BEYOND THE ENCODER: LATENT REPRESENTATION LEARNING VIA PARAMETRIC SOFT DISCRETE RANDOM VARIABLES

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose the Parametric Soft Discrete Random Variable Auto-Decoder (PSDR-AD), a novel encoder-free framework for learning latent representations through directly optimized Parametric Soft Discrete Random Variables (PSDR-Vars). Our approach is motivated by the hypothesis that encoder-produced representations may not be optimal for image reconstruction because they are restricted by the learned encoder mapping. Specifically, each image is associated with a dedicated PSDR-Var, a directly optimizable latent variable with a soft stochastic relaxation induced by noise injection. During training, the PSDR-Vars are fed into a shared decoder to reconstruct their corresponding images, while the PSDR-Vars and decoder are jointly optimized. During testing, the decoder parameters are fixed and noise injection is disabled, while newly initialized PSDR-Vars are assigned to unseen images and optimized to minimize their reconstruction errors. Unlike conventional autoencoders, which treat latent extraction as a forward function evaluation that maps an input image to the output of an encoder, PSDR-AD treats it as an inverse optimization problem: given a target image as the desired output of the decoder, it optimizes the decoder input to reproduce that output. This removes the additional constraint imposed by the encoder mapping and can improve reconstruction quality even when PSDR-AD is applied to a decoder from a pre-trained autoencoder. Comprehensive experiments demonstrate the effectiveness of PSDR-AD for latent representation learning. Using the same pre-trained FLUX-VAE decoder, PSDR-AD reduces the rFID from 0.176 to 0.005 on ImageNet through direct latent optimization, achieving a lower rFID than existing continuous reconstruction methods.

1 INTRODUCTION

Autoencoder is the fundamental framework for learning latent representations for downstream applications, such as image generation (Rombach et al., 2022). This framework is widely used to extract continuous latents for diffusion models and discrete tokens for Transformers (Li et al., 2026). Usually, an autoencoder is trained on a training set with a sufficient number of images and then applied to unseen images, based on the assumption that the training and testing images share the same distribution. In testing state, images are input into an encoder to extract image features, while a decoder is utilized to reconstruct the original images. However, from our perspective, although the encoder provides convenient inference for unseen images, it also restricts the representation ability of the latent space. This leads to an interesting hypothesis that the output of the encoder may not be the best latent representation for a fixed decoder. Therefore, directly optimizing the latent representations could be a better alternative for latent representation learning. Based on this insight, we propose to learn latent representations via Parametric Soft Discrete Random Variables (PSDR-Vars) with a decoder-only architecture, which we name the Parametric Soft Discrete Random Variable Auto-Decoder (PSDR-AD).

The main idea of the proposed approach is illustrated in Fig. 1. It is based on the hypothesis that the encoder output is not necessarily the globally optimal latent representation used as the input to the decoder. Therefore, directly optimizing the latent representation may provide better reconstruction

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

results. Based on this insight, we introduce Parametric Soft Discrete[1] Random Variables (PSDR-Vars), which are directly optimized as inputs to the decoder. Mathematically,

$$\mathcal{L}\big(\{\mathbf{x}^{(n)}\}_{n=1}^{N}\big) = \min_{\theta,\sigma,\{\mathbf{z}^{(n)}\}_{n=1}^{N}} \left\{ \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{\epsilon}}\Big[\big\|d_\theta\big(\tilde{\mathbf{z}}^{(n)}\big) - \mathbf{x}^{(n)}\big\|_2^2\Big] + (\sigma-\sigma_0)^2 \right\}. \tag{1}$$

where $\mathbf{z}^{(n)}$ is a directly optimizable parameter assigned to the n-th training instance $\mathbf{x}^{(n)}$, and $\tilde{\mathbf{z}}^{(n)} = \psi\big(\mathbf{z}^{(n)}\big) + \sigma \cdot \boldsymbol{\epsilon}$ is the corresponding PSDR-Var. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the Gaussian noise. $\psi(\cdot)$ is a controlling function, θ denotes the decoder parameters. σ is a learnable noise scale, and σ_0 is its user-specified target value. The regularization term $(\sigma-\sigma_0)^2$ encourages the learned noise scale to remain close to σ_0. In particular, Gaussian noise is injected into these latent representations to implicitly couple the local neighborhoods associated with different training samples. This mechanism encourages a smoother latent space and consequently improves both the generalization ability and learning capacity of the model. Sec. 5.1 evaluates how the target training noise scale affects reconstruction quality on test images.

The implementation of the proposed approach is straightforward. A latent variable is first initialized for each instance as a fully learnable parameter. It is then passed through a controlling function ψ, after which noise scaled by σ is added to the output. Note that the controlling function can be the identity function, i.e., $\psi(x) = x$, and σ can either be set to zero or treated as a learnable parameter. The resulting PSDR-Vars are fed into the decoder and optimized by reconstructing their corresponding training instances. After the PSDR-Vars and decoder are well trained, new PSDR-Vars are assigned to unseen images while the decoder is fixed. These new PSDR-Vars are then optimized for a sufficient number of iterations to obtain latent representations for the unseen images. Note that PSDR-Vars can also be applied to pre-trained autoencoder architectures. By bypassing the encoder of a pre-trained autoencoder, PSDR-Vars can be directly optimized as inputs to the pre-trained decoder, achieving better reconstruction results. This strategy avoids the heavy computational cost of training an autoencoder from scratch and can further improve the reconstruction performance of pre-trained autoencoders. Based on this approach, the proposed method achieves state-of-the-art performance compared with existing methods using only a single RTX 4080 SUPER GPU (16GB). The main contributions of this work are summarized as follows:

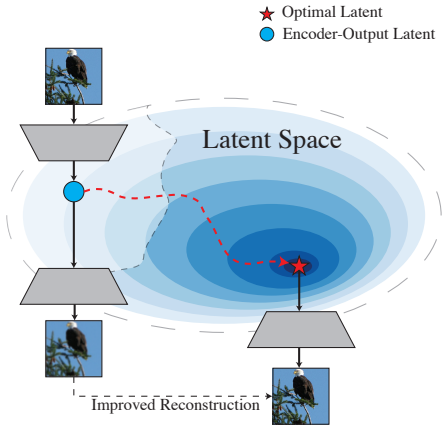

Figure 1: Even for a well-trained autoencoder, the encoder output may deviate from the reconstruction-optimal latent under the fixed decoder, resulting in suboptimal reconstruction.

- We provide a new perspective on latent representation learning by showing that the representations produced by a shared encoder may not be optimal for a fixed decoder. This motivates directly optimizing latent representations beyond the constraint imposed by encoder-based inference.
- We propose Parametric Soft Discrete Random Variables (PSDR-Vars) for direct latent representation optimization and develop the PSDR Auto-Decoder (PSDR-AD), a decoder-only framework that learns latent representations without an encoder. The proposed approach can be trained from scratch or directly applied to pretrained autoencoders by optimizing PSDR-Vars with a fixed pretrained decoder.
- We conduct extensive experiments to validate the proposed hypothesis and framework. The results demonstrate that direct latent optimization consistently improves reconstruction quality over encoder-based representations, including those produced by pretrained autoencoders, and achieves state-of-the-art reconstruction performance.

[1]Here "discrete" refers to the process of assigning a latent variable to each training instance, which is different from the continuous latent space of a conventional autoencoder.

2 Related Work

The autoencoder framework has been widely used to extract latent representations from high-dimensional data (Hinton & Salakhutdinov, 2006). Existing autoencoders can be broadly divided into two categories based on whether their latent representations are continuous or discrete. Continuous autoencoders encode input data into continuous latent representations and reconstruct the input through a decoder. Variational Autoencoders (VAEs) (Kingma & Welling, 2014) introduce a probabilistic latent space by regularizing the approximate posterior toward a predefined prior distribution, enabling both representation learning and generation. β-VAE (Higgins et al., 2017) encourages disentangled representations by increasing the weight of latent regularization, while FactorVAE (Kim & Mnih, 2018) and β-TCVAE (Chen et al., 2018) further investigate factorized and disentangled latent representations. Hierarchical architectures, including Ladder VAE (Sønderby et al., 2016), BIVA (Maaløe et al., 2019), and NVAE (Vahdat & Kautz, 2020), improve the representational capacity of VAEs through multiple levels of stochastic latent variables. More recently, continuous autoencoders have also become an important component of latent generative models. In particular, Latent Diffusion Models (LDMs) (Rombach et al., 2022) employ a continuous autoencoder to map high-dimensional images into a compact latent space, substantially reducing the computational cost of diffusion-based generation. Efficient autoencoder architectures such as DC-AE (Chen et al., 2025c) further explore highly compressed continuous latent representations for large-scale generative modeling.

In contrast, discrete autoencoders map input data into a finite set of discrete latent representations, which can naturally serve as tokens for subsequent generative models. In particular, VQ-VAE (van den Oord et al., 2017) introduced vector quantization into the autoencoder framework by mapping continuous encoder outputs to their nearest entries in a learned codebook. It is one of the most popular methods for discrete latent representation learning and has also been widely used in downstream applications such as image generation (Razavi et al., 2019). VQGAN (Esser et al., 2021) combined vector-quantized representations with perceptual and adversarial objectives, enabling high-fidelity image reconstruction and Transformer-based image synthesis. Several subsequent studies have focused on improving codebook utilization and quantization quality, including SQ-VAE (Takida et al., 2022), CVQ-VAE (Zheng & Vedaldi, 2023), and residual vector quantization approaches such as RQ-VAE (Lee et al., 2022). More recently, alternative quantization strategies have been proposed to simplify discrete representation learning and alleviate the limitations of conventional learned codebooks. Finite Scalar Quantization (FSQ) (Mentzer et al., 2024) replaces vector quantization with independent scalar quantization over a small number of dimensions, while Lookup-Free Quantization (LFQ) (Yu et al., 2024) removes explicit codebook lookup and represents tokens through binary latent dimensions. Binary Spherical Quantization (BSQ) (Zhao et al., 2025) further learns binary representations on a normalized latent space. These discrete autoencoders provide compact tokenized representations that can be directly modeled by autoregressive or other token-based generative models.

Limitations of encoder-based inference have been studied in variational inference(Cremer et al., 2018; Marino et al., 2018). In particular, Marino et al. (2018) refine approximate posterior estimates through learned iterative updates. In neural image compression, Yang et al. (2020) optimize latents of pretrained models under a rate–distortion objective using stochastic Gumbel annealing (SGA), while Perugachi-Diaz et al. (2024) improve its rounding probability parameterizations through SGA+. These methods establish relevant precedents for iterative latent inference. Encoder-free learning has also been explored in GLO (Bojanowski et al., 2018) and DeepSDF (Park et al., 2019). GLO jointly learns a generator and image-specific latent vectors through reconstruc- tion, with a norm-projection constraint on its latent vectors. The continuous identity-map configuration of PSDR-AD does not require this projection and instead incorporates additive Gaussian perturbations with a learnable, regularized noise scale. DeepSDF jointly learns shape codes and a coordinate-conditioned network that predicts signed distances; our decoder reconstructs images from their latent representations. Thus, these approaches share direct latent optimization but differ in their latent constraints, decoder formulations, and objectives.

Within this context, we propose PSDR-AD as a unified auto-decoder framework for continuous and vector-quantized image representation learning. PSDR-Vars provide independently optimizable latent parameters with a stochastic relaxation, whose noise regularization is evaluated by ablation. The framework supports both joint latent–decoder training and optimization of inputs to a fixed pre-

trained decoder. The latter is an application of the same framework, allowing us to evaluate modern image decoders without large-scale retraining. Our experiments examine held-out reconstruction, cross-dataset transfer, downstream generation, and reconstruction improvements at unchanged latent dimensionality.

3 METHOD

Notation: Let $\mathbb{X}$ denote the input space, and let $\mathbf{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N} \subset \mathbb{X}$ be a finite dataset containing N samples, such as images. Let $e_{\boldsymbol{\varphi}} : \mathbb{X} \to \mathbb{R}^d$ and $d_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{X}$ denote the encoder and decoder, respectively, where $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ denote their parameters and d denotes the latent dimensionality. For each sample $\mathbf{x}^{(n)}$, we denote its encoder-generated latent representation by $\mathbf{z}_{\boldsymbol{\varphi}}^{(n)} := e_{\boldsymbol{\varphi}}(\mathbf{x}^{(n)})$, where all samples share the same encoder parameters $\boldsymbol{\varphi}$. In contrast, the encoder-free latent representation is denoted by $\mathbf{z}_{\boldsymbol{\phi}}^{(n)} := \mathbf{z}(\boldsymbol{\phi}^{(n)})$, where $\boldsymbol{\phi}^{(n)}$ is independently and directly optimized for the n-th sample.

Definition 3.1 (Parametric Soft Discrete Random Variable) *Consider a discrete parametric variable $\mathbf{z}^{(k)} := \mathbf{z}(\boldsymbol{\phi}^{(k)}) \in \{\mathbf{z}(\boldsymbol{\phi}^{(1)}), \mathbf{z}(\boldsymbol{\phi}^{(2)}), \ldots, \mathbf{z}(\boldsymbol{\phi}^{(N)})\} \subset \mathbb{R}^d$, where $k \sim \mathrm{Cat}(\boldsymbol{\pi})$ with a uniform prior $\boldsymbol{\pi} = (\frac{1}{N}, \ldots, \frac{1}{N})$ and $\sum_{n=1}^{N} \pi_n = 1$. Let $\psi(\cdot)$ be a controlling function used to control the range of the output values. By injecting additive noise $\boldsymbol{\epsilon} \sim p_\epsilon(\cdot)$ in $\mathbb{R}^d$ into $\psi(\mathbf{z}^{(k)})$, independently of k, we obtain a continuous relaxation*

$$\tilde{\mathbf{z}}^{(k)} = \psi\big(\mathbf{z}(\boldsymbol{\phi}^{(k)})\big) + \sigma\boldsymbol{\epsilon}, \quad p_{\tilde{\mathbf{z}}}(\mathbf{u}) = \sum_{n=1}^{N} \pi_n \frac{1}{\sigma^d} p_\epsilon\left(\frac{\mathbf{u} - \psi\big(\mathbf{z}(\boldsymbol{\phi}^{(n)})\big)}{\sigma}\right), \tag{2}$$

where $\sigma > 0$ controls the noise scale, d denotes the dimensionality of $\tilde{\mathbf{z}}$, and ψ is a general controlling function. Then, $\tilde{\mathbf{z}}^{(k)}$ is defined as a Parametric Soft Discrete Random Variable (PSDR-Var).

Remark 3.2 *The Parametric Soft Discrete Random Variable is designed to enable latent representation learning without an encoder. Because the encoder is removed, a separate parametric variable must be initialized for each input and, after applying the controlling function and noise injection, fed into the decoder for reconstruction. The controlling function ψ is applied to regulate the values of these variables, which may otherwise become unstable during training. However, ψ is not required to be bounded during continuous training. For example, the identity function $\psi(\mathbf{z}) = \mathbf{z}$ may also be used, effectively removing the controlling operation. Noise is injected to provide a continuous relaxation of the discrete parametric variable. Empirically, noise injection results in a smoother latent space and improves generalization, as validated by the experiments in Sec. 5.1.*

Proposition 3.3 (Encoder-Induced Reconstruction Gap) *Let $\mathbf{X} = \{\mathbf{x}^{(k)}\}_{k=1}^{N}$ be a finite dataset, and let $e_{\boldsymbol{\varphi}}$ and $d_{\boldsymbol{\theta}}$ denote a deterministic encoder and decoder, respectively. Assume that the encoder and decoder have been well trained and their parameters are fixed. Suppose that the decoder inputs are replaced by the proposed PSDR-Vars $\tilde{\mathbf{z}}^{(k)}$ associated with the corresponding targets $\mathbf{x}^{(k)}$. we have:*

$$\frac{1}{N}\sum_{k=1}^{N} \min_{\boldsymbol{\phi}^{(k)},\sigma} \mathbb{E}_{\boldsymbol{\epsilon}\sim p_\epsilon}\left[\left\| d_{\boldsymbol{\theta}}\left(\tilde{\mathbf{z}}^{(k)}\right) - \mathbf{x}^{(k)}\right\|_2^2\right] \le \frac{1}{N}\sum_{k=1}^{N} \left\| d_{\boldsymbol{\theta}}\left(\psi\big(e_{\boldsymbol{\varphi}}(\mathbf{x}^{(k)})\big)\right) - \mathbf{x}^{(k)}\right\|_2^2. \tag{3}$$

The result follows directly from the inclusion of the feasible sets. Every collection of latent representations generated by the shared encoder can also be represented by independently optimized PSDR-Var parameters, whereas the converse does not necessarily hold. Therefore, the feasible set of encoder-constrained latent representations is a subset of that of independently optimized PSDR-Vars. Minimizing the reconstruction objective over the larger feasible set cannot produce a greater reconstruction error.

Remark 3.4 *Proposition 3.3 shows that the shared encoder mapping may restrict the latent representations from attaining the minimum reconstruction error permitted by the fixed decoder. Directly optimizing an individual PSDR-Var for each sample removes this shared parameter constraint and allows the decoder input to be optimized specifically for reconstruction. Therefore, directly optimizing PSDR-Vars as inputs to the decoder may provide room for further reconstruction improvement.*

Although the inequality is established based on a well-trained and fixed autoencoder, we further hypothesize that directly training the PSDR-Vars together with the decoder can provide better reconstruction results. This hypothesis motivates the proposed decoder-only optimization framework and is empirically supported by the experiments in Sec. 5.2.

4 Parametric Soft Discrete Random Variables Auto-Decoder

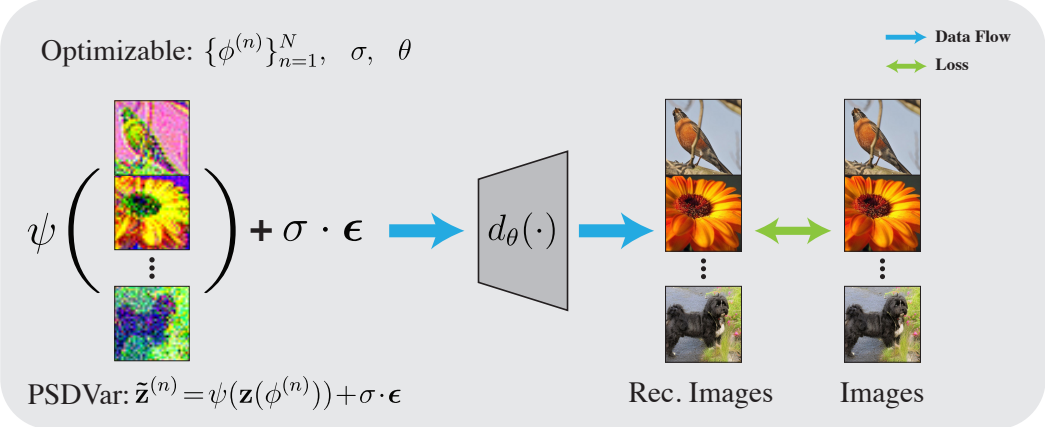

Figure 2: Illustration of the proposed PSDR-AD. During training, an independently parameterized PSDR-Var is assigned to each training image, and both the PSDR-Vars and decoder are jointly optimized for reconstruction. During testing, the decoder is fixed, while newly initialized PSDR-Vars are directly optimized to reconstruct unseen images.

The main framework of the proposed approach is shown in Fig. 2. A PSDR-Var is assigned to each training image, and both the decoder and Parametric Soft Discrete Random Variables (PSDR-Vars) are optimized during the training stage. Mathematically:

$$\mathcal{L}(\mathbf{X}) = \min_{\theta,\mathbf{\Phi},\sigma} \left\{ \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} \left[\left\| d_\theta\Big(\psi\big(\mathbf{z}(\phi^{(n)})\big) + \sigma\cdot\boldsymbol{\epsilon}\Big) - \mathbf{x}^{(n)} \right\|_2^2 \right] + (\sigma-\sigma_0)^2 \right\}. \tag{4}$$

where $\mathbf{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ denotes the set of training images, and $\mathbf{\Phi} = \{\phi^{(n)}\}_{n=1}^{N}$ denotes the parameters of the PSDR-Vars assigned to the individual images. σ is a learnable scalar optimized toward a user-specified target value σ_0.

After training is completed, the decoder is fixed for testing on unseen images. During the testing stage, new PSDR-Vars are initialized and directly optimized through the fixed decoder. The testing stage aims to find the optimal PSDR-Var that provides the best reconstruction. Mathematically:

$$\hat{\phi}^{(n)} = \underset{\phi^{(n)}}{\operatorname{argmin}}\, \mathbb{E}_{\epsilon} \left[\left\| d_\theta\Big(\psi\big(\mathbf{z}(\phi^{(n)})\big)\Big) - \hat{\mathbf{x}}^{(n)} \right\|_2^2 \right] \tag{5}$$

where $\hat{\mathbf{x}}^{(n)}$ denotes the n-th test image, and $\hat{\phi}^{(n)}$, with $\hat{\mathbf{z}}^{(n)} = \mathbf{z}(\hat{\phi}^{(n)})$, denotes the optimized parameters of its learned latent representation. This framework is applicable to both continuous and discrete tokens. For continuous tokens, the framework can be directly applied by setting ψ to the identity function, i.e., $\psi(x) = x$, to preserve the maximal freedom of the latent representations. For discrete tokens, traditional quantization methods such as VQ-VAE can be incorporated into our framework. We further evaluate the learned representations on the downstream task of image generation. For continuous tokens, a diffusion model is employed, whereas for discrete tokens, an autoregressive Transformer is utilized.

5 Experiments

All experiments in this paper are run on a single NVIDIA RTX 4080 SUPER (16GB). More details related to the implementation and experiments are available in the supplementary material. For

Table 1: Effect of the noise scale σ on continuous PSDR-AD reconstruction performance on CIFAR-10. AE+Opt. provides a separate reference obtained by optimizing latents with the fixed decoder of a conventional autoencoder. PSNR is reported in dB. Bold values indicate the best result for each metric.

	PSDR-AD											AE+Opt.
σ_0	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	
PSNR↑	26.13	26.16	27.39	27.66	**27.72**	27.68	27.55	27.48	27.38	27.26	27.15	27.02
SSIM↑	0.846	0.846	0.877	0.885	**0.886**	0.885	0.883	0.882	0.879	0.877	0.875	0.870
LPIPS↓	0.210	0.207	0.178	0.172	**0.171**	0.173	0.173	0.174	0.176	0.177	0.180	0.184

models trained in our experiments, reconstruction performance is evaluated on held-out test or validation images that are excluded from training state. All COCO2017 evaluations use the full official validation set (`val2017`), containing 5,000 images.

5.1 Effect of Noise on Latent Representation Learning

We examine noise regularization on CIFAR-10 by varying the target noise scale σ_0. The references are PSDR-AD with $\sigma_0 = 0$ and AE+Opt., which optimizes latents using a conventional autoencoder's fixed decoder. As shown in Tab. 1, $\sigma_0 = 0.20$ yields the best reconstruction scores: 27.72 dB PSNR, 0.8861 SSIM, and 0.1706 LPIPS. Its PSNR exceeds the two references by 1.59 and approximately 0.70 dB, respectively, with improvements in SSIM and LPIPS. These results support the contribution of appropriate noise regularization beyond direct latent optimization alone. By perturbing latents during training, noise injection encourages consistent reconstruction within local latent neighborhoods, providing a mechanism for learning a smoother latent-to-image mapping. The observed performance gains are consistent with this interpretation, while the decline at larger target scales indicates that excessive perturbation compromises reconstruction fidelity.

5.2 Comparison with Autoencoders and GLO

We compare PSDR-AD with conventional autoencoders on CIFAR-10, ImageNet-1K, and COCO2017 using a simple convolutional architecture following the VQ-VAE design. Continuous baselines include AE and GLO (Bojanowski et al., 2018); the discrete baseline is VQ-VAE. We set the target noise scale to $\sigma_0 = 0.20$ for the continuous variant and $\sigma_0 = 0$ for the vector-quantized variant. Tab. 2 shows that PSDR-AD outperforms the corresponding autoencoder baselines across all datasets and metrics. Continuous PSDR-AD improves PSNR over AE by 1.47, 0.84, and 0.85 dB, respectively, and outperforms the evaluated GLO configuration. On ImageNet and COCO2017, LPIPS decreases from 0.1332 to 0.0603 and from 0.1248 to 0.0581. The discrete variant improves PSNR over VQ-VAE by 0.62, 0.64, and 0.70 dB, alongside improvements in SSIM and LPIPS. These results establish the reconstruction benefits of the proposed framework across continuous and vector-quantized representations

5.3 Empirical Validation of the Reconstruction Gap

We assess the empirical implication of Proposition 3.3: whether encoder-produced representations leave reconstruction improvements attainable through latent optimization with the same decoder. We examine this reconstruction headroom across pretrained models and datasets, and assess its practical significance against state-of-the-art reconstruction methods.

5.3.1 Latent Optimization with Fixed Decoders

We evaluate LDM-AE ($f = 8$, KL), LDM-VQAE ($f = 8$, $Z = 256$, $d = 4$) (Rombach et al., 2022), and DCAE ($f = 32$, $c = 32$) (Chen et al., 2025c) across six datasets. Each latent is initialized from the corresponding encoder and optimized with the decoder fixed, using the reconstruction and perceptual loss terms of the original autoencoder. Tab. 3 shows improvements in PSNR, SSIM, and LPIPS across all 18 model–dataset pairs. For example, Latent optimization for LDM-AE increases ImageNet PSNR from 25.67 to 27.40 dB and reduces LPIPS from 0.1377 to 0.0933. Because the decoder remains unchanged, these gains demonstrate that its initial encoder outputs do not fully

Table 2: Reconstruction performance on CIFAR-10, ImageNet-1K, and COCO2017 with continuous and discrete latent representations. AE denotes a conventional autoencoder in the continuous setting and a VQ-VAE in the discrete setting. PSDRVars denotes the proposed PSDR-AD framework. PSNR is reported in dB. Bold values indicate the best result within each representation setting. ↑ and ↓ indicate higher and lower values are better, respectively.

Method	CIFAR-10			ImageNet-1K			COCO2017		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Continuous Latents									
AE	26.25	0.8522	0.2074	28.35	0.8264	0.1332	28.64	0.8508	0.1248
GLO	25.10	0.8167	0.2487	17.32	0.3811	0.6864	17.79	0.4103	0.6680
PSDR-AD	**27.72**	**0.8861**	**0.1706**	**29.19**	**0.8451**	**0.0603**	**29.49**	**0.8655**	**0.0581**
Discrete Latents									
AE	26.13	0.8678	0.1860	27.61	0.8348	0.1668	27.92	0.8491	0.1494
PSDR-AD	**26.75**	**0.8772**	**0.1671**	**28.25**	**0.8450**	**0.0893**	**28.62**	**0.8549**	**0.0814**

Table 3: Reconstruction performance before and after latent optimization with fixed pretrained decoders across six datasets. "+Opt." denotes direct optimization of encoder-initialized latent representations while keeping the corresponding decoder fixed. PSNR is reported in dB. Bold values indicate the better result within each model pair.

Method	ImageNet-1K			COCO2017			LSUN Bed.		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM-AE	25.67	0.7240	0.1377	25.43	0.7375	0.1344	28.36	0.8399	0.1032
LDM-AE+Opt.	**27.40**	**0.7751**	**0.0933**	**27.66**	**0.8026**	**0.0860**	**30.63**	**0.8861**	**0.0611**
DCAE	23.93	0.6593	0.1641	23.63	0.6718	0.1635	26.43	0.7919	0.1327
DCAE+Opt.	**25.89**	**0.7232**	**0.1386**	**26.15**	**0.7457**	**0.1203**	**28.90**	**0.8460**	**0.0908**
LDM-VQAE	22.67	0.6310	0.1906	22.58	0.6402	0.1886	24.88	0.7558	0.1636
LDM-VQAE+Opt.	**24.52**	**0.6991**	**0.1748**	**24.46**	**0.7065**	**0.1713**	**26.36**	**0.8024**	**0.1412**

Method	CelebA-HQ			CIFAR-10			Kodak		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM-AE	31.25	0.8685	0.0807	17.08	0.5180	0.2865	26.66	0.8536	0.1455
LDM-AE+Opt.	**33.25**	**0.9029**	**0.0501**	**22.29**	**0.6934**	**0.1481**	**28.87**	**0.8922**	**0.0923**
DCAE	29.13	0.8223	0.1068	16.36	0.3540	0.4243	25.32	0.7932	0.1614
DCAE+Opt.	**31.30**	**0.8672**	**0.0749**	**19.25**	**0.5086**	**0.3210**	**27.54**	**0.8425**	**0.1207**
LDM-VQAE	27.53	0.7971	0.1304	12.51	0.3041	0.4895	24.16	0.7775	0.1988
LDM-VQAE+Opt.	**28.67**	**0.8390**	**0.1124**	**18.82**	**0.4924**	**0.3456**	**25.90**	**0.8168**	**0.1776**

exploit its reconstruction capability. The results provide empirical support for the reconstruction headroom identified by Proposition 3.3 under the practical optimization objective used here.

5.3.2 Comparison with State-of-the-Art Reconstruction Methods

Tab. 4 compares continuous reconstruction methods on the ImageNet validation set at 256×256 resolution. The comparison reports latent scalar counts alongside different architectures and model sizes. Optimization preserves the representation budget within each model pair while allocating additional inference computation; Appendix reports these costs. Optimization improves all four metrics for each evaluated backbone while preserving its decoder and latent dimensionality. For LDM-f8 and LDM-f4, rFID decreases from 0.744 to 0.044 and from 0.284 to 0.017, respectively. Most notably, FLUX-VAE with optimized latents outperforms all compared state-of-the-art models across all four metrics, achieving 32.66 dB PSNR, 0.9144 SSIM, 0.0286 LPIPS, and 0.005 rFID. Relative to the original FLUX-VAE, PSNR improves by 1.60 dB and rFID decreases from 0.176 to 0.005. These paired improvements demonstrate that the reconstruction headroom associated with

Table 4: Comparison with State-of-the-Art image reconstruction Methods on the ImageNet-1K validation set at 256×256 resolution. Scalars denotes the total number of scalar elements in the latent representation of each image, computed as the product of all dimensions in Latent Shape. "+Opt." denotes latent optimization with the corresponding pretrained decoder fixed. PSNR is reported in dB. ↑ and ↓ indicate higher and lower values are better, respectively.

Scalars	Method	Latent Shape	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓
2,048	DC-AE f32c32 (Chen et al., 2025c)	$8 \times 8 \times 32$	23.93	0.659	0.164	0.774
	DC-AE f64c128 (Chen et al., 2025c)	$4 \times 4 \times 128$	23.69	0.651	0.172	0.810
	SoftVQ-L-64 (Chen et al., 2025b)	64×32	22.97	0.739	–	0.610
	DC-AE f32c32+Opt. (Chen et al., 2025c)	$8 \times 8 \times 32$	**25.89**	**0.723**	**0.139**	**0.175**
4,096	SD-VAE (Rombach et al., 2022)	$32 \times 32 \times 4$	26.26	0.745	0.133	0.606
	MAETok-B-128 (Chen et al., 2025a)	128×32	23.61	0.763	–	0.480
	Cosmos-CI1616 (Agarwal et al., 2025)	$16 \times 16 \times 16$	25.07	0.700	0.167	0.959
	LDM-f8 (Rombach et al., 2022)	$32 \times 32 \times 4$	25.67	0.7240	0.1377	0.744
	LDM-f8+Opt.	$32 \times 32 \times 4$	**27.40**	**0.775**	**0.093**	**0.044**
12,288	LDM-f4 (Rombach et al., 2022)	$64 \times 64 \times 3$	28.30	0.8313	0.0803	0.284
	LDM-f4+Opt.	$64 \times 64 \times 3$	**29.97**	**0.8625**	**0.0497**	**0.017**
16,384	SD3-VAE (Esser et al.)	$32 \times 32 \times 16$	29.56	0.856	0.069	0.212
	Qwen-Image VAE (Wu et al., 2025)	$32 \times 32 \times 16$	30.22	0.867	0.065	1.459
	Cosmos-CI88 (Agarwal et al., 2025)	$32 \times 32 \times 16$	30.50	0.875	0.073	1.145
	FLUX-VAE (Labs et al., 2025)	$32 \times 32 \times 16$	31.06	0.893	0.051	0.176
	FLUX-VAE+Opt.	$32 \times 32 \times 16$	**32.66**	**0.9144**	**0.0286**	**0.005**

Table 5: Downstream image generation on CIFAR-10 using discrete and continuous latent representations. An autoregressive Transformer is used for discrete codes, and a diffusion model is used for continuous latents. Reconstruction and generation metrics are reported separately.

Method	Tokens	Reconstruction				Generation	
		PSNR↑	SSIM↑	LPIPS↓	rFID↓	FID↓	IS↑
Transformer + VQ-VAE	64	26.13	0.8678	0.1860	44.85	63.66	5.903 ± 0.069
Transformer + PSDR-AD(VQ)	64	**26.75**	**0.8772**	**0.1671**	**42.65**	**59.10**	$\mathbf{6.240 \pm 0.063}$
Diffusion + AE	64	26.25	0.8522	0.2074	52.78	81.96	4.757 ± 0.026
Diffusion + GLO	64	25.10	0.8167	0.2487	41.28	133.26	2.865 ± 0.016
Diffusion + PSDR-AD	64	**27.72**	**0.8861**	**0.1706**	**33.78**	**68.02**	$\mathbf{5.189 \pm 0.038}$

encoder-based inference remains practically significant even for strong pretrained models. Direct latent optimization accesses this capacity without decoder retraining or additional latent dimensions.

5.4 Downstream Image Generation

We assess whether directly optimized representations support generative modeling on CIFAR-10. An autoregressive Transformer is trained on discrete codes from VQ-VAE and PSDR-AD(VQ), while a diffusion model is trained on continuous latents from AE, GLO, and PSDR-AD. Within each setting, the latent configuration, generative architecture, and training protocol are shared. We generate 50,000 images per method and report FID and IS. Prior work has shown that reconstruction FID is not a reliable predictor of downstream generation FID (Xu et al., 2026). As shown in Tab. 5, PSDR-AD(VQ) reduces autoregressive generation FID from 63.66 to 59.10 and increases IS from 5.903 to 6.240 relative to VQ-VAE. For diffusion generation, PSDR-AD reduces FID from 81.96 to 68.02 and increases IS from 4.757 to 5.189 relative to AE. Although GLO achieves a lower reconstruction rFID than AE, its generation FID is substantially worse, further illustrating the need for direct downstream evaluation. Under matched generative architectures and training protocols within each setting, these results provide empirical evidence that PSDR-AD representations benefit downstream generation under the evaluated experimental conditions, beyond improvements in reconstruction quality alone.

6 Limitations and Discussion

Representation budget and computational trade-off. A central objective of latent representation learning is to improve reconstruction quality under a given representation budget. For continuous latents, we characterize this budget by the number of scalar elements and the latent shape, which describe dimensional capacity rather than an entropy-coded bitrate or the actual information content. For discrete latents, the budget depends on both token count and codebook size. Network architecture and computational resources affect how representations are learned and inferred, but are distinct from the representation budget. It is under this definition of budget that methods with different architectures and computational costs can be compared in terms of reconstruction quality under a common representation budget, with computational cost evaluated separately. Our approach explores a different direction from architectural improvements or increased model size, trading additional latent-inference computation for improved reconstruction quality. The comparison between our approach and existing autoencoders is therefore fair in terms of representation budget.

Training scale and downstream generation. Our resources limit the scale of joint latent–decoder training. We use fixed pretrained decoders for the high-capacity reconstruction experiments; this is a supported use of PSDR-AD rather than a requirement of the framework. Training a comparably large PSDR-AD from scratch remains to be evaluated. We do train downstream autoregressive and diffusion models using representations from our own method on CIFAR-10 (Tab. 5), with shared generative architectures and training protocols within each comparison. These experiments show downstream benefits under the evaluated budget. We do not have the resources to train state-of-the-art-scale generative priors for every evaluated tokenizer, or to establish their fully converged performance. In particular, the strong pretrained-decoder reconstruction results do not by themselves establish state-of-the-art generation quality. Reconstruction rFID is not a reliable substitute for generation FID (Xu et al., 2026); the elevated absolute FID of our small-scale generation experiments limits the scope of that evidence.

Access to test images. The target image is the observed input to the reconstruction task for both AE and AD inference. In the latter, only its instance-specific latent is fitted; the decoder is not trained on test images. This is an optimization-based representation extraction procedure. The paired results in Tab. 3 compare these inference mechanisms using the same decoder and representation size, with the computational trade-off described above.

7 Conclusion

In this work, we revisited the conventional encoder-based paradigm for latent representation learning and investigated whether the latent representations produced by a shared encoder are optimal inputs to the decoder. Based on the observation that encoder-based inference may constrain the representation ability of the latent space, we proposed Parametric Soft Discrete Random Variables (PSDR-Vars), which enable latent representations to be directly optimized through a decoder. We further developed the Parametric Soft Discrete Random Variable Auto-Decoder (PSDR-AD), a decoder-only framework in which an independently parameterized PSDR-Var is assigned to each instance and optimized jointly with the decoder. For unseen images, new PSDR-Vars are directly optimized while the decoder is fixed. The proposed framework supports both continuous and discrete representations and can be trained from scratch or applied to pretrained autoencoders by bypassing the encoder and optimizing the inputs to their fixed decoders. Extensive experiments demonstrate that direct latent optimization consistently improves reconstruction quality over encoder-generated representations and achieves state-of-the-art reconstruction performance using only a single RTX 4080 SUPER GPU. These results support our hypothesis that directly optimizing decoder inputs provides an effective alternative to conventional encoder-based latent representation learning.

AI use statement

Generative AI tools were used solely to improve readability, correct grammatical and typographical errors, and assist with the LaTeX typesetting of equations. They were not used for research ideation, mathematical reasoning, methodology or experiment design, implementation, data analysis, or result

interpretation. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content.

Ethics statement

This work uses only publicly available benchmark datasets and does not involve human-subject studies or the collection of new personal or sensitive data. We are not aware of any specific ethical concerns associated with this work.

Reproducibility statement

The theoretical assumptions and justification for the encoder-induced reconstruction gap are provided in Proposition 3.3. Sec. 4 describes the proposed PSDR-AD framework and its training and test-time optimization procedures. Sec. A.3 specifies the model architectures, dataset splits, preprocessing procedures, evaluation metrics, and computational environment. The configurations for the noise ablation and comparisons with autoencoders and GLO are detailed in Sec. A.4 and Sec. A.5, respectively. Sec. A.6 and Sec. A.7 describe the pretrained models, latent initialization, optimization objectives, and evaluation protocols for the reconstruction comparisons. Downstream generation settings are provided in Sec. A.8, and optimization runtimes are reported in Sec. A.1. We include demo code in the supplementary material to illustrate the implementation and usage of the proposed method. The full source code will be released upon acceptance of the paper to facilitate reproducibility.

References

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 600–609. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/bojanowski18a.html.

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

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

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 96539–96560, 2025c.

Ricky T. Q. Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International conference on machine learning*, pp. 1078–1086. PMLR, 2018.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*.

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

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2649–2658. PMLR, 2018.

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

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.

Jindong Li, Yali Fu, Jiahong Liu, Linxiao Cao, Wei Ji, Menglin Yang, Irwin King, and Ming-Hsuan Yang. Discrete tokenization for multimodal llms: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Jiasen Lu, Liangchen Song, Mingze Xu, Byeongjoo Ahn, Yanjun Wang, Chen Chen, Afshin Dehghan, and Yinfei Yang. AToken: A unified tokenizer for vision. *arXiv preprint arXiv:2509.14476*, 2025. URL https://arxiv.org/abs/2509.14476v1. Version 1, Table 4.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pp. 3403–3412. PMLR, 2018.

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

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Yura Perugachi-Diaz, Arwin Gansekoele, and Sandjai Bhulai. Robustly overfitting latents for flexible neural image compression. In *Advances in Neural Information Processing Systems*, volume 37, pp. 106714–106742. Curran Associates, Inc., 2024. doi: 10.52202/079017-3388. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c1233cb30ef11a3fe6ab10aa96beabaf-Paper-Conference.pdf.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 29, pp. 3745–3753, 2016.

Yuhta Takida, Takashi Shibuya, Weihsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. Sq-vae: Variational bayes on discrete representation with self-annealed stochastic quantization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20987–21012. PMLR, 2022.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

van den Oord et al. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Tongda Xu, Mingwei He, Shady Abu-Hussein, Jose Miguel Hernandez-Lobato, Chunhang Zheng, Kai Zhao, Chao Zhou, Ya-Qin Zhang, and Yan Wang. Making reconstruction fid predictive of diffusion generation fid. *arXiv preprint arXiv:2603.05630*, 2026.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In *Advances in Neural Information Processing Systems*, volume 33, pp. 573–584. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/066f182b787111ed4cb65ed437f0855b-Paper.pdf.

Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation. In *International Conference on Learning Representations*, 2024.

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 90844–90868, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e25198b6a75f74277ee3a2bd4165d9ef-Paper-Conference.pdf.

Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22798–22807, 2023.

A Appendix

A.1 Computational Cost of Latent Optimization

Tab. 6 reports the configurations and per-update runtimes corresponding to the preceding reconstruction experiments on a single RTX 4080 SUPER (16 GB). The amortized optimization cost per image is computed as the per-update runtime multiplied by the number of updates and divided by the batch size.

Correspondence to reconstruction experiments. The runtime entries in Tab. 6 are linked below to the corresponding reconstruction results. Unless a single dataset is specified, each triple lists seconds per optimization update on ImageNet-1K, COCO2017, and CIFAR-10, in that order.

- **GLO** (0.51 seconds, 0.52 seconds, 0.27 seconds) corresponds to the GLO row in Tab. 2, Sec. 5.2.
- **PSDR-AD** (0.54 seconds, 0.54 seconds, 0.29 seconds) corresponds to the continuous PSDR-AD row in Tab. 2, Sec. 5.2. The CIFAR-10 configuration also corresponds to the $\sigma_0 = 0.20$ setting in Tab. 1, Sec. 5.1.
- **PSDR-AD(VQ)** (0.47 seconds, 0.47 seconds, 0.32 seconds) corresponds to the discrete PSDR-AD row in Tab. 2, Sec. 5.2.
- **LDM-f4** (0.49 seconds per update on ImageNet-1K) corresponds to LDM-f4+Opt. in Tab. 4, Sec. 5.3.2.
- **LDM-f8** (0.46 seconds, 0.44 seconds, 0.40 seconds) corresponds to LDM-AE+Opt. on the respective datasets in Tab. 3, Sec. 5.3.1. The ImageNet-1K entry also corresponds to LDM-f8+Opt. in Tab. 4, Sec. 5.3.2.
- **DCAE** (0.37 seconds, 0.36 seconds, 0.27 seconds) corresponds to DCAE+Opt. on the respective datasets in Tab. 3, Sec. 5.3.1. The ImageNet-1K entry also corresponds to DC-AE f32c32+Opt. in Tab. 4, Sec. 5.3.2.
- **FLUX-VAE** (0.38 seconds per update on ImageNet-1K) corresponds to FLUX-VAE+Opt. in Tab. 4, Sec. 5.3.2.

On ImageNet, continuous and discrete PSDR-AD with initialization from scratch require approximately 16.88 and 14.69 seconds per image, respectively. Encoder-initialized optimization requires 6.75–12.25 seconds per image on ImageNet and 0.23–0.64 seconds on CIFAR-10. In particular, latent optimization for FLUX-VAE takes 38 seconds per batch of four ImageNet images, or 9.50 seconds per image, to obtain the results in Tab. 4. These amortized values describe batch throughput rather than single-image latency.

Although iterative optimization incurs additional inference cost, the measured runtimes offer a practical trade-off for offline representation extraction and applications that prioritize reconstruction fidelity. More capable hardware could further reduce runtime, including through larger batches where memory permits. The framework thus converts additional inference computation into improved reconstruction using a fixed decoder.

A.2 Cross-Dataset Generalization

We evaluate whether PSDR-AD supports reconstruction across ImageNet-1K and COCO2017. Both settings use images unseen during decoder training: in-distribution (ID) evaluation uses the validation split of the training dataset, whereas out-of-distribution (OOD) evaluation uses that of the other dataset. Whenever COCO2017 is the evaluation dataset, both ID and OOD evaluation use the full official `val2017` validation set of 5,000 images. Only PSDR-Vars are optimized; the decoder remains fixed.

As shown in Tab. 7, cross-dataset transfer largely preserves reconstruction performance. Relative to ID evaluation on the same target dataset, continuous PSDR-AD incurs PSNR decreases of 0.36 dB on COCO2017 and 0.98 dB on ImageNet-1K. The discrete variant exhibits smaller, metric-dependent changes, with absolute PSNR differences of at most 0.20 dB. These findings support

Table 6: Runtime of latent optimization on ImageNet-1K, COCO2017, and CIFAR-10 with initialization from scratch or encoder outputs. Latent Size is reported as $H \times W \times C$. BS. denotes batch size, Steps denotes the number of optimization updates, and Time is measured in seconds per update.

Method	ImageNet-1K				COCO2017				CIFAR-10			
	LS.	BS.	Steps	Time	LS.	BS.	Steps	Time	LS.	BS.	Steps	Time
Initialization: from scratch												
GLO	64×64×3	32	1,000	0.51	64×64×3	32	1,000	0.52	8×8×3	1,024	5,000	0.27
PSDR-AD	64×64×3	32	1,000	0.54	64×64×3	32	1,000	0.54	8×8×3	1,024	5,000	0.29
PSDR-AD(VQ)	64×64×10	32	1,000	0.47	64×64×10	32	1,000	0.47	8×8×10	1,024	5,000	0.32
Initialization: encoder output												
LDM-f4	64×64×3	4	100	0.49	64×64×3	4	100	0.49	8×8×3	64	100	0.37
LDM-VQ-f8	32×32×4	4	100	0.27	32×32×4	4	100	0.26	4×4×4	64	100	0.15
LDM-f8	32×32×4	4	100	0.46	32×32×4	4	100	0.44	4×4×4	64	100	0.40
DCAE-f32c32	8×8×32	4	100	0.37	8×8×32	4	100	0.36	1×1×32	64	100	0.27
FLUX-VAE	32×32×16	4	100	0.38	32×32×16	4	100	0.38	4×4×16	64	100	0.41

LS.: Latent Size; BS.: Batch Size; Steps: Number of optimization updates; Time: seconds per update.

generalization through sample-specific latent optimization without adapting the decoder to the target dataset.

Table 7: Generalization evaluation of the proposed approach under in-distribution and out-of-distribution settings across COCO2017 and ImageNet-1K using PSNR, SSIM, and LPIPS.

	Continuous			Discrete		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
In-distribution						
ImageNet-1K	29.19	0.8451	0.0603	28.25	0.8450	0.0893
COCO2017	29.49	0.8655	0.0581	28.62	0.8549	0.0814
Out-of-Distribution						
ImageNet-1K-COCO2017	29.13	0.8542	0.0627	28.42	0.8512	0.0840
COCO2017-ImageNet-1K	28.21	0.8492	0.0628	28.38	0.8432	0.0867

ImageNet-1K-COCO2017: Trained on ImageNet-1K and evaluated on COCO2017; COCO2017-ImageNet-1K: Trained on COCO2017 and evaluated on ImageNet-1K.

A.3 Common Experimental Settings

Datasets and preprocessing. For experiments involving decoder training on ImageNet-1K and COCO2017, we randomly select 50,000 images from each training set. Evaluation uses the full 50,000-image ImageNet-1K validation set and the full official COCO2017 validation set (`val2017`, 5,000 images). For CIFAR-10, we use all 50,000 training images and all 10,000 test images. ImageNet-1K and COCO2017 images are center-cropped to 256×256, whereas CIFAR-10 images retain their original resolution of 32×32. Pixel values are normalized to $[-1, 1]$. All noise injected into latent representations in our experiments is Gaussian.

Reconstruction metrics. We evaluate reconstruction quality using PSNR, SSIM, and LPIPS, and report the mean of the per-image scores. For PSNR, images are mapped from $[-1, 1]$ to $[0, 255]$. The mean squared error is computed over all spatial locations and RGB channels for each image, with a peak intensity of 255. SSIM is computed using the `piq` implementation after mapping images to $[0, 1]$, with `data_range=1`. LPIPS is computed using the `lpips` package with its VGG backbone and image inputs in $[-1, 1]$. The LPIPS network is used in evaluation mode.

Table 8: Architecture of the continuous autoencoder for 32×32 RGB images. N denotes the batch size and d the latent channel dimension.

Stage	Operation	Channels	Kernel	Stride / Pad.	Output shape
Input	RGB image	3	–	–	$N \times 3 \times 32 \times 32$
Encoder	Conv2d + ReLU	$3 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	Conv2d + ReLU	$64 \to 128$	4×4	$2/1$	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \to d$	3×3	$1/1$	$N \times d \times 8 \times 8$
Decoder	Conv2d	$d \to 64$	3×3	$1/1$	$N \times 64 \times 8 \times 8$
	PosEmb.	$64 \to 64$	–	–	$N \times 64 \times 8 \times 8$
	ConvTranspose2d	$64 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	ConvTranspose2d + ReLU	$128 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	ConvTranspose2d	$64 \to 3$	4×4	$2/1$	$N \times 3 \times 32 \times 32$
	Tanh	$3 \to 3$	–	–	$N \times 3 \times 32 \times 32$

Each ResidualStack contains two residual layers with 128 input/output channels and 32 residual hidden channels. Pad. denotes padding. PosEmb refers to the positional embedding layer.

Table 9: Architecture of the continuous decoder-only model for 32×32 RGB reconstruction. Each image has an independently learnable latent tensor. The decoder has the same architecture as that in Tab. 8.

Stage	Operation	Channels	Kernel	Stride / Pad.	Output shape
Input	Learnable latent	d	–	–	$N \times d \times 8 \times 8$
Decoder	Conv2d	$d \to 64$	3×3	$1/1$	$N \times 64 \times 8 \times 8$
	PosEmb.	$64 \to 64$	–	–	$N \times 64 \times 8 \times 8$
	ConvTranspose2d	$64 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	ConvTranspose2d + ReLU	$128 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	ConvTranspose2d	$64 \to 3$	4×4	$2/1$	$N \times 3 \times 32 \times 32$
	Tanh	$3 \to 3$	–	–	$N \times 3 \times 32 \times 32$

The ResidualStack contains two residual layers with 128 input/output channels and 32 residual hidden channels. Training-time Gaussian perturbations follow the experimental settings described in the text. PosEmb refers to the positional embedding layer.

FID and reconstruction FID. We compute FID and reconstruction FID (rFID) using `pytorch-fid` version 0.2.1 with its default FID-specific pretrained Inception-v3 network, extracting 2,048-dimensional features from the final average pooling layer. The evaluation protocol is specified separately for each experiment.

Implementation and result reporting. Experiments are conducted using a single NVIDIA GeForce RTX 4080 SUPER GPU with 16 GB of memory. The software environment consists of Python 3.11.7, PyTorch 2.4.1 with CUDA 12.1, and cuDNN 9.1.0. Image tensors used for reconstruction metric evaluation are represented in 32-bit floating-point precision. PSNR, SSIM, and LPIPS are computed for individual image pairs and averaged over the evaluated images. FID and rFID are computed from feature statistics aggregated over their respective image sets.

A.4 Details of the Noise Ablation

This subsection provides the experimental details for the noise ablation in Sec. 5.1.

Table 10: Architecture of VQ-VAE for 32×32 RGB images. The codebook contains 1,024 embeddings of dimension 128. Quantization produces an 8×8 grid of discrete indices and corresponding embeddings for decoding.

Stage	Operation	Channels	Kernel	Stride / Pad.	Output shape
Input	RGB image	3	–	–	$N \times 3 \times 32 \times 32$
Encoder	Conv2d + ReLU	$3 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	Conv2d + ReLU	$64 \to 128$	4×4	$2/1$	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
Quantizer	Nearest-code assignment	–	–	–	$N \times 8 \times 8$
	Codebook lookup	128	–	–	$N \times 128 \times 8 \times 8$
Decoder	Conv2d	$128 \to 64$	3×3	$1/1$	$N \times 64 \times 8 \times 8$
	ConvTranspose2d	$64 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	ConvTranspose2d + ReLU	$128 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	ConvTranspose2d	$64 \to 3$	4×4	$2/1$	$N \times 3 \times 32 \times 32$
	Tanh	$3 \to 3$	–	–	$N \times 3 \times 32 \times 32$

Each ResidualStack contains two residual layers with 128 input/output channels and 32 residual hidden channels. At each spatial position, the quantizer selects the nearest codebook embedding by Euclidean distance. The assignment and lookup rows describe the two operations of the same vector quantizer.

Table 11: Architecture of PSDR-AD(VQ) for 32×32 RGB reconstruction. Each image has an independently learnable latent tensor, which is quantized using a codebook of 1,024 128-dimensional embeddings. The decoder has the same architecture as that in Tab. 10.

Stage	Operation	Channels	Kernel	Stride / Pad.	Output shape
Input	Learnable latent	128	–	–	$N \times 128 \times 8 \times 8$
Quantizer	Nearest-code assignment	–	–	–	$N \times 8 \times 8$
	Codebook lookup	128	–	–	$N \times 128 \times 8 \times 8$
Decoder	Conv2d	$128 \to 64$	3×3	$1/1$	$N \times 64 \times 8 \times 8$
	ConvTranspose2d	$64 \to 128$	3×3	$1/1$	$N \times 128 \times 8 \times 8$
	ResidualStack	$128 \to 128$	–	–	$N \times 128 \times 8 \times 8$
	ConvTranspose2d + ReLU	$128 \to 64$	4×4	$2/1$	$N \times 64 \times 16 \times 16$
	ConvTranspose2d	$64 \to 3$	4×4	$2/1$	$N \times 3 \times 32 \times 32$
	Tanh	$3 \to 3$	–	–	$N \times 3 \times 32 \times 32$

The ResidualStack contains two residual layers with 128 input/output channels and 32 residual hidden channels. Vector quantization follows the same assignment and lookup operations as in Tab. 10, yielding 64 discrete tokens per image.

Architecture and latent initialization. The noise ablation uses continuous PSDR-AD on CIFAR-10, with the architecture specified in Tab. 9 and latent channel dimension $d = 3$. Each image is associated with an independently learnable latent tensor of shape $3 \times 8 \times 8$ in channel–height–width order, initialized to zero. The controlling function is the identity mapping, $\psi(\mathbf{z}) = \mathbf{z}$.

Training configuration. We use Adam to optimize the decoder parameters, per-image latent parameters, and noise scale σ. The decoder learning rate is 3×10^{-4}, while the learning rates for the latent parameters and σ are both 3×10^{-3}. The batch size is 1,024. Training runs for 5,000 epochs, where one epoch denotes a complete pass over the training dataset. The reconstruction term is the

squared ℓ_2 loss, without an additional perceptual loss such as LPIPS. The noise-scale regularization term follows the training objective in the main text.

Noise-scale settings. For each target noise scale σ_0, we initialize $\sigma = \sigma_0$ and learn σ jointly with the decoder and latent parameters. Each setting is trained independently, with all other experimental settings held constant. In particular, σ remains learnable when $\sigma_0 = 0$; this setting specifies a zero target for the noise scale rather than fixing the training noise scale to zero.

Test-time latent optimization. At test time, the trained decoder in Tab. 9 is fixed and noise injection is disabled. New latent tensors for the test images are initialized to zero and optimized using the squared ℓ_2 reconstruction loss. Latent optimization runs for 5,000 epochs, where one epoch denotes a complete pass over the test dataset. Thus, the noise ablation evaluates the effect of training with different target noise scales while using noiseless latent optimization during evaluation.

AE latent optimization reference. The AE+Opt. reference uses the continuous autoencoder architecture in Tab. 8, with $d = 3$. Its decoder has the same architecture as the PSDR-AD decoder in Tab. 9, but its parameters are learned through autoencoder training. The autoencoder is trained for 5,000 epochs using Adam with a learning rate of 3×10^{-4}. At test time, latent parameters are initialized from the encoder outputs. The decoder is then fixed, and the latent parameters are optimized using Adam with a learning rate of 3×10^{-3} for 5,000 epochs over the test dataset. Both autoencoder training and test-time optimization use the squared ℓ_2 reconstruction loss without a perceptual loss. The remaining applicable experimental settings are shared with the noise-ablation experiments.

A.5 Details of Comparisons with Autoencoders and GLO

This subsection provides the experimental details for the comparisons in Sec. 5.2 and Tab. 2.

Architectures and representation configurations. The continuous AE uses the architecture in Tab. 8, while GLO and continuous PSDR-AD use the decoder-only architecture in Tab. 9. VQ-VAE and PSDR-AD(VQ) use the architectures in Tab. 10 and Tab. 11, respectively. Within each representation setting, the compared methods use the same decoder architecture, with parameters trained independently for each method.

For continuous representations, we use latent tensors of shape $3 \times 8 \times 8$ on CIFAR-10 and $3 \times 64 \times 64$ on ImageNet-1K and COCO2017, in channel–height–width order. For vector-quantized representations, both methods use a codebook containing 1,024 embeddings of dimension 128. The spatial token grids are 8×8 on CIFAR-10 and 64×64 on ImageNet-1K and COCO2017. The architecture tables show the CIFAR-10 configuration; the same convolutional architectures accommodate the larger spatial dimensions of the other datasets.

Training configuration. All methods are trained from scratch on each dataset, without transferring parameters between datasets. We use Adam with a learning rate of 3×10^{-4} for network parameters and 3×10^{-3} for directly optimized latent parameters and the learnable noise scale, where applicable, following Sec. A.4. AE and VQ-VAE learn their encoder and decoder parameters, whereas GLO and PSDR-AD jointly optimize the decoder and per-image latent parameters.

The batch size is 1,024 on CIFAR-10 and 32 on ImageNet-1K and COCO2017. All methods are trained for 5,000 epochs on CIFAR-10 and 100 epochs on ImageNet-1K and COCO2017. One epoch denotes a complete pass over the corresponding dataset. For PSDR-AD, we set the target noise scale to $\sigma_0 = 0.20$ in the continuous setting and $\sigma_0 = 0$ in the vector-quantized setting. Noise-scale initialization and learning follow Sec. A.4.

Reconstruction objective. On CIFAR-10, the reconstruction objective is the squared ℓ_2 loss without a perceptual loss, consistent with the noise ablation in Sec. A.4. On ImageNet-1K and COCO2017, we additionally include an Inception-v3-based perceptual loss with weight 0.1:

$$\mathcal{L}_{\mathrm{rec}} = \begin{cases} \mathcal{L}_{\ell_2}, & \text{CIFAR-10,} \\ \mathcal{L}_{\ell_2} + 0.1\,\mathcal{L}_{\mathrm{perc}}, & \text{ImageNet-1K and COCO2017.} \end{cases} \tag{6}$$

The corresponding dataset-specific reconstruction objective is used during training and, where applicable, test-time latent optimization for all compared methods. Method-specific regularization and quantization losses are included where applicable.

GLO configuration. We follow the original GLO formulation for the latent norm constraint and projection schedule. To control the comparison, GLO and continuous PSDR-AD use the same decoder architecture, latent dimensions, initialization procedure, optimizer settings, learning rates, batch sizes, and training and test-time optimization budgets, while retaining their respective latent constraints and regularization. The evaluated GLO models reached convergence within the specified training budgets.

Vector quantization. VQ-VAE quantizes the encoder-produced representations, whereas PSDR-AD(VQ) quantizes directly optimized latent representations. Both use the same codebook size and embedding dimension. The commitment and codebook losses, codebook update rule, and gradient propagation through quantization follow the original VQ-VAE formulation.

Test-time evaluation. AE and VQ-VAE reconstruct evaluation images using forward passes without test-time latent optimization. For GLO and PSDR-AD, the trained decoder is fixed and only the per-image latent parameters are optimized, using Adam with a learning rate of 3×10^{-3} and the dataset-specific reconstruction objective in Eq. 6. Noise injection is disabled during PSDR-AD evaluation.

Test-time latent optimization runs for 5,000 epochs on CIFAR-10 and 100 epochs on ImageNet-1K and COCO2017, with batch sizes of 1,024 and 32, respectively. Each test-time epoch traverses the complete evaluation dataset. The comparisons match decoder architectures and representation configurations within each setting, while the additional computation required for iterative latent inference is reported separately.

A.6 Details of Latent Optimization with Fixed Decoders

This subsection provides the experimental details for Sec. 5.3.1 and Tab. 3.

Pretrained models. We evaluate LDM-AE with KL regularization and spatial downsampling factor $f = 8$, LDM-VQAE with $f = 8$, codebook size $Z = 256$, and embedding dimension $d = 4$, and DC-AE with $f = 32$ and 32 latent channels.

The pretrained models are obtained from the corresponding official repositories. For DC-AE, we use the ImageNet-trained model. Specific repository and checkpoint download links will be provided upon acceptance.

Evaluation datasets and resolutions. We evaluate on ImageNet-1K, COCO2017, LSUN Bedrooms, CelebA-HQ, CIFAR-10, and Kodak, using the complete evaluation split for each dataset and the full Kodak image set. For COCO2017, this is the official `val2017` validation set containing all 5,000 images. CIFAR-10 images retain their original resolution of 32×32, while Kodak images are evaluated at their original resolutions. Images from the remaining datasets are evaluated at 256×256 resolution. All inputs are RGB images. Shared preprocessing and metric definitions follow Sec. A.3.

Latent initialization and frozen parameters. For each evaluation image, the latent representation is initialized from the output of the corresponding pretrained encoder. Only the image-specific latent parameters are optimized. All model parameters remain fixed, including the encoder, decoder, and, for LDM-VQAE, the codebook. Noise injection is disabled throughout latent optimization and evaluation.

For LDM-VQAE, we optimize the continuous latent representation before quantization. At each forward pass, this representation is passed through the fixed quantizer, and the resulting quantized embeddings are supplied to the decoder. A straight-through estimator (STE) propagates gradients through quantization to the continuous latent parameters. The discrete representation is obtained by assigning each optimized latent vector to its nearest entry in the unchanged codebook.

Optimization objective. All three models use the same form of reconstruction objective:

$$\mathcal{L}_{\text{ref}} = \mathcal{L}_{\ell_2} + 0.1\,\mathcal{L}_{\text{perc}}, \tag{7}$$

where $\mathcal{L}_{\ell_2}$ is the squared pixel-space reconstruction loss and $\mathcal{L}_{\text{perc}}$ is the perceptual loss. No additional constraints or regularization terms are imposed on the optimized latent parameters. For LDM-VQAE, the fixed quantizer remains part of the reconstruction forward pass, but no additional codebook or commitment loss is included in the latent optimization objective.

Optimization configuration. Latent parameters are optimized using Adam with a learning rate of 3×10^{-3} for 100 epochs. One epoch denotes a complete pass over the corresponding evaluation dataset. The batch size is 64 for CIFAR-10, 4 for datasets evaluated at 256×256, and 2 for Kodak. The reconstruction before optimization and the optimized reconstruction are evaluated using the same images, pretrained model parameters, and metric implementations.

A.7 Details of State-of-the-Art Reconstruction Comparisons

This subsection provides the experimental details for Sec. 5.3.2 and Tab. 4.

Sources of reported results. The results for SoftVQ-L-64 and MAETok-B-128 are taken from the 256×256 ImageNet comparison in Table 4 of Chen et al. (2025a). The results for SD-VAE and Cosmos-CI1616 are taken from Table 4 of the first arXiv version of AToken (Lu et al., 2025). All remaining entries in Tab. 4, including all results marked "+Opt.", are evaluated by us using publicly released pretrained models.

The four externally reported entries retain the evaluation protocols of their respective sources. Our common evaluation protocol, described below, applies to the entries evaluated by us. The paired comparisons before and after optimization use the same images, model parameters, preprocessing, and metric implementations.

Model configurations and official resources. LDM-f4 and LDM-f8 denote continuous autoencoders with spatial downsampling factors of 4 and 8, respectively. DC-AE f32c32 and f64c128 use spatial downsampling factors of 32 and 64, with 32 and 128 latent channels, respectively. Details of models shared with the preceding experiment are provided in Sec. A.6.

For models evaluated by us, pretrained checkpoints are obtained from the corresponding official repositories. Specific repository and checkpoint download links will be provided upon acceptance.

Cosmos-CI1616 and Cosmos-CI88 denote the continuous image tokenizer variants with spatial compression factors of 16×16 and 8×8, respectively.

SD-VAE denotes the externally reported baseline from AToken, while LDM-f8 denotes the model evaluated in our own reconstruction and optimization experiments. We retain separate entries to distinguish their result provenance; identical latent shapes alone do not establish identical checkpoints or evaluation protocols.

Evaluation protocol. For all models evaluated by us, we use the complete ImageNet-1K validation set containing 50,000 images. Images are center-cropped to 256×256 and processed using the common settings in Sec. A.3. PSNR, SSIM, and LPIPS are computed using the implementations described there.

Reconstruction FID is computed between all 50,000 reconstructed images and the corresponding 50,000 preprocessed original images, using `pytorch-fid` version 0.2.1 with 2,048-dimensional Inception-v3 features. The same reference image set is used before and after latent optimization.

Latent optimization. For each model marked "+Opt.", each image-specific latent representation is initialized from the output of the corresponding pretrained encoder. All model parameters, including those of the decoder, remain fixed throughout optimization. Only the latent parameters are updated, and no additional noise is injected.

We use Adam with a learning rate of 3×10^{-3} and a batch size of 4. Latent optimization runs for 100 epochs, where one epoch denotes a complete pass over the 50,000-image evaluation set. The

optimization objective is

$$\mathcal{L}_{\text{ref}} = \mathcal{L}_{\ell_2} + 0.1\,\mathcal{L}_{\text{perc}}, \tag{8}$$

where $\mathcal{L}_{\ell_2}$ is the squared pixel-space reconstruction loss and $\mathcal{L}_{\text{perc}}$ is the perceptual loss.

Representation budget. The latent shape and scalar count of each model are reported in Tab. 4. The scalar count is the product of the latent tensor dimensions. Optimization preserves the latent shape, scalar count, and pretrained decoder parameters within each before/after pair. It therefore improves reconstruction by allocating additional computation to latent inference without increasing the dimensionality of the representation. Comparisons between groups with different scalar counts correspond to different representation budgets.

A.8 Details of Downstream Image Generation

This subsection provides the experimental details for Sec. 5.4 and Tab. 5. We train autoregressive Transformers on discrete representations from VQ-VAE and PSDR-AD(VQ), and diffusion models on continuous representations from AE, GLO, and PSDR-AD. All experiments use CIFAR-10 images at their original resolution of 32×32.

Representation models and latent preparation. We use the corresponding CIFAR-10 representation models from the comparisons described in Sec. A.5. The continuous AE and decoder-only architectures are specified in Tab. 8 and Tab. 9, respectively. The vector-quantized architectures are specified in Tab. 10 and Tab. 11.

The complete CIFAR-10 training set, containing 50,000 images, is used to construct the latent training dataset for each generative model. For encoder-based models, each image's latent parameters are initialized from its encoder output. For decoder-only models, latent parameters are initialized to zero and optimized anew, rather than reusing the latent parameters saved during representation model training. Latent optimization runs for 5,000 epochs over the training set, using the CIFAR-10 latent optimization settings in Sec. A.5.

All representation model parameters remain fixed during latent optimization, including the decoder and, where applicable, the codebook. For vector-quantized models, optimization is performed on the pre-quantization latent parameters, with gradients propagated through the fixed quantizer using the straight-through estimator. Noise injection is disabled. The resulting latent dataset remains fixed throughout generative model training.

Continuous representations and standardization. Continuous representations have shape $3 \times 8 \times 8$ in channel–height–width order. For each method, we compute the mean and standard deviation of its latent training dataset and standardize the latents to zero mean and unit variance. The diffusion model is trained on these standardized representations. The same statistics are retained and used to invert the standardization before decoding sampled latents.

Discrete representations and tokenization. VQ-VAE and PSDR-AD(VQ) each use a codebook containing 1,024 embeddings of dimension 128. Their quantized latent tensors have shape $128 \times 8 \times 8$. The codebook index at each spatial position is used directly as a discrete token, giving 64 tokens per image. The same ordering of spatial positions is used during training and sampling.

Codebook indices are shifted by $+1$, while index zero is reserved for the beginning-of-sequence (BOS) token. The resulting vocabulary contains 1,025 entries. No binary-code conversion is applied.

Autoregressive model architecture. We use an unconditional decoder-only causal Transformer with six pre-layer-normalization blocks, a hidden dimension of 512, and eight attention heads with 64 dimensions per head. Each block uses an MLP expansion ratio of four and ReLU activation. No dropout is used. Position information is provided by learnable absolute positional embeddings initialized to zero. The sequence length is 64.

The output projection has no bias and does not share weights with the token embedding. The model has approximately 20 million parameters and does not use class labels.

Autoregressive training. Each Transformer is trained from scratch using fused AdamW with a constant learning rate of 3×10^{-4} and weight decay of 0.01. The remaining optimizer settings use the PyTorch defaults. The batch size is 1,024, and training runs for 100 epochs over the 50,000 training sequences. The final partial batch is retained, resulting in 49 parameter updates per epoch and 4,900 updates in total. No learning-rate warmup or scheduling is used.

The training objective is cross-entropy under teacher forcing. The input sequence is shifted right by one position and prepended with the BOS token. Training uses FP16 mixed precision with `GradScaler` and `torch.compile`, with random seed 2047. Linear and token embedding weights are initialized from $\mathcal{N}(0, 0.02^2)$, while positional embeddings are initialized to zero. Apart from weight decay, no additional regularization is applied; in particular, no dropout or label smoothing is used. The model at the end of epoch 100 is used without checkpoint selection.

Autoregressive sampling and decoding. Starting from the BOS token, the Transformer generates 64 tokens autoregressively. Tokens are drawn by multinomial sampling from the softmax distribution at temperature 1.0, without top-k or top-p filtering. No class conditioning is used.

For each method, we generate 50,000 sequences in 500 batches of 100. The token offset of $+1$ is removed to recover the codebook indices, which are reshaped into 8×8 grids. The corresponding embeddings are retrieved from the fixed codebook, yielding latent tensors of shape $128 \times 8 \times 8$. These tensors are passed to the corresponding fixed decoder. Decoded images use the decoder's $\tanh$ output activation and are clamped to $[-1, 1]$.

As a separate memorization sanity check, we inspect one greedily decoded sample and 100 multinomially sampled images for exact matches against the training set. These additional samples are excluded from metric evaluation.

Diffusion model architecture. For continuous representations, we use an unconditional U-Net with a base width of 64 and two spatial downsampling stages, $8 \times 8 \to 4 \times 4 \to 2 \times 2$. The corresponding channel dimensions are 64, 128, and 256. Residual blocks use GroupNorm and SiLU activations, and timesteps are represented by sinusoidal embeddings. The network contains no attention layers and has approximately five million parameters. All variants operate on standardized, continuous, unquantized $3 \times 8 \times 8$ latent tensors.

Diffusion objective and noise schedule. The diffusion model predicts the Gaussian noise added to a standardized latent training example. Let $\widetilde{\mathbf{z}}_0$ denote a standardized latent and define

$$\widetilde{\mathbf{z}}_t = \sqrt{\bar{\alpha}_t}\,\widetilde{\mathbf{z}}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{9}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. We minimize the mean-squared error between the predicted and sampled noise:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\widetilde{\mathbf{z}}_0, t, \boldsymbol{\epsilon}}\left[\text{MSE}\left(\epsilon_\theta(\widetilde{\mathbf{z}}_t, t), \boldsymbol{\epsilon}\right)\right]. \tag{10}$$

We use $T = 1,000$ diffusion timesteps and a linear variance schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.

Diffusion training. Each diffusion model is trained using AdamW with weight decay of 0.01, a constant learning rate of 2×10^{-4}, and a batch size of 128. Training runs for 101 epochs, corresponding to approximately 39,500 parameter updates. We use FP32 precision without an exponential moving average of model parameters. The final checkpoint is used without checkpoint selection.

Diffusion sampling and decoding. Sampling uses ancestral DDPM sampling with all 1,000 timesteps and no guidance. For each method, 50,000 latent tensors are sampled in the standardized latent space. Before decoding, we invert the standardization using the statistics computed from that method's latent training dataset. The restored latent tensors are then decoded using the corresponding fixed decoder.

Reconstruction and generation metrics. Reconstruction FID (rFID) is computed between the reconstructions of the 50,000 CIFAR-10 training images and the original training image set. Generation FID is computed using `pytorch-fid`, comparing 50,000 generated images with the 50,000 CIFAR-10 training images. Thus, rFID evaluates reconstruction quality, whereas generation FID evaluates images sampled from the learned generative model.

Inception Score (IS) is computed using `torch-fidelity` with ten splits. The reported IS is the mean over these splits, and the $\pm$ value denotes the standard deviation across splits. Each result is obtained from a single run with one set of 50,000 generated images. The reported deviation therefore measures variation across IS splits, rather than across independent training runs.

Shared configurations and training budgets. Within the autoregressive group, VQ-VAE and PSDR-AD(VQ) use the same Transformer architecture, tokenization procedure, optimizer, training budget, and sampling configuration. Within the diffusion group, AE, GLO, and PSDR-AD use the same U-Net architecture, standardization procedure, optimizer, training budget, and sampling configuration. Standardization statistics are computed separately for each method's latent dataset. Generative model parameters are trained independently for each representation method.

The training budgets are 4,900 updates per autoregressive model and approximately 39,500 updates per diffusion model. Training losses are monitored, but convergence is not formally established. The comparison evaluates downstream generation under these shared training budgets within each generative model family.