2.

4 Conclusion

We introduced Representation-Augmented Diffusions (RAD), a class of diffusion models that embed semantic representations as latent variables within the denoising process: at every step, the denoiser conditions on a representation predicted from its own noisy state. The framework recovers representation alignment methods as special cases and reveals a simple design choice they leave unused: feeding the aligned representation back into the denoiser. RAD converges in fewer iterations during training, is more data- and parameter-efficient, needs fewer sampling steps, and preserves discriminative representations deeper in the denoiser.

Limitations. The gains in sample quality come at the cost of diversity. RAD attains substantially higher precision than REPA but lower recall (e.g., 0.78 vs. 0.59 precision and 0.49 vs. 0.65 recall on SiT-B/2 at 400K iterations; Table 8), and its recall stays roughly constant as sample quality improves with training and scale. Improving this fidelity–diversity tradeoff, for instance through the variance of the representation policy or the sampler, is an important direction for future work.

References

- [1] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023.

- [2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.
- [3] Hila Chefer, Patrick Esser, Dominik Lorenz, Dustin Podell, Vikash Raja, Vinh Tong, Antonio Torralba, and Robin Rombach. Self-supervised flow matching for scalable multi-modal synthesis. In *International Conference on Machine Learning*, 2026.
- [4] Jiu-hai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, Le Xue, Caiming Xiong, and Ran Xu. BLIP3-o: A family of fully open unified multimodal models – architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025.
- [5] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. PixelFlow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.
- [8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.
- [9] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=CzceR82CYc>.
- [10] Dhruva Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, 2023.
- [11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.
- [12] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- [13] 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*, 2017.
- [14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2022.
- [15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.
- [16] Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (SiD2): 1.5 FID on ImageNet512 with pixel-space diffusion. *arXiv preprint arXiv:2410.19324*, 2024.
- [17] Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. Representation alignment for diffusion transformers without external components. In *International Conference on Learning Representations*, 2026.
- [18] 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*, 2024.
- [19] Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, 2023.
- [20] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.
- [21] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.
- [22] Theodoros Kousalis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyridon Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. In *Advances in Neural Information Processing Systems*, volume 38, pp. 16685–16714, 2025.

- [23] Felix Krause, Timy Phan, Ming Gui, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. TREAD: Token routing for efficient architecture-agnostic diffusion training. In *IEEE/CVF International Conference on Computer Vision*, pp. 15703–15713, 2025.
- [24] 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*, 2019.
- [25] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *Advances in Neural Information Processing Systems*, 2024.
- [26] 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*, pp. 18262–18272, 2025.
- [27] Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. In *Advances in Neural Information Processing Systems*, 2024.
- [28] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.
- [29] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=XVjTT1nw5z>.
- [30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [31] 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*, 2024.
- [32] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL <https://arxiv.org/abs/2603.19312>.
- [33] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. In *International Conference on Machine Learning*, 2021.
- [34] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.
- [35] Kushagra Pandey and Stephan Mandt. A complete recipe for diffusion generative models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4238–4249. IEEE, 2023.
- [36] Kushagra Pandey, Avideep Mukherjee, Piyush Rai, and Abhishek Kumar. DiffuseVAE: Efficient, controllable and high-fidelity generation from low-dimensional latents. *Transactions on Machine Learning Research*, 2022.
- [37] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, 2023.
- [38] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [39] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [40] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pp. 324–333. PMLR, 2016.
- [41] 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*, 2022.
- [42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, 2016.
- [43] Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In *International Conference on Learning Representations*, 2026.
- [44] Abhishek Sinha, Jiaming Song, Chenlin Meng, and Stefano Ermon. D2C: Diffusion-decoding models for few-shot conditional generation. In *Advances in Neural Information Processing Systems*, 2021.

- [45] Kihyuk Sohn, Honglak Lee, and Xinchun Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, 2015.
- [46] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- [47] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*, 2021.
- [48] Chenyu Wang, Cai Zhou, Sharut Gupta, Zongyu Lin, Stefanie Jegelka, Stephen Bates, and Tommi Jaakkola. Learning diffusion models with flexible representation guidance. In *Advances in Neural Information Processing Systems*, 2025.
- [49] Runqian Wang and Kaiming He. Diffuse and disperse: Image generation with representation regularization. *arXiv preprint arXiv:2506.09027*, 2025.
- [50] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025.
- [51] Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. InfoDiffusion: Representation learning using information maximizing diffusion models. In *International Conference on Machine Learning*, 2023.
- [52] Ziqiao Wang, Wangbo Zhao, Yuhao Zhou, Zekai Li, Zhiyuan Liang, Mingjia Shi, Xuanlei Zhao, Pengfei Zhou, Kaipeng Zhang, Zhangyang Wang, Kai Wang, and Yang You. REPA works until it doesn't: Early-stopped, holistic alignment supercharges diffusion training. In *Advances in Neural Information Processing Systems*, 2025.
- [53] Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *Advances in Neural Information Processing Systems*, 2025.
- [54] Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *International Conference on Computer Vision*, 2023.
- [55] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations*, 2022.
- [56] Shaodong Xu, Zhendong Wang, Litong Gong, Zexian Li, Wengang Zhou, Tiezheng Ge, and Houqiang Li. Beyond point-wise matching: Structural representation alignment for accelerating diffusion transformers. *arXiv preprint arXiv:2605.16949*, 2026.
- [57] Yilun Xu, Gabriele Corso, Tommi Jaakkola, Arash Vahdat, and Karsten Kreis. DisCo-Diff: Enhancing continuous diffusion models with discrete latents. In *International Conference on Machine Learning*, 2024.
- [58] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [59] 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 *International Conference on Learning Representations*, 2025.
- [60] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *International Conference on Computer Vision*, 2023.
- [61] Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2022.
- [62] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=hrkmlPhplu>.
- [63] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026.
- [64] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research*, 2024.
- [65] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, 2025.
- [66] Rui Zhu, Yingwei Pan, Yehao Li, Ting Yao, Zhenglong Sun, Tao Mei, and Chang Wen Chen. SD-DiT: Unleashing the power of self-supervised discrimination in diffusion transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

A Appendix

Contents

A.1	Background	15
A.1.1	Stochastic Interpolants	15
A.1.2	Representation Alignment	15
A.2	Derivations	16
A.2.1	Variational Bound	16
A.2.2	Parameterization	17
A.2.3	Simplified Training Objective	18
A.2.4	Practical Design Choices	19
A.3	Related Work	21
A.3.1	Contextualizing RAD within Generative Modeling	21
A.3.2	Prior Work as Special Cases under RAD	22
A.4	Experimental Details	23
A.5	Additional Generation Results	25
A.6	Guided Sampling	25
A.7	Additional Quantitative Results	26
A.8	Additional Interpolation Results	29
A.9	Additional Samples (Class Conditional ImageNet)	32
A.10	Additional Text-to-Image Samples	37

A.1 Background

A.1.1 Stochastic Interpolants

Flow matching or Stochastic Interpolants convert data to noise by defining a noising process $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon$, $t \in [0, 1]$. A commonly adopted choice is the straight-line interpolant [29] where $\alpha_t = 1 - t$ and $\sigma_t = t$. The marginals of $\mathbf{x}_t$ are transported by the velocity field $\mathbf{v}(\mathbf{x}_t, t) = \mathbb{E}[\dot{\alpha}_t \mathbf{x}_0 + \dot{\sigma}_t \epsilon \mid \mathbf{x}_t]$, which a network $\mathbf{v}_\theta$ learns by regressing onto the conditional velocity target,

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t, \mathbf{x}_0, \epsilon} [\|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\dot{\alpha}_t \mathbf{x}_0 + \dot{\sigma}_t \epsilon)\|_2^2]. \quad (12)$$

During inference, the *probability-flow* ODE $d\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t) dt$, started from $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, converts noise back to data. For all our experiments in this work, we use the widely adopted Scalable Interpolant Transformer (SiT) as the denoiser backbone in combination with the straight-line interpolants.

A.1.2 Representation Alignment

Representation alignment adds an auxiliary loss to the velocity matching objective in Eq. 12 that pulls the hidden states of the velocity field toward a semantic target. Let $\mathbf{h}_t = \mathbf{v}_\theta^{(\ell)}(\mathbf{x}_t, t)$ denote the hidden state of the denoiser $\mathbf{v}_\theta$ after ℓ SiT blocks. The main idea behind representation alignment is to align the outputs $\mathbf{y} = f_{\text{pre}}(\mathbf{x}_0) \in \mathbb{R}^{N \times D}$ with the intermediate layer representations $\mathbf{h}_t$, where f_{pre} is a frozen pretrained encoder such as DINOv2, applied to the clean image. A trainable MLP projector h_ϕ maps the hidden state to the encoder’s dimension, and the two are aligned token by token with a cosine loss,

$$\mathcal{L}_{\text{REPA}}(\theta, \phi) = -\mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\frac{1}{N} \sum_{n=1}^N \cos(\mathbf{y}^{[n]}, h_\phi(\mathbf{h}_t)^{[n]}) \right], \quad \mathcal{L} = \mathcal{L}_{\text{FM}} + \lambda \mathcal{L}_{\text{REPA}}, \quad (13)$$

where $\lambda > 0$ is a scalar hyperparameter. In most representation alignment methods, the projector is discarded after training i.e. at sampling time the denoiser receives $\mathbf{x}_t$ alone. In contrast, we propose a class of diffusion models where representations are natively embedded within the denoising process and help guide generation (see Sec. 2). We show that this can largely benefit a number of aspects during training and inference of diffusion models (see Sec. 3).

A.2 Derivations

A.2.1 Variational Bound

We derive the variational bound in Eq. 4, using the noising process in Eq. 3 as the variational distribution. Throughout, $t \in \{1, \dots, T\}$ indexes a uniform grid on $[0, 1]$, with $t = T$ corresponding to the terminal time 1 of the interpolant. Recall the generative process,

$$p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\phi}(\mathbf{u}_t | \mathbf{x}_t) p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t), \quad (14)$$

and the noising process,

$$q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}), \quad (15)$$

where $\mathbf{x}_{\leq t} = (\mathbf{x}_0, \dots, \mathbf{x}_t)$. The reference policy may depend on any subset $\mathbf{x}_S$ of these states, $S \subseteq \{0, \dots, t\}$, i.e., $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = q_{\theta'}(\mathbf{u}_t | \mathbf{x}_S)$; for example, $S = \{0\}$ for the pretrained encoder in Sec. 2.3. Conditioning on $\mathbf{x}_{\leq t}$ covers all such choices, and the derivation below holds for any of them, including a reference policy whose parameters θ' are not optimized, such as a frozen pretrained encoder. We also use that the noising process is Markov, i.e., $q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)$.

Using the noising process as the variational distribution and applying Jensen’s inequality, we obtain the evidence lower bound,

$$\begin{aligned} \log p(\mathbf{x}_0) &= \log \mathbb{E}_{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} \left[\frac{p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T})}{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} \right] \\ &\geq \mathbb{E}_{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} \left[\log \frac{p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T})}{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} \right]. \end{aligned} \quad (16)$$

Substituting the factorizations in Eqs. 14 and 15, the log-ratio in Eq. 16 becomes

$$\log \frac{p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T})}{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} = \log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_{\phi}(\mathbf{u}_t | \mathbf{x}_t) p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})}. \quad (17)$$

For $t \geq 2$, the Markov property and Bayes’ rule give $q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t | \mathbf{x}_0) / q(\mathbf{x}_{t-1} | \mathbf{x}_0)$. In the product over t , the ratios of marginals cancel and, together with the factor $q(\mathbf{x}_1 | \mathbf{x}_0)$ at $t = 1$, leave only $q(\mathbf{x}_T | \mathbf{x}_0)$:

$$\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_T | \mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0). \quad (18)$$

Substituting Eq. 18 into Eq. 17 and grouping the factors of each step,

$$\begin{aligned} \log \frac{p_{\theta, \phi}(\mathbf{x}_{0:T}, \mathbf{u}_{1:T})}{q_{\theta'}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | \mathbf{x}_0)} &= \sum_{t=1}^T \left[\log p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t) + \log \frac{p_{\phi}(\mathbf{u}_t | \mathbf{x}_t)}{q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})} \right] \\ &\quad - \sum_{t=2}^T \log q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) + \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T | \mathbf{x}_0)}. \end{aligned} \quad (19)$$

Reducing the expectations. The summand with index t in Eq. 19 depends only on $(\mathbf{x}_{\leq t}, \mathbf{u}_t)$, and no state of the noising process depends on a representation, so all other variables integrate out of the expectation in Eq. 16. We write $\mathbb{E}_{\mathbf{x}_{\leq t}, \mathbf{u}_t} := \mathbb{E}_{q(\mathbf{x}_{1:t}|\mathbf{x}_0)} \mathbb{E}_{q_{\theta'}(\mathbf{u}_t|\mathbf{x}_{\leq t})}$ for the resulting expectation and $\mathbb{E}_{\mathbf{x}_{\leq t}}$ for its outer expectation.

Simplifying the terms. The first term of Eq. 19 gives the denoising term $\mathbb{E}_{\mathbf{x}_{\leq t}, \mathbf{u}_t} [\log p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)]$. For the remaining terms,

$$\mathbb{E}_{\mathbf{x}_{\leq t}, \mathbf{u}_t} \left[\log \frac{p_{\phi}(\mathbf{u}_t | \mathbf{x}_t)}{q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})} \right] = -\mathbb{E}_{\mathbf{x}_{\leq t}} [D_{\text{KL}}(q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) \| p_{\phi}(\mathbf{u}_t | \mathbf{x}_t))], \quad (20)$$

$$-\mathbb{E}_{\mathbf{x}_{\leq t}, \mathbf{u}_t} [\log q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)] = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [\mathcal{H}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0))] =: \mathcal{H}_t, \quad (21)$$

$$\mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T | \mathbf{x}_0)} \right] = -D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T)). \quad (22)$$

Eq. 20 is the alignment term. Eq. 21 holds because the integrand depends only on $(\mathbf{x}_{t-1}, \mathbf{x}_t)$, whose marginal is $q(\mathbf{x}_t | \mathbf{x}_0) q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ by the chain rule.

Combining. Substituting these terms into Eq. 16, we arrive at

$$\begin{aligned} \log p(\mathbf{x}_0) \geq \sum_{t=1}^T & \left[\underbrace{\mathbb{E}_{\mathbf{x}_{\leq t}, \mathbf{u}_t} [\log p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)]}_{\text{Denoising}} \right. \\ & \left. - \underbrace{\mathbb{E}_{\mathbf{x}_{\leq t}} [D_{\text{KL}}(q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) \| p_{\phi}(\mathbf{u}_t | \mathbf{x}_t))]}_{\text{Alignment}} \right] + C, \end{aligned} \quad (23)$$

where $C = \sum_{t=2}^T \mathcal{H}_t - D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))$ is determined by the noising process alone and does not depend on (θ, ϕ) . The KL term in C vanishes when $q(\mathbf{x}_T | \mathbf{x}_0) = p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, as for the interpolant in Sec. 2.3. This reduces Eq. 23 to Eq. 4, which concludes the proof.

Remark. In standard diffusion models, the denoising term and the entropy $\mathcal{H}_t$ combine into $-\mathbb{E}[D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))]$. In RAD, the same holds for $t \geq 2$ only if $\mathbf{u}_t$ is conditionally independent of $\mathbf{x}_{t-1}$ given $(\mathbf{x}_t, \mathbf{x}_0)$, e.g., for $S = \{0\}$. In general, $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})$ depends on $\mathbf{x}_{t-1}$, and the denoising term keeps the log-likelihood form of Eq. 4.

A.2.2 Parameterization

We now specify the distributions in Eq. 4. We begin with the two distributions of the alignment term, the reference policy $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})$ and the representation policy $p_{\phi}(\mathbf{u}_t | \mathbf{x}_t)$. We then describe the representation-augmented denoiser $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)$ and the forward process, and close with the resulting objective.

Reference and representation policies. A frozen pretrained encoder f_{pre} extracts the intermediate representations $\mathbf{u}_t$ from the clean image $\mathbf{x}_0$. Following REPA, f_{pre} can be DINOv2 or any other pretrained model that produces semantically meaningful features. Since the supervision comes from $\mathbf{x}_0$ alone, the reference policy drops its dependence on the trajectory and carries no learnable parameters, $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = q(\mathbf{u}_t | \mathbf{x}_0)$. We parameterize it, together with the representation policy, as Gaussians:

$$q(\mathbf{u}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t f_{\text{pre}}(\mathbf{x}_0), \beta^2 \sigma_t^2 \mathbf{I}), \quad (24)$$

$$p_{\phi}(\mathbf{u}_t | \mathbf{x}_t) = \mathcal{N}(\alpha_t h_{\phi}(\mathbf{h}_t), \beta^2 \sigma_t^2 \mathbf{I}), \quad \mathbf{h}_t = \mathbf{v}_{\theta}^{(\ell)}(\mathbf{x}_t, t). \quad (25)$$

Here (α_t, σ_t) is the interpolant schedule of App. A.1.1, shared between $\mathbf{x}_t$ and $\mathbf{u}_t$ so that the representation shares the noise schedule of the state: $\mathbf{u}_t$ is pure noise at the terminal time and approaches the clean features $f_{\text{pre}}(\mathbf{x}_0)$ as $t \rightarrow 0$. On the generative side, $\mathbf{h}_t$ is the hidden state of the denoiser $\mathbf{v}_{\theta}$ after block ℓ , h_{ϕ} is an MLP projector that maps it to the feature dimension of the target, and we write $\mathbf{h}_{\phi}(\mathbf{x}_t, t) = h_{\phi}(\mathbf{h}_t)$ for the projection. Finally, $\beta > 0$ scales the standard deviation of both policies, so that the two share their covariance. The more expressive flow policy used in our experiments is described in App. A.2.4.

Generative transition and forward process. The generative transition is a Gaussian with fixed variance,

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, \mathbf{u}_t, t), \tilde{\sigma}_{t-1|t}^2 \mathbf{I}), \quad (26)$$

where $\tilde{\sigma}_{t-1|t}^2$ is the variance of the forward posterior $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$. It differs from the standard diffusion transition only in that the mean depends on $\mathbf{u}_t$. We parameterize $\boldsymbol{\mu}_\theta$ through a velocity network $\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t)$ by substituting the estimate of $\mathbf{x}_0$ implied by $\mathbf{v}_\theta$ into the mean of the forward posterior.

The forward process is the fixed Gaussian process of App. A.1.1, with marginals $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$.

Training objective. Since the reference and representation policies in Eqs. 24 and 25 are isotropic Gaussians with the same covariance, the KL term of Eq. 4 is available in closed form,

$$D_{\text{KL}}(q(\mathbf{u}_t | \mathbf{x}_0) \| p_\phi(\mathbf{u}_t | \mathbf{x}_t)) = \frac{\text{SNR}(t)}{2\beta^2} \|f_{\text{pre}}(\mathbf{x}_0) - h_\phi(\mathbf{h}_t)\|_2^2, \quad \text{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}, \quad (27)$$

Substituting it into Eq. 4 gives, after reweighting the time steps, the training objective in Eq. 9 (App. A.2.3 gives the derivation). No gradient is propagated through $f_{\text{pre}}(\mathbf{x}_0)$, and f_{pre} is not required at sampling time: each step draws $\mathbf{u}_t$ from the model’s own policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$.

A.2.3 Simplified Training Objective

We derive Eq. 9 from the bound in Eq. 23 under the parameterization of App. A.2.2. Here, $\alpha_t, \sigma_t, \dot{\alpha}_t, \dot{\sigma}_t$ denote the interpolant schedule and its time derivative at the corresponding grid time. Since in our experiments the reference policy depends on $\mathbf{x}_0$ alone, $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = q(\mathbf{u}_t | \mathbf{x}_0)$, the alignment term depends on $(\mathbf{x}_0, \mathbf{x}_t)$ only, and $\mathbf{u}_t$ is conditionally independent of $\mathbf{x}_{t-1}$ given $(\mathbf{x}_t, \mathbf{x}_0)$.

Alignment term. Both $q(\mathbf{u}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t f_{\text{pre}}(\mathbf{x}_0), \beta^2 \sigma_t^2 \mathbf{I})$ and $p_\phi(\mathbf{u}_t | \mathbf{x}_t) = \mathcal{N}(\alpha_t \mathbf{h}_\phi(\mathbf{x}_t, t), \beta^2 \sigma_t^2 \mathbf{I})$ are isotropic Gaussians with the same covariance, so their KL divergence reduces to the squared distance between the means scaled by the precision $1/(\beta^2 \sigma_t^2)$,

$$D_{\text{KL}}(q(\mathbf{u}_t | \mathbf{x}_0) \| p_\phi(\mathbf{u}_t | \mathbf{x}_t)) = \frac{\text{SNR}(t)}{2\beta^2} \|f_{\text{pre}}(\mathbf{x}_0) - \mathbf{h}_\phi(\mathbf{x}_t, t)\|_2^2, \quad \text{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}. \quad (28)$$

If the reference variance is chosen differently from $\beta^2 \sigma_t^2$, the KL changes only by terms that do not depend on ϕ , since the mean term involves the variance of p_ϕ alone.

Denoising term. The forward posterior of the Gaussian noising process is [21]

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_{t-1|t}^2 \mathbf{I}), \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = a_t \mathbf{x}_t + b_t \mathbf{x}_0, \quad (29)$$

with $a_t = \alpha_{t|t-1} \sigma_{t-1}^2 / \sigma_t^2$, $b_t = \alpha_{t-1} \sigma_{t|t-1}^2 / \sigma_t^2$ and $\tilde{\sigma}_{t-1|t}^2 = \sigma_{t|t-1}^2 \sigma_{t-1}^2 / \sigma_t^2$, where $\alpha_{t|t-1} = \alpha_t / \alpha_{t-1}$ and $\sigma_{t|t-1}^2 = \sigma_t^2 - \alpha_{t|t-1}^2 \sigma_{t-1}^2$ are the coefficients of the transition $q(\mathbf{x}_t | \mathbf{x}_{t-1})$. For $t = 1$, $\mathbf{x}_0$ is observed and $(a_1, b_1) = (0, 1)$; we then use a fixed decoder variance in place of $\tilde{\sigma}_{0|1}^2$. Since $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)$ is a Gaussian with the same variance, the expectation over $\mathbf{x}_{t-1}$ in the denoising term is

$$\mathbb{E}_{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)} [\log p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)] = -\frac{1}{2\tilde{\sigma}_{t-1|t}^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, \mathbf{u}_t, t)\|_2^2 + \text{const}, \quad (30)$$

where the constant collects the entropy of the posterior and the normalizer of p_θ , neither of which depends on θ . We parameterize $\boldsymbol{\mu}_\theta$ through the velocity network by substituting the estimate of $\mathbf{x}_0$ implied by $\mathbf{v}_\theta$ into the posterior mean. Solving $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$ and $\mathbf{v}_{\text{ref}} = \dot{\alpha}_t \mathbf{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$ for $\mathbf{x}_0$ gives

$$\mathbf{x}_0 = \frac{\sigma_t \mathbf{v}_{\text{ref}} - \dot{\sigma}_t \mathbf{x}_t}{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}, \quad \hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) := \frac{\sigma_t \mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) - \dot{\sigma}_t \mathbf{x}_t}{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}, \quad (31)$$

and we set $\mu_\theta(\mathbf{x}_t, \mathbf{u}_t, t) := \tilde{\mu}_t(\mathbf{x}_t, \hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{u}_t, t))$. For the straight-line interpolant, $\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t = -1$. Since $\tilde{\mu}_t$ is affine in its second argument, Eq. 31 yields $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, \mathbf{u}_t, t) = b_t(\mathbf{x}_0 - \hat{\mathbf{x}}_\theta) = b_t \sigma_t(\mathbf{v}_{\text{ref}} - \mathbf{v}_\theta)/(\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t)$, and Eq. 30 becomes a weighted velocity regression,

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} [\log p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t)] &= -w_t \|\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) - \mathbf{v}_{\text{ref}}\|_2^2 + \text{const}, \\ w_t &= \frac{b_t^2 \sigma_t^2}{2\tilde{\sigma}_{t-1|t}^2 (\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t)^2}. \end{aligned} \quad (32)$$

Combining. Substituting Eqs. 28 and 32 into Eq. 23 and writing the noising process as $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon$ with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the negative log-likelihood is bounded by

$$-\log p(\mathbf{x}_0) \leq T \mathbb{E}_{t, \epsilon, \mathbf{u}_t \sim q} \left[w_t \|\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) - \mathbf{v}_{\text{ref}}\|_2^2 + \frac{\lambda_t}{2} \|f_{\text{pre}}(\mathbf{x}_0) - \mathbf{h}_\phi(\mathbf{x}_t, t)\|_2^2 \right] + \text{const}, \quad (33)$$

where $\lambda_t = \text{SNR}(t)/\beta^2$, t is uniform on $\{1, \dots, T\}$, and $\mathbf{u}_t \sim q(\mathbf{u}_t | \mathbf{x}_0)$. Following common practice [19], we replace the weight w_t of the denoising term by one, which changes the relative weighting of the time steps in the denoising term. Since T only scales the bound, we drop it, and we take the continuous-time limit of the uniform grid by sampling $t \sim \mathcal{U}(0, 1)$. Dropping the constant and taking the expectation over $\mathbf{x}_0$ gives Eq. 9, which completes the derivation.

The alignment loss penalizes deviation of the representation policy from the reference policy much more at higher SNR levels. Intuitively, the alignment loss ensures that the representation policy recovers more semantic information when the current state is cleaner (i.e., higher SNR values), thus encoding the uncertainty in the diffusion process. Moreover, as $t \rightarrow 0$, the SNR and therefore the alignment weight λ_t grows unbounded. Therefore, we instead use a bounded version (between 0 and 1) of the alignment weight. In practice, we find that different choices of λ_t like $\alpha_t^2/(\beta^2(1 + \sigma_t^2))$ and $\text{SNR}(t)/(\beta^2 + \text{SNR}(t))$ work equally well without changing the overall semantics of the alignment loss weighting. We choose the latter for the remainder of our discussion.

A.2.4 Practical Design Choices

Choice of policy. We refer to the representation policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$ simply as the policy. Under the Gaussian parameterization of Eq. 6, a sample takes the form $\mathbf{u}_t = \alpha_t h_\phi(\mathbf{h}_t) + \beta \sigma_t \bar{\epsilon}$ with $\bar{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, a Gaussian policy is unimodal which limits expressivity. To obtain a more expressive policy, we replace the noise with the output of a conditional normalizing flow [8],

$$\mathbf{u}_t = \alpha_t h_\phi(\mathbf{h}_t) + \beta \sigma_t \mathbf{g}_\zeta(\mathbf{h}_t, \bar{\epsilon}), \quad \bar{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (34)$$

where $\mathbf{g}_\zeta(\mathbf{h}_t, \bar{\epsilon})$ is a stack of affine coupling layers whose conditioning is a linear map of $h_\phi(\mathbf{h}_t)$, and ζ is trained jointly with ϕ . Consequently,

$$\bar{\epsilon} = \mathbf{g}_\zeta^{-1}\left(\mathbf{h}_t, \frac{\mathbf{u}_t - \alpha_t h_\phi(\mathbf{h}_t)}{\beta \sigma_t}\right). \quad (35)$$

Intuitively, the second argument in Eq. 35 represents the whitened version of $\mathbf{u}_t$ which we denote as $\mathbf{z}_t$, for brevity. Next, the KL term of Eq. 4 can be simplified as,

$$D_{\text{KL}}(q(\mathbf{u}_t | \mathbf{x}_0) \| p_\phi(\mathbf{u}_t | \mathbf{x}_t)) = -\mathbb{E}_{\mathbf{u}_t \sim q} [\log p_\phi(\mathbf{u}_t | \mathbf{x}_t)] + \text{const}. \quad (36)$$

The above result follows directly from the definition of the KL divergence. The additional constant term is the negative entropy of the reference policy, which does not depend on the parameters of the representation policy and can be ignored during optimization. The KL can be further simplified as,

$$-\mathbb{E}_{\mathbf{u}_t \sim q} [\log p_\phi(\mathbf{u}_t | \mathbf{x}_t)] = -\mathbb{E}_{\mathbf{u}_t \sim q} \left[\log p_{\bar{\epsilon}}(\bar{\epsilon} = \mathbf{g}_\zeta^{-1}(\mathbf{h}_t, \mathbf{z}_t)) + \log \left| \det \frac{\partial \mathbf{g}_\zeta^{-1}}{\partial \mathbf{z}_t} \right| \right] + \text{const}, \quad (37)$$

$$= \mathbb{E}_{\mathbf{u}_t \sim q} \left[\frac{1}{2} \|\mathbf{g}_\zeta^{-1}(\mathbf{h}_t, \mathbf{z}_t)\|_2^2 - \log \left| \det \frac{\partial \mathbf{g}_\zeta^{-1}}{\partial \mathbf{z}_t} \right| \right] + \text{const}, \quad (38)$$

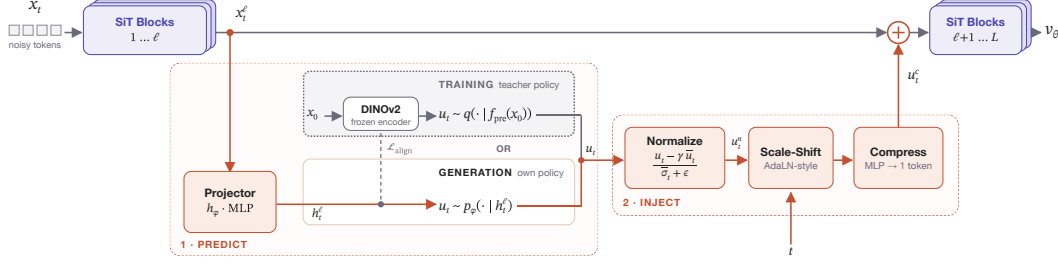


Figure 7: RAD architecture. (1) Predict: the projector h_ϕ maps the hidden state $\mathbf{x}_t^\ell$ after the first ℓ SiT blocks to the representation policy. During training, $\mathbf{u}_t$ is sampled from the teacher policy $q(\cdot | f_{\text{pre}}(\mathbf{x}_0))$, and during generation from the model’s own policy $p_\phi(\cdot | h_t^\ell)$. (2) Inject: $\mathbf{u}_t$ is normalized, compressed to a single token and modulated (Eq. 8), then added to the hidden state before the remaining SiT blocks predict $\mathbf{v}_\theta$.

where the constants collect $d \log(\beta \sigma_t)$ from the change of variables from $\mathbf{u}_t$ to $\mathbf{z}_t$ and the normalizer of $p_{\bar{\epsilon}}$, neither of which depends on (ϕ, ζ) . Therefore,

$$\ell_{\text{flow}}(\phi, \zeta) = \mathbb{E}_{\mathbf{u}_t \sim q} \left[\frac{1}{2} \|\mathbf{g}_\zeta^{-1}(\mathbf{h}_t, \mathbf{z}_t)\|_2^2 - \log \left| \det \frac{\partial \mathbf{g}_\zeta^{-1}}{\partial \mathbf{z}_t} \right| \right], \quad \mathbf{z}_t = \frac{\mathbf{u}_t - \alpha_t h_\phi(\mathbf{h}_t)}{\beta \sigma_t}, \quad (39)$$

At the start of training, the flow is initialized to the identity, i.e., $\mathbf{g}_\zeta^{-1}(\mathbf{h}_t, \mathbf{z}_t) = \mathbf{z}_t$. The log-determinant then vanishes and, in expectation over $\mathbf{u}_t$, the first term equals the alignment term of Eq. 27 up to an additive constant. In practice, we find that keeping the explicit alignment term helps training stability. Therefore, we combine the two with complementary time weights. The final RAD training objective with a flow policy then reads,

$$\mathcal{L}(\theta, \phi, \zeta) = \mathbb{E} \left[\left\| \mathbf{v}_\theta(\mathbf{x}_t, \mathbf{u}_t, t) - (\dot{\alpha}_t \mathbf{x}_0 + \dot{\sigma}_t \epsilon) \right\|_2^2 + \frac{\lambda_t}{2} \|\mathbf{f}_{\text{pre}}(\mathbf{x}_0) - h_\phi(\mathbf{h}_t)\|_2^2 + (1 - \lambda_t) \ell_{\text{flow}}(\phi, \zeta) \right] \quad (40)$$

where $\lambda_t = \text{SNR}(t) / (\beta^2 + \text{SNR}(t))$ and $\mathbf{u}_t \sim q(\mathbf{u}_t | \mathbf{x}_0)$ is drawn from the reference policy as before, the outer expectation is over t , $\mathbf{x}_0$ and the noise. It is worth noting that the objective in Eq. 40 is only a practical surrogate and doesn’t correspond to the exact ELBO bound. During training, we fix $\beta = 1.0$ and vary it during inference to control the temperature of the policy which improves sample quality without affecting diversity (Fig. 9).

Architecture. We build on SiT. We first compute the hidden state $\mathbf{h}_t$, from which the policy draws $\mathbf{u}_t \in \mathbb{R}^{N \times D}$ from the representation policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$. We then process the controls as follows before feeding it back into the denoiser.

- **Control Normalization.**

$$\mathbf{u}_t^n = \frac{\mathbf{u}_t - \gamma \bar{\mathbf{u}}_t}{\bar{\sigma}_t + \delta},$$

where $\bar{\mathbf{u}}_t$ and $\bar{\sigma}_t$ are the mean and standard deviation of $\mathbf{u}_t$ across its N tokens, $\gamma \in [0, 1]$ controls how much of the mean is removed, and δ is a small constant. This helps remove global representation information from the injected representation which helps with downstream sample quality [43]. Throughout this work, we set $\gamma = 0.6$ and $\delta = 10^{-6}$.

- **Compression.** We then aggregate the normalized representation along the token axis using an MLP of $\mathbf{u}_t^n$ to M tokens,

$$\mathbf{u}_t^c = \text{MLP}(\mathbf{u}_t^n) \in \mathbb{R}^{M \times D},$$

where we use $M = 1$. A linear control projection then maps $\mathbf{u}_t^c$ from the encoder dimension D to the embedding dimension of the backbone, so that it can be added to the hidden state.

- **Modulation.**

$$\mathbf{u}_t^f = (1 + \mathbf{a}(c)) \odot \mathbf{u}_t^c + \mathbf{b}(c),$$

where $\mathbf{a}$ and $\mathbf{b}$ are linear functions of the conditioning embedding $\mathbf{c}$, as in adaLN, and are zero-initialized so that training starts from the unmodulated representation.

The result is added to every token of the hidden state, $\bar{\mathbf{h}}_t^{[n]} = \mathbf{h}_t^{[n]} + \mathbf{u}_t^f$. We inject early, with ℓ within the first third of the backbone, which we find to be the most influential architectural choice (Table 6).

A.3 Related Work

A.3.1 Contextualizing RAD within Generative Modeling

External Representation Alignment methods align the intermediate representations of the diffusion denoiser with the features of a frozen pretrained vision encoder like DINOv2 [34]. REPA [59] pulls the hidden states of a diffusion transformer toward the DINOv2 features of the clean image, which leads to an order-of-magnitude gain in training efficiency. Consequently, different aspects of external alignment have received a lot of recent interest. iREPA [43] shows that the spatial structure of the target features, rather than their global semantic content, drives these gains, and speeds up convergence by replacing the projector h_ϕ with a convolutional layer and normalizing the target tokens across space to suppress their shared global component. REPA-E [26] trains the VAE and the denoiser end to end using the REPA loss. In all these methods, the representation acts as a training-time regularizer, and the learned representations are never injected back into the denoiser during sampling. In contrast, RAD additionally conditions every denoising step on a latent $\mathbf{u}_t$ drawn from the representation policy $p_\phi(\cdot | \mathbf{x}_t)$, and the alignment loss arises as a KL term in a likelihood bound instead of as a heuristic addition to the denoising loss. Another line of work aligns the autoencoder’s latent space with the encoder features [58] or replaces the autoencoder with a frozen encoder and a trained decoder [63]. Exploring design choices related to the VAE and other architectural modifications as proposed in these works in the context of RAD could be an interesting direction for future work.

Closest to our work, REED [48] also proposes a general variational framework for representation alignment. The main idea is to introduce a *global representation* $\mathbf{z}$, sampled once along the reverse process, on which subsequent denoising steps are conditioned. However, in practice, REED drops this explicit conditioning (along with a normalization term), reducing its objective to the standard REPA objective with multimodal alignment targets and a training curriculum. This is supported by an ablation (their Table 10) in which the denoiser is conditioned on the ground-truth encoder representation during training but sampled from the unconditional branch; this variant performs on par with REPA. In contrast, RAD samples a latent $\mathbf{u}_t$ for each transition from a learned policy whose noise level matches that of the state, and conditions the denoiser on it explicitly. Because $\mathbf{u}_t$ is stochastic, each transition is a continuous mixture of Gaussians, whereas REED’s transitions at sampling time, conditioned implicitly on a point estimate, remain single Gaussians. Our bound has no normalization term and recovers REPA as an exact special case when the denoiser ignores $\mathbf{u}_t$, using the same von Mises–Fisher argument as in Wang et al. [48] (see also App. A.3.2 for a more formal discussion).

Finally, another line of work makes the encoder representation part of the diffusion state and denoises it jointly with the image latents: REG [53] denoises the encoder’s class token alongside the image tokens, and ReDi [22] jointly diffuses VAE latents and dimension-reduced DINOv2 features. In these methods the representation follows its own noising trajectory and is generated together with the image. In RAD, the representation is not part of the diffusion state: a fresh latent $\mathbf{u}_t$ is predicted from the current state at every step and conditions only that transition. The two ideas are complementary, and combining RAD with REG further improves convergence (Sec. 3.4).

Internal Representation Alignment methods remove the external encoder and let the model supervise itself. MaskDiT [64] drops a subset of tokens and asks the network to reconstruct them while denoising. SRA [17] aligns the features of an earlier block at a higher noise level with those of a deeper block of an EMA teacher at a lower noise level. Another line of work [3, 66] noises the student’s tokens and pulls its features toward an EMA teacher that sees the cleaner view, which yields a self-supervised signal. SRA-like methods share the design of external alignment: an auxiliary loss on the hidden states that leaves the generative model unchanged. While the RAD framework allows for incorporating internal representation alignment via parameterization of the reference policy $q_{\theta'}$, in our initial experiments, we find that adapting SRA with RAD yields only modest

gains (2–3 FID points on the B/2 model scale for class-conditional ImageNet generation) and therefore restrict our discussion of RAD to external alignment methods. However, adapting RAD for internal alignment would be an interesting direction for further work. Lastly, approaches based on dispersive loss [49] instead spread the hidden states apart with a contrastive-style regularizer that needs no positive pairs, which could be interesting to explore in the context of RAD.

Combining VAEs with Diffusion Models. LSGM [47] and D2C [44] train a VAE with a diffusion prior over its latents. Diffusion autoencoders [38], PDAE [61], and InfoDiffusion [51] learn a global latent with an auxiliary encoder and condition the entire denoising trajectory on it, whereas DiffuseVAE [36] pairs a VAE with a diffusion decoder that refines its output. DisCo-Diff [57] similarly augments the denoiser with a few discrete global latents, inferred by an encoder that is trained end to end with the denoiser, to simplify the noise-to-data mapping, and fits an autoregressive prior over these latents in a second stage. More recently, RCG [27] generates a self-supervised representation with a diffusion prior and conditions image generation on it. RAD shares the idea of a latent that steers generation but differs in two respects. First, the latent $\mathbf{u}_t$ is per-step with its own prior $p_\phi(\cdot | \mathbf{x}_t)$ conditioned on the current state, so the model is a conditional VAE [45] at every transition rather than a single global code. Second, the per-step prior p_ϕ is trained jointly with the denoiser under a single bound, whose KL term recovers the representation-alignment loss. Methods like RCG, DisCo-Diff, and DiffuseVAE instead train the latent prior or the image generator in a separate stage, and DisCo-Diff learns its latents from scratch rather than aligning them with a pretrained representation. Another work DDGAN [55] models each denoising transition with a conditional GAN so that the transition can be multimodal and large denoising steps remain accurate. RAD shares this motivation (Eq. 2), but its per-step latent has a learned, state-conditioned prior, is trained under a variational bound rather than adversarially, and is aligned with a semantic representation rather than being unstructured noise.

Augmented Diffusion Models such as CLD [9] and PSLD [35] extend the diffusion state with auxiliary velocity or momentum variables that are coupled to the data through a fixed forward SDE, which smooths the diffusion paths, simplifies score learning, and improves sample quality at a given sampling budget. As in REG and ReDi, these auxiliary variables are part of the diffusion state and are carried across steps. In contrast, RAD’s latent $\mathbf{u}_t$ is drawn at every step from a learned, state-conditioned policy and aligned with a semantic representation.

A.3.2 Prior Work as Special Cases under RAD

Prior work on representation alignment can be broadly categorized by the choice of the reference policy $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t})$: an external pretrained model such as DINOv2 (referred to as external alignment) or the denoiser itself (internal alignment). Our framework accommodates both, and the two settings differ only in the choice of the reference policy.

External alignment. In external alignment, a frozen pretrained encoder f_{pre} such as DINOv2 extracts the target representation from the clean image $\mathbf{x}_0$, so the reference policy depends on $\mathbf{x}_0$ alone, $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = q(\mathbf{u}_t | \mathbf{x}_0)$. With this assumption in place, REPA is recovered by two choices. First, we parameterize the denoiser to ignore $\mathbf{u}_t$, $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t) = p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$, so that the denoising term of Eq. 4 reduces to the velocity-matching objective of Eq. 12. Second, we parameterize both the reference and the representation policy distributions as von Mises–Fisher (vMF) distributions on the unit sphere, token by token, with a shared concentration $\kappa > 0$,

$$q(\mathbf{u}_t^{[n]} | \mathbf{x}_0) = \text{vMF}(\bar{\mathbf{y}}^{[n]}, \kappa), \quad p_\phi(\mathbf{u}_t^{[n]} | \mathbf{x}_t) = \text{vMF}(\bar{h}_\phi(\mathbf{h}_t)^{[n]}, \kappa), \quad (41)$$

where $\mathbf{y} = f_{\text{pre}}(\mathbf{x}_0)$, $\mathbf{h}_t$ is the hidden state of the denoiser after block ℓ , h_ϕ is the projector of Eq. 25, and the bar denotes ℓ_2 normalization. Since $\text{vMF}(\boldsymbol{\mu}, \kappa) \propto \exp(\kappa \boldsymbol{\mu}^\top \mathbf{u})$, the KL divergence between two vMF distributions with the same concentration can be stated as,

$$D_{\text{KL}}(\text{vMF}(\boldsymbol{\mu}_1, \kappa) \| \text{vMF}(\boldsymbol{\mu}_2, \kappa)) \propto (1 - \boldsymbol{\mu}_1^\top \boldsymbol{\mu}_2). \quad (42)$$

Summing over tokens, the alignment term of Eq. 4 becomes $\sum_n (1 - \cos(\mathbf{y}^{[n]}, h_\phi(\mathbf{h}_t)^{[n]}))$, which is REPA’s cosine alignment loss of Eq. 13 up to an additive constant and a positive scale, which plays the role of λ . The bound therefore reduces to the REPA objective and REPA becomes a special case in our framework.

Internal alignment. In this setting, the representations are produced by the denoiser itself, as in SRA. The target is the hidden state $\mathbf{v}_{\theta'}^{(\ell')}(x_{t-k}, t-k)$ after block ℓ' of an exponential moving average (EMA) copy $\mathbf{v}_{\theta'}$ of the denoiser, evaluated at a less noisy state $\mathbf{x}_{t-k}$, $k \in \{1, \dots, t\}$, of the same trajectory. Since the target depends on $\mathbf{x}_{t-k}$ alone, the reference policy reduces to $q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{\leq t}) = q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{t-k})$, which we parameterize as

$$q_{\theta'}(\mathbf{u}_t | \mathbf{x}_{t-k}) = \mathcal{N}(\alpha_t \mathbf{v}_{\theta'}^{(\ell')}(x_{t-k}, t-k), \beta^2 \sigma_t^2 \mathbf{I}). \quad (43)$$

The parameters θ' track θ through the EMA update and receive no gradient. The representation policy $p_\phi(\mathbf{u}_t | \mathbf{x}_t)$ is unchanged from Eq. 25: the model at the noisier state $\mathbf{x}_t$ predicts the features that its EMA copy extracts from the cleaner state $\mathbf{x}_{t-k}$. For $k = t$ the target is computed from the clean image, as in the external setting, with $\mathbf{v}_{\theta'}^{(\ell')}(x_0, 0)$ in place of $f_{\text{pre}}(x_0)$.

A.4 Experimental Details

Data and latents. For class-conditional generation we use the ImageNet training set [6] of 1.28M images at 256×256 , center-cropped as in ADM [7], without further augmentation. Following REPA, images are encoded once with the Stable Diffusion VAE [41] into $32 \times 32 \times 4$ latents. At every training step, a latent is drawn from the stored posterior and scaled by 0.18215. The external supervisor f_{pre} is a frozen DINOv2-B/14 [34]: the 256×256 image is normalized and resized to 224×224 , and the 256 patch tokens of dimension 768 (the class token is dropped) serve both as alignment targets and as the clean representation $f_{\text{pre}}(x_0)$ from which the training-time controls $\mathbf{u}_t$ are noised. For the encoder ablation (Table 6), DINOv2-L and SigLIP-L [60] use the standard preprocessing of the respective models.

Backbones and injection depth. We use the unmodified SiT-B/2 (12 blocks, width 768), SiT-L/2 (24 blocks, width 1024), and SiT-XL/2 (28 blocks, width 1152) architectures with a 2×2 patch size, so that the latent is processed as 256 tokens. The hidden state after block ℓ is aligned with the encoder and, for RAD, $\mathbf{u}_t$ is injected after the same block. RAD-B/2 uses $\ell = 2$ for sample quality (Table 6) and $\ell = 4$ for linear probing. RAD-L/2 and RAD-XL/2 use $\ell = 8$, the depth at which REPA aligns these backbones. In case the control dimension differs from the backbone width (for L/2 or XL/2 backbones), we use a linear layer to map the controls to the backbone width.

Optimization. Following the same settings in REPA, all RAD models are trained with AdamW [30] at a constant learning rate of 10^{-4} without warmup, $(\beta_1, \beta_2) = (0.9, 0.999)$, no weight decay, gradient clipping at norm 1, and a global batch size of 256, in fp16 mixed precision, for 400K iterations unless stated otherwise. We keep an exponential moving average (EMA) of the weights with decay 0.9999, updated every step, and use the EMA weights for every evaluation. The noising process is the linear interpolant $\alpha_t = 1 - t$, $\sigma_t = t$ with $t \sim \mathcal{U}(0, 1)$, the denoiser predicts the velocity, and the class label is dropped with probability 0.1. We use a random seed of 0 throughout.

RAD hyperparameters. We highlight different RAD hyperparameters in Table 7. The projector h_ϕ is the REPA MLP (width 2048, two hidden layers, SiLU) applied to every token. Models with the flow policy are trained with Eq. 40: the alignment term is weighted by $0.5 \lambda_t$ and the flow likelihood by $1 - \lambda_t$, with $\beta \sigma_t$ clamped below by $\sigma_{\min} = 10^{-3}$ in the whitened residual. The adaptive weighting of the alignment loss multiplies the per-sample error e by $1/(\text{sg}(e) + 10^{-3})^p$, where sg stops the gradient. Setting $p = 0$ recovers the unweighted loss, and $p = 1$ is used for the final models. The policy scale is $\beta = 1$ during training. The control pathway of App. A.2.4 uses $\gamma = 0.6$ and $\delta = 10^{-6}$ for the normalization, $M = 1$ control token, and zero-initialized modulation. The flow policy has four affine coupling layers with a hidden width of 512 and a 256-dimensional context computed from $h_\phi(x_t, t)$ by one linear layer. Its last layers are zero-initialized, so training starts from the Gaussian policy. At inference we draw fresh policy noise at every step, keep the

Table 7: RAD Hyperparameters on class-conditional ImageNet at each model scale. Values spanning all three columns are shared across scales.

	RAD-B/2	RAD-L/2	RAD-XL/2
<i>Backbone</i>			
SiT blocks	12	24	28
Hidden width	768	1024	1152
Attention heads	12	16	16
Patch size / tokens		$(2 \times 2) / 256$	
#Params	142M	472M	690M
<i>Representation pathway</i>			
Encoder f_{pre}	DINOv2-B/14 (768-dim patch tokens)		
Alignment and injection depth ℓ	2/4	8	8
Control projection	–	$768 \rightarrow 1024$	$768 \rightarrow 1152$
Projector h_ϕ	MLP, width 2048, two hidden layers, SiLU		
Control Normalization (γ / δ)		$0.6 / 10^{-6}$	
Control tokens M		1	
Flow policy	4 affine couplings, width 512, context 256		
Alignment weight		$0.5 \lambda_t$	
Adaptive weighting p	1.0	1.0	1.0
Training policy scale β		1.0	
<i>Optimization</i>			
Optimizer	AdamW, lr 10^{-4} , (0.9, 0.999), no weight decay		
Batch size / iterations		256 / 400K	
EMA decay		0.9999	
Label dropout		0.1	
Training time (4 GPUs)	11 h	16 h	22 h
<i>Inference</i>			
β_{inf}	0.7	0.8	0.8
Extra FLOPs over REPA	11.8%	3.5%	2.5%

controls active over the whole trajectory, and sweep $\beta_{\text{inf}} \in \{0.5, 0.6, \dots, 0.9\}$ for each checkpoint (Fig. 4c). The best value, 0.7 for B/2 and 0.8 for L/2 and XL/2, is the same at every training budget.

Samplers. The Euler–Maruyama SDE sampler follows REPA and uses 250 steps with diffusion coefficient $w_t = 2t$. The Euler ODE sampler uses N uniform steps from $t = 1$ to $t = 0$, so that N steps cost N network evaluations. UniPC [62] is run with the same step budgets at orders 2 and 3 (Table 15). No classifier-free guidance is used unless stated; the guided sampling protocol is described in App. A.6.

Evaluation. We report FID, sFID, Inception Score, precision, and recall computed with the ADM evaluation suite [7] against its ImageNet 256×256 reference batch, on 50k samples with uniformly drawn class labels, except for the guided results in Table 2, which use class-balanced sampling (App. A.6).

Data efficiency. For the data-efficiency experiment (Fig. 3c, Table 12), we train both REPA-B/2 and RAD-B/2 ourselves with the recipe above on random subsets of 3%, 10%, and 30% of ImageNet and on the full dataset, for the same 400K iterations at batch size 256. We evaluate every checkpoint with the Euler ODE sampler with 50 steps and report the best FID of each run.

Linear Probing. Following REPA, we probe pre-trained EMA models for RAD and REPA with the null class label. At B/2, both REPA and RAD are aligned at block 4. For a diffusion time t , every image is noised once, and the hidden state after block k is mean-pooled over its tokens. A parameter-free batch normalization followed by a linear layer is then trained on the ImageNet training set for 90 epochs with Adam (learning rate 10^{-3} , cosine decay, batch size 16384) and scored by top-1 accuracy on the 50k validation images. No augmentation is used. For RAD, the controls are sampled from the policy as at inference, with $\beta_{\text{inf}} = 0.7$ for B/2 (Fig. 6a) and 0.8 for XL/2. Fig. 6b reports the best block for each t .

REG within RAD. REG denoises the class token of the encoder jointly with the image tokens and weights its class-token loss by 0.03. We keep this recipe unchanged and add the control pathway of RAD with the hyperparameters above. However, following REG we fix the injection depth at $\ell = 4$ for B/2 and $\ell = 8$ for L/2. The projection of the noisy patch tokens serves as the mean of the representation policy, and REG’s class token is not part of the control.

Text-to-image. We train on the pre-training captions of BLIP3o [4] (the JourneyDB and short-caption subsets, about 9M image–caption pairs) at 256×256 with the same VAE latents. We encode the captions using a frozen Qwen3-0.6B [39] model. Its final hidden states (dimension 1024) pass through an RMSNorm and a linear layer and are prepended in context to the 256 patch tokens with a key mask. A pooled caption embedding is added to the adaLN conditioning through a small MLP. We cap the number of caption tokens to 64, which truncates 3.5% of the total captions. The choice of this cap is primarily driven by computational constraints. Similar to the class labels on ImageNet, we drop the entire caption with probability 0.1 by replacing it with a null caption whose tokens are all masked out of attention. The alignment loss, the flow policy, and the controls act on the patch tokens only. RAD-B/2 injects after block 2 and RAD-XL/2 after block 8, both with $p = 1$. At XL/2, both REPA and RAD additionally use QK-normalization in attention, without which we find training to be unstable. Pre-training follows the class-conditional ImageNet recipe (400K iterations with a batch size of 256). For evaluation, we use the Euler ODE sampler with 50 steps. FID is computed against 50k images sampled from the BLIP3o pre-training set such that their captions are diverse. For GenEval [10] benchmarking, we generate four images for each of its 553 prompts and report the mean of its six task accuracies.

For the qualitative results in Figs. 28–36, we perform additional finetuning of RAD-XL/2 on the BLIP3o-60k dataset. We initialize from the pretrained EMA weights and train for 11,250 steps (50 epochs) with the learning rate 5×10^{-4} . We use a CFG scale of 3.0, $\beta = 1.0$ and the SDE sampler with 250 steps for all qualitative samples.

Compute. All experiments run on a single node with 4 NVIDIA RTX PRO 6000 Blackwell GPUs. At the observed throughputs of roughly 10, 7, and 5 iterations per second, 400K iterations take about 11, 16, and 22 hours for B/2, L/2, and XL/2, respectively. The text-to-image XL/2 model takes about 37 hours, and fine-tuning it on BLIP3o-60k takes about 1.5 hours. Linear probing takes about one hour for B/2 and a few hours for XL/2.

A.5 Additional Generation Results

Tables 8 and 9 report the full ImageNet 256×256 results of RAD at each training budget, alongside the numbers reported in the official papers of REPA and REG at the same budgets. All results use the Euler–Maruyama SDE sampler with 250 steps and no classifier-free guidance.

A.6 Guided Sampling

Sampling protocol. All guided results use RAD-XL/2 trained for 400K iterations (80 epochs). We use the Euler–Maruyama SDE sampler with 50 steps on a uniform time grid from $t = 1$ to $t = 0.04$, followed by a final noise-free step to $t = 0$, with $\beta_{\text{inf}} = 0.8$ and fixed policy noise (Sec. 2). Token coupling (Eq. 11) uses $\rho = 0.12$; ξ and η_n are drawn once per sample together with $\mathbf{x}_1$. For CFG, the conditional and unconditional branches share the same representation $\mathbf{u}_t$, sampled from the class-conditional policy, and we use a guidance scale of $s = 3.25$. Autoguidance [18] uses the EMA checkpoint of the same training run at 100K iterations as the weak model $v_{\theta'}$, with the same β_{inf} and the same policy noise, and is applied on top of CFG:

$$\tilde{\mathbf{v}} = \mathbf{v}_{\text{cfg}} + (w - 1)(\mathbf{v}_{\theta}(\mathbf{x}_t, \mathbf{u}_t, t, y) - \mathbf{v}_{\theta'}(\mathbf{x}_t, \mathbf{u}'_t, t, y)), \quad (44)$$

$$\mathbf{v}_{\text{cfg}} = \mathbf{v}_{\theta}(\mathbf{x}_t, \mathbf{u}_t, t, \emptyset) + s(\mathbf{v}_{\theta}(\mathbf{x}_t, \mathbf{u}_t, t, y) - \mathbf{v}_{\theta}(\mathbf{x}_t, \mathbf{u}_t, t, \emptyset)), \quad (45)$$

with $w = 1.5$, where $\mathbf{u}'_t$ is sampled from the weak model’s own policy. Both guidance terms are applied only for $t \leq 0.75$ [25]. Of the 50 steps, 37 fall in this interval and require three network evaluations each, so a sample costs 124 network evaluations in total.

Table 8: Full ImageNet 256×256 results of SiT, REPA, and RAD at every training budget (numbers behind Table 1, Fig. 1c, and Fig. 3a,b). Baseline numbers are from the respective papers. All metrics on 50k samples, Euler–Maruyama SDE sampler with 250 steps, no guidance. RAD uses $\beta_{\text{inf}} = 0.7$ for B/2 and 0.8 for L/2 and XL/2 at every budget.

Model	#Params	Iter.	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
SiT-B/2	130M	400K	33.0	6.46	43.7	0.53	0.63
+ REPA	130M	50K	78.2	11.71	17.1	0.33	0.48
+ REPA	130M	100K	49.5	7.00	27.5	0.46	0.59
+ REPA	130M	200K	33.2	6.68	43.7	0.54	0.63
+ REPA	130M	400K	24.4	6.40	59.9	0.59	0.65
+ RAD (ours)	142M	50K	55.6	21.94	25.4	0.44	0.46
+ RAD (ours)	142M	100K	23.5	9.05	64.8	0.64	0.51
+ RAD (ours)	142M	200K	12.0	7.49	110.5	0.74	0.50
+ RAD (ours)	142M	400K	7.7	6.72	148.0	0.78	0.49
SiT-L/2	458M	400K	18.8	5.29	72.0	0.64	0.64
+ REPA	458M	50K	55.4	24.0	23.0	0.43	0.53
+ REPA	458M	100K	24.1	6.25	55.7	0.62	0.60
+ REPA	458M	200K	14.0	5.18	86.5	0.67	0.64
+ REPA	458M	400K	10.0	5.20	109.2	0.69	0.65
+ RAD (ours)	472M	50K	28.2	16.88	48.4	0.64	0.48
+ RAD (ours)	472M	100K	9.3	8.24	120.1	0.78	0.48
+ RAD (ours)	472M	200K	5.2	7.10	169.7	0.82	0.49
+ RAD (ours)	472M	400K	3.9	5.59	203.7	0.84	0.49
SiT-XL/2	675M	7M	8.3	6.32	131.7	0.68	0.67
+ REPA	675M	50K	52.3	31.24	24.3	0.45	0.53
+ REPA	675M	100K	19.4	6.06	67.4	0.64	0.61
+ REPA	675M	200K	11.1	5.05	100.4	0.69	0.64
+ REPA	675M	400K	7.9	5.06	122.6	0.70	0.65
+ REPA	675M	4M	5.9	5.73	157.8	0.70	0.69
+ RAD (ours)	690M	50K	24.4	19.65	56.7	0.65	0.45
+ RAD (ours)	690M	100K	7.8	6.36	134.3	0.82	0.47
+ RAD (ours)	690M	200K	4.6	6.38	185.3	0.84	0.49
+ RAD (ours)	690M	400K	3.7	5.66	217.8	0.85	0.50

Evaluation. The FID-50k in Table 2 uses class-balanced sampling (50 images per class), following Zheng et al. [63]. All sampling hyperparameters were selected with FID-10k, using uniformly drawn class labels and a single seed; Table 10 reports these values shifted to the FID-50k scale (FID -3.1 , recall -0.10). With uniform labels they are not directly comparable to the class-balanced FID-50k in Table 2.

Vanilla CFG vs. token coupling. Table 10a shows that vanilla CFG mainly increases precision and IS while lowering recall, which improves FID by only 0.46. Token coupling has the opposite effect: it increases recall from 0.51 to 0.59 at the cost of precision, and on its own slightly worsens FID. The two are complementary: token coupling supplies the sample-level diversity that guidance removes, and combined with CFG it lowers FID from 3.51 to 2.18. Increasing ρ trades precision for recall monotonically (Table 10b), with the best FID for ρ between 0.09 and 0.12.

Autoguidance. Autoguidance alone outperforms vanilla CFG and also benefits from token coupling, but it is best used as a complement to CFG, where it adds a small further improvement (2.18 to 2.08). For the weak model, we compared EMA checkpoints of the same run at 50K, 100K, 120K, and 150K iterations and found 100K to work best (FID 3.01, 2.39, 2.60, and 2.84 with autoguidance alone, $w = 2.0$, $\rho = 0.1$).

A.7 Additional Quantitative Results

Tables 11–19 list the values plotted in the figures of Sec. 3 that are not already covered by Tables 1, 8, and 9.

Table 9: Full ImageNet 256×256 results of REG and REG + RAD at every training budget (numbers behind Table 3). REG numbers are from its paper. All metrics on 50k samples, best β_{inf} per row, Euler–Maruyama SDE sampler with 250 steps, no guidance.

Model	#Params	Iter.	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
SiT-B/2	130M	400K	33.0	6.46	43.7	0.53	0.63
+ REG	132M	50K	64.7	9.47	23.2	0.40	0.51
+ REG	132M	100K	36.1	7.74	45.5	0.53	0.61
+ REG	132M	200K	22.1	7.19	72.2	0.60	0.63
+ REG	132M	400K	15.2	6.69	94.6	0.64	0.63
+ REG + RAD (ours)	143M	50K	50.5	20.54	28.1	0.45	0.51
+ REG + RAD (ours)	143M	100K	22.4	7.11	67.3	0.64	0.54
+ REG + RAD (ours)	143M	200K	12.9	6.60	102.8	0.70	0.54
+ REG + RAD (ours)	143M	400K	8.9	5.76	124.6	0.74	0.53
SiT-L/2	458M	400K	18.8	5.29	72.0	0.64	0.64
+ REG	460M	50K	30.1	8.92	52.6	0.58	0.57
+ REG	460M	100K	11.4	5.36	108.8	0.70	0.60
+ REG	460M	200K	6.6	5.16	145.4	0.73	0.63
+ REG	460M	400K	4.6	5.21	167.6	0.75	0.63
+ REG + RAD (ours)	474M	50K	23.7	11.03	61.9	0.62	0.53
+ REG + RAD (ours)	474M	100K	7.8	6.13	136.1	0.77	0.52
+ REG + RAD (ours)	474M	200K	4.6	5.81	183.6	0.80	0.52
+ REG + RAD (ours)	474M	400K	3.9	4.78	211.0	0.84	0.52

Table 10: Guided sampling ablations for RAD-XL/2 (400K iterations). FID-50k with uniformly drawn class labels, Euler–Maruyama SDE sampler with 50 steps, $\beta_{\text{inf}} = 0.8$. (a) CFG and autoguidance (AG) applied to independent ($\rho = 0$) and token-coupled ($\rho = 0.1$) policy noise; CFG uses $s = 3.0$ and AG $w = 2.0$. (b) Token coupling strength ρ with CFG ($s = 3.0$).

(a) Guidance with independent and token-coupled noise									(b) Coupling strength			
Guidance	Independent ($\rho = 0$)				Token-coupled ($\rho = 0.1$)				ρ	FID↓	Prec.↑	Rec.↑
	FID↓	IS↑	Prec.↑	Rec.↑	FID↓	IS↑	Prec.↑	Rec.↑				
None	3.97	206.5	0.84	0.51	4.31	160.5	0.77	0.59	0	3.51	0.87	0.49
CFG	3.51	296.8	0.87	0.49	2.18	241.2	0.80	0.58	0.05	2.77	0.85	0.51
AG	3.12	247.7	0.85	0.52	2.39	192.7	0.78	0.60	0.09	2.22	0.81	0.57
									0.10	2.18	0.80	0.58
									0.11	2.22	0.78	0.58
									0.15	2.81	0.72	0.63

Table 11: Training convergence (Fig. 1c and Fig. 3a). FID-50k on ImageNet 256×256 against training iterations. Baseline numbers are from the respective papers. A dash marks a budget that is not reported in the respective paper (baselines) or not evaluated (RAD). Euler–Maruyama SDE sampler with 250 steps, no guidance.

Model	50K	100K	150K	200K	400K	1M	4M	7M
SiT-XL/2 [31]	–	–	–	–	17.2	–	–	8.3
REPA-B/2 [59]	78.2	49.5	–	33.2	24.4	–	–	–
REPA-XL/2 [59]	52.3	19.4	–	11.1	7.9	6.4	5.9	–
RAD-B/2 (ours)	55.6	23.5	–	12.0	7.7	–	–	–
RAD-XL/2 (ours)	24.4	7.8	5.7	4.6	3.7	–	–	–

Table 12: Data efficiency (Fig. 3c). FID-50k of SiT-B/2 models trained for 400K iterations on a random fraction of ImageNet 256×256 , best checkpoint of each run. Euler ODE sampler with 50 steps, no guidance.

Fraction of ImageNet	3%	10%	30%	100%
REPA-B/2 [59]	33.8	18.8	22.4	22.9
RAD-B/2 (ours)	32.6	13.0	8.9	7.8

Table 13: Sampling efficiency against REPA (Fig. 4a). FID-50k on ImageNet 256×256 against the number of Euler ODE steps for RAD-XL/2 trained for 150K iterations and the official REPA-XL/2 checkpoint trained for 4M iterations without CFG. With the 250-step Euler–Maruyama SDE sampler the same models reach 5.7 and 5.9, respectively.

Model	5	10	15	20	25	50	100	200
REPA-XL/2 (4M) [59]	62.3	21.9	15.0	12.2	10.5	8.2	7.4	7.2
RAD-XL/2 (150K, ours)	23.7	9.1	7.3	6.7	6.4	6.2	–	–

Table 14: Stochastic sampling efficiency against REPA (Fig. 8). FID-50k on ImageNet 256×256 against the number of Euler–Maruyama SDE steps for RAD-XL/2 trained for 150K iterations and the official REPA-XL/2 checkpoint trained for 4M iterations without CFG.

Model	10	25	50	100	250
REPA-XL/2 (4M) [59]	101.0	22.0	10.3	7.4	5.9
RAD-XL/2 (150K, ours)	46.9	7.9	5.9	5.4	5.7

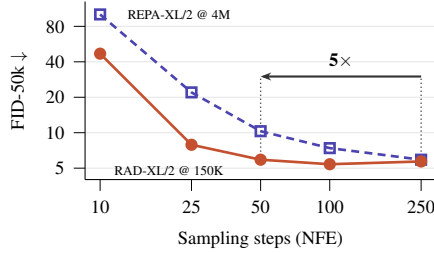


Figure 8: RAD vs. REPA, Euler–Maruyama SDE. FID-50k vs. sampling steps for RAD-XL/2 (150K) and REPA-XL/2 (4M) with the Euler–Maruyama SDE sampler. No guidance. Values are listed in Table 14.

Table 15: ODE samplers and policy noise (Fig. 4b). FID-50k on ImageNet 256×256 against the number of sampling steps for RAD-XL/2 trained for 400K iterations with the Euler ODE sampler and UniPC [62] of order 2 and 3 without CFG, drawing fresh policy noise at every step or keeping it fixed over the trajectory. The best value per step budget is in bold.

Sampler	Policy noise	5	10	25	50
Euler	Fresh	20.6	6.8	4.1	4.0
	Fixed	20.0	6.5	4.2	3.9
UniPC, order 2	Fresh	6.5	4.5	3.7	3.8
	Fixed	6.7	4.4	3.8	3.8
UniPC, order 3	Fresh	6.4	4.5	3.8	3.8
	Fixed	6.4	4.4	3.9	3.9

Table 16: Inference policy scale (Figs. 4c and 9). FID-50k, precision, and recall on ImageNet 256×256 of RAD-XL/2 trained for 400K iterations against β_{inf} . Training uses $\beta = 1$. Euler ODE sampler with 50 steps, no guidance. The best value per column is in bold.

β_{inf}	FID↓	Prec.↑	Rec.↑
0.2	45.9	0.37	0.59
0.4	8.1	0.72	0.55
0.6	4.3	0.82	0.51
0.75	4.2	0.84	0.49
0.8	4.0	0.85	0.50
1.0 (train)	4.5	0.84	0.50

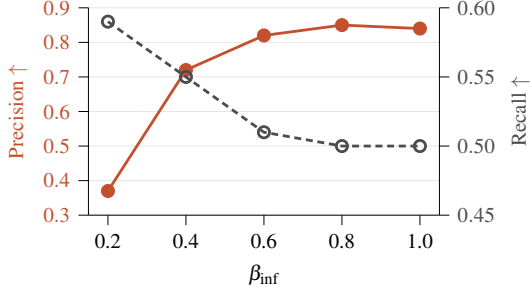


Figure 9: Effect of the inference policy scale β_{inf} on precision and recall. Precision and recall on ImageNet 256×256 of RAD-XL/2 trained for 400K iterations as β_{inf} varies (FID in Fig. 4c). Training uses $\beta = 1$. Euler ODE sampler with 50 steps, no guidance. Values are listed in Table 16.

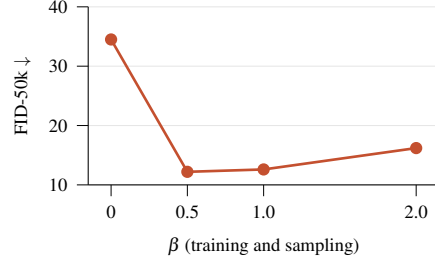


Figure 10: Effect of the training policy scale β . FID-50k on ImageNet 256×256 of RAD-B/2 trained for 400K iterations with policy scale β and sampled with $\beta_{\text{inf}} = \beta$. Euler-Maruyama SDE sampler with 250 steps, no guidance. Values are listed in Table 17.

Table 17: Training policy scale (Fig. 10). FID-50k on ImageNet 256×256 of RAD-B/2 trained for 400K iterations with policy scale β and sampled with $\beta_{\text{inf}} = \beta$. Euler-Maruyama SDE sampler with 250 steps, no guidance. The best value is in bold.

β	FID-50k $\downarrow$
0.0	34.5
0.5	12.2
1.0	12.6
2.0	16.2

Table 18: Linear probing per block (Fig. 6a). ImageNet top-1 accuracy (%) of a linear classifier on the hidden state at block k of REPA-B/2 and RAD-B/2 trained for 400K iterations, for a noised input at diffusion time t . Both models are aligned at block 4, after which RAD injects $\mathbf{u}_t$ (sampled with $\beta_{\text{inf}} = 0.7$). The better value per block and t is in bold.

Block	$t = 0.1$		$t = 0.25$		$t = 0.5$		$t = 0.75$	
	REPA	RAD	REPA	RAD	REPA	RAD	REPA	RAD
4	63.0	62.7	59.1	59.0	37.2	37.8	8.3	8.6
5	60.9	63.4	57.4	59.2	36.2	36.4	8.3	6.6
6	58.3	62.1	54.9	58.1	35.4	36.0	8.2	6.8
8	52.3	56.7	50.2	53.9	33.8	34.6	8.1	6.8
10	46.1	48.8	44.6	47.6	31.7	32.4	8.2	6.9
12	40.7	43.3	40.2	43.3	29.3	30.5	7.8	6.9

Table 19: Linear probing across model scale (Fig. 6b). Best ImageNet top-1 accuracy (%) over all blocks of RAD-B/2 and RAD-XL/2 trained for 400K iterations, at each diffusion time t , with the block that attains it in parentheses. B/2 is probed with $\beta_{\text{inf}} = 0.7$ and XL/2 with $\beta_{\text{inf}} = 0.8$.

Model	$t = 0.1$	$t = 0.25$	$t = 0.5$	$t = 0.75$
RAD-B/2	63.4 (5)	59.2 (5)	37.8 (4)	8.6 (4)
RAD-XL/2	71.7 (9)	68.1 (9)	46.1 (8)	9.5 (8)

A.8 Additional Interpolation Results

Protocol. Given two real ImageNet images $\mathbf{x}_0^A$ and $\mathbf{x}_0^B$ with labels y_A and y_B , we take the mean of the learned policy on each noised image as its representation, $\bar{\mathbf{u}} = \mathbf{h}_\phi(\mathbf{x}_{t_c}, t_c)$ with $t_c = 0.1$, and invert each image to noise $\mathbf{x}_1$ with the Euler ODE sampler (50 steps). For a weight $m \in [0, 1]$, we then interpolate

$$\bar{\mathbf{u}}_m = m \bar{\mathbf{u}}^A + (1 - m) \bar{\mathbf{u}}^B, \quad \mathbf{x}_{1,m} = \text{slerp}(\mathbf{x}_1^A, \mathbf{x}_1^B; m), \quad \mathbf{e}_m = m \mathbf{e}_{y_A} + (1 - m) \mathbf{e}_{y_B}, \quad (46)$$

where e_y is the class embedding. Each sample is generated from $\mathbf{x}_{1,m}$ and e_m with the same sampler, where at every step $\mathbf{u}_t$ is sampled from the flow policy in Eq. 34 with $\beta_{\text{inf}} = 0.8$ and $\mathbf{h}_\phi(\mathbf{x}_t, t)$ replaced by $\bar{\mathbf{u}}_m$. Hence, $m = 1$ and $m = 0$ approximately reconstruct A and B, respectively. REPA does not pass the representations back into the denoiser, so its generations cannot be manipulated through them directly. Hence, we interpolate $\mathbf{x}_1$ and the class embedding in the same way.

Additional samples. We show additional interpolations between real ImageNet images across classes (Fig. 11) and within the same class (Fig. 12), using the protocol above and the checkpoints of Sec. 3.3. For same-class pairs, the class embedding is fixed, so REPA interpolates $\mathbf{x}_1$ alone. As in Fig. 5, RAD preserves the pose and framing of the endpoints while the appearance changes gradually, e.g., the iguana’s pale scales darken into the textured skin of the Komodo dragon, the eye of the green chameleon grows into the turreted eye of the African chameleon, and the white-lit castle turns into the orange-lit palace while the night scene and its reflection on the water are kept. Within the same class, RAD keeps the side profile of the wood rabbit as the background changes from dry brush to green foliage. REPA again does not remain consistent: the head of the iguana dissolves into a full-body lizard among leaves, the close-up of the chameleon is replaced by full-body chameleons on branches, and the wood rabbit is replaced by a rounder, crouched rabbit facing the opposite direction for $0.35 \leq m \leq 0.5$.

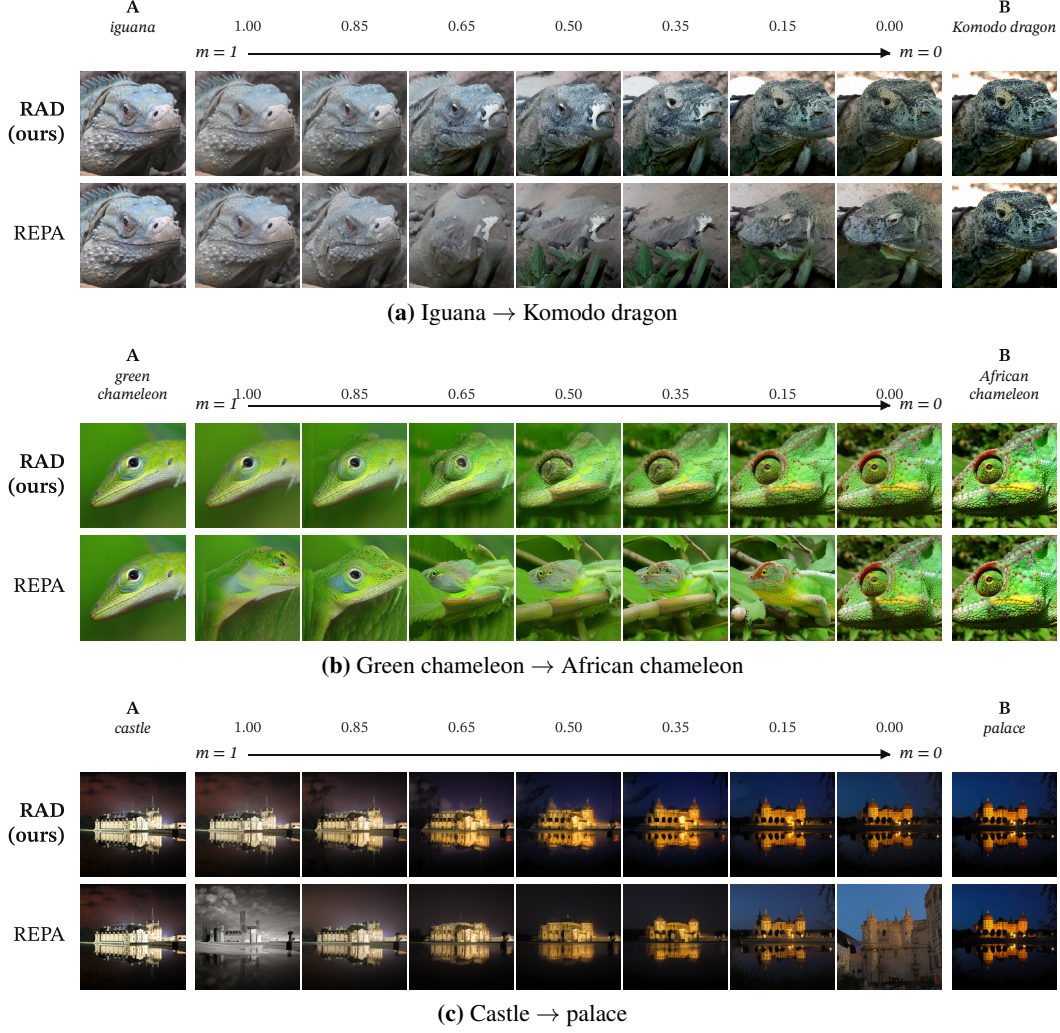


Figure 11: Additional interpolations across classes. Samples interpolated from image A ($m = 1$) to image B ($m = 0$), with the real images shown in the outer columns. Top rows: RAD-XL/2 trained for 400K iterations, interpolating the representation, $\mathbf{x}_1$, and the class embedding (Eq. 46). Bottom rows: REPA-XL/2 trained for 4M iterations, i.e., $10\times$ longer than RAD, interpolating $\mathbf{x}_1$ and the class embedding. No guidance.

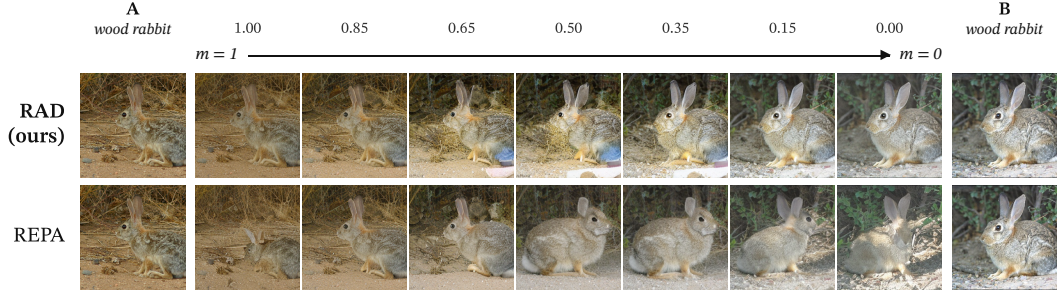


Figure 12: Additional interpolation within the same class. Samples interpolated between two wood rabbits, from image A ($m = 1$) to image B ($m = 0$), with the real images shown in the outer columns. Top: RAD-XL/2 trained for 400K iterations, interpolating the representation and $\mathbf{x}_1$. Bottom: REPA-XL/2 trained for 4M iterations, i.e., $10\times$ longer than RAD, interpolating $\mathbf{x}_1$ only. The class embedding is fixed in both. No guidance.

A.9 Additional Samples (Class Conditional ImageNet)



Figure 13: Class-conditional samples from ImageNet (Macaw, class 88).



Figure 14: Class-conditional samples from ImageNet (Lorikeet, class 90).



Figure 15: Class-conditional samples from ImageNet (Red panda, class 387).



Figure 16: Class-conditional samples from ImageNet (Giant panda, class 388).



Figure 17: Class-conditional samples from ImageNet (Red fox, class 277).



Figure 18: Class-conditional samples from ImageNet (Anemone fish, class 393).



Figure 19: Class-conditional samples from ImageNet (Agaric, class 992).



Figure 20: Class-conditional samples from ImageNet (Coral fungus, class 991).



Figure 21: Class-conditional samples from ImageNet (Strawberry, class 949).



Figure 22: Class-conditional samples from ImageNet (Teapot, class 849).



Figure 23: Class-conditional samples from ImageNet (Schooner, class 780).



Figure 24: Class-conditional samples from ImageNet (Church, class 497).



Figure 25: Class-conditional samples from ImageNet (Arctic fox, class 279).



Figure 26: Class-conditional samples from ImageNet (Valley, class 979).



Figure 27: Class-conditional samples from ImageNet (Cheeseburger, class 933).

A.10 Additional Text-to-Image Samples



Figure 28: “A fox wearing a knitted scarf, reading a book by a cozy fireplace, warm storybook illustration.”



Figure 29: “A cat astronaut floating inside a spaceship, looking at the Earth through a round window.”



Figure 30: “A tiny island with a lighthouse inside a teacup, ocean waves swirling, whimsical 3D render.”



Figure 31: “A golden retriever wearing a yellow raincoat, splashing in puddles on a rainy street.”



Figure 32: “A miniature village inside a glass terrarium with moss, tiny houses and a winding path, macro photo.”



Figure 33: “A stained glass window depicting a phoenix rising from flames, vivid red and gold.”



Figure 34: “A floating castle above the clouds at sunset, waterfalls pouring off its edges, fantasy art.”



Figure 35: “An isometric 3D render of a cozy bedroom with a bed, plants and fairy lights, pastel colors.”



Figure 36: “A small cactus wearing a straw hat and neon sunglasses in the Sahara desert.”